16th International Conference on Project Management and Scheduling
Rome, April 17-20, 2018

CONFERENCE PROCEEDINGS

Proceedings of the 16th International Conference on Project Management and Scheduling
PMS 2018 - April 17-20, 2018 - Rome, Italy

ISBN: 9788894982022

Editors:
Massimiliano Caramia
Lucio Bianco
Stefano Giordani

Address of the Editors:
University of Rome "Tor Vergata"
Dipartimento di Ingegneria dell'Impresa
Via del Politecnico, 1
00133 Roma - Italy
Tel. +39 06 72597360
Fax. +39 06 72597305
email: caramia@dii.uniroma2.it

Published by:
TexMat
Via di Tor Vergata, 93-95
00133 Roma
Tel. +39 06 2023572
www.texmat.it
e-mail: info@texmat.it

Place and Date of Publication:
Rome (Italy), March 31, 2018

Preface

This volume contains the papers that will be presented at PMS 2018, the 16^{th} International Conference on Project Management and Scheduling to be held on April, 17-20 2018 in Rome - Italy.

The EURO Working Group on Project Management and Scheduling was established by Professors Luís Valadares Tavares and Jan Weglarz during the EURO VIII Conference, Lisbon, in September 1986. It was decided to organize a workshop every two years. Gathering the most promising theoretical and applied advances in Project Management and Scheduling, and assessing both the state-of-the-art of this field and its potential to support management systems are the main objectives of these workshops.

76 extended abstracts have been submitted to the PMS 2018 Confenrece. These valuable contributions were reviewed by 2 referees who are members of the International Program Committee and distinguished researchers of the associated fields. The proceedings at hand contain the 65 papers that were finally accepted for presentation at the conference. There papers involves 165 authors from 23 different countries.

The 16^{th} edition of PMS has four plenary speakers: Professor Jacques Carlier (Université de Technologie de Compiègne) will present the talk "Comparing event-node graphs with nonrenewable resources and activity-node graphs with renewable resources", Professor Erwin Pesch (University of Siegen) will discuss on "Optimization problems in intermodal transport", Professor Ruben Ruiz (University of Valencia) and Professor Erik Demeulemeester (Katholic University of Leuven) will delight us talking on "Simple metaheuristics for flowshop scheduling: all you need is local search" and "On the construction of optimal policies for the RCPSP with stochastic activity durations", respectively.

The scientific program and the social events will give to all the participants an opportunity to share research ideas and debate on recent advances on project managment and scheduling. I am sure that together we will contribute to make PMS 2018 a great success.

Welcome in Rome and have an enjoyable stay!

Rome, 17th April 2018 Massimiliano Caramia (Conference Chair)

Organizing Committee

Massimiliano Caramia (Chairman)	Università di Roma "Tor Vergata" (Italy)
Lucio Bianco	Università di Roma "Tor Vergata" (Italy)
Stefano Giordani	Università di Roma "Tor Vergata" (Italy)

Program Committee

Alessandro Agnetis	Università di Siena (Italy)
Ali Allahverdi	Kuwait University (Kuwait)
Christian Artigues	LAAS-CNRS (France)
Francisco Ballestrin	Universitat de València (Spain)
Fayez Boctor	Université Laval (Canada)
Jazek Błażewicz	Poznań University of Technology (Poland)
Massimiliano Caramia	Università di Roma "Tor Vergata" (Italy)
Jacques Carlier	Université de Technologie de Compiègne (France)
Erik Demeulemeester	Katholieke Universiteit Leuven (Belgium)
Joanna Józefowska	Poznań University of Technology (Poland)
Sigrid Knust	Universität Osnabrück (Germany)
Rainer Kolisch	Technische Universität München (Germany)
Mikhail Kovalyov	National Academy of Sciences of Belarus (Belarus)
Wieslaw Kubiak	Memorial University (Canada)
Erwin Pesch	Universität Siegen (Germany)
Chris Potts	University of Southampton (United Kingdom)
Rubén Ruiz	Universitat Politècnica de València (Spain)
Avraham Shtub	Technion - Israel Institute of Technology (Israel)
Funda Sivrikaya Şerifoğlu	Istanbul Bilgi Üniversitesi (Turkey)
Vincent T'Kindt	Université François Rabelais Tours (France)
Norbert Trautmann	Universität Bern (Switzerland)
Mario Vanhoucke	Ghent University (Belgium)
Jan Weglarz	Poznań University of Technology (Poland)
Jürgen Zimmermann	Technische Universität Clausthal (Germany)
Linet Özdamar	Yeditepe Üniversitesi (Turkey)

Table of Contents

Author Index

Keyword Index

Scheduling energy-consuming jobs on parallel machines with piecewise-linear costs and storage resources

A lot-sizing and scheduling perspective

Nabil Absi[1], Christian Artigues[2], Safia Kedad-Sidhoum[3], Sandra U. Ngueveu[2], Janik Rannou[2], et Omar Saadi[3]

[1] Mines Saint-Etienne and UMR CNRS 6158 LIMOS, Gardanne, France
absi@emse.fr
[2] LAAS-CNRS, Université de Toulouse, CNRS, INP, Toulouse, France
sungueve,artigues@laas.fr
[3] Sorbonne Université, UPMC, UMR 7606, LIP6, Paris, F-75005, France
safia.kedad-sidhoum@lip6.fr

Keywords: Scheduling under energy constraints and costs, Mixed-integer programming, Lot-sizing, Dynamic programming

1 Problem formulation and complexity

In this paper, a scheduling and energy source assignment problem is studied. The problem is abstracted from several applications including data centers (Guérout *et al.* 2017), smart buildings (Desdouits *et al.* 2016), hybrid vehicles (Caux *et al.* 2017, Ngueveu *et al.* 2017) and manufacturing (Haouassi *et al.* 2016). A set of jobs $\mathcal{J}$ has to be scheduled on a set of energy consuming machines $\mathcal{M}$. The energy consumed by a machine k has a fixed part $\underline{D}_k$ depending on whether the machine is switched on or off and a variable part depending on the tasks that are currently in process. Each task j requires an amount D_{jk} of energy at each time period it is processed on machine k. We are interested in optimizing the total energy cost induced by energy production required to satisfy the total energy demand of a given schedule over a fixed discrete horizon $T = \{1, \ldots, |T|\}$. At each time period, the energy required by the schedule can be supplied by two energy sources. One source is reversible, such as batteries and super-capacitors. Such a source is able not only to produce energy but also to retrieve it assuming a limited capacity $\overline{Q}$. Such a resource is equivalent to a continuously-divisible storage resource in the scheduling terminology. The second source is a non-reversible source, only able to produce energy, such as the external power grid (assuming here that energy cannot be sold to the network). This source is assumed of infinite capacity, but its usage comes with a cost (see below). In this paper, we consider a parallel machine environment, such that p_j units of each task j must be scheduled preemptively inside a time window $[r_j, d_j]$ and have to be assigned at each time period to one and only one machine. Job units also require resources from a set $\mathcal{R}$ (e.g. CPU, RAM) on their assigned machines. A job requires a non-negative amount c_{jr} on each resource $r \in \mathcal{R}$. On each machine k, resource r is available in a limited amount C_{kr}. The energy cost for a period of the scheduling horizon is a piecewise linear (PWL) function f_t, $t \in T$, of the required amount of energy on the non-reversible source. The PWL function is assumed to be time dependent. This allows to model time dependent electricity prices as well as previsions of photovoltaic production since the cost can be zero up to a required amount of energy corresponding to the expected photovoltaic production on the considered time period. We introduce variables $y_{jkt} \in \{0, 1\}$ indicating whether one unit of job j is assigned to machine k at time t and $z_{kt} \in \{0, 1\}$ indicating whether machine k is switched on at time t. Continuous variables are used for energy amounts: $x_t \geq 0$ gives the amount of energy used on the non-reversible source at time t, $s_t \geq 0$ is the level of energy remaining in the

non-reversible source at time t (s_0 is a constant indicating the initial energy level). Based on these variables, we define a MILP formulation of the problem aiming at minimizing the total energy cost.

$$\text{minimize} \sum_{t\in T} f_t(x_t) \tag{1}$$

subject to

$$\sum_{k\in\mathcal{M}} y_{jkt} \leq 1, \qquad j \in \mathcal{J}, t \in T \tag{2}$$

$$\sum_{k\in\mathcal{M}} \sum_{t\in T} y_{jkt} = p_j, \qquad j \in \mathcal{J} \tag{3}$$

$$\sum_{j\in\mathcal{J}} c_{jr} y_{jkt} \leq C_{kr}, \qquad k \in \mathcal{M}, r \in \mathcal{R}, t \in T \tag{4}$$

$$z_{kt} - y_{jkt} \geq 0, \qquad k \in \mathcal{M}, j \in \mathcal{J}, t \in T \tag{5}$$

$$x_t + s_t - s_{t-1} - \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{M}} D_{jk} y_{jkt} - \sum_{k\in\mathcal{M}} \underline{D}_k z_{kt} = 0, \; t \in T \tag{6}$$

$$s_{|T|} - s_0 \geq 0, \tag{7}$$

$$s_t - \overline{Q} \leq 0, \qquad t \in T \tag{8}$$

$$s_t \geq 0 \qquad t \in T \tag{9}$$

$$x_t \geq 0 \qquad t \in T \tag{10}$$

$$y_{jkt} \in \{0,1\} \qquad j \in \mathcal{J}, k \in \mathcal{M}, t \in T \tag{11}$$

$$z_{kt} \geq 0 \qquad k \in \mathcal{M}, t \in T \tag{12}$$

The total energy cost minimization objective (1) is considered. Constraints (2) state that a job may be in process on only one machine at a given time. Constraints (3) enforce each unit of a job to be scheduled on one machine. Constraints (4) are the resource constraints. Constraints (5) enforce a machine to be switched on at each time it processes at least one job. Constraints (6) are the energy balance constraints between the schedule demand, the energy provided by the non-reversible source (x_t) and the energy taken from or provided to the reversible source ($s_t - s_{t-1}$, that can be positive or negative). Constraint (7) enforces the final energy level in the reversible source to be at least the initial one. Constraints (8) are the reversible source capacity constraints (storage limit). Scheduling preemptive jobs with PWL energy costs is NP-hard, even with an unlimited number of machines and single non-reversible source (Ngueveu *et al.* 2016). Therefore the proposed MILP becomes intractable as the problem size increases.

2 A lot-sizing and scheduling matheuristic

We propose a natural decomposition of the problem. Let $d_t = \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{M}} D_{jk} y_{jkt} + \sum_{k\in\mathcal{M}} \underline{D}_k z_{kt}$ denote the total energy demand of a fixed schedule at time t. Then constraints (6) can be rewritten

$$x_t + s_t - s_{t-1} - d_t = 0, \quad t \in T \tag{13}$$

Now observe that for fixed d_t, problem LSP: $\min \sum_{t\in T} f_t(x_t)$ s.t. (7–10), (13) is a single-item (continuous) lot sizing problem with PWL production costs where d_t is the demand for period t, x_t is the production variable for period t, and s_t is the variable giving the amount of inventory at the end of period t. In Absi *et al.* (2017), the problem is shown to be NP-hard but for integer inventory levels, a pseudo-polynomial dynamic programming (DP) algorithm of complexity $O(T^2\overline{q}\overline{d})$ where $\overline{d}$ is the average demand and $\overline{q}$ is the average number of breakpoints of the PWL functions f_t is given, generalizing the results of Shaw and Wagelmans (1998). On the other hand, if the variables s_t are fixed, by performing change of variables $x_t \leftarrow x_t - s_t + s_{t-1}$ for all t, we obtain

$$x_t = \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{M}} D_{jk} y_{jkt} + \sum_{k\in\mathcal{M}} \underline{D}_k z_{kt}, \quad t \in T \tag{14}$$

and problem MSP: $\min \sum_{t\in T} f'_t(x_t) = \sum_{t\in T} f_t(x_t - s_t + s_{t-1})$ s.t. (2–5), (10–12), (14), which is a parallel machine scheduling problem with PWL costs and a single non-reversible source, NP-hard in the strong sense as shown in Ngueveu *et al.* (2016).

A matheuristic is obtained by solving alternatively MSP and LSP. Starting with initial reversible source transferred amounts $s_t - s_{t-1} = 0$ for all t, MSP is solved and the output energy demand $(d_t)_{t\in\mathcal{T}}$ is used as input of LSP. The output inventory levels are used to update the PWL functions f'_t. Then, MSP is solved again, and so on until no improvement is observed in the objective function.

We first compare the MILP (solved with Cplex) and the pseudo polynomial DP algorithm with fixed demands (only LSP), see table 1. The merits and the drawback of the two approaches are illustrated on 4 instances with $T = 1000$, $\overline{q} = 10$ breakpoints, and varying average maximal capacities Q and demands $\overline{d}$. Under a 300s time limit for the MILP, the CPU times (in seconds) and obtained costs are compared. It appears that no algorithm dominates the other one in terms of CPU time, while the DP is more impacted by the maximal available capacity for the reversible source. However the MILP shows a more erratic and unpredictable behavior. Note that due to the integrity requirement of the inventory levels, the DP costs are higher than the MILP costs.

Table 1. Comparison of MILP and DP on the lot sizing problem (LSP)

T	$\overline{q}$	Q	$\overline{d}$	MILP cost	MILP CPU	DP cost	DP CPU
1000	10	1000	100	6661	300	6916	14.16
1000	10	10000	100	4396	2.90	4508	279
1000	10	100	2000	975523	0.57	975617	1.31
1000	10	1000	2000	940165	300	937886	15

Finally we compare the MSP/LSP decomposition matheuristic (MH) with the full MILP to solve the global problem. We also illustrate the cost and CPU time differences on 4 instances with varying horizon, number of machines, number of jobs (see table 2). On 3 instances (marked with a $*$) the full MILP reached the time limit. The matheuristic is only slightly faster than the MILP except on the last instance, where it is much faster. The costs can be close to the MILP ones although important gaps can also be observed. In parenthesis, the maximal CPU time per iteration and the iteration number assigned to MH is indicated. A closer analysis of the CPU times between MSP and LSP reveals that 90% of the CPU time is spent on solving the scheduling problem MSP.

Table 2. Comparison of full MILP and matheuristic on the global problem

T	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{J}\rvert$	MILP cost	MILP CPU	MH cost	MH CPU
30	2	50	11280	450^*	11280	154 (150× 3)
60	4	150	25966	2000^*	26183	1893 (1000× 2)
120	2	150	5944	2000^*	6972	1855 (1000× 2)
120	1	150	11255	1353	11265	228 (200× 10)

3 Conclusion

We have proposed an original lot sizing and scheduling decomposition approach to solve an energy management and scheduling problem on parallel machines. The lot sizing subproblem can be solved considerably faster than the scheduling subproblem and consequently, further research on the problem should focus on improvement of the scheduling solution procedure. To improve the decomposition heuristic, optimality cuts issued from lot sizing could be designed for the scheduling problem. Another interesting issue is to consider a non ideal yield of the reversible source. In practice, due to energy conversion and losses only a fraction of $s_t - s_{t-1}$ is available to fulfill the demand and a possibly non linear efficiency function $g(s_t - s_t - 1)$ has to be used to compute the obtained energy. It remains to know whether efficient lot sizing procedure can be devised with such efficiency functions.

Acknowledgements

This research benefited from the support of the FMJH Program PGMO and from the support of EDF, Thales, Orange. It also benefited from funding from the Cellule Energie of the CNRS.

References

Absi N., C. Artigues, S. Kedad-Sidhoum, S.U. Ngueveu and O. Saadi, 2017, "Lot-sizing models for energy management". *LAAS report*.

Caux S., Y. Gaoua, and P. Lopez, 2017, "A combinatorial optimisation approach to energy management strategy for a hybrid fuel cell vehicle". *Energy*, Vol. 133, pp.219-230.

Desdouits C., M. Alamir, R. Giroudeau and C. Le Pape, 2016, "The Sourcing Problem - Energy Optimization of a Multisource Elevator", *ICINCO*, Vol. 1, pp. 19-30.

Guérout T., Y. Gaoua, C. Artigues, G. Da Costa, P. Lopez and T. Monteil, 2017, "Mixed integer linear programming for quality of service optimization in Clouds" Future Generation Computer Systems 71: 1–17.

Haouassi M., C. Desdouits, R. Giroudeau and C. Le Pape, 2016, "Production scheduling with a piecewise-linear energy cost function" *IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1-8.

Ngueveu S. U., C. Artigues and P. Lopez, 2016, "Scheduling under a non-reversible energy source: An application of piecewise linear bounding of non-linear demand/cost functions", *Discrete Applied Mathematics*, Vol. 208, pp. 98-113.

Ngueveu S.U., S. Caux, F. Messine and M. Guemri, 2017, "Heuristics and lower bound for energy management in hybrid-electric vehicles", *4OR*, to appear.

Shaw D. X., A. P. Wagelmans, 1998, "An algorithm for single-item capacitated economic lot sizing with piecewise linear production costs and general holding costs", *Management Science*, Vol. 44, pp. 831-838.

The truck scheduling problem at cross docking terminals: Formulations and valid inequalities

A. Agnetis[1], L. Berghman[2] and C. Briand[3]

[1] A. Agnetis, Università degli Studi di Siena, DIISM, Siena, Italy
agnetis@diism.unisi.it
[2] L. Berghman, Université de Toulouse - Toulouse Business School,
20 BD Lascrosses – BP 7010, 31068 Toulouse Cedex 7, France
l.berghman@tbs-education.fr
[3] C. Briand, LAAS-CNRS, Université de Toulouse, UPS, Toulouse, France
briand@laas.fr

Keywords: crossdocking, truck scheduling, parallel machine scheduling, integer linear programming.

1 Introduction

Crossdocking is a warehouse management concept in which items delivered to a warehouse by inbound trucks are immediately sorted out, reorganized based on customer demands and loaded into outbound trucks for delivery to customers, without requiring excessive inventory at the warehouse (J. van Belle *et al.* 2012). If any item is held in storage, it is usually for a brief period of time that is generally less than 24 hours. Advantages of crossdocking can accrue from faster deliveries, lower inventory costs, and a reduction of the warehouse space requirement (U.M. Apte and S. Viswanathan 2000, N. Boysen *et al.* 2010). Compared to traditional warehousing, the storage as well as the length of the stay of a product in the warehouse is limited, which requires an appropriate coordination of inbound and outbound trucks (N. Boysen 2010, W. Yu and P.J. Egbelu 2008).

The truck scheduling problem, which decides on the succession of truck processing at the dock doors, is especially important to ensure a rapid turnover and on-time deliveries. The problem studied concerns the operational level: trucks are allocated to the different docks so as to minimize the storage usage during the product transfer. The internal organization of the warehouse (scanning, sorting, transporting) is not explicitly taken into consideration. We also do not model the resources that may be needed to load or unload the trucks, which implies the assumption that these resources are available in sufficient quantities to ensure the correct execution of an arbitrary docking schedule.

In this abstract, we present a time-indexed formulation, a network formulation and some valid inequalities. Experimental results will be presented during the talk at the conference.

2 Detailed problem statement

We examine a crossdocking warehouse where incoming trucks $i \in I$ need to be unloaded and outgoing trucks $o \in O$ need to be loaded (where I is the set containing all inbound trucks while O is the set containing all outbound trucks). The warehouse features n docks that can be used both for loading and unloading. The processing time of truck $j \in I \cup O$ equals p_j. This processing time includes the loading or unloading but also the transportation of goods inside the crossdock and other handling operations between dock doors. It is assumed that there is sufficient workforce to load/unload all docked trucks at the same time. Hence, a truck assigned to a dock does not wait for the availability of a material handler.

The products on the trucks are packed on unit-size pallets, which move collectively as a unit: re-packing inside the terminal is to be avoided. Each pallet on an inbound truck i needs to be loaded on an outbound truck o, which gives rise to a start-start precedence constraint $(i, o) \in P \subset I \times O$, with P the set containing all couples of inbound trucks i and outbound trucks o that share a precedence constraint. Each truck j has a release time r_j (planned arrival time) and a deadline $\tilde{d}_j$ (its latest departure time).

Products can be transshipped directly from an inbound to an outbound truck if the outbound truck is placed at a dock. Otherwise, the products are temporarily stored and will be loaded later on. Each couple $(i, o) \in P$ has a weight w_{io}, representing the number of pallets that go from inbound truck i to outbound truck o. The problem aims at determining time-consistent start times s_i and s_o of unload and load tasks i and o so as to minimize the weighted sum of sojourn times of the pallets stocked in the warehouse. Remark that the time spent by a pallet in the storage area is equal to the flow time of the pallet: the difference between the start of loading the outbound trailer and the start of unloading the inbound trailer (i.e., $s_o - s_i$).

Our problem can be modeled as a parallel machine scheduling problem with release dates, deadlines, and precedence constraints, denoted by $Pm|r_i, \tilde{d}_i, prec|-$. As this problem is a generalization of the $1|r_j, \tilde{d}_j|-$ problem which is NP-complete (J.K. Lenstra *et al.* 1977), even finding a feasible solution for the problem is NP-complete.

3 Time-indexed formulation

A time-indexed formulation discretizes the continuous time space into periods $\tau \in \mathcal{T}$ of a fixed length. Let period τ be the interval $[t-1, t[$. It is well known that time-indexed formulations perform well for scheduling problems because the linear programming relaxations provide strong lower bounds (M. E. Dyer and L. A. Wolsey 1990).

For all inbound trucks $i \in I$ and for all time periods $\tau \in \mathcal{T}_i$, we have

$$x_{i\tau} = \begin{cases} 1 & \text{if the unloading of inbound truck } i \text{ is} \\ & \text{started during time period } \tau, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

with $\mathcal{T}_i = \{r_i + 1, r_i + 2, \ldots, \tilde{d}_i - p_i + 1\}$, the relevant time window for inbound truck i. Additionally, for all outbound trucks $o \in O$ and for all time periods $\tau \in \mathcal{T}_o$, we have

$$y_{o\tau} = \begin{cases} 1 & \text{if the loading of outbound truck } o \text{ is} \\ & \text{started during time period } \tau, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

with $\mathcal{T}_o = \{r_o + 1, r_o + 2, \ldots, \tilde{d}_o - p_o + 1\}$, the relevant time window for outbound truck o.

A time-indexed formulation for the considered truck scheduling problem is the following:

$$\min \quad z = \sum_{(i,o)\in P} \sum_{\tau \in \mathcal{T}} w_{io}\tau \left(y_{o\tau} - x_{i\tau}\right) \tag{3}$$

subject to

$$\sum_{\tau \in \mathcal{T}_i} x_{i\tau} = 1 \qquad \forall i \in I \tag{4}$$

$$\sum_{\tau \in \mathcal{T}_o} y_{o\tau} = 1 \qquad \forall o \in O \tag{5}$$

$$\sum_{\tau \in \mathcal{T}} \tau \left(x_{i\tau} - y_{o\tau} \right) \leq 0 \qquad \forall (i,o) \in P \tag{6}$$

$$\sum_{i \in I} \sum_{u=\tau-p_i+1}^{\tau} x_{iu} + \sum_{o \in O} \sum_{u=\tau-p_o+1}^{\tau} y_{ou} \leq n \qquad \forall \tau \in \mathcal{T} \tag{7}$$

$$x_{i\tau} \in \{0,1\} \qquad \forall i \in I; \forall \tau \in \mathcal{T}_i \tag{8}$$

$$y_{o\tau} \in \{0,1\} \qquad \forall o \in O; \forall \tau \in \mathcal{T}_o \tag{9}$$

The objective function (3) minimizes the total weighted usage of the storage area. Constraints (4) and (5) demand each truck to be assigned to exactly one gate. Constraints (6) ensure that if there exists a precedence constraint between inbound truck i and outbound truck o, then o cannot be processed before i. Constraints (7) enforce the capacity of the docks for any period $\tau \in \mathcal{T}$.

4 Network formulation

The formulation below makes use of the well-known concept of a *critical set* (see e.g. (M. Lombardi and M. Milano 2012)), i.e., a set of tasks which cannot all be performed in parallel. We introduce a pair of disjunctive precedence constraints for each task pair $(u,v) \in (I \cup O)^2$ with $[r_u, d_u] \cap [r_v, d_v] \neq \emptyset$, belonging to a critical set (the set of these task pairs is further referred as C). We let E be the set of all critical sets. Additionally, we also refer to e_k as a specific critical set of k elements and to $E^m \subset E$ as the set of all minimal critical sets. To model the disjunction, binary variables α_{uv} are introduced such that $\alpha_{uv} = 1 \equiv u \prec v$. Our problem can be modelled as follows:

$$\min \sum_{o \in O} s_o p_o - \sum_{i \in I} s_i p_i \tag{10}$$

subject to

$$s_o - s_i \geq 0 \qquad \forall (i,o) \in P \tag{11}$$

$$s_v - s_u + \alpha_{uv}(M_{uv} - p_u) \geq M_{uv} \qquad \forall (u,v) \in C \tag{12}$$

$$s_v - s_0 \geq r_v \qquad \forall v \in I \cup O \tag{13}$$

$$s_0 - s_u \geq p_u - d_u \qquad \forall u \in I \cup O \tag{14}$$

$$\sum_{(u,v) \in e_{n+1}} \alpha_{uv} \geq 1 \qquad \forall e_{n+1} \in E^m \tag{15}$$

$$\alpha_{uv} \in \{0,1\} \qquad \forall (u,v) \in C \tag{16}$$

$$s_u \in \mathcal{R} \qquad \forall u \in I \cup O \tag{17}$$

with $M_{uv} = p_u - d_u + r_v$ and 0 a dummy vertex, which is introduced to represent the time origin $s_0 = 0$. Note that obviously $\alpha_{uv} + \alpha_{vu} \leq 1$, even though this constraint is not mandatory for the formulation accuracy.

Remark that constraints (15) express the limited capacity of the crossdocking terminal. Their number is exponential, as the number of minimal critical sets is exponential. Even

though including only minimal critical sets is sufficient, we can also consider the non-minimal critical sets, generalizing (15) as:

$$\sum_{(u,v)\in e_k} \alpha_{uv} \geq k - n \qquad \forall e_k \in E \tag{18}$$

with $n + 1 \leq k \leq \left| I \cup O \right|$.

We will show that this family of constraints can be strengthened by augmenting the right-and-side, so that it can be replaced by:

$$\sum_{(u,v)\in e_k} \alpha_{uv} \geq \frac{(k-n)(k-n+1)}{2} \qquad \forall e_k \in E \tag{19}$$

5 Solving methodology framework

Intuitively, only a small number of constraints (19) may be required into the formulation to obtain a feasible (optimal) solution. Consequently, we consider the following cutting-plane method which consists in introducing progressively constraints of type (19). First, the problem is solved without any constraint of type (19) using a MILP solver. Then, violated constraints of type (19) are added for some $k > n$ involving a critical set e_k and the solver is launched again. Now, each time a feasible solution is found by the solver in course of the branch-and-cut process, violated constraints of type (19) are added on-the-fly. Note that if such a solution is feasible with respect to the resource capacity, then it is an upper bound of the initial problem. When the MILP solver ends up with an optimal solution also capacity-feasible, it is also optimal. Otherwise, violated constraints of type (19) can be added again and another MILP is ran. Within various computational time limitation assumptions, the above methodology will be compared in terms of performance (quality of the upper and lower bounds) with the time-indexed linear programming approach on a set of artificial problem instances.

References

U.M. Apte and S. Viswanathan, Effective cross docking for improving distribution efficiencies, International Journal of Logistics: Research and Applications, 3 (3), 291–302.

N. Boysen, Truck scheduling at zero-inventory cross docking terminals, Computers & Operations Research, 37, 32–41.

N. Boysen and M. Fliedner and A. Scholl, Scheduling inbound and outbound trucks at cross docking terminals, OR Spectrum, 32, 135–161.

M. E. Dyer and L. A. Wolsey, Formulating the single machine sequencing problem with release dates as a mixed integer problem, Discrete Applied Mathematics, 26, 255–270.

J.K. Lenstra and A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics, 1, 343–362.

M. Lombardi and M. Milano, A min-flow algorithm for Minimal Critical Set detection in Resource Constrained Project Scheduling, Artificial Intelligence, 182-183, 58–67.

J. van Belle and P. Valckenaers and D. Cattrysse, Cross docking: State of the art, Omega,40 (6), 827–846.

W. Yu and P.J. Egbelu, Scheduling of inbound and outbound trucks in cross docking systems with temporary storage, International Journal of Production Economics,, 184, 377–396.

The Price of Fairness in a Two-Agent Single-Machine Scheduling Problem

Alessandro Agnetis[1], Bo Chen[2], Gaia Nicosia[3], et Andrea Pacifici[4]

[1] University of Siena, Italy
agnetis@diism.unisi.it
[2] University of Warwick, UK
Bo.Chen@wbs.ac.uk
[3] University of Roma Tre, Italy
nicosia@dia.uniroma3.it
[4] University of Roma Tor Vergata, Italy
pacifici@disp.uniroma2.it

Keywords: multiagent scheduling, bargaining problems, fairness.

1 Introduction

Fairness issues arise in several real-world contexts and are investigated in different research areas of mathematics, game theory and operations research. In classical two-player bargaining problems, the notion of *fairness* has been introduced to compare the utility of one agent to the other agent's.

Here we address fairness concepts in the context of *classical single-machine scheduling*. There are two agents, called A and B, each owning a set of jobs, which must be scheduled on a common processing resource. Each schedule implies a certain *utility* for each agent. We adopt the sum of the agents' utilities as an index of collective satisfaction (*system utility*) and we refer to any solution maximizing system utility as a *system optimum*. Even if it maximizes system utility, a system optimum may well be highly unbalanced and therefore possibly unacceptable by the worse-off agent. Rather, a solution that incorporates some criterion of fairness may be more acceptable. The problem we investigate is how much system utility must be sacrificed in order to reach a fair solution. The quantity that captures this concept is known as *price of fairness* (PoF). Given an instance of a bargaining problem, and a certain definition of fair solution, PoF is the relative loss in overall utility of a fair solution with respect to the system optimum. Depending on the specific problem setting and also on the agent perception of what a fair solution is, assorted definitions of fair solution can be found in the scientific literature. In our study we focus on two of the most popular fairness definitions.

Caragiannis et al. (Caragiannis et al. 2012) introduced the concept of PoF in the context of fair allocation problems: In particular, they compare the value of total agents' utility in a system optimum with the maximum total utility obtained over all fair solutions (they make use of several notions of fairness, namely proportionality, envy-freeness and equitability). In (Bertsimas *et al.* 2011), Bertsimas *et al.* focus on proportional fairness and max-min fairness and provide a tight characterization of PoF for a broad family of allocation problems with compact and convex agents' utility sets. In (Nicosia et al. 2017) the authors prove a number of properties on the price of fairness which hold for any general multi-agent problem without any special assumption on the agents' utilities, focusing on max-min, Kalai-Smorodinski and proportional fairness. Situations in which the agents pursue the minimization of their costs (rather than the maximization of their utility) have been dealt with by Ertogral and Wu (Ertogral and Wu 2000), who derive a measure of fairness among a set of supply chain members. Another example of fairness in the presence of cost allocations can be found in

(Bohm and Larsen 1994). Our view of fairness is related to scheduling problems, but it is worth observing that fairness issues arise in other contexts, such as fair representation problems (Balinski and Young 2001), or the apportionment problem (Lucas 1983). These need to be dealt with by different methods and algorithms than those presented in this talk.

2 Scheduling bargaining problems

Bargaining problems address situations in which two players (*agents*) are faced with a set of possible agreements (*resource set*), and must reach a compromise over one of them. Here we are concerned with the following *scheduling bargaining problem*. There are two agents, namely A and B. Each agent has a set of jobs, which have to be processed by a single machine. Agents A (B) has jobs $J_1^A, \ldots, J_{n_A}^A$ ($J_1^B, \ldots, J_{n_B}^B$), of length $p_1^A, \ldots, p_{n_A}^A$ ($p_1^B, \ldots, p_{n_B}^B$). Let $P^A = \sum_{j=1}^{n_A} p_j^A$ ($P^B = \sum_{j=1}^{n_B} p_j^B$). Jobs cannot be preempted, and the machine can only process one job at a time. We use the terms A-jobs and B-jobs to refer to the two agents' respective jobs.

Given a feasible schedule σ, we let $f^A(\sigma)$ and $f^B(\sigma)$ denote the *cost values* for the two agents. Here we consider as resource set the set Σ_P of *Pareto optimal* schedules, as they include all sensible compromise schedules. For each $\sigma \in \Sigma_P$, we want to define utility values $u^A(\sigma)$ and $u^B(\sigma)$, so that, for $i = A, B$, $u^i(\sigma) \geq 0$ and $u^i(\sigma)$ increases as $f^i(\sigma)$ decreases. For this purpose, we propose the following definition of utility. Let $f_\infty^i = \max\{f^i(\sigma) | \sigma \in \Sigma_P\}$ (for regular functions $f^i(\sigma)$, this is the minimum cost the agent i bears if its jobs are scheduled *after* all the jobs of the other agent). Then

$$u^i(\sigma) = f_\infty^i - f^i(\sigma) \quad i = A, B$$

In other words, we consider that an agent's utility is the *saving* achieved with respect to the worst schedule for that agent. We also let $U(\sigma) = u^A(\sigma) + u^B(\sigma)$ denote the overall utility of schedule σ and let σ^* denote the schedule that maximizes $U(\sigma)$ (*system optimum*), i.e.

$$U(\sigma^*) = \max_{\sigma \in \Sigma_P} \{U(\sigma)\}.$$

In the following we also let $f^{i*} = \min\{f^i(\sigma) | \sigma \in \Sigma_P\}$.

As for the definition of a fair solution, we consider the following two concepts.

1. *Kalai-Smorodinsky* solution (Kalai and Smorodinsky 1975). Given $\sigma \in \Sigma_P$, let

$$\bar{u}^i(\sigma) = \frac{u^i(\sigma)}{f_\infty^i - f^{i*}}$$

be the *normalized utility* of σ for agent i. A *Kalai-Smorodinsky schedule* σ_{KS} (briefly, KS schedule) is defined as

$$\sigma_{KS} = \arg\max_{\sigma} \min_{i=A,B} \{\bar{u}^i(\sigma)\}.$$

The idea is that in σ_{KS} the normalized utility of the agent who is worse-off is maximized. So, in σ_{KS} the two agents' normalized utility values are typically quite close. Obviously, under this definition a KS schedule always exists. In the literature, another definition of fair solution is the *max-min* solution (Bertsimas *et al.* 2011). Kalai-Smorodinsky and max-min solutions coincide if $f_\infty^A - f^{A*} = f_\infty^B - f^{B*}$.

2. *Proportionally fair* solution. A schedule σ_{PF} is proportionally fair if, for any other Pareto optimal schedule σ, it holds

$$\left(\frac{u^A(\sigma) - u^A(\sigma_{PF})}{u^A(\sigma_{PF})}\right) + \left(\frac{u^B(\sigma) - u^B(\sigma_{PF})}{u^B(\sigma_{PF})}\right) \leq 0. \quad (1)$$

The idea behind such definition is the following. When considering moving from schedule σ_{PF} to any other schedule σ, the relative benefit that one agent may obtain is balanced by a not smaller utility decrease for the other agent. This is actually the same rationale behind the concept of Nash solution (Nash 1950). However, the Nash solution was introduced only with respect to compact and convex resource sets, while Definition (1) is more general. In fact, while a Nash solution always exists, a proportionally fair solution may not exist. If it does exist, then it coincides with the solution maximizing the product of utilities, and hence, if the resource set is compact and convex, with the Nash solution.

Our study investigates how much (global) utility should be given up in order to have a fair solution. This is captured by the *price of fairness*, defined as the relative loss of utility in a fair solution with respect to maximum utility. Notice that there may be more than one fair solution, differing in terms of global utility. Here we adopt the same viewpoint as in (Karsu and Morton 2015, Naldi et al. 2016), i.e., whenever this occurs, we measure the price of fairness with respect to the *best* fair solution. In formal terms, letting $\mathcal{I}$ denote the set of all instances of a given problem, I one of them, $\sigma^*(I)$ the system optimum, Σ_F the set of all fair schedules and $\sigma_F(I)$ one of them, we define the price of fairness as:

$$PoF = \sup_{I \in \mathcal{I}} \left\{ \min_{\sigma_F \in \Sigma_F} \left\{ \frac{U(\sigma^*(I)) - U(\sigma_F(I))}{U(\sigma^*(I))} \right\} \right\}. \quad (2)$$

Notice that this is a similar definition to the well-known *Price of Stability* (Anshelevich et al. 2004), replacing the role of Nash equilibrium with fairness. Hereafter, we indicate with PoF_{KS} and PoF_{PF} the price of Kalai-Smorodinsky fairness and proportional fairness, respectively.

3 Scenario addressed

In this talk we address the value of PoF in the following scheduling scenario. Agent A pursues the minimization of the sum of its jobs' completion times, while agent B is interested in minimizing the maximum tardiness of its jobs with respect to a common due date d. Following the usual Graham's notation, we denote this scenario as $1|d_j^B = d|\sum C_j^A, T_{\max}^B$. Note that this scenario includes the case in which B wants to minimize its jobs' makespan, obtained for $d = 0$. Note that in this scenario, in any Pareto optimal solution all B-jobs are scheduled consecutively, so one can assume that B owns a single job of length $P^B = \sum_{j=1}^{n_B} p_j^B$. In this scenario, the values $f^{A*}, f_\infty^A, f^{B*}, f_\infty^B$ can all be easily computed. In fact, f^{A*} is the total completion time of the A-jobs when they are sequenced in SPT order and $f_\infty^A = f^{A*} + n_A P^B$, while $f^{B*} = \max\{0, P^B - d\}$ and $f_\infty^B = \max\{0, P^A + P^B - d\}$.

Our contributions to this scenario are summarized in Table 1. In particular, we show that $PoF_{KS} = 2/3$ and $PoF_{PF} = 1/2$. Moreover, we show that, if the A-jobs are already ordered by nondecreasing length, in time $O(\log n_A)$ a proportionally fair solution can be computed or proved that it does not exist.

Table 1. Results for scenario $1|d_j^B = d|\sum C_j^A, T_{\max}^B$ (*) if A-jobs are given in SPT order.

	Proportionally fair solution	**Kalai-Smorodinsky solution**
PoF value	1/2	2/3
Existence	Established in $O(\log n_A)^*$	(always exists)

References

Anshelevich, E., Dasgupta, A., Kleinberg, J., Tardös, E., Wexler, T. and T. Roughgarden (2004), The Price of Stability for Network Design with Fair Cost Allocation, in 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 59-73.

Balinski, M.L., Young H.P. (2001), *Fair Representation: Meeting the Ideal of One Man, One Vote*, Brookings Institution Press, Washington.

Bertsimas D., V. Farias, N. Trichakis (2011), The price of fairness, *Operations Research*, 59(1), 17–31.

Bohm, P., B. Larsen (1994), Fairness in a tradeable-permit treaty for carbon emissions reductions in Europe and the former Soviet Union, *Environmental and Resource Economics*, 4, 219–239.

Caragiannis I., C. Kaklamanis, P. Kanellopoulos, M. Kyropoulou (2012), The efficiency of fair division, 2012, *Theory of Computing Systems*, 50(4), 589–610.

Ertogral K., D. Wu (2000), "Auction-theoretic coordination of production planning in the supply chain", *IIE Transactions*, 32(10), 931-940.

Kalai E., M. Smorodinsky (1975), Other solutions to Nash bargaining problem,*Econometrica*, 43, 513–518.

Karsu Ö., A. Morton (2015), Inequity averse optimization in operational research, *European Journal of Operational Research*, 245(2), 343–359.

Lucas, W.F., (1983), The Apportionment Problem, in: Brams S.J., Lucas W.F., Straffin P.D. (eds), *Political and Related Models*, Modules in Applied Mathematics. Springer, New York, NY, 358-396.

Naldi M., G. Nicosia, A. Pacifici, U. Pferschy (2016), Maximin Fairness in Project Budget Allocation, *Electronic Notes in Discrete Mathematics*, 55, 65–68.

Nash, J. (1950), The bargaining problem, *Econometrica*, 18(2), pp. 155–162.

Nicosia G., A. Pacifici, U. Pferschy, 2017, Price of Fairness for allocating a bounded resource, *European Journal of Operational Research*, 257, pp. 933-943.

A MILP formulation for an operating room scheduling problem under sterilizing activities constraints

H. Al Hasan[1,2,3], C. Guéret[1], D. Lemoine[2] and D. Rivreau[3]

[1] Université d'Angers, LARIS, Angers, France
christelle.gueret@univ-angers.fr
[2] IMT Atlantique, LS2N, Nantes, France
david.lemoine@imt-atlantique.fr
[3] Université Catholique de l'Ouest, LARIS, Angers, France
alhasan, rivreau@uco.fr

Keywords: operating rooms in health services, sterilization unit, scheduling, MILP

1 Introduction

Operating rooms have been recognized to be the main income source for hospitals as it generates around 60% of hospital revenues (Macario *et. al.* 1995) but it counts for around 40% of hospital costs (Jackson 2002) throughout the use of facilities (operating rooms, etc.) and the personnel costs. This huge financial factor makes the operating rooms management a priority for hospital managers in order to achieve an efficient and effective use of the operating rooms. Exhaustive literature reviews on the Surgical Case Scheduling (SCS) problem are reported in (Cardoen *et. al.* 2010, Guerriero *et. al.* 2011).

In this paper, we study a real scheduling problem which consists in scheduling a set of elective surgical cases which require surgical instruments and tools to several operating rooms with the objective of minimizing the operating costs while taking into account the activities of the sterilizing unit. To the best of our knowledge there are very few literature on such particular problem. For instance (Beroule *et. al.* 2016) study an operating room scheduling problem including medical devices sterilization but with the objective of reducing the number of medical devices needed at the same time. And a branch-and-price technique is applied in (Cardoen *et. al.* 2009) to find the best order for surgeries in a day care center in order to optimize several objectives (peak use of recovery beds, occurrence of recovery overtime, ...) while satisfying the medical devices sterilizing constraints.

This research was performed in collaboration with the University Hospital of Angers in France (CHU Angers), which has also provided historical data for the experiments. We propose a mixed integer linear programming model for the problem which is solved in a lexicographic way. We show that our solutions provide competitive results in terms of number of operating rooms, and significantly improve those operationally implemented in terms of overtime and emergencies at the sterilizing unit.

2 Problem description

The CHU Angers includes several blocks and a sterilizing unit which centralizes all the sterilizing activities. In this study, we focus only on the activities of the Orthopedic Surgery Block (OSB) and the Sterilizing Unit (SU). There are around 2500 surgeries that are performed at the OSB per year in its 3 operating rooms. The opening hours for these 3 ORs are different as: room 1 and 2 are open 5 days a week from 8:15 to 17:00, and room 3 is open only 4 days a week from 8:15 to 14:30. Between 10 and 14 surgeons share these rooms according to a planning indicating the days when they operate, and the list of rooms that each surgeon can use each day.

Each surgeon has to perform a list of surgeries on an horizon of one month : some of them can be scheduled anytime during the opening hours of the rooms, whereas others (ambulatory surgeries) have to be completed before 15:00 to allow the patient to go home at the end of the day. Each of these surgeries is characterized by an estimated duration time and requires a list of surgical instruments which are organized in small boxes called kits. These kits are available in limited quantities. After each surgery, the used kits are kept into water for 30 minutes for pre-disinfection. Then they are collected at the predefined periods given in table 1 and sent to the SU for sterilization.

Table 1. SU's pickups and deliveries to the OSB.

Pick-up	07:00	11:30	13:00	14:30	16:00	17:30	18:30
Delivery	07:00	-	-	14:30	-	17:30	-

At the SU, the sterilization process is being performed in several steps : the instruments are first cleaned by automatic washers, then reassigned in their corresponding kit before being processed through sterilization machines. Finally, the kits are kept at the SU to cool off before being returned to the block. On average, when a kit arrives at the SU, the whole sterilization process takes around 4h30. From these delivery/collect hours in Table 1 and from the average kits processing time at the SU, we have the following situations:

1. A kit collected at 11:30, 13:00 or 14:30 on day (t) can be used in the morning on day $(t+1)$ if it is treated as a priority at the SU (**priority kit**, case 1). If it is not treated as a priority, it is considered that it cannot be used before 14:30 on day $(t+1)$.
2. A kit collected at 16:00 on day (t) can be used in the morning on day $(t+1)$ if it is treated urgently at the SU (**urgent kit**). If it is not treated urgently, it can be used from 14:30 on day $(t+1)$.
3. A kit collected at 17:30, 18:30 on day (t) or 7:00 on day $(t+1)$ can be used on day $(t+1)$ from 14:30 if it is treated as a priority at the SU (**priority kit**, case 2). If it is not treated as a priority, it will be available on day $(t+1)$ from 17:30.

In the current decision process at the CHU, it is only checked whether the surgeries scheduled each day are compatible with the number of kits owned by the block (the sterilizing courses for kits are neglected during the assignment of surgeries to the shifts). Consequently, the number of urgent and priority kits in SU to process remains substantial. The impact on the activity for the SU is immediate. In particular, each urgent kit implies to stop a machine in order to process it immediately (inducing a need to re-process removed kits afterward). On the other hand, priority kits are taken into account upon their arrival by moving them to the start of the queue to be treated first, against the first-in first out classical policy of SU. Ultimately, in rare cases, some kits may even be requested outside the delivery hours (violated kits). In that case, it is necessary to request a special shuttle.

The purpose of this work is to schedule all surgeries at the OSB while taking into account the sterilizing process in order to reduce the pressure on the SU staff.

In terms of objectives, according to the CHU, the first priority remains to schedule all surgeries in order to minimize the total overtime of the staff members of the OSB. The second priority consists in minimizing the number of used operating rooms. Finally, the third objective is to keep the number of urgent and priority kits as low as possible.

In the next section, we briefly sketch the basis of a mathematical model for solving this integrated OSB-SU problem.

3 Mathematical formulation

In order to model this problem, we propose a mathematical formulation based on the decomposition of the day in four periods : period 1 from 8:15 to 14:00, period 2 from 14:00 to 14:30, period 3 from 14:30 to 15:30 and period 4 from 15:30 to 17:00. The starting and ending hours of these periods are obtained by the combination of the critical pickup and delivery hours at the OSB, the opening and closing hours of the operating rooms, and the fact that surgeries must end 30 minutes before the collect of their medical devices.
We then introduce the following decision variables:

w_{itr}	binary variable equal to 1 if operation i is scheduled at day t in room r
x_{itr}^{bf}	binary variable equal to 1 if surgery i begins at period b and finishes at f, on day t, in room r
ε_{tr}	integer variable representing the total overtime in room r at day t
L_{tr}	binary variable equal to 1 if room r is used at day t
E_{tk}	integer variable representing the total urgent kits of type k at day t
Y_{tk}^1 (Y_{tk}^2)	integer variable representing the total priority kits of case 1 (resp. case 2) of type k at day t

In addition to the regular scheduling constraints (all surgeries have to be scheduled in the horizon, etc), our model includes other constraints such as:

- the expressions of urgent (1) and priority kits (both cases) (2)-(3):

$$\sum_{i=1}^{O}\sum_{r=1}^{R} q_{ik}.\left(\sum_{f=2}^{J}\sum_{b=1}^{f} x_{itr}^{bf} + \sum_{b=1}^{2}\sum_{f=b}^{J} x_{i(t+1)r}^{bf}\right) - Q_k \leq E_{tk} \qquad \forall t \in \{1,..,T\}, \forall k \in \{1,..,K\} \quad (1)$$

$$\sum_{i=1}^{O}\sum_{r=1}^{R} q_{ik}.\left(\sum_{f=1}^{J}\sum_{b=1}^{f} x_{itr}^{bf} + \sum_{b=1}^{2}\sum_{f=b}^{J} x_{i(t+1)r}^{bf}\right) - Q_k - E_{tk} \leq Y_{tk}^1 \qquad \forall t \in \{1,\ldots,T\}, \forall k \in \{1,\ldots,K\} \quad (2)$$

$$\sum_{i=1}^{O}\sum_{r=1}^{R} q_{ik}.\left(\sum_{f=2}^{J}\sum_{b=1}^{f} x_{itr}^{bf} + \sum_{b=1}^{J}\sum_{f=b}^{J} x_{i(t+1)r}^{bf}\right) - Q_k - E_{tk} \leq Y_{tk}^2 \qquad \forall t \in \{1,..,T\}, \forall k \in \{1,..,K\} \quad (3)$$

  where R is the total number of ORs, q_{ik} is the total number of kits of type $k \in \{1..K\}$ that surgery $i \in \{1..O\}$ requires, J represents the total number of periods (4), Q_k is the total available quantity of kit k and T is the total number of days in the horizon.

- the control of the workload of surgeries for each interval of the day and each room (4):

$$\sum_{i=1}^{O}\sum_{b=\beta}^{\gamma}\sum_{f=b}^{\gamma} p_i \cdot x_{itr}^{bf} + \sum_{i=1}^{O}\sum_{b=1}^{\beta-1}\sum_{f=\gamma+1}^{J} d_{rt}^{\beta\gamma} \cdot x_{itr}^{bf} \leq d_{rt}^{\beta\gamma} \cdot L_{tr} + u_\gamma \cdot \varepsilon_{tr} \qquad (4)$$
$$\forall \beta \in \{1,\ldots,J\}, \forall \gamma \in \{\beta,\ldots,J\}, \forall t \in \{1,\ldots,T\}, \forall r \in \{1,\ldots,R\}$$

  where p_i is the duration of surgery i, $d_{rt}^{\beta\gamma}$ is the duration from period β to γ in room r on day t, u_γ is a binary parameter equal to 1 if $\gamma = J$.

The multiple objective functions are taken into account by using a lexicographic method (5) and after each objective is solved, its value is added as an upper bound to the model. First, f_1 minimizes the total over time and then f_2 minimizes the number of used rooms and finally f_3 minimizes the total penalty cost of urgent (cu) and priority (cp) kits.

$$\text{Minimize } Lex\left(f_1 : \sum_{t=1}^{T}\sum_{r=1}^{R} \varepsilon_{tr}\,;\; f_2 : \sum_{t=1}^{T}\sum_{r=1}^{R} L_{tr}\,;\; f_3 : \sum_{t=1}^{T}\sum_{k=1}^{K}\left[cu \cdot E_{tk} + cp \cdot (Y_{tk}^1 + Y_{tk}^2)\right]\right) \qquad (5)$$

4 Experimental results

To test and validate our model, we used a 10 instances benchmark provided by the CHU. Each instance corresponds to the activity of the OSB during one month. The number of surgeries in these instances varies from 164 to 220. We used CPlex 12.6.1 to solve the model and a time limit of 3600 seconds was set for each of the three objective functions.

As shown in table 2, which compares the schedule of the CHU with the schedule from the MILP, our model managed to eliminate the ambulatory surgeries that finish after 15:00

Table 2. Results comparison

Instance	CHU schedule						MILP schedule					
	#late ambs	#violated kits	over time	#rooms	#urgent	#priorities	#late ambs	#violated kits	over time	#rooms	#urgent	#priorities
1	12	0	1043	59	9	40	0	0	0	58	0	2
2	10	5	1336	59	4	44	0	0	440	55	0	3
3	7	10	766	48	3	56	0	0	131	46	0	12
4	10	7	1204	48	15	66	0	0	211	48	0	8
5	15	5	1165	59	14	58	0	0	142	58	0	9
6	4	3	1138	52	0	65	0	0	55	52	0	9
7	5	26	1474	59	1	95	0	0	461	59	0	11
8	9	0	1011	49	2	51	0	0	234	48	0	0
9	6	12	580	46	8	45	0	0	86	43	0	0
10	6	3	714	60	5	31	0	0	133	53	0	0
Average	8.4	7.1	1043.1	53.9	6.1	55.1	0	0	189.3	52	0	5.4

and the need of kits that cannot be delivered in the normal hours (violated kits). In addition, our model decreased the overtime by around 81.85% (from 14h14 to 3h09) per month and it closed around 2 rooms in average each month. Finally, our model was able to decrease the number of urgent and priority kits by around 91.17% (from 61.2 to 5.4) per month.

5 Conclusions and perspectives

This work focuses on a real surgical case scheduling including sterilization activity constraints, and three objective functions. We propose an MILP formulation which is solved in a lexicographic fashion. Our solutions provide competitive results in terms of used rooms, and significantly improve those operationally implemented in terms of overtime and urgent and priority kits at the SU. Still in line with the needs of the CHU Angers, the next step is to address the online version of the problem.

Acknowledgements

The authors would like to thank L. Hubert, A.V. Lebelle and A. Robelet from the CHU for submitting us this challenging problem and kindly providing us with real instances. This research is partially founded by Angers Loire Metropole (ALM) and IMT Atlantique.

References

Beroule, B., Grunder, O., Barakat, O., Aujoulat, O., Lustig, H., 2016, Operating room scheduling including medical devices sterilization: towards a transverse logistic. *IFAC-PapersOnLine*, 49(12), pp.1146-1151.

Cardoen, B., Demeulemeester, E., Beliën, J., 2009, Sequencing surgical cases in a day-care environment: an exact branch-and-price approach. *Computers & Operations Research*, 36(9), pp.2660-2669.

Cardoen, B., Demeulemeester, E., Beliën, J., 2010, Operating room planning and scheduling: A literature review. *European journal of operational research*, 201(3), pp.921-932.

Guerriero, F., Guido, R., 2011, Operational research in the management of the operating theatre: a survey. *Health care management science*, 14(1), pp.89-114.

Jackson, R. L., 2002, "The business of surgery. Managing the OR as a profit center requires more than just IT. It requires a profit-making mindset, too", *Health management technology*, 23(7), 20.

Macario, A., Vitez, T., Dunn, B., McDonald, T., 1995, "Where are the costs in perioperative care?: Analysis of hospital costs and charges for inpatient surgical care", *Anesthesiology: The Journal of the American Society of Anesthesiologists*, 83(6), pp.1138-1144.

Modeling and solving a two-stage assembly flowshop scheduling problem with buffers

Andrés C. and Maheut J.

Universitat Politècnica de València, Spain
candres@omp.upv.es, juma2@upv.es

Keywords: Assembly flowshop, limited buffer, scheduling.

1 Abstract

The two-stage assembly scheduling problem is a well-known problem from the literature with a lot of practical applications. It consists of a system with two stages where a set of n jobs must be processed in a given sequence. First stage is composed of m machines, each one produces a component to be assembled in one single machine at the second stage. One typical assumption in all the previous literature is the absence of buffer limitations between both stages but this is too unrealistic from a practical point of view. Under Lean Manufacturing paradigm, it is interesting to minimize the time that all jobs wait in the buffers and total machine blocking time. This study presents a mathematical model to two stage assembly flowshop scheduling problem with finite storage conditions together with a complete enumeration study for small instances. The aim of the research is to show the effect of buffer size over the total blocking time plus inventory time.

2 Introduction

Since the seminal paper of Johnson (1954) an extensive amount of papers has been published related with scheduling problem. Most of them with the unrealistic assumption of no buffer limitations between machines This is too unrealistic under Lean Manufacturing paradigm and it has been less studied in the literature. The first paper about limited buffer scheduling was (Dutta and Cunningham 1975) who studied a flowshop problem with capacitated buffers using dynamic programming. Later, (Papadimitriou and Kanellakis 1980) probed this problem is NP-hard in the strong sense and developed a relation between a heuristic developed for the problem and buffer size. Due to the complexity of the problem, several authors developed heuristic approaches. First one was the paper of Leinsten (1990) what showed a general framework for scheduling problem with capacitated buffer (limited buffer size, blocking and no wait problems) and studied several heuristic rules, concluding the high performance of NEH rule of Nawaz *et al.* (1983). Later Nowicki (1999) developed a Tabu Search approach using some job properties from graph representation to accelerate the local search by eliminating sets of solutions that do not improve the current solution. Other approaches were Tabu Search of Brucker *et al.* (2003), Genetic Algorithm of Wang *et al.* (2006), Particle Swarm Algorithm of Liu *et al.* (2008), immune system algorithm of Hsieh *et al.* (2009) or the Ant Colony Algorithm of Rossi and Lanzetta (2013).

Simultaneously, other kind of scheduling problems called assembly flowshop has attracted the interest of the researchers. Regarding makespan minimization, Lee *et al.* (1993) and Potts *et al.* (1995) probed this problem is NP-hard even for two and M machines at the first stage. The best approach up to now to solve the problem with makespan was proposed by Hariri and Potts, (1997) using Branch and Bound techniques. Regarding total completion time minimization in assembly flow shops, Framinan and Perez-Gonzalez

(2017) proposed a constructive heuristic and a metaheuristic that outperform all the previous heuristics.

However, there are no previous research about limited storage two stage assembly flowshop with buffering and blocking time as objective function. So our aim is to study the effect of buffer size between both stages. First, a complete enumeration study will be presented and used to investigate the effect of buffer size changes over small size problems based on Taillard (1993) instances. Later, a mathematical model will be presented and used to optimize the sequence for medium instances. The results let us confirm the interest of this kind of problem and the necessity to develop procedures for study realistic instances.

3 Complete enumeration study

In order to show the effect of buffer size over makespan for small instances (up to nine jobs), a test based in complete enumeration has been carried out considering that there is an identical buffer of size b between each component machine and the assembly one. We used some instances from Taillard's set where first machine was used to represent processing time in assembly machine and the rest represents processing time in component manufacturing machines. All the sequences for the same instance have been computed for buffer size between 0 and 4.

The following figures represent the results for all the 9! sequences from Ta004 instance (in our case a shop with one assembly machine plus four component machines) and the empirical distribution of blocking plus buffering times depending of buffer size. It can be seen from both figures that there are difference between each solution space depending on buffer size.

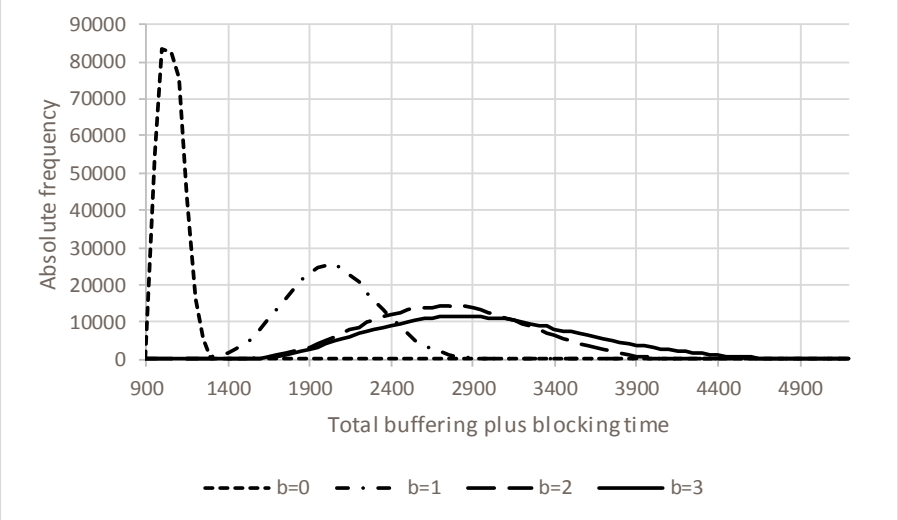


Fig. 1. Distribution for $N = 9$ jobs and different buffer size.

It can be seen that maximum number of solutions decreases for each objective function value when buffer size increases. On the other hand, when buffer size increases, distribution width increases too.

Figure 2, shows that when buffer size increases is more difficult to find a good solution randomly. For example, for zero buffer size all solutions are under 50% from optimal value

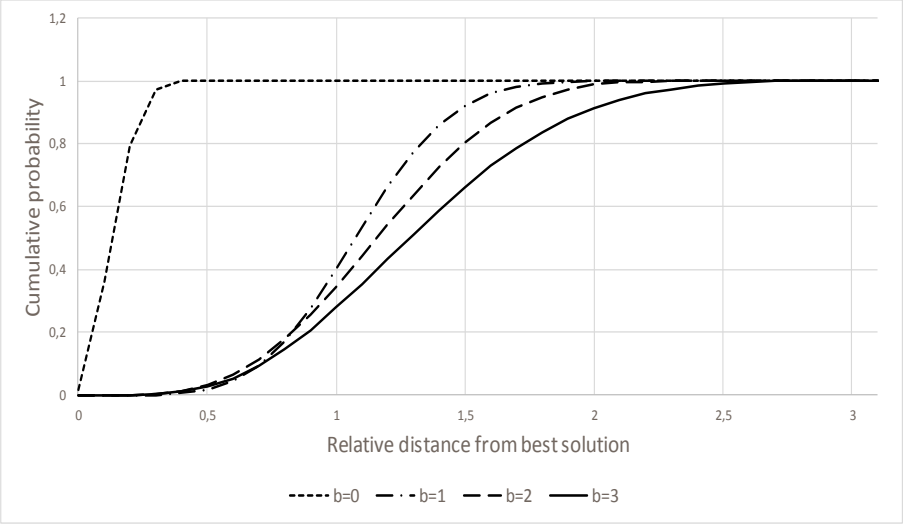


Fig. 2. Empirical cumulative distribution for $N = 8$ jobs.

but for buffer size equal to 3, there are only 2.3% of solutions. That support the idea that it is "easier" to find a good solution randomly in blocking assembly shops for these objective function.

4 Mathematical model

A second step to study the problem for medium size instances is to develop a mathematical model. It is necessary the following notation to formalize the model.

Let:

i, index for jobs or components $i = 1, \ldots, n$
j, index for sequence positions $j = 1, \ldots, n$
k, index for machines and buffers $k = 1, \ldots, m$
MC_k, component manufacturing machines $k = 1, \ldots, m$. Each machine has a capacitated buffer BC_k, $k = 1, \ldots, m$.
$pc_{j,k}$ is the processing time of job in position j in machine k
p_j is the processing time of job in position j at assembly machine
b_k is the capacity of buffer k
SC_{jk} is the starting time of component to be assembled at position j in machine k
S_j is the starting time of job to be assembled at position j in the assembly machine
$x_{ij} = \begin{cases} 1, \text{ if component/job } j \text{ is sequenced on position } j \\ 0, \text{ otherwise.} \end{cases}$

Thus, mathematical model can be stated as follows:

$$\min z = \sum_{j=1}^{n}\sum_{k=1}^{m}(S_{j-b} - SC_{jk} - pc_{jk}) + \sum_{j=1}^{n}\sum_{k=2}^{m}(S_j - S_{j-b})$$

s.t.

$$
\begin{aligned}
& SC_{1k} = 0, && k = 1, \ldots, m \\
& S_1 \geq SC_{1k} + \sum_{i=1}^{n} x_{i1}\, pc_{ik}, && k = 1, \ldots, m \\
& SC_{jk} \geq SC_{j-1,k} + \sum_{i=1}^{n} x_{i,j-1}\, pc_{ik}, && j = 2, \ldots, n, \;\; k = 1, \ldots, m \\
& SC_{jk} \geq S_{j-bc_k-1,k}, && j > 2, \ldots, bc_k + 1, \;\; k = 1, \ldots, m \\
& S_j \geq S_{j-1} + \sum_{i=1}^{n} x_{i,j-1}\, p_i, && j = 2, \ldots, n \\
& S_j \geq SC_{jk} + \sum_{i=1}^{n} x_{ij}\, pc_{ik}, && j = 2, \ldots, n, \;\; k = 1, \ldots, m \\
& \sum_{i=1}^{n} x_{ij} = 1, && j = 1, \ldots, n \\
& \sum_{j=1}^{n} x_{ij} = 1, && i = 1, \ldots, n \\
& x_{ij} \in \{0, 1\}, && i, j = 1, \ldots, n \\
& S_j, SC_{jk} \geq 0, && j = 1, \ldots, n, \;\; k = 1, \ldots, m.
\end{aligned}
$$

First term on the objective function represents total blocking time while the second one computes total buffering time for a given sequence. Set of constraints represents the relations between starting time of every operations under finite storage assumption.

Mathematical model was tested with some Taillard's instances (Ta001 to Ta020 and Ta031 to Ta050) adapting them to the assembly flowshop problem. The results show that it is possible to solve optimally instances until 20 jobs and 4 component machines. More results about this study will be presented at the conference.

5 Conclusions and future work

In this paper we presented a study about two stage assembly flowshop scheduling problem with limited buffers of size b. Instead of classical objective functions like makespan or total flowtime, we study a composed function of total buffering time plus total blocking time. A complete enumeration study shows that solution space shape changes with the size of the buffers and it seems a promising field for researchers due to the relation of objective function with the improvement in production systems under Lean Manufacturing paradigm.

Moreover, a new mathematical model is described and some results are presented. Our aim is to develop competitive heuristic procedures to solve realistic instances and get more insights about the relationship between buffer size and assembly flowshop performance under finite storage conditions.

References

Brucker, P., S. Heitmann and J. Hurink, 2003, "Flow-shop problems with intermediate buffers", *OR Spectrum*, Vol.25, pp. 549–574.

Dutta, S., A. Cunningham, 1975, "Sequencing Two-Machine Flow-Shops with Finite Intermediate Storage", *Management Science*, Vol. 21, pp. 989–996.

Framinan, J.M., P. Peez-Gonzalez, 2017, "The 2 atage assembly flowshop scheduling problem with total completion time: efficient constructive heuristic and metaheuristic", *Computers & Operations Research*, Vol. 88, pp. 237–246.

Hsieh, Y., P. You and C. Liou, 2009, "A note of using effective immune based approach for the flow shop scheduling with buffers", *Applied Mathematics and Computation*, Vol. 215, pp. 1984–1989.

Johnson, S. M., 1954, "Optimal two-and three-stage production schedules with setup times included", *Naval Research Logistics Quarterly*, Vol. 1, pp. 61–68.

Lee, C.-Y., T.C.E. Cheng, and B.M.T. Lin, 1993, "Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem", *Management Science*, Vol. 39, pp. 616–625.

Leisten R., 1990, "Flowshop sequencing problems with limited buffer storage", *International Journal of Production Research*, Vol. 28, pp. 2085–2100.

Liu, B., L. Wang and Y. Jin, 2008. "An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers", *Computers & Operations Research*, Vol. 35, pp. 2791–2806.

Nawaz, M., E. E. Enscore and I. Ham, 1983, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem". *Omega*, Vol. 11, pp. 91–95.

Papadimitriou, C.H., P.C. Kanellakis, 1978, "Flow shop scheduling with limited temporary storage", *Proceedings of Annual Allerton Conference on Communication, Control, and Computing*, pp. 214–223.

Potts, C. N., S. V. Sevast'janov, V. A. Strusevich, L. N. Van Wassenhove, and C.M. Zwaneveld, 1995, "The Two-Stage Assembly Scheduling Problem: Complexity and Approximation", *Operations Research*, Vol. 43, pp. 346–355.

Rossi, A. and M. Lanzetta, 2013, "Scheduling flow lines with buffers by ant colony digraph", *Expert Systems with Applications*, Vol. 40, pp. 3328–3340.

Taillard, E. 1993, "Benchmarks for basic scheduling problems", European Journal of Operational Research , Vol. 64(2), pp. 278–285.

Wang, L., L. Zhang and D. Zheng, 2006, "An effective hybrid genetic algorithm for flow shop scheduling with limited buffers", *Computers & Operations Research*, Vol. 33, pp. 2960–2971.

A new polynomial-time algorithm for calculating upper bounds on resource usage for RCPSP problem

Dmitry Arkhipov[1,2], Olga Battaïa[1] and Alexander Lazarev[2,3,4,5]

[1] Department of Complex Systems Engineering, ISAE-SUPAERO, Université de Toulouse, Toulouse, France;
[2] V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russian Federation;
[3] Lomonosov Moscow State University, Moscow, Russian Federation;
[4] Moscow Institute of Physics and Technology, Dolgoprudny, Russian Federation;
[5] International Laboratory of Decision Choice and Analysis, National Research University Higher School of Economics, Moscow, Russian Federation.

Keywords: project planning, polynomial algorithms, scheduling, constraint programming, propagator.

1 Introduction

The Resource-Constrained Project Scheduling Problem (RCPSP) is considered. This problem is NP-hard in strong sense (Garey and Johnson 1975). In this paper, a new polynomial-time approach is developed to find an upper bound on resource consumption. This bound can be also used to calculate a lower bound for makespan. Furthermore, the procedure helps to increase the efficiency of existing propagators and to improve constraint programming model performances by tightening decision variables domains.

We consider the following formulation of RCPSP with continuous time. There is a set of tasks N and a set of renewable resources R. The capacity of resource $X \in R$ is defined by non-negative piecewise constant function $c_X(t)$ which consists of constant functions $c_X^1(t) = c^1, \forall t \in [0, t_1); c_X^2(t) = c^2, \forall t \in [t_1, t_2); \ldots, c_X^m(t) = c^m, \forall t \in [t_{m-1}, T]$. For any task $j \in N$, the following parameters are given: p_j – processing time and a_{jX} – required amount of resource $X \in R$ for task j.

Precedence relations between tasks are given by a directed acyclic graph $G = (N, E)$. If an edge $e_{ji} \in E$ exists, it means that task j must be finished before the starting time of task i ($j \to i$).

Time horizon T is defined and for each $j \in N$ the parameters of task processing domain are given: r_j – release time, the earliest time from which task j can be started and D_j – deadline, the latest time for finishing task j. In case if this parameters are not given we can set $r_j = 0$, $D_j = T$ for each $j \in N$, and then use some polynomial-time propagators to tighten domains $[r_j, D_j]$ of task processing.

We consider the decision version of RCPSP without objective function, but we have to find a schedule which satisfies precedence relations and resource constraints with makespan value lower than T.

The paper is organised as follows. In section 2 the overview of resourced-based propagators is presented. In the section 3 we give the main idea of our approach and theorems on which it is based on. Then, we discuss some generalizations of RCPSP problem for which presented approach is applicable and make some conclusion remarks in section 4.

2 State of the art

Our research is focused on improving constraint programming model and propagators, which use resource constraints to make decision variable intervals tighter. There are a lot of

propagators, based on resource consumption constraints. (Lahrichi 1982) firstly calculated «resource compulsory part», (Le Pape 1988) created «time tables». Then (Fox 1990) introduced the term «resource profile» and (Caseau and Laburthe 1996) presented «resource histogram». Sweep algorithm to calculate resource profi le was presented by (Beldiceanu and Carlsson 2001). Several efficient propagators based on time-tabling algorithms were developed in literature (Schutt *et. al.* 2011), (Ouellet and Quimper 2013). Other propagators were discussed in (Baptiste *et. al.* 2001), (Vilim 2007) and in makespan lower bound surveys (Neron *et. al.* 2006) and (Knust 2015).

3 Calculating an upper bound on highest possible resource consumption

For each resource $X \in R$ and any time $t \in [0, T]$ we define an upper bound on highest possible resource consumption in time interval $[0, t)$ by $U_X(t)$. In (Arkhipov *et. al.* 2017) we presented an algorithm for a discrete version of RCPSP to estimate $U_X(t)$ in $O(n^2 r(n + m + r)T \log T)$ operations, where n – number of tasks, r – number of resources, m – the highest number of breakpoints in resource capacity function, T – time horizon. The main idea of this algorithm was as follows. First of all, the algorithm uses some polynomial-time propagators to tighten processing intervals $[r_j, D_j]$ for each $j \in N$. Then, for each task $j \in N$, resource $X \in R$ and timeslot $t \in 1, \ldots, T$ the highest possible consumption of X by j in interval $[0, t]$ is calculated and defined by $A_{jX}(t)$.

Then the RCPSP problem is considered for each pair of resources $X, Y \in R$ and the following task processing constraints:

- preemptions of task processing are allowed;
- tasks are able to continue processing after deadline;
- amount of resources X and Y consumed by task j in timeslot $[t, t + 1]$ can be not equal to a_{jX} and a_{jY} respectively, but functions $u_{jX}(t)$ and $u_{jY}(t)$ – total amounts of resources X and Y in time interval $[0, t]$ should satisfy the following constraints:

$$u_{jX}(t) \leq A_{jX}(t),$$

$$u_{jY}(t) \leq A_{jY}(t),$$

$$\frac{u_{jX}(t)}{u_{jY}(t)} = \frac{a_{jX}}{a_{jY}}.$$

The objective is to find the functions $u_{jX}(t)$ and $u_{jY}(t)$ for any $t \in [0, T]$ subject to maximizing the objective functions

$$U_{X|Y}(t) = \sum_{j \in N} u_{jX}(t),$$

$$U_{Y|X}(t) = \sum_{j \in N} u_{jY}(t)$$

for any $t \in [0, T]$.

The Master Algorithm which finds an optimal solution for this problem iterates on timeslots $t = 1, \ldots, T$, solving the following optimization problem for each pair of resources X, Y.

Timeslot problem. *For each $j \in N$ values $u_{jX}(t-1)$ and $u_{jY}(t-1)$ are given and functions $A_{jX}(t)$, $A_{jX}(t)$ are defined. Determine $u_{jX}(t) \geq u_{jX}(t-1)$ and $u_{jY}(t) \geq u_{jY}(t-1)$ for all tasks $j \in N$ such that*

$$\max U_X(t), U_Y(t)$$

subject to resource capacities

$$\sum_{j\in N}(u_{jX}(t)-u_{jX}(t-1))\le c_X(t),$$

$$\sum_{j\in N}(u_{jY}(t)-u_{jY}(t-1))\le c_Y(t)$$

and constraints

$$\frac{u_{jX}(t)-u_{jX}(t-1)}{u_{jY}(t)-u_{jY}(t-1)}=\frac{a_{jX}}{a_{jY}},$$

$$u_{jX}(t)\le A_{jX}(t),\ u_{jY}(t)\le A_{jY}(t).$$

If for any time slot there is more than one solution satisfying these conditions, choose the one using the following criterion:

$$\min\sum_{j\in N}\sqrt{(u_{jX}(t)-u_{jX}(t-1))^2+(u_{jY}(t)-u_{jY}(t-1))^2}.$$

The developed geometric algorithm solves this optimization problem in $O(n^2)$ operations.

The objective of this paper is to show that this algorithm can be used for RCPSP formulation with continuous time. We will call a time point $T_i\in[0,T]$ a *breakpoint* if T_i is a release time of any task $j\in N$, i.e. $T^i=r_j$ or T_i is a $T^i=t_k$ – start or end of any segment of function $c_X(t)$. Total number of breakpoints is $b\le n+m$. We assume that the breakpoints are ordered in ascending order: $0=T_1<T_2<\ldots<T_b=T$. Note that the size of a timeslot does not matter for the Master Algorithm. Therefore, we consider intervals $[T_1,T_2),\ldots,[T_{b-1},T_b)$ like timeslots and use Master Algorithm to find functions $u_{jX}(t),u_{jY}(t)$ for each $j\in N$ and each timeslot. According to the theorem proved in (Arkhipov *et. al.* 2017) obtained functions $U_{X|Y}(t)$ and $U_{Y|X}(t)$ would be an upper bound on resource X and Y consumption respectively for any $t=T_1,\ldots,T_b$ if for any timeslot and any $t\in[T_k,T_{k+1}]$ $u'_{jX}(t)=const$. The following lemma proves that these conditions can be simplified.

Lemma 1.
Suppose there is a set of functions $u_{1X}(t),\ldots,u_{nX}(t)$ defined on timeslot $[T_k,T_{k+1}]$ which satisfy the constraints of the Timeslot problem for any $t\in[0,T]$. Then function

$$uu_{jX}(t)=\frac{1}{T_{k+1}-T_k}\int_{T_k}^{T_{k+1}}u_{jX}(t)dt$$

defined for all $j\in N$ satisfies the constraints of the Timeslot problem and $uu'_{jX}(t)=const$.

Each feasible schedule defines a set of integrable functions $u_{1X}(t),\ldots,u_{nX}(t)$. Since $uu_{jX}(T_k)\le u_{jX}(T_k)$, Lemma 1 implies that functions $U_{X|Y}(t)$ and $U_{Y|X}(t)$ provided by the Master Algorithm give an upper bound on amount of resources consumed by the tasks belonging to set N in time interval $[0,T_{k+1})$. The complexity of the Geometric Algorithm to solve Timeslot problem is $O(n^2)$. Hence the Master Algorithm complexity equals to $O(n^2(n+m))$, i.e. total number of cycled timeslots is not more than $n+m$. Therefore functions $U_{X|Y}(t)$ and $U_{Y|X}(t)$ could be found for all pairs of resources $(X,Y)\in R^2$ in $O(r^2n^2(n+m))$ operations, where r – number of resources. An upper bound on resource consumption in interval $[0,t)$ can be defined correctly for any $t=T_1,\ldots,T_n$:

$$U_X(t)=\min_{Y\in R}U_{X|Y}(t).$$

4 Applications & Generalizations

In this paper, a polynomial algorithm to estimate an upper bound on resource amount used in time interval $[0, t]$ is presented. This approach can be applied not only to the classical RCPSP formulation but for other RCPSP statements with segment-constant resource capacity functions, i.e. RCPSP/max.

Obtained functions $U_X(t)$ can be used in resource-based propagators to evaluate resource-using profiles. Our future research will be focused on development of direct methods to improve existing resource-based propagators and to create new techniques of bounding resource usage function.

Acknowledgements

This research was supported by ISAE-SUPAERO Foundation and the Russian Science Foundation (grant 17-19-01665).

The authors are grateful to Emmanuel Hebrard and Pierre Flener for useful advises on resource-based propagators.

References

Arkhipov D., O. Battaia and A. Lazarev, 2017, "Long-term production planning problem: scheduling, makespan estimation and bottleneck analysis", *IFAC-PapersOnLine*, Vol. 50, I. 1, pp. 7970–7974.

Baptiste P., C. Le Pape and W. Nuijten, 2001, " Constraint-Based Scheduling", *Kluwer Academic Publisher.*

Beldiceanu N., M. Carlsson, 2001, "Sweep as a Generic Pruning Technique Applied to the Non-overlapping Rectangles Constraint", *Proceedings of the seventh International Conference on Principles and Practice of Constraint Programming*, pp. 377-391.

Caseau Y., F. Laburthe, 1996, "Cumulative Scheduling with Task Intervals", *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pp. 363-377.

Fox, B. R., 1991, "Non-chronological scheduling", *In: Proceedings of AI, Simulation and Planning in High Autonomy Systems*, pp. 72-77.

Garey M.R., D.S. Johnson, 1975, "Complexity results for multiprocessor scheduling under resource constraints", *SIAM Journal on Computing*, Vol. 4, pp. 397–411.

Knust S., 2015, "Lower Bounds on the Minimum Project Duration", In:*Schwindt C., Zimmermann J.: Handbook on Project Management and Scheduling*, Vol. 01, pp. 3–56.

Kolisch R., A. Sprecher, 1997, "PSPLIB – a project scheduling problem library", *Eur J Oper Res*, Vol. 96(1), pp. 205-216, http://www.om-db.wi.tum.de/psplib/.

Lahrichi A., 1982, "Scheduling: the Notions of Hump, Compulsory Parts and their Use in Cumulative Problems", *In: C.R. Acad. Sc. Pairs*, pp. 209-211.

Le Pape C., 1988, "Des systemes d'ordonnancement exibles et opportunistes", *PhD thesis. Universite Paris XI.*

Neron E., C. Artigues, P. Baptiste, J. Carlier, J.Damay, S.Demassey and P. Laborie 2006, "Lower bounds for Resource Constrained Project Scheduling Problem", In:*Jozefowska J., Weglarz J.: Perspectives in Modern Project Scheduling*, pp. 167–204.

Ouellet P., C.G. Quimper, 2013, 'Time-Table Extended-Edge-Finding for the Cumulative Constraint", *Proceedings of the Nineteenth International Conference on Principles and Practice of Constraint Programming*, pp. 562-577.

Schutt A., Feydy T., Stuckey P., Wallace M. 2011, "Explaining the cumulative propagator", *Constraint*, Vol. 16(3) pp. 250–282.

Vilim P., 2007, "Global Constraints in Scheduling", *PhD thesis. Charles University in Prague, Faculty of Mathematics, Physics, Department of Theoretical Computer Science, and Mathematical Logic.*

Assembly Flowshops Scheduling Problem to Minimize Maximum Tardiness with Setup Times

Asiye Aydilek[1], Harun Aydilek[1], and Ali Allahverdi[2]

[1] Gulf University for Science and Technology, Kuwait
{aydilek.a,aydilek.h}@gust.edu.kw
[2] Kuwait University, Kuwait
ali.allahverdi@ku.edu.kw

Keywords: Assembly flowshop, simulated annealing, maximum tardiness, setup time.

1 Introduction

Manufacturing of a product consisting of different parts, where parts are processed by different machines in parallel at the first stage, and all the processed parts of the product are assembled at the second stage can be considered as a two-stage assembly flowshop scheduling problem.

The problem was addressed with respect to different performance measures, e.g., makespan, total completion time, and maximum lateness. Moreover, the problem was addressed with zero and non-zero setup times. There are some applications where assumption of zero setup times is valid. However, the assumption is not valid for some other applications, e.g., Allahverdi (2015).

The problem with total tardiness performance measure was addressed by Allahverdi and Aydilek (2015) assuming zero setup times. They proposed several heuristics including a genetic algorithm.

Allahverdi and Al-Anzi (2006) provided a dominance relation and presented a few heuristics for the problem with respect to maximum lateness performance measure for zero setup times while Al-Anzi and Allahverdi (2007) presented several heuristics, including Particle Swarm Optimization (*PSO*), Tabu search, and Self-Adaptive Deferential Evolution (*SDE*), for the problem with setup times. Al-Anzi and Allahverdi (2007) showed that the *SDE* heuristic outperforms the others not only in terms of the error but also in terms of the CPU time.

In this paper, we address the problem with maximum tardiness performance measure with non-zero setup times. The maximum tardiness performance measure is used in the scheduling literature. This is because for some applications completing a job after its due date results in a penalty which increases as the gap between the due date and completion time widens.

It should be noted that a sequence that minimizes maximum lateness also minimizes maximum tardiness. As a result, the *SDE* heuristic of Al-Anzi and Allahverdi (2007) is not only the best existing heuristic for the problem with maximum lateness performance measure but also the best existing heuristic with respect to the maximum tardiness performance measure. Therefore, we compare the performance of our newly proposed algorithm, to be developed in this paper, with the *SDE* algorithm of Al-Anzi and Allahverdi (2007), which is known to be the best existing algorithm for the problem. We show that performance of the newly developed algorithm in this paper significantly outperforms that of the *SDE* algorithm of Al-Anzi and Allahverdi (2007).

2 The proposed Simulated Maximum Insertion (*SMI*) algorithm

Al-Anzi and Allahverdi (2007) presented several algorithms and showed that the algorithm of self-adaptive differential evolution (*SDE*) outperforms the others for the two-stage assembly flowshop scheduling problem with non-zero setup times to minimize maximum lateness. The algorithm *SDE* of Al-Anzi and Allahverdi (2007) is the only benchmark existing algorithm, in the literature, to compare our algorithm with. The parameter values of the algorithm *SDE* are taken as the ones given by Al-Anzi and Allahverdi (2007). However, for a fair comparison, the number of generations in the *SDE* is selected so that the algorithm proposed in this paper and the *SDE* algorithm have the same computational time. Our algorithm, called *SMI*, is explained next.

Simulated Maximum Insertion (*SMI*) algorithm is a hybrid of simulated annealing algorithm and maximum insertion algorithm. In the maximum insertion algorithm, for a given sequence, the job with the maximum tardiness is inserted to certain positions in the sequence and the sequence is updated if the insertion decreases the maximum tardiness. It is observed that repeating this procedure decreases the maximum tardiness of a given sequence significantly. Therefore, we combined the maximum insertion algorithm with the simulated annealing algorithm in order to obtain the hybrid algorithm. In the hybrid algorithm, given a sequence and given initial parameters, we apply the swap and insertion operators to obtain two new sequences and the better one among the two is selected. The current sequence is updated whenever one of these new sequences, the better one, improves the objective function. If the objective function is not improved, then the current sequence is updated with the better one with certain probability. When the temperature is high, this probability is large and as the temperature decreases the probability of choosing an inferior sequence decreases. In order not to trap to a local solution, the solution space is explored for high temperatures and exploited for low temperatures. Then, the maximum insertion algorithm is applied. At high temperatures, the job with the maximum tardiness is inserted to every z-th position rather than every position in the sequence and this is repeated z times. Thus, this gives more chances to explore alternative solutions. As the temperature decreases, the value of z decreases which helps to exploit the sequence. Once the temperature drops below the final temperature, the maximum insertion algorithm is applied such that the job with the maximum tardiness is inserted to every position in the sequence and the procedure is repeated certain times in order to improve the solution further. In short, inserting the job with maximum tardiness strengthens the exploration step of the simulated annealing algorithm at the beginning when the temperature is high and reinforces the exploitation step of the simulated annealing algorithm towards the end when the temperature is low. The hybrid algorithm requires an initial sequence, which affects the performance of the algorithm. We construct some initial sequences as follows. We first convert the problem to a single machine scheduling problem by aggregating the processing times and setup times at both stages. The aggregation can be performed in several ways. Four of the aggregated processing times are

$$
\begin{aligned}
AP_0(i) &= \max\{\max_{j=1,\ldots,m}(t_{ij}+s_{ij}),(s_i+p_i)\},\\
AP_1(i) &= \max_{j=1,\ldots,m}(t_{ij}+s_{ij})+(s_i+p_i),\\
AP_2(i) &= \max_{j=1,\ldots,m}(t_{ij}+s_{ij}),\\
AP_3(i) &= \max\{(\max_{j=1,\ldots,m}(t_{ij}+s_{ij})+\min_{j=1,\ldots,m}(t_{ij}+s_{ij}))/2,(s_i+p_i)\}.
\end{aligned}
$$

Then, by applying the shortest processing time (SPT) rule to the aggregated processing times, we obtain a sequence for each one. In addition to these sequences, we also considered the sequence obtained from earliest due date (EDD) rule and the best performing sequence among the five sequences is taken as the initial sequence and denoted as seq_b.

Simulated annealing algorithm has parameters, which need to be calibrated for the problem which are initial temperature, TP_i, final temperature, TP_f, cooling factor, cf, and number of repetitions, N_r. The following table presents the values considered for the calibration and the selected values for the parameters followed by the steps of the SMI algorithm.

Table 1. Considered and selected values for the parameters of *SMI*

Parameters	Tested values	Selected values
Initial temperature (TP_i)	0.10, 0.11, 0.12, 0.13, 0.14, 0.15	0.12
Final temperature (TP_f)	0.0001, 0.0005, 0.0010, 0.0020	0.0010
Cooling factor (cf)	0.970, 0.975, 0.980, 0.985, 0.990	0.975
Number of repetitions (N_r)	20, 30, 40	30

3 Algorithm Evaluation

The performances of the existing algorithm SDE and the proposed algorithm SMI are compared in this section. Computations were executed on a PC with Intel Core i7-3520M CPU processor of 2.9 GHz with 8 GB RAM.

A uniform distribution $U(1, 100)$ is used to generate processing times on all the machines including the assembly machine. Similarly, setup times at both stages are generated from a uniform distribution $U(0, k \cdot 100)$ where the parameter k denotes the expected ratio of setup times to processing times. Job due dates are generated from a uniform distribution over the interval of $[L(1-T-R/2), L(1-T+R/2)]$ where L denotes an approximate value for makespan. The parameter R denotes relative range of due dates while the parameter T denotes tardiness factor. Therefore, as T increases the due dates become smaller. On the other hand, the difference between job due dates increases as R increases. The generation of due dates by using this method is common in the scheduling literature. The values of T and R are usually taken to be between 0 and 1 in the literature. Therefore, we have also selected R and T values in the same range. In the experimentations, the following LB value was first used instead of L where

$$LB = \max\left(\max_{k=1,\ldots,m}\left\{\sum_{r=1}^{n}\left(t_{[k,r]}+s_{[k,r]}\right)\right\}+\min_{j}\{p_j+s_j\},\right.$$
$$\left.\max_{k=1,\ldots,m}\left\{\min_{r=1,\ldots,n}\left(t_{[k,r]}+s_{[k,r]}\right)\right\}-\min_{j}\{s_j\}+\sum_{r=1}^{n}\left(p_{[r]}+s_{[r]}\right)\right).$$

Nevertheless, the aforementioned LB may lead to an environment where no job is tardy. Thus, we have generated n random sequences, and computed the average makespan, which may be considered as an upper bound, denoted by LU. Subsequently, the average of the LB and LU is computed to obtain the value of L as an approximate makespan.

The values of parameters utilized in the computational experiments are summarized in Table 2.

Table 2. Parameter values

Parameter	Considered values
N	30, 40, 50, 60, 70
M	3, 5, 8
K	0.4, 0.8, 1.2
R	0.3, 0.5, 0.7
T	0.2, 0.4, 0.6

There are a total of 405 combinations of n, m, k, R, and T values. For each combination of the parameter values, fifty replicates are generated. Therefore, a total of 20,250 problems are considered.

The existing and proposed algorithms are assessed by using the performance measure of percentage error (*Error*). The *Error* is defined as $100(T_{\max}$ of the algorithm $-$ $T_{\max}$ of the best algorithm$)/T_{\max}$ of the best algorithm where $T_{\max}$ denotes maximum tardiness.

Figure 1 indicates the error versus the number of jobs for both the *SDE* and *SMI* algorithms. It is obvious from the figure that the proposed *SMI* algorithm performs significantly better than the existing *SDE* algorithm. The gap between the performances of *SDE* and *SMI* algorithms monotonically increases as n increases. This is another advantage of *SMI* over *SDE*.

Figure 2 summarizes the errors of *SMI* and *SDE* algorithms versus the setup to processing time ratio k. The figure clearly indicates that the *SMI* algorithm performs significantly better than the *SDE* algorithm for k values. The performances of both algorithms *SMI* and *SDE* do not seem to be sensitive to k value.

Given that the CPU times of both algorithms are the same, the error of *SMI* is negligible compared to the error of *SDE* algorithm as the overall average error of the *SMI* algorithm is 0.057 while that of the *SDE* algorithm is 4.17. Therefore, the proposed *SMI* algorithm reduces the error of the best existing *SDE* algorithm by 98.6%.

We also performed statistical tests to verify the conclusions stated above. For example, Figure 3 shows 95% confidence interval graph for the case of $n = 70$, $m = 8$, $R = 0.3$, $T = 0.6$, and $k = 0.8$ for which the performances of the algorithms are the closest. Even in this case, the p-value is less than 0.01, which implies that the error of *SMI* is statistically less than that of *SDE*.

4 Conclusion

We investigate a two-stage assembly flowshop scheduling problem where setup times are considered as separate from processing times. The objective is to minimize maximum tardiness. The literature reveals that the algorithm of Self-Adaptive Differential Evolution (*SDE*) performs as the best for the problem. We propose a new hybrid simulated annealing and insertion algorithm, called *SMI*. We compare the performance of the proposed *SMI* algorithm with that of the best existing algorithm, *SDE*. The computational experiments indicate that the proposed *SMI* algorithm performs significantly better than the existing *SDE* algorithm. More specifically, under the same CPU time, the proposed *SMI* algorithm, on average, reduces the error of the best existing *SDE* algorithm over 90%, which indicates the superiority of the proposed *SMI* algorithm.

References

Al-Anzi, F.S., Allahverdi, A., 2007, "A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times", *European Journal of Operational Research*, Vol. 182, pp. 80–94.

Allahverdi, A., 2015, "Third comprehensive survey on scheduling problems with setup times/costs", *European Journal of Operational Research*, Vol. 246, pp. 345–378.

Allahverdi, A, Al-Anzi, F.S., 2006, "A PSO and a Tabu Search Heuristics for Assembly Scheduling Problem of the Two-Stage Distributed Database Application", *Computers & Operations Research*, Vol. 33, pp. 1056–1080.

Allahverdi, A., Aydilek, H., 2015, "The two stage assembly flowshop scheduling problem to minimize total tardiness", *Journal of Intelligent Manufacturing*, Vol. 26, pp. 225–237.

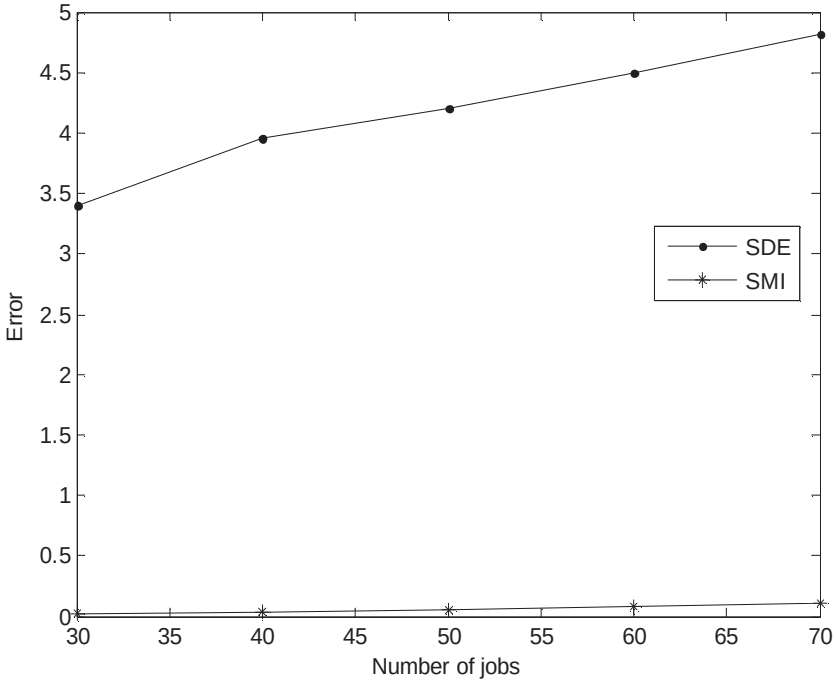


Fig. 1. Error versus number of jobs.

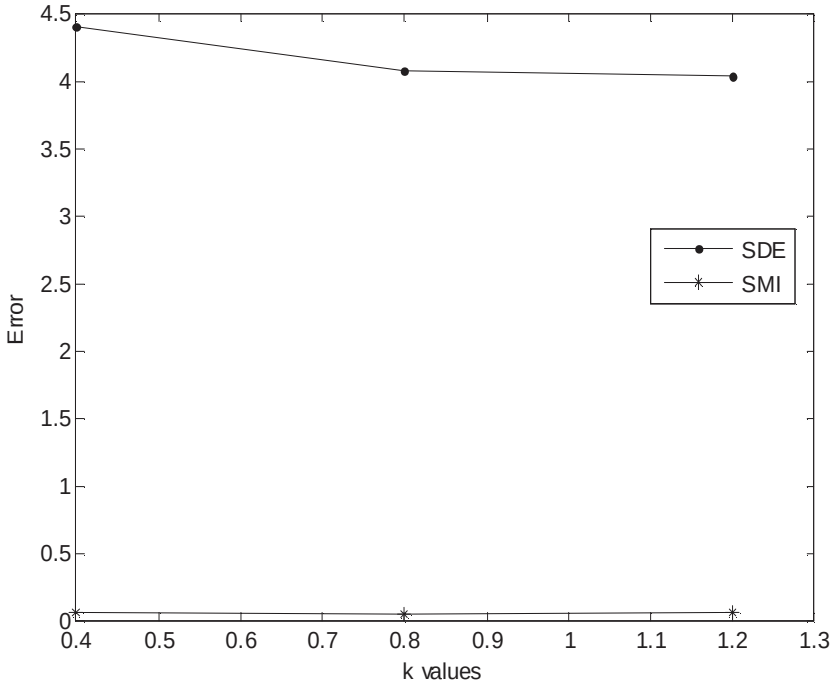


Fig. 2. Error versus k values.

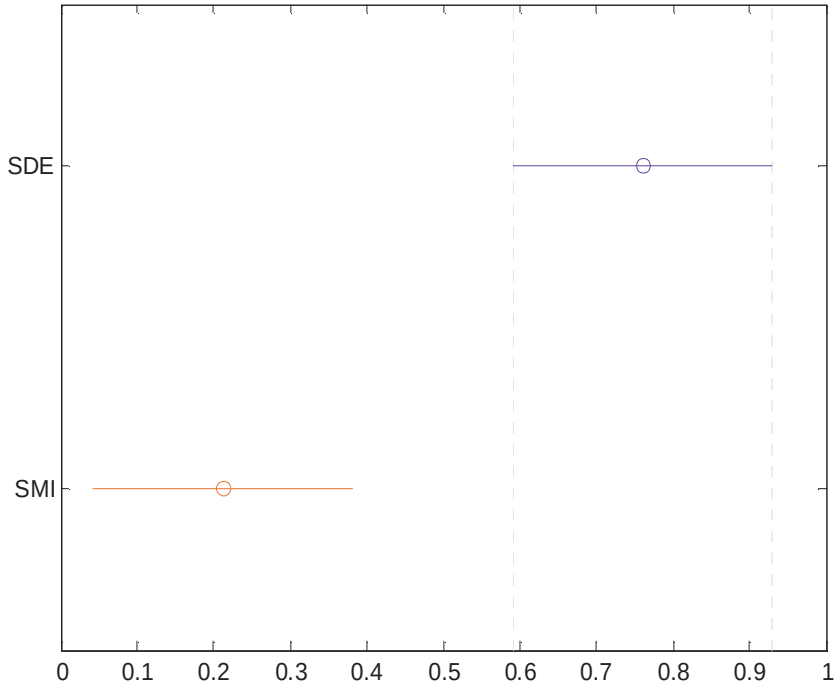


Fig. 3. Mean confidence interval ($n = 70, m = 8, R = 0.3, T = 0.6, k = 0.8$).

No-Wait Flowshop Scheduling Problem to Minimize Total Tardiness Subject to Makespan

Harun Aydilek[1], Asiye Aydilek[1], and Ali Allahverdi[2]

[1] Gulf University for Science and Technology, Kuwait
{aydilek.h,aydilek.a}@gust.edu.kw
[2] Kuwait University, Kuwait
ali.allahverdi@ku.edu.kw

Keywords: Total tardiness, makespan, no-wait, algorithm.

1 Introduction

We address the no-wait flowshop scheduling problem. The no-wait flowshop scheduling problem is applicable in many industries, such as plastic, chemical, and pharmaceutical, e.g. Hall and Sriskandarajah (1996) and Allahverdi (2016).

The total tardiness and makespan performance measures are considered in this paper. Today's fierce global competition makes the total tardiness performance measure important since customer satisfaction is affected by the fulfillment of due dates. On the other hand, the makespan performance measure is directly related to resource utilization as many resources are scarce and efficient utilization of such scarce resources is important for many manufacturing firms. Therefore, both performance measures are critical.

The m-machine no-wait flowshop scheduling problem with the makespan performance measure has been addressed widely in the literature. For example, Tseng and Lin (2010) presented a hybrid genetic algorithm (GA), which hybridizes a novel local search scheme and the GA. Tseng and Lin (2010) indicated that their hybrid GA performs better than the heuristics the earlier algorithms Furthermore, Jarboui *et al.* (2011) presented another hybrid GA algorithm where the variable neighborhood search is utilized to further improve (in the last step) their GA. On the other hand, Lin and Ying (2016) proposed a three-phase heuristic. In the first phase, two constructive heuristics are used to obtain an initial sequence. In the second phase, the problem is transformed into an asymmetric traveling salesman problem and an algorithm is used to improve the initial solution. In the last phase, a mathematical model is used to further improve the solution.

The m-machine no-wait flowshop scheduling problem with a total tardiness (TT) performance measure has also been addressed in the literature. Aldowaisan and Allahverdi (2012) presented several dispatching rules for the problem with respect to total tardiness. They also proposed a simulated annealing (SA) and a genetic algorithm (SA). Furthermore, Liu *et al.* (2013) presented dispatching rules and constructive heuristics, including a modified *NEH*, for the problem. They indicated that the modified *NEH* performs better than the dispatching rules and the constructive heuristics. Moreover, Ding *et al.* (2015) studied the problem and explored the objective function evaluation incremental properties. They presented an accelerated *NEH* and iterated greedy algorithms based on the incremental properties. They indicated that the accelerated algorithms perform much faster than the original algorithms. They further showed that their proposed algorithms perform better than those of Aldowaisan and Allahverdi (2012) and Liu *et al.* (2013).

The aforementioned research addressed a single criterion while many scheduling environments require considering multi criteria. We address m-machine no-wait scheduling problem to minimize total tardiness subject to the constraint that makespan is less than a certain value.

2 Algorithms

We propose an algorithm and adapt three existing algorithms to our problem. The existing algorithms are given in the next subsection while the proposed algorithm is presented in the following subsection.

2.1 Existing algorithms

The m-machine no-wait flowshop scheduling problem to minimize total tardiness was addressed by Aldowaisan and Allahverdi (2012) who presented an algorithm, called *FISA*, which was shown to perform as the best out of the six algorithms they considered. Moreover, Liu *et al.* (2013) also considered the same problem and proposed six heuristic approaches and indicated that the heuristic *MNEH* is the best. In addition, Ding *et al.* (2015) provided three algorithms and indicated that the algorithm *AIG1* performs the best. We adapt the algorithms *FISA*, *MNEH*, and *AIG1* to our problem, which are denoted by *A-FISA*, *A-MNEH*, and *A-AIG1*. We propose a new algorithm, which is called Algorithm *HA*, in the next subsection and compare our algorithm with the existing best algorithms of *FISA*, *MNEH*, and *AIG1*.

2.2 The proposed algorithm (*HA*)

The algorithm *HA* utilizes both the simulated annealing algorithm and the insertion algorithm.

Algorithm *HA*

1. Obtain a C value, and choose an initial sequence s_i, set the parameters t_i, t_f, λ, N and I, set the sequence $st = s_i$, and $i = 1$
2. Set the intermediate temperature $tt = t_i$
3. Generate a sequence by swapping two random jobs of st and call it ss
4. If $TT(ss) < TT(st)$ then update st with ss. Otherwise, update st with ss if $rand < e^{-\frac{D}{tt}}$ where $D = (TT(ss) - TT(st))/TT(st)$ and $rand$ is $U[0, 1]$
5. Update the intermediate temperature tt such that, $tt = tt \cdot \lambda$
6. If $tt < t_f$, go to Step 7, otherwise go to Step 3
7. If $C_{\max}(st) < C$, update $i = i + 1$ and go to Step 15. Otherwise, go to Step 8
8. Set $pi = n$
9. Set $pj = 1$
10. Insert the job in position pi of the sequence st to position pj and call the new sequence sm
11. Evaluate $C1 = C_{\max}(sm)$, and $C2 = C_{\max}(st)$. If $C1 < C$, update st with sm, and update $i = i + 1$, then go to Step 15. Otherwise, go to Step 12
12. Update $pj = pj + 1$. If $C1 < C2$ then update st with sm. Then go to Step 10 if $pj < n$. Otherwise, go to Step 13
13. Update $pi = pi - 1$, and go to Step 9 if $pi > 0$. Otherwise, go to Step 14
14. Update st with si if $C_{\max}(st) > C$. Update $i = i + 1$ and go to Step 15
15. If $i < I$, go to Step 2

The parameters of the simulated annealing part of the algorithm are calibrated based on the values given in the following table. Selected values are 0.14 for t_i, 0.001 for t_f, 0.98 for λ, and 20 for N.

3 Algorithm evaluation

Computations were conducted on a PC with Intel Core i7-3520M CPU processor of 2.9 GHz with 8 GB RAM. An appropriate C value is usually given by the scheduler as stated earlier. However, there is a need to know the C value for the computational experiments. First we reduce the m-machine problem into a two-machine problem such that the processing time of machine one is the sum of the processing times on the first $m/2$ ($(m+1)/2$ if m is odd) machines while the processing time of machine two is the sum of the processing times on the remaining $m/2$ ($(m-1)/2$ if m is odd) machines. Then, we apply Johnson's algorithm to the two machine problem to obtain a sequence s. Next, we take the first job in the sequence s and insert it in all the n positions of the sequence s which results in n different sequences. We take the minimum $C_{\max}$ of all the n sequences, which is the C value.

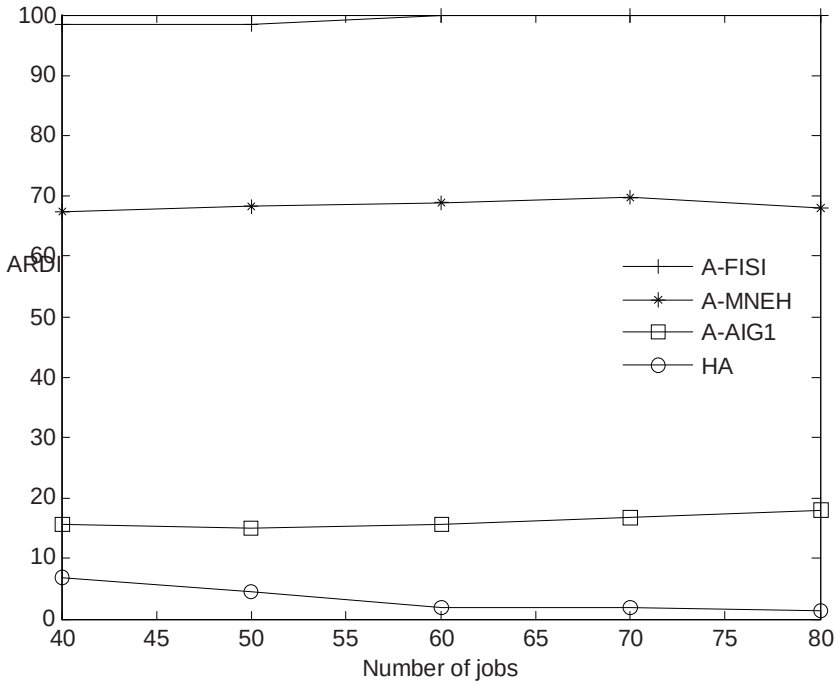


Fig. 1. *ARDI* values of algorithms with respect to n values.

The uniform distribution of $U(1, 99)$ was used to generate processing times on all the m machines. The uniform distribution of $U[LB(1 - T - R/2), LB(1 - T + R/2)]$ is used in generating job due dates where LB denotes an approximate value for makespan. The parameter R indicates a relative range of due dates while the parameter T denotes a tardiness factor, i.e., a larger T value results in a smaller due date. In contrast, as the R value increases, the difference between job due dates increases. The values of T and R are usually taken to be between 0 and 1 in the literature. Thus, we have also selected R and T values between 0 and 1. We use a lower bound on makespan LB which is used by Aldowaisan and Allahverdi (2012). The utilized values of n, m, R, and T are summarized in Table 1.

Table 1. Parameter values

Parameter	Considered values
n	40, 50, 60, 70, 80
m	3, 5, 10, 12
R	0.2, 0.6, 1.0,
T	0.2, 0.4, 0.6,

The performance measure utilized in evaluating the algorithms is the Average Relative Deviation Index ($ARDI$) as a percentage, which is

$$ARDI = \frac{100}{N_r} \sum_{k=1}^{N_r} \frac{TT_k - TT_{best}}{TT_{worst} - TT_{best}}$$

Figure 1 summarizes the $ARDI$ values of the proposed algorithm HA, and the adapted algorithms of *A-FISA*, *A-MNEH*, and *A-AIG1* with respect to n. The figure clearly shows that the algorithms HA performs much better than the others.

The overall average $ARDI$ values of the algorithms *A-FISA*, *A-MNEH*, *A-AIG1*, and HA are 98.5, 65.1, 17.6, and 4.6, respectively. Therefore, the proposed algorithm HA reduces the error of the best adapted algorithm *A-AIG1* by 74%. It should be noted that the CPU times (less than two minutes) of the algorithms are same.

The aforementioned conclusions are statistically tested by using the Tukey Honest Significant Difference (HSD) test at $\alpha = 0.025$. Figures 2 shows the results for a combination of the parameters, which is representative of the vast majority of the combinations. The statistical results, in general, validate the earlier conclusions.

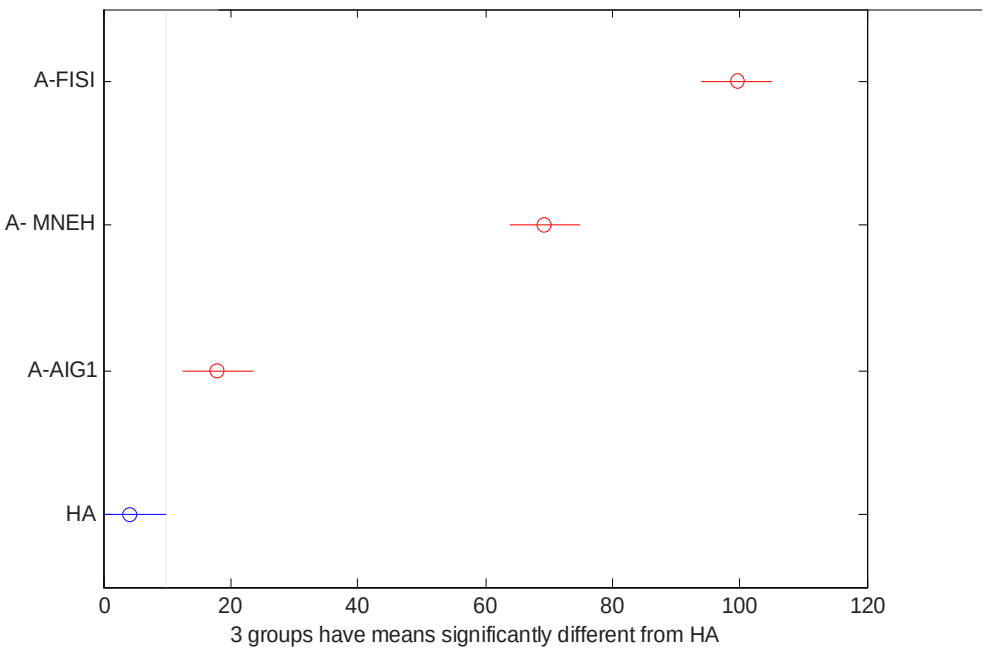


Fig. 2. Confidence intervals for $n = 50$, $m = 10$, $R = 0.2$, $T = 0.6$.

4 Conclusion

We consider the m-machine no-wait flowshop scheduling problem to minimize total tardiness subject to the constraint that the makespan is less than a given value. We propose an algorithm, which is a combination of simulated annealing and insertion algorithm. Moreover, we adapt three best existing algorithms for minimizing total tardiness to our problem. We conduct extensive computational experiments to compare the performance of our proposed algorithm with the three best existing algorithms under the same *CPU* times. The computational analysis indicates that the error of algorithm is 74 percent smaller than that of the best of the three adapted algorithms. All the results are statistically verified.

References

Allahverdi, A., 2016, "A survey of scheduling problems with no-wait in process", *European Journal of Operational Research*, Vol. 255, pp. 665–686.

Aldowaisan, T., Allahverdi, A., 2012, "Minimizing total tardiness in no-wait flowshops", *Foundations of Computing and Decision Sciences*, Vol. 37, pp. 149–162.

Ding, J.-Y., Song, S., Zhang, R., Gupta, J.N.D., Wu, C., 2015, "Accelerated methods for total tardiness minimization in no-wait flowshops", *Int. Journal of Production Research*, Vol. 53, pp. 1002–1018.

Hall, N.G., and Sriskandarajah, C. 1996, "A survey of machine scheduling problems with blocking and no-wait in process", *Operations Research*, Vol. 44, pp. 510–525.

Jarboui, B., Eddaly, M., Siarry, P., 2011, "A hybrid genetic algorithm for solving no-wait flowshop scheduling problems", *Int. Journal of Advanced Manufacturing Technology*, Vol. 54, pp. 1129–1143.

Lin, S.-W., Ying, K.-C., 2016, "Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics", *Omega*, Vol. 64, pp. 115–125.

Liu, G., Song, S., Wu, C., 2013, "Some heuristics for no-wait flowshops with total tardiness criterion", *Computers and Operations Research*, Vol. 40, pp. 521–525.

Tseng, L.Y., Lin, Y.T., 2010, "A hybrid genetic algorithm for no-wait flowshop scheduling problem", *Int. Journal of Production Economics*, Vol. 128, pp. 144–152.

R.H. Möhring, F.J. Radermacher and G. Weiss, "Stochastic scheduling problems II - Set strategies", ZOR - *Zeitschrift für Operations Research*, vol. 29, pp. 65–104, 1985.

F. Stork, Stochastic resource-constrained project scheduling, Ph.D. Thesis, Techische Universität Berlin, Germany, 2001.

S. Van de Vonder, E. Demeulemeester, W. Herroelen and R. Leus, "The trade-off between stability and makespan in resource-constrained project scheduling", *International Journal of Production Research*, vol. 44, no. 2, pp. 215–236, 2006.

S. Van de Vonder, F. Ballestin, E. Demeulemeester and W. Herroelen, "Heuristic procedures for reactive project scheduling", *Computers & Industrial Engineering*, vol. 52, no. 1, pp. 11–28, 2007.

G. Yu and X. Qi, *Disruption management - Framework, models and applications*, New Jersey: World Scientific, 2004.

J. Wang, "Constraint-based schedule repair for product development projects with time-limited Constraints", *International Journal of Production Economics*, 95, 399–414, 2005.

G. Zhu, J.F. Bard and G. Yu, "Disruption management for resource-constrained project scheduling", *Journal of the Operational Research Society*, vol. 56, pp. 365–381, 2005.

A Robust Optimization Model for the Multi-mode Resource Constrained Project Scheduling Problem with Uncertain Activity Durations

Noemie Balouka[1] and Izack Cohen[2]

Technion - Israel Institute of Technology, Haifa, Israel
[1]nbalouka@tx.technion.ac.il
[2] izikc@technion.ac.il

Keywords: project scheduling, uncertainty, robust optimization.

1 Introduction and Motivation

The multi-mode resource constrained project scheduling problem (MRCPSP) aims to minimize the makespan by selecting activities' modes and scheduling project activities under precedence and resource constraints. It extends the resource constrained project scheduling problem (RCPSP), by assuming that activities can be performed under one of several modes, where a mode determines an activity's duration and its resource requirements. RCPSP and MRCPSP are NP-hard (Blazewicz et al., 1983), so real problems are often solved by heuristic methods. Most researches consider the RCPSP and MRCPSP parameters as deterministic although some of the research treats uncertainty. For example, Herroelen and Leus (2005) review procedures for generating feasible baseline schedules with respect to a specific objective (e.g. makespan minimization or net present value maximization) under uncertainty, and mention four approaches: reactive scheduling, stochastic optimization, fuzzy project scheduling and robust or proactive scheduling. We focus on the latter; contrary to reactive scheduling that revises the baseline schedule after an unexpected event, proactive scheduling plans robust baseline schedules. This approach origins from the concept of robust optimization, a relatively recent optimization approach (Ben-Tal et al., 2009) that aims to construct a solution that is feasible for any realization within a given uncertainty set. Proactive scheduling develops a robust schedule that anticipates the variability during project execution. Daniels and Kouvelis (1995) are among the first to introduce the approach of robust optimization in a scheduling environment. However, they consider a single-machine where the job's processing times are uncertain and the objective is to minimize the total flow time over all jobs. Cohen et al. (2007) apply the robust optimization approach to the stochastic time—cost tradeoff problem and show that the price of robustness is relatively small when using ellipsoid uncertainty sets. To the best of our knowledge, there are no papers presenting a robust optimization model for the MRCPSP. There are two recent papers developing a robust optimization model for the RCPSP. In the first paper (Artigues et al., 2013), a minimax absolute regret problem is presented to find a schedule minimizing the maximum absolute regret aver all duration scenarios. Bruni et al. (2017) develop an adjustable robust optimization problem to find the schedule that minimizes the worst case makespan over all duration realizations varying through a polyhedral uncertainty set. A Benders approach is considered and a polynomially solvable case is identified for a specific uncertainty set. In the present paper, we extend this robust model to the MRCPSP.

2 Model Description

The project network is modeled through a directed graph defined over the set of nodes $V = 0, ..., n + 1$. The dummy nodes 0 and $n + 1$ represent the start and end of the project,

receptively. The other nodes are non-dummy activities. The set of arcs E represents precedence relations between the activities. We assume K types of renewable resources, with a finite capacity per period denoted by R_k. Each activity j can be performed in one of $|M_j|$ modes, where each mode $m_j \in M_j$ is characterized by a duration d_{jm} and a resource requirement of type k, r_{jm_jk}. A solution to the MRCPSP is a vector of mode combinations $(m_1, ..., m_n)$ and a vector of non negative starting times $(S_0, ..., S_{n+1})$ which result in a schedule that satisfies the precedence and resource constraints. Given a mode combination $\underline{m} = (m_1, ..., m_n)$, we define $F_{\underline{m}} \subseteq V$ to be any subset of activities without precedence relations between them such that $\sum_{i \in F_{\underline{m}}} r_{im_ik} > R_k$ for at least one $k \in K$. This set is called forbidden since its activities cannot be performed in parallel because of resource conflicts. We denote $\mathcal{F}_{\underline{m}}$ as a minimal forbidden set that corresponds to a mode combination $\underline{m}$, such that each of its subsets is not a forbidden set. The MRCPSP solution can be reduced to an optimal mode combination $\underline{m}$ and a optimal selection of the set $X_{\underline{m}} \subseteq (V \times V) \backslash E$ of extra precedences such that the extended graph $G'(V, E \cup X_{\underline{m}})$ is acyclic and $\mathcal{F}_{\underline{m}}(T(E \cup X_{\underline{m}})) = \varnothing$ where $T(A)$ denotes the transitive closure of the set A. We assume that the uncertain data varies within a so-called uncertainty set. A robust feasible solution guarantees that there are no violations of constraints for all possible realizations within a considered uncertainty set. An optimal robust solution is one that solves the robust optimization problem. This new optimization problem is called the robust counterpart. Tractability of robust counterparts strongly depends on the uncertainty set's nature. Ben-Tal et al. (2009) show that a robust counterpart of an uncertain linear problem is also linear under a polyhedral uncertainty set. A typical example of a polyhedral set is the case of interval uncertainty, also called a Box. For a non-polyhedral set, such as the case of ellipsoidal uncertainty, Ben-Tal et al. (2009) show that a robust counterpart of an uncertain linear problem is quadratic. Since we formulate our problem with integer variables, we assume that uncertainty sets are polyhedral in order to maintain linear constraints. In our model, uncertain durations are defined over the polyhedral uncertainty set $\theta \subseteq \mathbb{N}^{n \times M}$. For convenience, we denote the subset $\theta_{\underline{m}}$ as the uncertain duration's support according to a given mode assignment. Indeed, for a given mode combination $\underline{m}$, the corresponding durations vector is $d_{\underline{m}} = (d_{jm_1}, ..., d_{jm_n})$ which is included in $\mathbb{N}^J$. The set of combination modes is denoted by $M \subseteq \mathbb{N}^n$.
We define the robust multi-mode resource constrained scheduling problem (RMRCPSP) as a robust optimization problem. The objective is to find a mode assignment and a sufficient selection that minimizes the worst case makespan under uncertainty:

$$\min_{\underline{m} \in M, X_{\underline{m}} \in \mathcal{X}_{\underline{m}}, S(\cdot)} \max_{d_{\underline{m}} \in \theta_{\underline{m}}} S_{n+1}(d_{\underline{m}}) \tag{1}$$

$$S_0 = 0 \tag{2}$$

$$S_j(d_{\underline{m}}) - S_i(d_{\underline{m}}) \geq d_{im_i}\,, \forall (i,j) \in E \cup X_{\underline{m}}\,, \forall d_{\underline{m}} \in \theta_{\underline{m}} \tag{3}$$

The mode and selection decisions, $\underline{m} \in M$ and $X_{\underline{m}} \in \mathcal{X}_{\underline{m}}$ respectively, represent a first-stage decisions that made before the project's execution. That is, before activity durations are known. The second-stage decisions concern the starting times $S_j(d_{\underline{m}})$ of each activity under the duration realization $d_{\underline{m}} \in \theta_{\underline{m}}$. When the uncertainty set is a box, it can be shown that solving the RMRCPSP is equivalent to solving a deterministic MRCPSP for the worst-case activity duration vector.

3 Development of an Analytical Solution Approach

The structure of the RMRCPSP encourages us to use a Benders' solution approach for solving it (Benders, 1962). Bender's decomposition algorithm is an iterative algorithm; at the initial iteration, the lower bound of the objective equals $-\infty$ and its upper bound equals ∞. At each iteration, we solve a master problem that provides an updated lower bound and a subproblem that provides an updated upper bound. Once the subproblem

is solved, valid cuts are calculated and added to the master problem formulation. The algorithm stops when the lower bound converges to the upper bound.

3.1 The Master Problem

The master problem determines mode and sufficient selections, and its objective is to minimize the lower bound of the worst case makespan. The mode selection decisions variables are binary and denoted by x_{jm_j} (equals to 1 if activity j is executed under mode $m_j \in M_j$). The variables about sufficient selections are modeled by resources flow variables, f_{ijk}, corresponding to the number of resources k units transfered from activity i to activity j and by binary variables y_{ij}, representing all the precedence relations in $E \cup X_{\underline{m}}$ (including its transitive closure). In order to improve the computational performance of the master problem, we incorporate relaxation.

3.2 The Subproblem

After selecting the modes and sequencing activities in the master problem, without the necessity to consider their durations and the uncertainty set, now, we have to schedule the activities in the subproblem. The objective is to minimize the worst case makespan when uncertain durations are defined over a polyhedral uncertainty set. The optimal solution of the master problem at iteration t determines an acyclic subgraph $G'(V, U^t)$ where the set U^t is defined as follows: $U^t = \{(i,j) \in V \times V : y_{ij}^{*t} = 1\}$. We accordingly update $X_{\underline{m}^{*t}}$, the optimal selections at iteration t, when $\underline{m}^{*t} = (m_1^{*t}, ..., m_n^{*t})$ denotes the optimal mode assignment resulted from the master problem at t. Then, a feasible solution for the RMRCPSP can be determined by solving the following subproblem:

$$\min_{S(\cdot)} \max_{d \in \theta_{\underline{m}^{*t}}} S_{J+1}(d) \tag{4}$$

$$S_0 = 0 \tag{5}$$

$$S_j(d) - S_i(d) \geq d_{jm_j}, \forall (i,j) \in U^t, \forall d \in \theta_{\underline{m}^{*t}} \tag{6}$$

We can rewrite the subproblem as:

$$\max_{d \in \theta_{\underline{m}^{*t}}} \min_{S \in \Omega(X_{\underline{m}^{*t}}, \theta_{\underline{m}^{*t}})} S_{J+1}(d) \tag{7}$$

where: $\Omega(X_{\underline{m}^{*t}}, \theta_{\underline{m}^{*t}}) = \{S \in R_+^{n+2} : S_0 = 0, S_j - S_i \geq d_{jm_j^{*t}}, \forall (i,j) \in U^t\}$, is a set of activities' starting times. Using a strong duality result (Beck and Ben-Tal, 2009), we state that the optimizing under the worst-case makespan in the primal (7), at a generic iteration t, is equivalent to optimizing under the best case in the dual. The objective function of the dual problem is non-linear. Then, we focus on the budgeted uncertainty set inspired by Bertsimas and Sim (2003). The advantage of this polyhedral set is in its flexibility to adjust the level of conservatism and robustness through the budget Γ, representing the number of activities which are allowed to deviate from their nominal durations. The parameter Γ can vary between 0 and n. We assume that each activity duration j performed in mode m_j has a nominal value, $\hat{d}_{jm_j}$ and a maximal deviation denoted by $\tilde{d}_{jm_j}$. Given an optimal mode combination $\underline{m}^{*t} = (m_1^{*t}, ..., m_n^{*t})$, the uncertainty set is defined as:

$$\theta_{\underline{m}^{*t}} = \{d_{jm_j^{*t}} | j \in V, d_{jm_j^{*t}} = \hat{d}_{jm_j^{*t}} + \xi_j \tilde{d}_{jm_j^{*t}}, 0 \leq \xi_j \leq 1, \sum_{j \in V} \xi_j \leq \Gamma\}.$$

Under this uncertainty set, we can reformulate subproblem (7) as a mixed-integer linear problem.

3.3 Optimality Cuts

Once the subproblem is solved, two valid cuts are calculated and incorporated to the master problem.

Proposition 1. *Given a finite global lower bound L of the problem (1)-(3), and the optimal solutions at iteration t, x^{*t}, y^{*t}, M^{*t}, the following constraints are valid optimality cuts.*

$$\eta \geq (M^{*t} - L) \cdot \sum_{(i,j) \in X_{\underline{m}^{*t}}} [1/3 \cdot (y_{ij} + x_{m_i^{*t}} + x_{m_j^{*t}}) - N \cdot (3 - y_{ij} - x_{m_i^{*t}} - x_{m_j^{*t}})] \quad (8)$$
$$- (M^{*t} - L)(|X_{\underline{m}^{*t}}| - 1) + L \,,$$

when N is a large number.

Proof. It always holds that:
$\sum_{(i,j) \in X_{\underline{m}^{*t}}} [1/3 \cdot (y_{ij} + x_{m_i^{*t}} + x_{m_j^{*t}}) - N \cdot (3 - y_{ij} - x_{m_i^{*t}} - x_{m_j^{*t}})] \leq |X_{\underline{m}^{*t}}|$, with equality only when $x = x^{*t}$ and $y = y^{*t}$.
In this case, we have that:
$\sum_{(i,j) \in X_{\underline{m}^{*t}}} [1/3 \cdot (y_{ij} + x_{m_i^{*t}} + x_{m_j^{*t}}) - N \cdot (3 - y_{ij} - x_{m_i^{*t}} - x_{m_j^{*t}})] - |X_{\underline{m}^{*t}}| = 0$, and then the right-hand side takes the value M^{*t}.
Otherwise, $\sum_{(i,j) \in X_{\underline{m}^{*t}}} [1/3 \cdot (y_{ij} + x_{m_i^{*t}} + x_{m_j^{*t}}) - N \cdot (3 - y_{ij} - x_{m_i^{*t}} - x_{m_j^{*t}})] < |X_{\underline{m}^{*t}}|$, and then the right-hand side takes a value less than or equal to L.

□

Proposition 2. *The number of cuts that can be added to the master problem is finite, and then the procedure is finite.*

Proof. Proposition 1 in Laporte and Louveaux (1993). We can apply this result here because x and y are integer variables. □

Summary

This research formulates, for the first time to the best of our knowledge, the robust MRPCPSP and develops an analytical solution approach.

Acknowledgements

We thank the Israeli Ministry of Science and Technology for supporting our research.

References

Christian Artigues, Roel Leus, and Fabrice Talla Nobibon. Robust optimization for resource-constrained project scheduling with uncertain activity durations. *Flexible Services and Manufacturing Journal*, 25(1-2):175–205, 2013.

Amir Beck and Aharon Ben-Tal. Duality in robust optimization: primal worst equals dual best. *Operations Research Letters*, 37(1):1–6, 2009.

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

Jacques F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical programming*, 98(1-3):49–71, 2003.

Jacek Blazewicz, Jan Karel Lenstra, and AHG Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.

Maria Elena Bruni, Luigi Di Puglia Pugliese, Patrizia Beraldi, and Francesca Guerriero. An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations. *Omega*, 71:66–84, 2017.

Izack Cohen, Boaz Golany, and Avraham Shtub. The stochastic time–cost tradeoff problem: a robust optimization approach. *Networks*, 49(2):175–188, 2007.

Richard L Daniels and Panagiotis Kouvelis. Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, 41(2):363–376, 1995.

Willy Herroelen and Roel Leus. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2):289–306, 2005.

Gilbert Laporte and François V Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142, 1993.

Scheduling data gathering with limited base station memory

Joanna Berlińska

Faculty of Mathematics and Computer Science, Adam Mickiewicz University in Poznań,
Umultowska 87, 61-614 Poznań, Poland
Joanna.Berlinska@amu.edu.pl

Keywords: scheduling, data gathering network, limited memory, flow shop.

1 Introduction

Data gathering networks are widely used in many types of contemporary applications. Distributed computing introduces the need for collecting the results obtained by remote workers. Data gathering wireless sensor networks find environmental, military, health and home applications (Akyildiz *et al.* 2002). Scheduling algorithms for data gathering networks were proposed, e.g., by Moges and Robertazzi (2006), Choi and Robertazzi (2008), Berlińska (2014) and Berlińska (2015).

In this work, we analyze gathering data in a network with limited base station memory. A dataset being received or processed by the base station occupies a block of memory of given size. The total size of coexisting memory blocks cannot exceed the base station buffer capacity. Our goal is to gather and process all data within the minimum possible time.

2 Problem formulation and complexity

We study a data gathering network consisting of m identical worker nodes $P_1, \ldots, P_m$ and a single base station. Node P_i has to transfer dataset D_i of size α_i directly to the base station. When dataset D_i starts being sent, a memory block of size α_i is allocated at the base station. The base station has limited memory of size $B \geq \max_{i=1}^{m}\{\alpha_i\}$. The transfer of dataset D_i may start only if the amount of available memory is at least α_i. Sending dataset D_i takes time $C\alpha_i$, where C is the network communication rate (inverse of speed). After dataset D_i is transferred, it has to be processed by the base station. This takes time $A\alpha_i$, where A is the base station computation rate. Datasets are processed in the order in which they were received. As soon as processing a dataset finishes, the corresponding memory block is released. It is assumed that both communication and computation on a dataset are non-preemptive. The base station can communicate with at most one node at a time and it can process at most one dataset at a time. The scheduling problem is to organize dataset transfers so that the total data gathering and processing time is minimized.

We prove that the above problem is strongly NP-hard even if $A = C = 1$, using a pseudopolynomial reduction from the bin packing problem (Garey and Johnson 1979).

3 Related work

As only one node can communicate with the base station at a time, our data gathering network can be seen as a two-machine flow shop, where the communication network is the first machine, and the base station is the second machine. Job i consists of two operations: sending and processing dataset D_i, and requires α_i units of base station memory resource. Thus, we solve a resource-constrained flow shop scheduling problem (Błażewicz *et al.* 1983). It may seem similar to two-machine flow shop scheduling with limited buffer storage (see,

e.g., Leisten (1990)), but there are substantial differences between them. In a flow shop with limited buffer storage, the buffer can hold a fixed number of jobs, and a job is stored in the buffer when it has already been processed on the first machine but not yet started on the second machine. In our problem, the buffer can hold a fixed amount of data (for example, only one big dataset, but up to three small datasets), and the buffer is occupied by a dataset not only between, but also during its transfer and processing.

Lin and Huang (2006) proposed a relocation problem with a second working crew for resource recycling. Each job was executed on two machines in a flow shop style. The i-th job required α_i units of a resource, and returned β_i units of this resource on completion. The goal was to minimize the makespan while not exceeding the available amount of the resource. This problem, denoted by $F2|rp|C_{max}$, was shown to be strongly NP-hard, and heuristic algorithms for solving it were proposed. The problem was further analyzed by Cheng *et al.* (2012), who formulated it as an integer linear program. Complexity results for a number of special cases of the problem were also presented.

Our data gathering scheduling problem is equivalent to yet another special case of problem $F2|rp|C_{max}$, which can be denoted by $F2|rp, p_i = C\alpha_i, q_i = A\alpha_i, \beta_i = \alpha_i|C_{max}$, and was not studied in the earlier literature.

4 Algorithms

In our problem, a schedule is determined by the order in which the datasets are transferred to the base station. Each dataset is sent without unnecessary delay, as soon as sufficient amount of memory is available.

We first observe the following symmetry property. Suppose that $A = kC$, where $k \geq 1$, and Σ is a schedule of length T for given values of B and $(\alpha_i)_{i=1}^m$. Then, by reversing schedule Σ, we obtain a schedule of length T for the same values of B and $(\alpha_i)_{i=1}^m$, communication rate $C' = kC$ and computation rate $A' = C$. In consequence, we can assume without loss of generality that $A \geq C$.

We propose three "simple" heuristics. Algorithm **Inc** sorts the datasets in the order of increasing sizes. Since $A \geq C$, this is the order that would be returned by the Johnson's algorithm for problem $F2||C_{max}$ (Johnson 1954), and hence, algorithm Inc delivers optimum solutions if the memory limit B is big enough. Algorithm **Alter** starts with sending the smallest dataset, then the greatest one, the second smallest, the second greatest, etc., thus alternating big and small datasets. Finally, algorithm **Rnd** transfers the datasets in a random order. This algorithm will be used to verify the quality of the results delivered by the remaining heuristics.

The second group of algorithms are "advanced" heuristics **IncLocal**, **AlterLocal** and **RndLocal**. Each of them starts with generating a schedule using the corresponding simple heuristic, and then applies the following local search procedure. For each pair of datasets, we check if swapping their positions in the current schedule leads to decreasing the makespan. The swap that results in the shortest schedule is executed, and the search is continued until no further improvement is possible.

Note that our algorithms cover the three heuristics H_1, H_2, H_3 proposed by Lin and Huang (2006) for solving problem $F2|rp|C_{max}$. In our special case, both H_1 and H_2 return the same results as IncLocal, and H_3 is equivalent to RndLocal.

To finish this section, let us observe that the length of a schedule obtained for an arbitrary dataset sequence does not exceed $(A + C)\sum_{i=1}^m \alpha_i$, and $A\sum_{i=1}^m \alpha_i$ is a lower bound on a schedule length. Thus, the approximation ratio of any algorithm for solving our problem is at most $1 + C/A$. Hence, we can say that our problem becomes easier to solve when A gets large in comparison to C.

5 Experimental results

In this section, we compare the quality of the solutions and the computational costs of the proposed heuristics. The algorithms were implemented in C++ and run on an Intel Core i5-2500K CPU @ 3.30 GHz with 6GB RAM. The test instances were constructed as follows. The communication rate was $C = 1$ and the computation rate was $A \in \{1, 2, 5, 10\}$. We generated "small" tests with $m = 10$ and "big" tests with $m = 100$. The dataset sizes α_i were chosen randomly from the interval $[1, 2]$. For a given set of α_i, we computed the minimum amount of memory that allows to hold more than one dataset in the buffer, $B_{min} = \min_{i \neq j}\{\alpha_i + \alpha_j\}$. Then, the memory limit was set to $B = \delta_B B_{min}$, where $\delta_B = 1 + i/10$, for $i = 1, 2, \ldots, 7$. For each triple of m, A and δ_B values, 30 instances were generated. Due to limited space, we report here only on a small subset of the obtained results.

The makespans returned by the heuristics for the small tests were compared to the optimum values computed using the ILP formulation from Cheng *et al.* (2012). It turns out that the local search procedure is very effective, as for each tested setting, the average relative errors of all the advanced heuristics were below 0.5%. The average relative errors of the simple algorithms were between 3% and 20% for the most difficult tests (with $A = 1$).

Constructing optimum solutions for instances with $m = 100$ was not possible because of the exponential complexity of the exact algorithm. Therefore, the obtained makespans were compared to the lower bound computed by disregarding the memory limit and solving problem $F2||C_{max}$ for given A and $(\alpha_i)_{i=1}^m$. The results are summarized in Table 1.

The solutions obtained by the advanced algorithms are much better than those of the simple algorithms. The differences between algorithms IncLocal and AlterLocal are very small, while RndLocal performs slightly worse. However, it is clear that for $\delta_B = 1.2$ the best choice among the simple heuristics is the Inc algorithm, and the results of algorithm Alter are even worse than those of the random algorithm. For $\delta_B = 1.5$ we have the reverse situation: algorithm Alter is the winner, and Inc is even worse than Rnd if $A > 1$.

Table 1. Average relative distance of the solutions from the lower bound, for $m = 100$.

A	δ_B	**Inc**	**Alter**	**Rnd**	**IncLocal**	**AlterLocal**	**RndLocal**
1	1.2	0.814	0.974	0.907	0.702	0.711	0.727
	1.5	0.557	0.463	0.586	0.216	0.211	0.245
2	1.2	0.411	0.495	0.456	0.340	0.347	0.361
	1.5	0.262	0.082	0.231	0.015	0.015	0.039
5	1.2	0.166	0.198	0.183	0.135	0.137	0.143
	1.5	0.109	0.047	0.097	0.009	0.010	0.017
10	1.2	0.084	0.099	0.093	0.070	0.072	0.075
	1.5	0.055	0.020	0.049	0.005	0.004	0.009

We report on the execution times of the algorithms in Table 2. Here we group the results for all tested values of A together. The running time of all simple algorithms is very short, and the advanced algorithms are five orders of magnitude slower. The slowest heuristic is RndLocal, and the relation between IncLocal and AlterLocal depends on δ_B. For $\delta_B = 1.2$, algorithm IncLocal is much faster than AlterLocal, and for $\delta_B = 1.5$ we have the opposite situation. Thus, conforming the (initial) dataset sequence to δ_B value leads to obtaining better schedules in the case of the simple heuristics, and to shorter execution time in the case of the advanced heuristics.

Table 2. Average algorithm running time (in seconds), for $m = 100$.

δ_B	**Inc**	**Alter**	**Rnd**	**IncLocal**	**AlterLocal**	**RndLocal**
1.2	3.18E−3	2.90E−3	3.33E−3	2.08E+2	3.11E+2	3.89E+2
1.5	2.68E−3	2.58E−3	2.40E−3	3.11E+2	2.06E+2	6.30E+2

6 Conclusions

In this work, we analyzed scheduling data gathering with limited base station memory. As we showed that this problem is strongly NP-hard, groups of simple and advanced heuristics were proposed. Their performance was tested in computational experiments. The simple algorithms are very fast, but the results they obtain are not very good in most cases. The advanced heuristics produce high quality schedules, but their execution times are long. We showed that sorting datasets according to their sizes is a good idea if the base station memory limit is rather small. If the base station buffer is big enough to hold the smallest and the biggest dataset together, then alternating small and big datasets allows to obtain better results.

Acknowledgements

This research has been partially supported by the National Science Centre, Poland, grant 2016/23/D/ST6/00410.

References

Akyildiz I.F., W. Su, Y. Sankarasubramaniam and E. Cayirci, 2002, "Wireless sensor networks: a survey", *Computer Networks*, Vol. 38, pp. 393-422.

Berlińska J., 2014, "Communication scheduling in data gathering networks with limited memory", *Applied Mathematics and Computation*, Vol. 235, pp. 530-537.

Berlińska J., 2015, "Scheduling for data gathering networks with data compression", *European Journal of Operational Research*, Vol. 246, pp. 744-749.

Błażewicz J., J.K. Lenstra and A.H.G. Rinnooy Kan, 1983, "Scheduling subject to resource constraints: classification and complexity", *Discrete Applied Mathematics*, Vol. 5, pp. 11-24.

Cheng T.C.E., B.M.T. Lin and H.L. Huang, 2012, "Resource-constrained flowshop scheduling with separate resource recycling operations", *Computers & Operations Research*, Vol. 39, pp. 1206-1212.

Choi K., T.G. Robertazzi, 2008, "Divisible Load Scheduling in Wireless Sensor Networks with Information Utility", In: *IEEE International Performance Computing and Communications Conference 2008: IPCCC 2008*, pp. 9-17.

Garey M.R., D.S. Johnson, 1979. "Computers and intractability: A guide to the theory of NP-completeness", Freeman, San Francisco.

Johnson S.M., 1954, "Optimal two- and three-stage production schedules with setup times included", *Naval Research Logistics Quarterly*, Vol. 1, pp. 61-68.

Leisten R., 1990, "Flowshop sequencing problems with limited buffer storage", *International Journal of Production Research*, Vol. 28, pp. 2085-2100.

Lin B.M.T., H.L. Huang, 2006, "On the relocation problem with a second working crew for resource recycling", *International Journal of Systems Science*, Vol. 37, pp. 27-34.

Moges M., T.G. Robertazzi, 2006, "Wireless Sensor Networks: Scheduling for Measurement and Data Reporting", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 42, pp. 327-340.

A Chance Constrained Optimization Approach for Resource Unconstrained Project Scheduling with Uncertainty in Activity Execution Intensity

Lucio Bianco, Massimiliano Caramia and Stefano Giordani

University of Rome "Tor Vergata", Italy
{bianco,caramia,giordani}@dii.uniroma2.it

Keywords: Chance constrained optimization, PERT, Project scheduling.

1 Introduction

A project network is a directed graph $G = (V, A)$, where the set of nodes V corresponds to the set of activities of the project, and the set A of its arcs represents the set of (classical) finish-to-start precedence relations among pairs of activities, i.e., $A = \{(i, j) : i, j \in V\}$. The graph G is acyclic and models the project in the so called Activity-On-Node (AON) representation; nodes can be topologically numbered assuming a single source dummy node (activity 1) and a single sink dummy node (activity n). Each activity $i \in V \backslash \{1, n\}$ has a duration d_i. No resource constraint is taken into account.

The problem of computing the minimum value of the project completion time (project duration or makespan), under the assumption that d_i's are deterministic values, is known to be polynomially solvable. If activities' durations are uncertain and are modeled by random variables, the project network becomes *stochastic* and the objective changes in determining the project makespan distribution or some characteristic thereof, e.g., its expectation.

In the literature, stochastic project networks are often referred to as PERT-networks, since PERT (Project Evaluation and Review Technique) was one of the first techniques to analyze the stochastic behavior of such networks (see, e.g., Clark, 1962). The originators of PERT proposed to use three estimates for the duration of each activity i, i.e., an optimistic value $\bar{a}_i$, a most likely value $\bar{m}_i$, and a pessimistic estimate $\bar{b}_i$. They modeled each activity duration as a stochastic variable with an appropriate Beta distribution and proposed a simple approximate method to calculate its expectation. The assumption of a Beta density function was a matter of convenience that allowed the derivation of nice approximations for the expected value and the variance of activity durations, but, in practice, these estimates may be far form the actual expected value and variance of a Beta distributed stochastic variable. Besides this, PERT model suffers from several other shortcomings: it assumes that activity durations are independent stochastic variables; it uses the Central Limit theorem assuming the number of critical activities being large enough; moreover, it suffers from the *merge event bias problem* leading PERT to an optimistically biased estimate of the earliest expected activity starting times, and then also of the project duration.

In this paper we try to cope with uncertainty in activity durations in a novel way in order to overcome PERT limitations. In Section 2, we present our approach, while in Section 3 we compare the latter to the PERT model.

2 The Proposed Chance Constrained Optimization Approach

In the following, we present a new approach where activity durations uncertainty is not modeled by means of stochastic variables directly associated with activity durations, as done in most of the stochastic approaches in the literature. We assume that the planning

horizon is $[0, T)$ (all the activities have to be completed within T), discretized into T unit time periods $[0, 1), [1, 2), \ldots, [T-1, T)$, indexed by $t = 1, 2, \ldots, T$. Moreover, for each (non-dummy) activity $i \in V \backslash \{1, n\}$ and for each time period t, we introduce the *execution intensity* x_{it} as a stochastic variable, with $0 \leq x_{it} \leq 1$, representing the fraction of activity i executed in time period t. This means that activity i is completely executed when the sum of the variables x_{it} over time is equal to one. Typically, in classical project scheduling, one assumes that the amount of activity carried out during its execution is flat, i.e., $x_{it} = \frac{1}{d_i}$.

Let s_i be a non-negative variable defining the start time of activity $i \in V$. Moreover, let Δ_i be an integer non-negative variable representing the number of time periods necessary to complete activity i with a probability at least equal to θ.

Since the project makespan can be expressed as s_n, where n is the dummy sink activity of the project network, the problem of minimizing the makespan of our stochastic project network has the following conceptual mathematical program

$$\min s_n \tag{1}$$

$$Prob\left\{\sum_{t=s_i+1}^{s_i+\Delta_i} x_{it} - 1 \geq 0\right\} \geq \theta, \qquad \forall i \in V \tag{2}$$

$$s_i + \Delta_i \leq s_j, \quad \forall (i,j) \in A \tag{3}$$

$$s_i \geq 0, \qquad \forall i \in V \tag{4}$$

$$\Delta_i \geq 0 \text{ and integer}, \forall i \in V \backslash \{1, n\}. \tag{5}$$

The objective function (1) minimizes the start time of the dummy end activity n and thus the makespan of the project. The constraints have the following meaning: Constraints (2) regulate the total amount processed of an activity $i \in V$ over time, i.e., the probability that the summation of the execution fractions of i after Δ_i unit time periods from the start time of i is greater than or equal to 1 must not be less than θ. Constraints (3) model *finish-to-start* precedence relations between i and j, $\forall (i,j) \in A$. Constraints (4) and (5) limit the range of variability of the problem variables.

The above model can be interpreted in terms of *Chance constrained programming* (see, e.g., Prekopa, 1995) that is one of the main approaches for dealing with stochastic optimization problems. The latter has the following form

$$\min_{y \in Y} f(y), \text{ s.t. } Prob\{G(y, \xi) \geq 0\} \geq \theta, \tag{6}$$

where $Y \subseteq \mathcal{R}^r$, ξ is a random vector with probability distribution P supported on a set $\chi \subseteq \mathcal{R}^q$, $f : \mathcal{R}^r \to \mathcal{R}$ is a real valued function, and $G(y, \xi) \geq 0$, with $G : \mathcal{R}^r \times \chi \to \mathcal{R}^s$, refers to a finite system of inequalities. $\theta \in (0, 1)$ is called the probability level and it is chosen by the decision maker in order to model the safety requirements. Sometimes, the probability level is strictly fixed from the very beginning (e.g., $\theta = 0.95, \ldots, 0.99$). Next we first show how to cope with constraints (2) in order to write the chance constrained program (1)–(5) as a deterministic (linear) program and then solving the latter.

2.1 Phase 1: Estimating Δ_i to cope with constraints (2)

Assume that the durations of the activities are sufficiently large, and consider a generic (non-dummy) activity $i \in V \backslash \{1, n\}$. Assuming the stochastic variables x_{it} being independent and identically distributed, by the Central Limit theorem we can state that

$\bar{X}_i = \sum_{t=s_i+1}^{s_i+\Delta_i} x_{it}$ is a Normal stochastic variable. That is, $\bar{X}_i$ is such that the (standardized) stochastic variable $Z_i = \frac{\frac{\bar{X}_i}{\Delta_i} - \mu_i}{\sigma_i}\sqrt{\Delta_i} \sim N(0,1)$, where N is the Normal distribution, and μ_i and σ_i are the mean value and the standard deviation of x_{it}, respectively.

In the following, we show how to calculate the minimum Δ_i such that $Prob\{\bar{X}_i \geq 1\} \geq \theta$, i.e., such that activity i being completed with probability at least equal to θ. The latter can be written as $Prob\left\{Z_i \geq \frac{\frac{1}{\Delta_i} - \mu_i}{\sigma_i}\sqrt{\Delta_i}\right\} \geq \theta$, which means that $\theta \leq Prob\left\{Z_i \geq \frac{\frac{1}{\Delta_i} - \mu_i}{\sigma_i}\sqrt{\Delta_i}\right\} = \Phi\left(-\frac{\frac{1}{\Delta_i} - \mu_i}{\sigma_i}\sqrt{\Delta_i}\right)$ where we exploited the fact that the Normal probability density function is an even function and Φ is the quartile of the Normal distribution which is the inverse of the repartition function in the case of absolutely continuous density function, and, therefore $\Phi^{-1}(\theta) \leq -\frac{\frac{1}{\Delta_i} - \mu_i}{\sigma_i}\sqrt{\Delta_i}$. This inequality can be rewritten as $\Delta_i\mu_i - \Phi^{-1}(\theta)\sigma_i\sqrt{\Delta_i} - 1 \geq 0$, and by solving it in $\sqrt{\Delta_i}$, considering the latter being non-negative, we have that

$$\sqrt{\Delta_i} \geq \frac{\Phi^{-1}(\theta)\sigma_i + \sqrt{[\Phi^{-1}(\theta)]^2\sigma_i^2 + 4\mu_i}}{2\mu_i} = \sqrt{\frac{[\Phi^{-1}(\theta)]^2\sigma_i^2}{4\mu_i^2}} + \sqrt{\frac{[\Phi^{-1}(\theta)]^2\sigma_i^2}{4\mu_i^2} + \frac{1}{\mu_i}}. \quad (7)$$

Assume now that for each (non-dummy) activity i and for all $t = s_i + 1, \ldots, s_i + \Delta_i$, the stochastic variables x_{it} in $[0,1]$, are identically distributed with a Beta (prior) probability density function with parameters $\alpha_i, \beta_i > 1$. Let us therefore consider the additional parameters $a_i = 0$, $b_i = 1$, and $m_i = \frac{\alpha_i - 1}{\alpha_i + \beta_i - 2}$. Parameter a_i identifies the pessimistic value of stochastic variable x_{it}, i.e., an estimate of the minimum fraction of activity i that can be executed in time period t, while parameter b_i identifies its optimistic value, i.e., an estimate of the maximum fraction of activity i that can be executed in time period t, and finally parameter m_i identifies the most likely value or modal value of x_{it}. Accordingly, we have that $\mu_i = \frac{\alpha_i}{\alpha_i + \beta_i}$ and $\sigma_i^2 = \frac{\alpha_i\beta_i}{(\alpha_i+\beta_i)^2(\alpha_i+\beta_i+1)}$.

After some easy calculations and substitutions, equation (7) can be written as,

$$\Delta_i \geq \left[\sqrt{\frac{[\Phi^{-1}(\theta)]^2}{4}\frac{(\alpha_i - 1)/m_i - \alpha_i + 2}{\alpha_i((\alpha_i - 1)/m_i + 3)}} + \sqrt{\frac{[\Phi^{-1}(\theta)]^2}{4}\frac{(\alpha_i - 1)/m_i - \alpha_i + 2}{\alpha_i((\alpha_i - 1)/m_i + 3)} + \frac{(\alpha_i - 1)/m_i + 2}{\alpha_i}}\right]^2. \quad (8)$$

Choosing the minimum integer value of Δ_i fulfilling inequality (8) guaranties that activity i will be completed in Δ_i time periods with probability not less than θ.

2.2 Phase 2: Solving the resulting deterministic problem

With the above choice for Δ_i, for each activity i, constraints (2) are satisfied and hence it can be removed, along with constraints (5), from the chance constrained program (1)–(5). The latter therefore reduces to the well known *natural-date* (linear) problem formulation of the (deterministic) resource unconstrained project scheduling problem, with finish-to-start precedence relations and where Δ_i assumes the role of the duration of activity i, that can be solved in linear time with respect the number of precedence relations.

The optimal solution value of the latter program provides the (minimum) project duration assuring that the project itself is completed with probability not less than $p = \theta^\ell \geq \theta^{n-2}$, where ℓ, with $1 \leq \ell \leq n-2$, is the number of non-dummy activities of the largest activity chain of the project network. Since typically θ is assumed to be very close to 1, i.e. $\theta = 1 - \epsilon$ with $\epsilon > 0$ being a value sufficiently close to 0, we have that $p = (1-\epsilon)^\ell \simeq 1 - \ell\epsilon$, hence the project will be completed with probability not less than $1 - \ell\epsilon$.

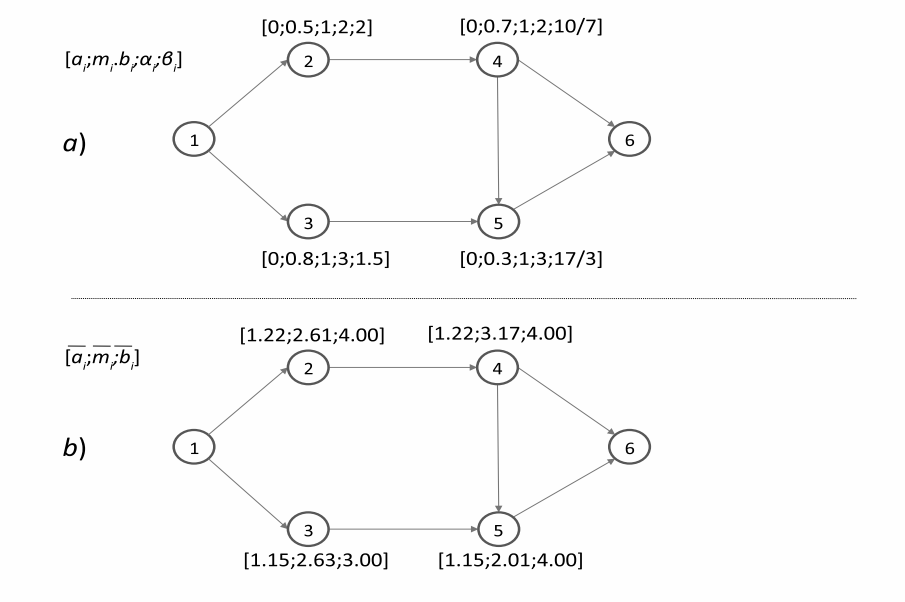


Fig. 1. Example of project network data in the proposed model (a) and in the PERT model (b).

3 Example

In the following example we compare our approach with PERT. In particular we are interested in comparing the project duration estimated by our approach with that provided by the PERT, given the probability p of project completion. The comparison is done by considering, for each non-dummy activity i, the optimistic estimation $\bar{a}_i$ and the pessimistic estimation $\bar{b}_i$ of the duration d_i equal to the value of Δ_i calculated with the approach described in Section 2.1 with $\theta_a = 0.01$ and with $\theta_b = 0.99$, respectively. Finally, given $\bar{a}_i$ and $\bar{b}_i$ and the modal value m_i of x_{it}, we calculate the modal value $\bar{m}_i$ of d_i as $\bar{m}_i = \bar{a}_i + m_i(\bar{b}_i - \bar{a}_i)$. In Figure 1 we depict a project network with (a) the input data for our model and (b) the corresponding input data for the PERT model evaluated as described above; nodes 1 and 6 are dummies. By computing the values of Δ_i with $\theta = 0.99$ for every activity i, with $i = 2, \ldots, 5$, we have $\Delta_2 = \lceil 3.43 \rceil$, $\Delta_3 = \lceil 2.43 \rceil$, $\Delta_4 = \lceil 3.16 \rceil$, and $\Delta_5 = \lceil 3.16 \rceil$. By solving the deterministic model of Phase 2 with these data we have that the project duration is 12 with a probability (approximately) equal to 0.97. As for the PERT model (see part b of Figure 1), we have that the project duration, with the same probability of 0.97, is $7.79 + 1.88 \cdot 0.81 = 9.31$, since its mean value is 7.79 an its standard deviation is 0.81 (1.88 is the number of standard deviations to be added to the mean value to get a project duration estimation with probability equal to 0.97). Hence, for this example, PERT underestimates project duration by more than 22.4% with respect to our approach. We will also compare the project duration given by our approach and by PERT with that calculated with Monte Carlo simulation on test problems with different sizes.

References

1. Clark, C.E., The PERT model for the distribution of an activity time, Operations Research, 10, 405–406 (1962).
2. Prekopa, A., Stochastic Programming. Kluwer Acad. Publ., Dordrecht, Boston (1995).

Single-machine capacitated lot-sizing and scheduling with delivery dates and quantities

Fayez F. Boctor

Centre interuniversitaire de recherche sur les réseaux d'entreprises, la logistique et le transport (CIRRELT)
Faculté des sciences de l'administration, Université Laval, Canada G1V 0A6
fayez.boctor@fsa.ulaval.ca

Keywords: Lot-sizing and scheduling, Heuristics, Integer programming.

1 Introduction

The standard *Capacitated Single-machine Lot-sizing Problem* (CSLP) assumes that the planning horizon is divided into a number of time periods of equal lengths and makes three implicit and simplifying assumptions. First, it implicitly assumes that a lot that starts within a given time period should be finished within this same period. This assumption is needed to simplify the mathematical expression of the capacity constraints. Second, it assumes that setups cannot be carried over from one period to the next. Thus if the last run of a period and the first of the next one process the same product, a setup cost for each of these two runs are included in the cost function. This overestimates the overall setup cost as, in most cases, these two lots can be processed with one setup (see Jans *et al.* 2008). Third, it is assumed that delivery occurs only at the end (or the beginning) of each time period. However, the objective function does not include the inventory holding cost between the time a lot is finished and the beginning of the next period. This assumption is also needed to simplify the mathematical expression of the objective function and to make the determination of processing dates unnecessary. However this underestimates the real inventory holding cost.

Some research publications (see Sox *et al.*, 1999; Gopalakrishnan, 2000; Suerie *et al.*, 2003; Porkka *et al.*, 2003) propose to relax the second assumption and allow setup carryover. However the resulting formulation still: (1) does not include the inventory holding cost between the time a lot is finished and the beginning of the next period; (2) does not allow a lot to be finished beyond the end of its starting period; and (3) does not allow delivery between the beginning and the end of a time period.

A more realistic version of the problem, called hereafter the *single-machine capacitated lot-sizing and scheduling problem with delivery dates and quantities* (SCLSP-DDQ), is studied in this paper. To the best of my knowledge, this problem is not studied in any previous publication. In this version each product has a set of delivery dates and the quantity to deliver at each of these dates is known. It is allowed to start the production of a lot at any time and finish it at any time later within the finite planning horizon provided that the required demands are delivered at the required dates. Also it allows setups to be carried over from one period to the next. Finally, the objective function of this new formulation includes the inventory holding cost of each produced lot from the end of its processing until the delivery of all its units.

Arranging delivery dates in their ascending order, denoted, $t_1, t_2, \ldots, t_L$, we consider that the planning horizon is divided into L time intervals of unequal lengths, where the length of interval l is the time interval between t_{l-1} and t_l. We also allow a lot that starts within a given time interval, say interval l, be finished within the same interval or within a later interval, say interval $l + r$.

2 Mathematical formulation

As mentioned above, the proposed formulation allows for setup carryover, takes into account the inventory holding cost for each produced lot between the finish time and the delivery of all its units. Also, it determines the sequence and processing dates of all lots to process.

2.1 Assumptions:

1. A number of products are to be produced by a single machine (or a production line);
2. Each lot of each product is composed of a number of units of the product;
3. A finite planning horizon and the machine cannot processes more than one product at a time;
4. For each product there is an upper limit on its lots size;
5. Lots of the same product are not necessarily of same size;
6. Processing time of a lot of a given product is composed of the processing time of its units plus a known sequence-independent setup time;
7. Processing time of a lot of a given product can be either proportional to the quantity to produce or constant (e.g. in chemical industries). Both cases are modeled in Boctor (2016); however this paper model the constant processing time case only;
8. Delivery dates and quantities to deliver at these dates are known and deterministic;
9. No backlogging is allowed;
10. Two cost elements are considered: setup cost and inventory holding cost;
11. For each product, unit inventory holding cost per time unit and setup cost are constant.

2.2 Notation:

N number of different products to produce; indexed i

T set of delivery dates indexed in the ascending order; $T = \{t_l; l = 1, \ldots, L\}$

d_{il} quantity of product i to deliver at t_l. This demand is nil if it is not required to deliver any quantity of product i at t_l

P_i processing time of a lot of product i including its setup time

c_i setup cost of a lot of product i

h_i inventory holding cost of a unit of product i per time unit

Q_i upper limit on the size of a lot of product i

F_i the required inventory level of product i at the end of the planning horizon

x_{ipl} a binary that takes the value 1 if a lot of product i is in position p among those to finish between t_{l-1} and t_l (even if it starts before t_{l-1}). Notice that, as we have an upper limit Q_i on the lot size of i, more than one lot of product i may be processed and finished between t_{l-1} and t_l but in different positions in the sequence

q_{ipl} the quantity of product i if produced in the p-th position and finishes between t_{l-1} and t_l

f_{ipl} the finish date of product i if produced in the p-th position and finishes between t_{l-1} and t_l

I_{il} inventory level of product i at t_l just after delivering the demand d_{il}

2.3 The SCLSP-DDQ Model

Find $x_{ipl} \in \{0, 1\}$, $q_{ipl} \geq 0$, $f_{ipl} \geq 0$, and $I_{il} \geq 0$; $i = 1, \ldots, N$; $p = 1, \ldots, N$; $l = 1, \ldots, L$, which:

$$Minimize: \sum_{i=1}^{N}\sum_{l=1}^{L} h_i I_{i,l-1}(t_l - t_{l-1}) + \sum_{i=1}^{N}\sum_{p=1}^{N}\sum_{l=1}^{L} h_i q_{ipl}(t_l - f_{ipl}) + \sum_{i=1}^{N}\sum_{p=1}^{N}\sum_{l=1}^{L} c_i x_{ipl} \quad (1)$$

$$Subject\ to: \sum_{i=1}^{N} x_{ipl} \leq 1,\ p = 1,\ldots,N,\ l = 1,\ldots,L \quad (2)$$

$$\sum_{i=1}^{N} x_{ipl} \leq \sum_{i=1}^{N} x_{i,p-1,l},\ p = 2,\ldots,N,\ l = 1,\ldots,L \quad (3)$$

$$q_{ipl} \leq x_{ipl} Q_i,\ i = 1,\ldots,N, p = 1,\ldots,N,\ l = 1,\ldots,L \quad (4)$$

$$I_{il} = I_{i,l-1} + \sum_{p=1}^{N} q_{ipl} - d_{il},\ i = 1,\ldots,N,\ l = 1,\ldots,L-1 \quad (5)$$

$$I_{i,L-1} + \sum_{p=1}^{N} q_{ipL} - d_{iL} = F_i,\ i = 1,\ldots,N \quad (6)$$

$$f_{i11} \geq x_{i11} P_i,\ i = 1,\ldots,N \quad (7)$$

$$f_{i1l} \geq x_{i1l} t_{l-1},\ i = 1,\ldots,N,\ l = 2,\ldots,L \quad (8)$$

$$f_{ipl} \geq x_{ipl} f_{j,p-1,l} + P_i x_{ipl} q_{ipl},\ i = 1,\ldots,N,\ j = 1,\ldots,N, p = 2,\ldots,N, l = 1,\ldots,L \quad (9)$$

$$\sum_{i=1}^{N}\sum_{p=2}^{N} P_i x_{ipj} \leq t_l - \sum_{i=1}^{N} x_{i1l} x_{i1l},\ l = 1,\ldots,L. \quad (10)$$

The first term in the objective function (1) gives the inventory holding cost of items over the time intervals t_{l-1} and t_l. The second term gives the inventory holding cost of the produced items between the end of their processing and the following delivery date. The third term gives the setup cost of the processed lots. Constraints (2) require that there is at most one product in each position of each time interval (i.e., the interval between two consecutive delivery dates). Constraints (3) make sure that if there is no product in a position then there are no products in the next position. Constraints (4) make sure that the produced quantities do not exceed the lot-size upper limit. Constraints (5) and (6) determine the inventory levels and assure that the demands are fulfilled without backlogs. Constraints (7), (8) and (9) determines the finish times of the lots to produce. Finally, constraints (10) make sure that we have enough time to produce the required quantities in each time interval (capacity constraints).

This model is difficult to solve as it contains a large number of variables and constraints. It is composed of N^2L binary variables, $NL(2N+1)+L$ continuous variables and $NL(2N+1)+L$ constraints. Thus if we have 10 products and 20 delivery dates our model has 2000 binary variables, 4220 continuous variables and 4220 constraints. It is also important to note that the objective functions (1) as well as constraints (9) and (10) are quadratic which adds to the difficulty of solving the model. Actually, we were not able to solve this model even for instance including 10 products and 8 delivery dates.

A necessary and sufficient condition for the feasibility of this model is:

$$\sum_{i=1}^{N} \left(P_i \left\lceil \frac{\sum_{j=1}^{l} d_{il}}{Q_i} \right\rceil \right) \leq t_l,\ l = 1,\ldots,L \quad (11)$$

3 A decomposition solution approach

A first approach to solve the above introduced problem consists of decomposing the problem into two sub-problems to be solved consecutively. The first sub-problem is the one of determining the product lots to be processed before each delivery date without determining their sequence of production (i.e., without specifying their position p). The second sub-problem is to determine the sequence for the obtained production lots.

The first sub-problem can be formulated as mixed integer linear program (see Boctor, 2016). The optimal solution of this model gives us the number of lots of each product to be finished by each date t_l. Once this optimal solution is obtained we solve a sequencing problem to determine the starting dates of the required production lots. This second sub-problem is also modeled by a mixed integer program. Unfortunately, solving this second model is very time consuming and has a weak LP relaxation. For more details see Boctor (2016).

4 A hybrid heuristic

This proposed heuristic is a hybrid one composed of a solution construction procedure followed by 3 improvement procedures. The construction phase is an iterative, backward-pass heuristic. To construct a production plan, the heuristic starts by setting $t = t_L$, the latest delivery date. The main iteration of the heuristic is as follows. At each date t we list the products to deliver at this date and chose from this list the product k that has the largest value of h_{kq_k} where $q_k = \min\{d_{kt}, Q_k\}$. Then we schedule the processing of a lot of size q_k of the product k to finish at t. If $q_k = d_{kt}$ we remove product k from our list otherwise we reduce its demand by q_k. Next, we set $t = t - P_k$. In other words we put t equal the starting time of the just scheduled job k, and add to the list all the orders for which the due date is between t and $t + P_k$ if any. If the resulting list contains more than one order for a given product, we group them into one order. Now, if the list is not empty, repeat the above and choose a job to schedule. Otherwise, move backward to the latest date where we have some orders to deliver. The heuristic stops when there are no more orders to schedule.

The rational of this construction heuristic is to schedule the production of the orders to deliver at the latest possible time in order to minimize the sum of inventory holding costs.

This procedure may produce a non-feasible schedule where the starting time of some jobs is before the beginning of the planning horizon (date 0). Even in such a case we apply the improvement procedures as they may modify this solution in a way that makes it feasible.

The first improvement procedure attempts to reduce the number of production lots in order to reduce the number of setups. The procedure considers one product at a time and repeat the following until no more gain can be achieved. For each lot of the considered product, determine if a gain can be made by removing this lot and adding its units to the preceding lots of the same product. The lot leading to the highest gain is removed and we repeat the same for the remaining lots. This improvement procedure stops if we cannot achieve any more gain.

The second improvement procedure attempts to move the remaining production lots to the latest possible date without causing any backlogs. This may be possible as the previous improvement procedure may remove some production lots making room for processing other lots in the time interval originally occupied by some of the removed lots.

The third improvement procedure exchanges the position of pairs of production lots as long as this can lead to cost reduction without backlogs.

5 Computational experiment

To assess its performance, the solutions of the hybrid heuristic were compared to those obtained by the decomposition approach using 100 randomly generated problem instances. Unfortunately, it is not possible to obtain the optimal solutions of these instances. In addition the literature does not provide any solution method that can be used to assess the performance of the proposed heuristics.

For all the generated instances, the number of products is 10 and the number of delivery dates is 8. Delivery dates are: 40, 60, 80, 100, 120, 150, 180 and 200. For each instance, the values of h_i, c_i, P_i and Q_i are randomly drawn from uniform distributions. The limits for these uniform distributions are respectively from 0.05 to 0.25 for h_i, from 80 to 200 for c_i, from 1 to 5 for P_i, and from 40 to 80 for Q_i. To determine the total demand of a product, a random value is drawn from a uniform distribution between 120 and 180. For each product, one to four delivery dates are randomly drawn among the 8 possible dates. The total demand is then partitioned and a quantity is randomly determined for each delivery date. Note that all test instances are generated in a way to satisfy the necessary and sufficient feasibility condition (11).

The proposed hybrid heuristic succeeded to solve all test instances with an average time of less than 1 second. The 3 improvement procedures reduce the total cost obtained at the construction step by 8.01% in average with a minimum improvement of 2.12% and a maximum improvement of 16.76%. In more details, the first improvement procedure, the grouping procedure, produced an improvement of 4.42% in average while the second and third improvement procedures yielded 1.05% and 2.73% improvement in average. The hybrid heuristic produced a better solution than the decomposition approach for 75 instances while the decomposition approach produced a better solution for the other 25 instances. Over the entire 100 instances set, in average, the hybrid heuristic produced solutions with a total cost of 3.38% less than the decomposition approach.

6 Acknowledgements

This research work was partially supported by grant OPG0036509 from the National Science and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

References

Boctor FF., 2016, "The Generalized single-facility capacitated lot-sizing and scheduling problem". Document de travail 2016-011; Faculté des sciences de l'administration, Université Laval.

Gopalakrishnan M., 2000, "A modified framework for modeling set-up carryover in the capacitated lot sizing problem". *International Journal of Production Research*, 38: 3421–3424.

Jans R. and Degraeve Z., 2008, "Modeling industrial lot sizing problems: a review". *International Journal of Production Research*, 46: 1619–1643.

Porkka P., Vepsalainen A.P.J., and Kuula M., 2003, "Multi-period production planning carrying over set-up time". *International Journal of Production Research*, 41: 1133–1148.

Sox C.R. and Gao Y., 1999, "The capacitated lot sizing problem with setup carry-over". *IIE Transactions*, 31: 173–181.

Suerie C. and Stadtler H., 2003, "The capacitated lot-sizing problem with linked lot sizes". *Management Science*, 49: 1039–1054.

Single machine scheduling with m:n relations between jobs and orders: Minimizing the sum of completion times and its application in warehousing

Nils Boysen[1], Konrad Stephan[1] and Felix Weidinger[1]

Friedrich-Schiller-University Jena, Germany
{nils.boysen,konrad.stephan,felix.weidinger}@uni-jena.de

Keywords: Single machine scheduling, Warehousing, Order consolidation.

1 Single machine scheduling with m:n job-order relations

In this paper we treat an elementary extension of single machine scheduling problem $1||\sum C_j$ where m:n relations among jobs and orders exist. This problem, which we dub $1|m{:}n|\sum C_o$, is defined as follows:

We have a set of jobs $J = \{1, \ldots, n\}$ with processing times p_j and a single machine to successively process these jobs. Furthermore, we have a set $O = \{1, \ldots, m\}$ of customer orders. Completing an order $o \in O$ requires the completion of job subset $J_o \subseteq J$. Job sets J_o are not disjunct, so that we have a m:n relation among jobs and orders. A solution can be encoded by a sequence ϕ, i.e., a permutation of jobs $j = 1, \ldots, n$, with $\phi(k)$ returning the job at sequence position $k = 1, \ldots, n$. Let $\kappa(\phi, o) = \max\{k = 1, \ldots, n : \phi(k) \in J_o\}$ be the sequence position of the last job required for completing order o. Among all sequences ϕ, problem $1|m{:}n|\sum C_o$ seeks a job sequence which minimizes the sum of order completion times, i.e.,

$$Z(\phi) = \sum_{o \in O} \sum_{k=1}^{\kappa(\phi,o)} p_{\phi(k)}.$$

Other than in our problem setting, traditional problem $1||\sum C_j$ presupposes a 1:1 relation among jobs and orders and is well known to be solvable in polynomial time by ordering jobs according to the shortest-processing-times rule (Smith 1956). In this paper, we show that this result no longer holds for $1|m{:}n|\sum C_o$, which is shown to be strongly NP-hard in Section 2. Section 3 elaborates on the application of $1|m{:}n|\sum C_o$ in distribution centers of large online retailers, e.g., Amazon and Zalando. Here, bins (jobs) containing multiple items for different orders need to be manually sorted into a rack (dubbed the put wall) by a human logistics worker (machine), such that human packers on the other side of the put wall receive customer orders quickly and do not run idle while stowing orders into cardboard boxes.

2 Computational complexity

In this section, we investigate the complexity status of $1|m{:}n|\sum C_o$ and reiterate the complexity proof initially presented by Boysen *et. al.* (2018). The transformation is from the linear arrangement problem (LAP), which is well-known to be NP-complete in the strong sense, see Garey and Johnson (1979).

LAP: Given a graph $G = (V, E)$ and a positive integer K. Is there a one-to-one-function $\vartheta : V \rightarrow \{1, 2, \ldots, |V|\}$, i.e., a numbering of nodes V with integer values from 1 to $|V|$,

such that $\sum_{(u,v)\in E} |\vartheta(u) - \vartheta(v)| \leq K$?

Theorem: $1|m{:}n|\sum C_o$ *is strongly NP-hard even if all jobs have unit processing time.*

Proof: Within our transformation of LAP to $1|m{:}n|\sum C_o$ we introduce a job for each node, so that $n = |V|$. The integer value $\vartheta(u)$ assigned to a node u within LAP corresponds to the sequence position $\phi^{-1}(i)$ assigned to job i within $1|m{:}n|\sum C_o$. Given the maximum degree $\delta(G) = \max_{u\in V}\{|\{v \in V : (u,v) \in E\}|\}$ of the LAP graph, we introduce $\delta(G)$ orders for each node $u \in V$: First, an order $\{u,v\}$ is generated for each adjacent node v, so that for each edge $(u,v) \in E$ two orders $\{u,v\}$ and $\{v,u\}$ are generated. Then, for each node having a degree less than $\delta(G)$ we extend the order set by additional single-job-orders $\{u\}$ until $\delta(G)$ orders per node are generated. In total, $\delta(G) \cdot |V|$ single- and two-job-orders are introduced. The question we ask is whether we can find a solution for $1|m{:}n|\sum C_o$ with objective value

$$Z = \delta(G) \cdot \frac{|V| \cdot (|V|+1)}{2} + K.$$

Obviously, this transformation can be done in polynomial time. The $\delta(G)$ orders associated with each job u are either single-job-orders, which have completion time $\phi^{-1}(u)$, or two-job-orders. Each order $\{u,v\}$ of the latter kind always exists twice, because in the name of each edge (u,v) two identical orders are introduced, i.e., one when generating the $\delta(G)$ orders for node u and the other when generating orders for v. The unit processing times allow us to measure completion time in sequence positions of the job sequence. The sum of completion times for both of these orders is, thus, twice the sequence position of the later of both jobs u and v. If $\phi^{-1}(u) < \phi^{-1}(v)$, this amounts to $2\phi^{-1}(v)$. Due to the inequality of $\phi^{-1}(u)$ and $\phi^{-1}(v)$, we can rearrange $2\phi^{-1}(v)$ to $\phi^{-1}(v)+\phi^{-1}(u)+(\phi^{-1}(v)-\phi^{-1}(u))$. If we assign the former two time spans $\phi^{-1}(v)$ and $\phi^{-1}(u)$ to jobs v and u, respectively, then it becomes obvious that to each sequence position $i = 1,\ldots,n$ exactly $\delta(G)$ time spans are assigned. Thus, we have an inevitable amount of completion time, i.e., independent of the sequence positions of jobs, of

$$\delta(G) \cdot \sum_{u\in V} \phi^{-1}(u) = \delta(G) \cdot \sum_{i=1}^{|V|} i = \delta(G) \cdot \frac{|V| \cdot (|V|+1)}{2}.$$

The remaining time spans $\phi^{-1}(v) - \phi^{-1}(u)$ within $1|m{:}n|\sum C_o$, which are dependent of the sequence positions of jobs, exactly equal the difference in the node numbers assigned to each edge within LAP, so that both problems are directly transferable from each other and the theorem holds. □

3 Application in the warehouses of online retailers

Online retailers like Amazon Europe and Zalando, typically, structure their order fulfilment process into three basic steps:

– Picking: First, the items demanded by customer orders need to be retrieved from the shelves of a warehouse. Most online retailers apply a picker-to-parts order picking in a batching and zoning environment where, additionally, a mixed-shelves policy (also denoted as scattered storage (Weidinger and Boysen 2018)) is applied. Under this policy unit loads of stock keeping units (SKUs) are purposefully broken down and single items are scattered all over the shelves of a warehouse. In this way, there is always some item of a demanded SKU close by irrespective of the current picker location. In such a

setting, large online retailers apply dozens of pickers, which are typically assigned to specific zones of the warehouse. They pick batched orders in parallel into bins each finally containing partial orders for multiple customers.
- Intermediate storage: Completed bins are handed over to the central conveyor system where they are stored until all bins belonging to the same batch have arrived from their zones. Once a batch is complete, the respective bins are released from storage and conveyed toward the consolidation area.
- Order consolidation: The bins of a batch successively arrive at a conveyor segment of the consolidation area. Here, a logistics worker we call the *putter* resides. The putter successively removes the items from the current bin and puts them into the put wall. The put wall is a simple reach-trough rack separated into multiple shelves, which are accessible from both sides. Each shelf is temporarily assigned to a separate order and once the putter scans the current item a put-to-light mechanism indicates into which shelf the current item is to be put. In this way, bin after bin is sorted into the wall. On the other side of the wall reside the *packers*. Here, another put-to-light mechanism indicates completed orders, so that a packer can empty an indicated shelf and pack the respective items into a cardboard box. Packed orders are, finally, handed over to another conveyor system bringing them towards the shipping area.

Our problem $1|m{:}n|\sum C_o$ can directly be applied to determine the sequence of bins, in which a batch is released from intermediate storage. Jobs equal bins and the processing sequence of jobs on the single machine corresponds to the release sequence of the batch from intermediate storage, which is also the sequence in which bins are sorted into the put wall. The processing times p_j depend on the number of items contained in each bin j. By minimizing the sum of completion times orders are quickly sorted into the put wall by the putter, so that the packers on the other side of the wall receive orders sooner and idle times are avoided. With the help of a comprehensive simulation study Boysen *et. al.* (2018) show that optimized bin sequences considerably reduce the packers' idle times.

Future research should consider our single machine scheduling problem with $m{:}n$ job-order relations for other objectives. There may be other cases where the traditional scheduling problem, i.e., with a 1:1 relation among jobs and orders, is solvable in polynomial time, whereas the corresponding problem with $m{:}n$ job-order relations turns out NP-hard.

Acknowledgements

This research has been supported by the German Science Foundation (DFG) through the grant "Planning and operating sortation conveyor systems" (BO 3148/5-1).

References

Boysen, N., K. Stephan and F. Weidinger, 2018, "Manual order consolidation with put walls: The batched order bin sequencing problem", *EURO Journal on Transportation and Logistics* (to appear).

Garey, M.R., D.S. Johnsom, 1956, "Computers and intractability: A guide to the theory of NP-completeness", Freeman, New York.

Smith W.E., 1956, "Various optimizers for single-stage production", *Naval Research Logistics Quarterly*, Vol. 03, pp. 59-66.

Weidinger, F., N. Boysen, 2018, "Scattered storage: How to distribute stock keeping units all around a mixed-shelves warehouse", *Transportation Science* (to appear).

A MILP formulation for multi-robot pick-and-place scheduling

Briand C.[1], Ndiaye J.C.[1] and Parlouar R.[2]

[1] LAAS-CNRS, Université de Toulouse, UPS, Toulouse, France
email: briand@laas.fr, ndiaye@laas.fr
[2] NOVALYNX, 35 Bis Route de Bessières, 31240 L'Union, France
email: remi.parlouar@novalynx.fr

Keywords: Assignment and Scheduling, Mixed Integer Linear Programming, Multi-robot pick-and-place.

1 Introduction

The 21st century is marked by the fourth industrial revolution, which embraces many technologies and concepts. Among them, robotization is often viewed as the most promising avenues of progress in the field of automated production. Indeed the use of more and more sophisticated machines and robots is able to bring improvements in production costs, rates, quality and operators safety.

Among the various types of robots frequently used in production systems are the handling robots, which basically pick parts somewhere in the shop and place them elsewhere. More specifically, this study focuses on automated packaging systems involving several handling robots. A packaging system is generally composed of two conveyor belts conveying products and boxes, respectively. A handling robot picks one or several products on the former conveyor and places them in a box on the latter. The conveyor belts may have several possible shapes: parallel, perpendicular or circular. The parallel one is the most common and product and box flows can go in the same or opposite direction, as illustrated in Figure 1 (taken from (Blanco Rendon 2013)). A pick/place task can only be carried out by one robot when the corresponding product/box is present inside the working area of this robot. The normal speed of each conveyor being assumed known, a time window can be associated with each robot task. Moreover, the pick/place task duration can vary and depends either on the product or the robot.

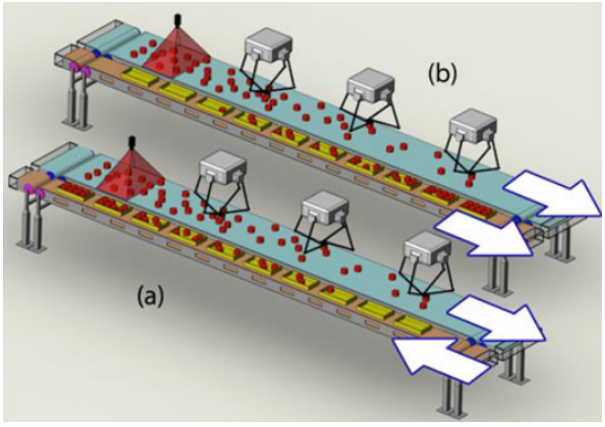


Fig. 1. Conveyor belts configurations

The problem considered in this paper, further referred to as the Multi-robot-Pick-and-Place Scheduling Problem (MPPSP), consists in i) assigning products and boxes to robots and ii) defining a *consistent* starting time for each pick/place task, so that the filling rate is maximized (or equivalently, the number of filled boxes over a given time horizon is maximized). In the case of a pick-task, the starting-time consistency only requires that the task is performed by the robot during its time execution window (i.e., when the product is present inside the robot workspace). In the case of a place-task, the previous condition should obviously hold *and*, additionally, there are flow constraints: if k products should be placed inside each box in a one-shot operation, one has to ensure that k pick tasks have been achieved before the place task can be carried out. Finally, note that in the case the conveyor speed can be controlled (which is assumed impossible in the present study), the filling rate can be further improved, which gives rise to a third MPPSP dimension consisting in the determination of the optimal conveyor speed profiles.

In many existing systems, a vision system is integrated in front of the conveyor entries to locate the various parts, which allows predicting the working-area entry or exit events a few seconds before their occurrence. Moreover, in the context of the industry 4.0, all information about production and packaging processes may be known in advance so that execution windows of pick/place tasks could be either predicted earlier. Under the assumption of predictability of the product/box flows, the MPPSP is studied in its offline version in this paper and a compact Mixed-Integer Linear Programming (MILP) formulation is proposed.

The paper is structured as follows. First, a brief literature overview is made that particularly put into evidence some relationships between MPPSP and some other well-known problems of the scheduling literature. Then, our MILP formulation is established that takes benefits from specific dominance rules, which allows characterizing all the dominant solutions on a robot within a single *master-sequence*. Some conclusions are drawn in the last section.

2 Literature overview

A vast majority of the paper of the literature tackles the online version of the problem, taking interest in designing efficient rules or cooperation mechanisms between robots that maximize the filling rate, while balancing the working load between robots, e.g., (Blanco Rendon 2013, Bouchrit 2016, Huang *et al.* 2015, Humbert *et al.* 2015). In the OR literature, Daoud *et al.* (2014) took interest in designing pick-and-place robotic systems and propose fast metaheuristics to determine the best schedule rule to be applied to each robot.

For the offline version of MPPSP, the literature is scarcer. In (A. Bouchrit 2016), a network-based MILP formulation is proposed to solve the offline MPPSP in the case of a homogeneous product/box flow (each product/box is separated from the next one on the conveyor by a constant distance). Products are considered as nodes within a network and the problem amounts to find for each robot the best path to collect the maximum possible number of products, which gives a pick-and-place task sequence. Many constraints are taken into account such as conveyor belt velocities, robot load balance, time windows and flow constraints. Nevertheless, the implementation of this formulation on commercial solver does not provide satisfying performances as finding optimal solution turns out to be too time-demanding.

In the scheduling literature, MPPSP is sharing some similarities with the parallel machine problem with time windows that aims at minimizing the number of tardy jobs, (denoted as $P|r_j|\sum U_j$ in (Pinedo 2008)). This problem is known to be NP-Hard in the strong sense even for one single machine. Nevertheless, still under the assumption of a single machine environment, it is polynomially solvable when execution windows have a

staircase structure. As a specific feature of MPPSP, we observe that there are several possible time windows for the execution of a task (depending on the robot implementing it), which tends to indicate that MPPSP is also related to the Runway Scheduling Problem (RSP) (Artiouchine *et. al.* 2008) that consists in sequencing aircraft landing. Note that RSP is also NP-hard.

3 MILP formulation

This chapter takes an interest in finding a job sequence that maximizes the number of filled boxes assuming the product and box flows predictable. We consider three sets $\mathcal{B}$, $\mathcal{P}$ and $\mathcal{R}$ of B boxes, P products and R robots, respectively. In the notations used below, index b (p and r, respectively) refers to a box $b \in \mathcal{B}$ (a product $p \in \mathcal{P}$ and a robot $r \in \mathcal{R}$, respectively). The pick and place processing times are denoted D_{pr} and D_{br}, which depend on robot r. We refer to $[S_{pr}, F_{pr}]$ and $[S_{br}, F_{br}]$ as the execution windows of product p (box b, respectively) on robot r.

In the remainder of this paper, as the conveyor speed is assumed constant, we set $F_{pr} - S_{pr} = \Delta_{pr}$, $\forall (p,r) \in \mathcal{P} \times \mathcal{R}$ ($F_{br} - S_{br} = \Delta_{br}$, $\forall (b,r) \in \mathcal{B} \times \mathcal{R}$, resp.). Moreover, without loss of generality, we assume that $\Delta_{pr} > \Delta_{br}$ (products stay longer in the robot working area than boxes) but, as explained below, it could be the reverse.

Once an assignment of products and boxes to robots is decided (note that a product/box can possibly not be assigned), the problem left is to find a pick-and-place sequence on each robot that i) is time feasible and ii) respects the constraint that k picks should always precede any *place* operation. For ensuring time feasibility, following the idea proposed by Briand and Ourari (2013), a *master sequence* can be considered that characterizes a set of dominant sequences. This master sequence uses the notion of a *top-job*, i.e. a job such that its execution window does not (strictly) include the execution window of any other job. In our case, as there are only two kinds of time intervals (the pick and place ones) and because $\Delta_{pr} > \Delta_{br}$, any place operation is a top job. Therefore, a master sequence Θ_r having the form below can be defined for each robot r.

$$\Theta_r = \sigma^-_{1r} \quad 1 \quad \sigma^+_{1r} \quad \sigma_{12} \quad \sigma^-_{2r} \quad 2 \quad \sigma^+_{2r} \cdots i-1 \quad \underbrace{\sigma^+_{i-1r} \;\; \sigma_{i-1,i} \;\; \sigma^+_{ir}}_{\theta_{i-1r}} \quad i \quad \sigma^+_{ir} + \cdots$$

Each place task i has two sets σ^-_{ir} and σ^+_{ir} of pick tasks at its left and its right, respectively. More specifically, σ^+_{i-1r} represents products which intervals overlap place interval $i-1$ but not place interval i. Similarly, σ^-_{ir} gathers pick tasks such that their intervals overlap box interval i but not box interval $i-1$. Eventually, $\sigma_{i-1,i}$ gathers product intervals which overlap both box intervals $i-1$ and i. We refer to θ_{i-1r} as the subset of pick tasks located between place task $i-1$ and i, with θ_{0r} (θ_{Br}) the subset located at the left (the right) of box 1 (of box B, resp.). Note that the same pick task can belong to several sets θ and one has to decide whether the task is performed and, if it is performed, in which set θ. One advantage of a master sequence lies in the fact that, once the previous decisions made, the time feasibility of the resulting pick-and-place sequence can easily be assessed.

The following formulation takes benefit of the master sequence notion and introduces the following binary variables. A box b is filled by robot r if binary variable $y_{br} = 1$ (0 otherwise). A product p is picked in subset θ_{br} if $x_{bpr} = 1$.

$$max\ z = \sum_b \sum_r y_{br}$$

$$\sum_{b} \sum_{p \in \theta_{br}} x_{bpr} \leq 1 \quad , \quad \forall p \quad \forall r \tag{1}$$

$$\sum_{r} y_{br} \leq 1 \quad , \quad \forall b \tag{2}$$

$$k y_{br} \leq -k \sum_{i<b} y_{ir} + \sum_{i<b} \sum_{p \in \theta_{ir}} x_{pir} \leq k \quad , \quad \forall b \quad \forall r \tag{3}$$

$$\text{The master sequence } \Theta_r \text{ is time feasible} \quad , \quad \forall r \tag{4}$$

$$\begin{aligned} x_{bpr} &\in \{0,1\} \quad , \quad \forall b \quad \forall p \quad \forall r \\ y_{br} &\in \{0,1\} \quad , \quad \forall p \quad \forall r \end{aligned}$$

The formulation aims at maximizing the number of filled boxes. Constraints (1-2) enforce any product/box to be picked/filled once at the most. Constraints (3) aim at satisfying the (flow) constraint, i.e. k product at the most should be picked before any place operation. As in (C. Briand and S. Ourari 2013), high level constraints (4) can be implemented using a set of big-M linear constraints (not stated here for matter of conciseness) that use integer variables s_{br} and f_{br}. Theses variables refer to as the earliest starting time and the latest finishing time, respectively, of place task b on robot r (this value linearly depending on the values of other binary variables), provided that $s_{br} + D_{br} \leq f_{br}$.

4 Conclusion

This paper sketches a formulation for solving the offline MPPSP. This formulation has been tested and validated using some academic instances. A more systematic experimental study is currently in progress to assess the efficiency of our approach. The special case where the processing times of the pick/place tasks are identical (i.e., $D_{pr} = Dpick_r$ and $D_{br} = Dplace_r$) will also be considered.

References

Artiouchine K., P. Baptiste, C. Durr, 2008,"Runway Sequencing with Holding Patterns", *European Journal of Operational Research*, Vol. 189(3), pp.1254-1266.

Blanco Rendon D.P., 2013, "Modelling and Simulation of a Scheduling Algorithm for a Pick-and-Place Packaging System", MastersThesis, Polytechnic of Milan.

Bouchrit A., 2016, "Optimal Scheduling for Robotized Pick and Place Packaging Systems", MastersThesis, Polytechnic of Milan.

Briand C. and S. Ourari, , 2013, "Minimizing the number of tardy jobs for the single machine scheduling problem: MIP-based lower and upper bounds", *RAIRO - Operations Research*, Vol. 47, pp. 33-46.

Daoud S., H. Chehade, F. Yalaoui and L. Amodeo, 2014, "Efficient metaheuristics for pick and place robotic systems optimization", *Journal of Intelligent Manufacturing*, Volume 25, pp. 27-41.

Huang Y., R. Chiba, T. Arai, T. Ueyama, J. Ota, 2015, "Robust multi-robot coordination in pick-and-place tasks based on part-dispatching rules", *Robotics and Autonomous Systems*, Volume 64, 2015, pp. 70-83.

Humbert G., M.T. Pham, X. Brun, M. Guillemot and D. Noterman, 2015, "Comparative analysis of pick & place strategies for a multi-robot application", *Proc. IEEE 20th Conference on Emerging Technologies and Factory Automation (ETFA)*, Luxembourg.

Pinedo, M.L., 2008, "Scheduling: Theory, Algorithm and Systems.", 3rd Edition, Springer-Verlag, New York.

Minimizing resource management costs in a portfolio with resource transfer possibilities

Jerome Bridelance[1], Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
jerome.bridelance@ugent.be, mario.vanhoucke@ugent.be
[2] Technology and Operations Management Area, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, UK

Keywords: Resource availability, Resource transfers, Portfolio management.

1 Introduction

The research presented in this abstract is located in the multi-project environment. Two approaches can be followed when working with a portfolio of projects, each with their own methodologies. First of all the different projects can be combined into one large super-project. This is done by adding additional precedence arcs and one dummy-start and end-activtiy. In that way the problem is again reduced to a resource constrained project scheduling problem (RCPSP), consequently this is called the single-project approach. There is also a second way to deal with those multiple projects, namely the multi-project approach. Within this method every project remains an entity by itself, with its own critical path (Kurtulus, I. and Davis, E.W. 1982). This second approach is preferred above the first one, for multiple reasons. To begin, the first approach is nothing more than solving a single project and takes distance from the multi-project environment. Secondly, up to now less research has been done on this topic, which creates more opportunities for improvement (Herroelen, W. 2005). Finally the second approach is a more realistic view of how multiple projects are dealt with in practice (Browning, T.R. and Yassine, A.A. 2010). Next to those two different ways of dealing with the schedule part of a portfolio of projects, there are also different approaches of how the management of resources can be organized. First there is the easiest method where the resources are all collected in one large resource pool. Those can then be freely shared among the activities in the portfolio. This method is called the resource sharing policy. Secondly, on the opposite end of the spectrum, there is the resource dedication policy (Besikci, U. *et. al.* 2013). With this approach resources are dedicated to a particular project at the beginning of the planning horizon. This method does not allow to share resources between projects and consequently not between activities of different projects. The policy is applied when sharing resources between projects is not feasible for example if those projects are geographically too far distributed (Besikci, U. *et. al.* 2013). As already mentioned those two ways are both ends of the resource management spectrum. In between these two, multiple combinations are possible and are probably more realistic approaches. An example of such an in between methodology is the dedication of resources to projects but also allowing the transfer of these resources to other projects. According to research on this topic, resource transfers should already be included in the scheduling part (Kruger, D. and Scholl, A. 2009).

2 Problem description

This research deals with multi-project management, more precisely the scheduling and resource management part. We have chosen to work with the multi-project approach. Consequently every project is a separate entity and of course this decision also has an

influence on the used methodology and its accompanied assumptions. The objective of this research is minimizing the resource costs, including availability and transfer costs. We are working with a static number of projects which have to be scheduled and resources have to be assigned to them in order to be executed. Because not all projects have the same due date, it is not necessary to start all of them as early as possible. All resources should be dedicated to a particular project and stay unified with it until the project is completely finished, after that resources can be transferred to other projects. The general renewable resource availabilities are positioned as low as possible. To accomplish this, projects are shifted further in time and resources are transferred between them. All this is done while taking the precedence relations between projects and the projects´ due dates into account. As a consequence the following assumptions have to be incorporated into the methodology:

- Resources can be transferred between projects, but only when the first project is finished and the second project should still be started.
- Transfer time is depending on the two projects between which the resources are transferred and on the amount transferred.
- Projects can not be interrupted in time.
- The due date of every project should be met.
- Precedence relations between projects have to be satisfied.
- Project activities have fixed durations.

3 Extensions on existing literature

This research idea originates from existing literature and is created as an extension on the combination of those research topics. Liberatore, M.J. and Pollack-Johnson, B. (2003) came up with a new methodology to minimize the resource availability costs in a single project setting, more precisely solving the resource availability cost problem (RACP). By doing this the project´s due date and the activities´ resource requirements have to be satisfied. This methodology obtains the minimum resource availabilities for the different resource types by deriving them from the solution of resource-constrained project scheduling decision problems (RCPSDP). These RCPSDPs are solved with only one or two resource types, all the others are supposed to have an unlimited availability. Resource dedication is also an important part of this research topic. The first ones to introduce resource dedication in a multi-mode and multi-project environment were Besikci, U. *et. al.* (2013). They presented two solution approaches to solve this problem, which can be divided into two sub-problems. First there is the dedication of resource capacities to a particular project, secondly the activities of the projects itself are scheduled. The first methodology works with a genetic algorithm in combination with a new local improvement heuristic, namely combinatorial auction. The second methodology employs a langrangian based heuristic and a subgradient optimization method to find a solution for the resource dedication problem.

The research presented in this abstract combines, adapts and extends the above introduced research studies. Like in Demeulemeester, E. (1995), also in this paper one of the goals is to minimize the renewable resource availability costs. With the difference that we now have a portfolio of projects at our disposal between which resources can be transferred. That is the reason why numerous solutions for the general resource availabilities are possible. The solution of this problem is not the summation of the optimal RACP solutions of every project separately. The general resource levels will be lower because transferring resources is allowed now. Of course these transfers bring along costs as well and shifting projects further in time can not be done endlessly because of every project´s due date. Previous research has already investigated the implementation of resource transfers in project

scheduling problems. Like in (Lacomme, P. *et. al.* 2017) where the resource transfers are introduced in the scheduling problem via routing operations, with the ultimate goal of minimizing the overall makespan. Another more practical example is the study of (Froger, A. *et. al.* 2017). Here resource transfers are implicitly incorporated in the methodology by only allowing that employees shift work locations on the same day, if these locations are compatible. A Location is seen as compatible if the travel time between them is negligible in comparison to a time unit. With the presented research the added value is the combination of the resource transfers with the undetermined resource availability levels. In contrary to Besikci, U. *et. al.* (2013) renewable resources are not only dedicated to projects at the beginning of the planning horizon. After a project is terminated, the renewable resources can be assigned to a new project after they are transferred. Which makes the problem a trade-off between availability and transferring costs, while still satisfying the projects´ due dates. In figure 1 a comparison is made between on the left side the method presented in this abstract and on the right side the methodology when every project is scheduled as early as possible. The considered transfer times are indicated by the arrows in figure 1. With this latter approach the portfolio´s cost is not optimized as a whole. Underneath figure 1 the cost difference between the methods is presented. Assuming a transfer cost of 10 euro/unit and an availability cost of 20 euro/week. Information about the projects can be found in table 1.

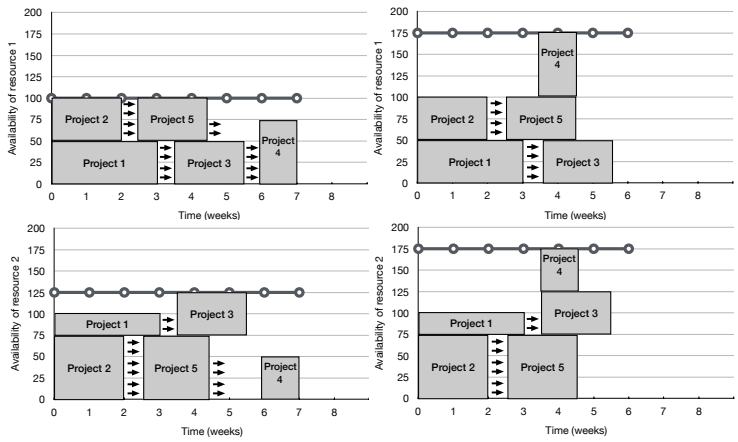


Fig. 1. Example: comparison between methods

Calculations

Left approach

Transfer costs: ((50 + 50 + 25 +50) + (75 + 25 + 50)) * 10 euro/unit = 3250 euro
Availability costs: (100 + 125) units * 7 weeks * 20 euro/week = 31500 euro
Total cost = 34750 euro

Right approach

Transfer costs: ((50 + 50) + (75 + 25)) * 10 euro/unit = 2000 euro
Availability costs: (175 + 175) units * 5,5 weeks * 20 euro/week = 38500 euro
Total cost = 40500 euro

Table 1. Project portfolio information

Project	Duration	Res.1	Res.2	Predecessor	Due date
Project 1	3 weeks	50	25	/	week 3
Project 2	2 weeks	50	75	/	week 3
Project 3	2 weeks	50	50	project 1	week 6
Project 4	1 week	75	50	project 1	week 8
Project 5	2 weeks	50	75	project 2	week 8

4 Methodology

The research papers presented in the previous part were used to come up with this new research problem and gave inspiration of which different methods can be applied to solve the specific problem. As a consequence, first a full factorial design is set up to conduct a complete analysis of multiple priority rules. The test problems used to perform this analysis are generated with different network-, project- and resource-related characteristics, including network complexity, the level of parallelism in the project portfolio and difference in resource type usage by the projects. All this is done to decide in which situation which priority rule should be used. Priority rule heuristics stay important for multiple reasons. In comparison to meta-heuristics the computational complexity is lower, which makes them interesting for larger problems. Next to this, priority rules are often employed to construct initial solutions for meta-heuristics. Next to these priority rules, a meta-heuristic is constructed to test various experiments and provide some managerial insights. The influence of the following situations on the objective function value is investigated:

- The ratio between resource availability costs and resource transfer costs.
- The ratio between the range in the projects´ due dates and the mean project duration.
- The diversity in usage of different resource types by the projects.

References

Besikci, U., Bilge, U. and Ulusoy, G., 2013, "Resource dedication problem in a multi-project environment", *Flexible Services and Manufacturing Journal*, Vol. 25(1-2), pp. 206-229.

Browning, T.R. and Yassine, A.A., 2010, "Resource-constrained multi-project scheduling: Priority rule performance revisited", *International Journal of Production Economics*, Vol. 126(2), pp. 212-228.

Demeulemeester, E., 1995, "Minimizing resource availability costs in time-limited project networks", *Management Science*, Vol. 41(10), pp. 1590-1598.

Froger, A., Gendreau, M., Mendoza, J.E., Pinson, E. and Rousseau, L.M., 2017, "A branch-and-check approach for a wind turbine maintenance scheduling problem", *Computers & Operations Research*, Vol. 88, pp. 117-136.

Herroelen, W., 2005, "Project scheduling-Theory and practice", *Production and operations management*, Vol. 14(4), pp. 413-432.

Kruger, D. and Scholl, A., 2009, "A heuristic solution framework for the resource constrained (multi-) project scheduling problem with sequence-dependent transfer times", *European Journal of Operational Research*, Vol. 197(2), pp. 492-508.

Kurtulus, I. and Davis, E.W., 1982, "Multi-project scheduling: Categorization of heuristic rules performance", *Management Science*, Vol. 28(2), pp. 161-172.

Lacomme, P., Moukrim, A., Quilliot, A. and Vinot, M., 2017, "A new shortest path algorithm to solve the resource-constrained project scheduling problem with routing from a flow solution", *Engineering Applications of Artificial Intelligence*, Vol. 66, pp. 75-86.

Liberatore, M.J. and Pollack-Johnson, B., 2003, "Factors influencing the usage and selection of project management software", *IEEE transactions on Engineering Management*, Vol. 50(2), pp. 164-174.

Vehicle sequencing at transshipment terminals with handover relations

Dirk Briskorn[1], Malte Fliedner[2] and Martin Tschöke[2]

[1] Department of Production und Logistics, University of Wuppertal, Germany
briskorn@uni-wuppertal.de
[2] Department of Operations Management, University of Hamburg, Germany
{malte.fliedner,martin.tschoeke}@uni-hamburg.de

Keywords: Transshipment terminals; vehicle sequencing; handover relations; computational complexity.

1 Introduction

The planning of transshipment operations is a crucial task in today's global supply chains, since the responsiveness of the supply chain as well as its cost structure are often heavily influenced by the operational efficiency at transshipment nodes. Even though specific transshipment processes may differ considerably with respect to specific problem characteristics, for instance due to the involved technologies or modes of transportation, at the core of many more involved transshipment problems lie some fundamental decisions that need to be taken irrespective of the specific application.

Generally speaking, at a transshipment node commodities are exchanged between different transport relations using some set of resources for (un)loading and transportation. Typically, the commodity exchange can be modeled as a strict handover relation, in the sense that the receiving vehicle cannot leave the system before the vehicle that supplies the commodity has arrived. In order to facilitate an easy access to incoming vehicles, transshipment nodes often provide a special set of docking resources where vehicles are processed. This can be rail-tracks in rail-rail or rail-road terminals (see (Boysen et al. 2011)), berths in seaports ((Bierwirth and Meisel 2015)), flight gates in airport hubs ((Dorndorf et al. 2007)) and dock doors at cross docks ((Boysen and Fliedner 2010)). Whenever these resources are scarce, there is a fundamental decision problem to assign vehicles to docking resources over time, such that all handover relations are satisfied. Typically, these assignment problems are solved under some time- or efficiency-oriented objective function while considering several additional constraints with respect to the loading and transshipment resources, storage strategy, due dates, etc.

One important organizational constraint in cross-docking reflects whether commodities can be stored on the dock floor or have to be transported directly to from one truck to the next. While temporary storage up to 24hrs is typically possible in most applications, it might restricted to reduce double handling or ensure that cooling requirements are met, see (Boysen 2010) and (Boysen et al. 2012). Further, in some cross docks the loading process of a truck may be interrupted, to clear the dock door for another more urgent vehicle, e.g. see (Alpan et al. 2011) and (Alpan et al. 2011b). Finally, the handover relations themselves can be subject to structural constraints. For instance, if the cross-dock is run in an exclusive service mode for in- and outbound trucks, e.g. see (Boysen and Fliedner 2010) and (Chmielewski et al. 2009), no inbound truck receives any commodity from outbound trucks. In addition to that, dock doors are partitioned into disjoint sets, such that inbound (outbound) trucks can only be docked to specific inbound (outbound) doors. Such grouping constraints are also encountered in applications where doors are assigned to specific transport relations, e.g. local or long distance transport relations.

In the following we will analyze the structure of finding feasible docking assignments under handover relations while considering the aforementioned problem characteristics. This assignment problem typically has to be solved as an integral part of any solution strategy that solves dock door scheduling problems under other time- or efficiency-oriented objective. In this sense, we study a core decision problem that can be responsible for much of the computational challenge that is encountered in various applications. For this purpose, we will introduce a formal framework for such decision problems in Section 2. Further, in Section 3, we outline the results of a comprehensive complexity analysis of the different problem versions covered by the framework.

2 Formal Problem Definition

We consider a set D of doors partitioned into q groups $D_1, \ldots, D_q$. Doors in the same subset are assumed to be identical. Moreover, we consider a set V of vehicles partitioned into groups $V_1, \ldots, V_q$. Vehicles in V_g, $g = 1, \ldots, q$, can be docked only at doors in D_q.

We distinguish between problem settings where each vehicle can be docked only once ("one"), each vehicle can be docked multiple times at the same door ("interrupt"), and each vehicle can be docked multiple times at different doors ("revisits"). We refer to this parameter as the docking strategy in the following.

Moreover, we have a set $H \subseteq V \times V$ of handover relations (HRs). HR $(v, w) \in H$ represents vehicle v handing over (part of) its delivery to w. We say that v supplies w in the following. With regard to the structure of HRs we adress specific problem settings using two parameters, namely the pair structure and the group structure. First, according to the pair structure we distinguish between settings where $(v, w) \in H$ whenever $(w, v) \in H$ ("sym"), where $(w, v) \notin H$ whenever $(v, w) \in H$ ("asym"), and where we have no restriction on H ("gen"). Second, according to the group structure we distinguish between settings where $(v, w) \in H$ only if v and w are in different groups of vehicles ("inter") and where, additionally, $(v, w) \in H$ with v and w in the same group is possible ("inner"). Note that restricting HRs to pairs of vehicle in the same group would be a natural third option but yields a problem setting which decomposes into group-specific subproblems.

If $(v, w) \in H$, v and w need to be docked such that these goods can be unloaded from v, transported through the terminal to the door where w is docked, and loaded onto w. In order to reduce the problem setting to the very core we ignore durations for unloading or loading and transportation times. We distinguish, however, between storage strategies where goods can be intermediately stored in the terminal ("sto") and where this is not allowed ("noSto").

This gives us a family of 36 different parameter settings. In each of the resulting problem settings we are interested in sequences of docking operations (DS). Such an operation (v, d) is specified by vehicle v and the door d involved. For such an operation to be feasible there has to be a group index $g = 1, \ldots, q$ such that $v \in V_g$ and $d \in D_g$. A DS is feasible with respect to the door allocation if each operation is feasible and it represents the order in which docking operations are carried out.

Let σ be a DS, $l(\sigma)$ its length, and $\sigma(k)$ the kth operation in σ. We say that operation $\sigma(k) = (v, d)$, $k = 1, \ldots, l(\sigma)$, is active in k and, furthermore, in $k' > k$ if for each $k'' = k + 1, \ldots, k'$ we have $\sigma(k'') = (w, d')$ with $w \neq v$ and $d' \neq d$. That is, a docking operation (v, d) is active as long as v is not docked at an other door and no other vehicle is docked at d. Let $e(\sigma, k) = k'$ if $\sigma(k)$ is active in k' and (i) $k' = l(\sigma)$ or (ii) $\sigma(k)$ is not active in $k' + 1$. We say $[k, e(\sigma, k)]$ is the activity interval of $\sigma(k)$.

A DS σ is feasible with regard to the docking strategy only if

- the docking strategy is "one visit" and σ contains exactly one operation for each vehicle,

– the docking strategy is "interrupt" and for any two operations (v, d) and (w, d') in σ we have $v \neq w$ or $d = d'$, or
– the docking strategy is "revisits".

A DS σ is feasible with regard to the storage strategy only if

– the storage strategy is "storage" and for each $(v, w) \in H$ there are operations $\sigma(k) = (v, d)$ and $\sigma(k') = (w, d')$ with $k \leq e(\sigma, k')$ or
– the storage strategy is "no storage" and for each $(v, w) \in H$ there are operations $\sigma(k) = (v, d)$ and $\sigma(k') = (w, d')$ with $[k, e(\sigma, k)]$ and $[k', e(\sigma, k')]$ overlapping.

A DS σ is feasible if it is feasible with regard to the door allocation, the docking strategy, and the storage strategy.

Definition 1. *Given a docking strategy, a storage strategy, $D_1, \ldots, D_q$, $V_1, \ldots, V_q$, and H, the dock operation sequencing problem (DOSP) is to determine whether a feasible DS exists.*

We will refer to DOSP with a specific parameter setting by a quadruplet

(docking strategy|storage strategy|group structure|pair structure).

For example, (interrupt|sto|inner|asym) refers to the problem setting where vehicles may approach the same door multiple times, goods can be stored, two vehicles do not supply each other, and HR within a group are possible.

3 Computational Complexity

We give an overview of results in Table 1. In those cases where an entry in the quadruplet specifying a problem setting is not given the corresponding result holds for any possible entry. Horizontal solid lines separate problem settings differing in the docking strategy.

No.	Problem	Comment	Complexity
1	(one|sto|inner|asym)	generalization of 2	NP-complete for $q = 1$
2	(one|sto|inner|sym)	equivalent to PATH WIDTH for $q = 1$	NP-complete for $q = 1$
3	(one|sto|inner|gen)	generalization of 2	NP-complete for $q = 1$
4	(one|sto|inter|asym)	generalization of 2	NP-complete for $q = 2$
5	(one|sto|inter|sym)	generalization of 2	NP-complete
6	(one|sto|inter|gen)	generalization of 5	NP-complete
7-9	(one|noSto|inner|−)	equivalent to 2	NP-complete for $q = 1$
10-12	(one|noSto|inter|−)	generalization of MIN CUT LINEAR ARRANGEMENT	NP-complete
13-18	(interrupt|sto|−|−)	q doors sufficient,	in P
19-21	(interrupt|noSto|inner|−)	equivalent to VERTEX COLORING	NP-complete for $q = 1$
22-24	(interrupt|noSto|inter|−)	q doors sufficient	in P
25-30	(revisits|sto|−|−)	q doors sufficient,	in P
31-33	(revisits|noSto|inner|−)	$2q$ doors sufficient	in P
34-36	(revisits|noSto|inter|−)	q doors sufficient	in P

Table 1. Computational complexity

References

Alpan, G.; Larbi, R.; Penz, B. (2011): A bounded dynamic programming approach to schedule operations in a cross docking platform. Computers & Industrial Engineering 60, 385-396.

Alpan, G.; Ladier, A.-L.; Larbi, R.; Penz, B. (2011): Heuristic solutions for transshipment problems in a multiple door cross docking warehouse. Computers & Industrial Engineering 61, 402-408.

Bierwirth, C.; Meisel, F. (2015): A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. European Journal of Operational Research 244, 675-689.

Boysen, N.; Fliedner, M. (2010): Cross dock scheduling: Classification, literature review and research agenda. Omega 38, 413-422.

Boysen, N. (2010): Truck scheduling at zero-inventory cross docking terminals. Computers & Operations Research 37, 32-41.

Boysen, N.; Jaehn, F.; Pesch, E. (2011): Scheduling Freight Trains in Rail-Rail Transshipment Yards, Transportation Science 45, 199-211.

Boysen, N.; Briskorn, D.; Tschöke, M. (2012): Truck scheduling in cross-docking terminals with fixed outbound departures. OR Spectrum 32, 135-161.

Chmielewski, A.; Naujoks, B.; Janas, M.; Clausen, U. (2009): Optimizing the Door Assignment in LTL-Terminals. Transportation Science 42, 198-210.

Dorndorf, U.; Drexl, A.; Nikulin, Y.; Pesch, E. (2007): Flight gate scheduling: State-of-the-art and recent developments. Omega 35, 326-334.

Synchronous flow shop scheduling with pliable jobs

Matthias Bultmann[1], Sigrid Knust[1], Stefan Waldherr[2]

[1] University of Osnabrück, Germany
{mbultmann,sknust}@uni-osnabrueck.de
[2] Technical University of Munich, Germany
stefan.waldherr@in.tum.de

Keywords: flow shop, synchronous movement, pliability

1 Introduction

In this work we consider synchronous flow shop scheduling problems where the processing times of the operations are not fixed in advance. Instead, for each job a total processing time is given which can be distributed freely among the machines, respecting some lower and/or upper bounds on the processing times of the operations.

A synchronous flow shop (also called a "flow shop with synchronous movement") (cf. Kouvelis and Karabati (2011)) is a variant of a non-preemptive permutation flow shop where transfers of jobs from one machine to the next take place at the same time after the operations on all machines are finished. If the processing time of an operation on one machine is smaller than the maximum processing time of the operations started on the other machines at the same time, the corresponding machine is idle until the job may be transferred to the next machine. In contrast, in a classical flow shop the transfer of jobs is asynchronous: Jobs may be transferred to the next machine as soon as their processing on the current machine is completed and processing on the next machine immediately starts as soon as this machine is available.

The term "pliability" was first introduced in Weiß et al. (2016). Within this model, the processing times of the individual operations of a job are not fixed in advance but may be determined with some flexibility. They must respect given lower/upper bounds and add up to the given total processing time of each job. For example, this allows to model situations where the processing time of an operation can deviate from a fixed amount by some margin, defined by the lower and upper bounds. Such a model occurs in practice if several machines are able to process an operation and it is possible to distribute the processing time of an operation among these machines. For example, workers at an assembly line might be trained to not only be able to perform a single, dedicated operation, but to also be skilled enough to work on additional ones. Then, instead of waiting for the next job to be transported to them, they may continue working on the current job, which may lead to reduced idle times and hence a better productivity of the assembly line.

2 Problem formulation

We consider a permutation flow shop with m machines $M_1, \ldots, M_m$ and n jobs where job j consists of m operations $O_{1j} \to O_{2j} \to \ldots \to O_{mj}$. Operation O_{ij} has to be processed without preemption on machine M_i for p_{ij} time units. In a feasible schedule each machine processes at most one operation at any time, each job is processed on at most one machine at any time, and the operations of each job are processed in the predefined order.

The processing is organized in synchronized cycles where jobs are moved from one machine to the next by an unpaced synchronous transportation system. This means that in a cycle all current operations start at the same time on the corresponding machines.

Only after all operations have finished processing, all jobs are moved to the next machine simultaneously. The job processed on the last machine M_m leaves the system, a new job (if available) is put on the first machine M_1. As a consequence, the processing time of a cycle t (its so-called "cycle time" c_t) is determined by the maximum processing time of the operations contained in it. Furthermore, only permutation schedules are feasible, i.e., the jobs have to be processed in the same order on all machines.

Let C_j be the completion time of job j, i.e., the time when j has been processed on all machines and leaves the system. The goal is to find a permutation of the jobs such that the makespan $C_{\max} = \max_j C_j$ is minimized. With each permutation a corresponding (left-shifted) schedule is associated in which each operation starts as early as possible.

Huang (2008) introduced the notation "synmv" in the β-field of the well-known $\alpha|\beta|\gamma$ scheduling classification scheme to indicate synchronous movement. Hence, the basic synchronous flow shop problem with the makespan objective is denoted by $F|synmv|C_{\max}$.

In this work, the jobs are "pliable" in such a way that instead of a fixed individual processing time p_{ij} for operation O_{ij} on M_i we are only given a total processing time p_j of job j. Then, in addition to finding a job permutation, we also have to determine actual processing times $x_{ij} \geq 0$ for operations O_{ij} such that

$$\sum_{i=1}^{m} x_{ij} = p_j \qquad (j = 1, \ldots, n). \tag{1}$$

In the unrestricted model, there are no constraints on the actual processing times, i.e., we only have to fulfill

$$0 \leq x_{ij} \leq p_j \qquad (i = 1, \ldots, m; j = 1, \ldots, n). \tag{2}$$

To indicate this situation, we add "*plbl*" in the β-field of the $\alpha|\beta|\gamma$-notation.

In a more realistic, restricted scenario, additionally lower and upper bounds $\underline{p}_{ij}, \overline{p}_{ij}$ are given, and the actual processing times have to satisfy

$$\underline{p}_{ij} \leq x_{ij} \leq \overline{p}_{ij} \qquad (i = 1, \ldots, m; j = 1, \ldots, n). \tag{3}$$

To indicate this situation, we add "$plbl(\underline{p}_{ij}, \overline{p}_{ij})$" in the β-field. We also consider the special case that only lower bounds $\underline{p}_{ij}$ are given, indicated by "$plbl(\underline{p}_{ij})$".

We assume all input data (processing times, lower and upper bounds) to be integer and usually allow that the actual processing times x_{ij} may take arbitrary real values. However, in some applications, the processing times must also be integer. A similar distinction has been made for scheduling problems with preemption where usually continuous preemption is allowed, but in some situations jobs can only be split at integer points in time. For some special cases it was shown that always an optimal preemptive schedule exists where all interruptions and all starting/completion times occur at integer time points (cf. Baptiste et al. (2011)). Dealing with the same question for pliability, in the absence of upper bounds allowing real-valued processing times does not lead to better schedules since we can show that for problem $F|synmv, plbl(\underline{p}_{ij})|C_{\max}$ always an optimal schedule with integer-valued processing times exists. Hence, in this case, when looking for an optimal schedule we may restrict ourselves to schedules with integer processing times. On the other hand, in the more general situation $F|synmv, plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$ with lower and upper bounds, allowing non-integer processing times can lead to better solutions.

Concerning complexity, problem $F2|synmv, plbl|C_{\max}$ without any bounds on the processing times is already $\mathcal{NP}$-hard.

3 Solution approach

Since the problem is $\mathcal{NP}$-hard, we cannot expect a polynomial time exact algorithm. In preliminary tests, mixed integer linear programs could only be solved to optimality for very small instances. To achieve good results, we use a two-stage heuristic. It can be shown that for a fixed job permutation optimal corresponding processing times can be obtained in polynomial time by linear programming. The problem is decomposed by employing a local search procedure using the set of all job permutations as search space. For each permutation corresponding optimal processing times can be calculated with the LP. Unfortunately, for larger problem instances solving this LP is quite time-consuming. Since in the local search approach, usually many permutations should be evaluated, it is more efficient to use a direct combinatorial algorithm with a better run time than an LP solver. Depending on the size of the problem, even if no such direct algorithm is known, it may be more efficient to determine only near-optimal processing times heuristically instead of solving this subproblem to optimality. Then more neighbors can be evaluated in the same amount of time.

Thus, the subproblem of determining actual processing times for a fixed job permutation is of special interest. For problem $F|synmv, plbl(\underline{p}_{ij})|C_{\max}$ without upper bounds, we propose a polynomial-time direct combinatorial algorithm to obtain optimal actual processing times. On the other hand, the situation for problem $F|synmv, plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$ is more involved. While the case where arbitrary real-valued processing times are allowed can be still solved in polynomial time, the problem becomes $\mathcal{NP}$-hard if all actual processing times are required to be integer.

In the first stage of the two-stage approach, we use a tabu search procedure with a simple swap neighborhood. In a tabu list we store pairs of swapped jobs. A move is tabu if it involves two jobs which are currently in the tabu list. In each iteration of the tabu search, we consider the whole neighborhood (i.e., we evaluate all possible swaps of two jobs) and perform the best non-tabu move or the best overall move if it results in a schedule with a new best objective value (aspiration criterion). As inital solution we used a job permutation calculated by the NEH heuristic (Nawaz et al. (1983)).

4 Computational results

To evaluate the two-stage approach, we simulated a scenario in which for each job we are first given a "base" processing time for each operation (which corresponds to the processing of the operation in the model without pliability) and then introduce flexibility in such a way that we are allowed to deviate from these processing times on each machine by some amount as long as the total processing time of the job remains the same. For our test sets, we randomly generated instances of synchronous flow shops with 2 to 5 machines and 10 to 150 jobs. For each operation O_{ij} we chose a base processing time p^B_{ij} uniformly distributed over the interval $[0, 100]$. Using these base processing times, we generated lower and upper bounds by defining two real-valued parameters $\alpha \geq 0$ and $\beta \geq 1$ and setting $\underline{p}_{ij} = \alpha p^B_{ij}$ and $\overline{p}_{ij} = \beta p^B_{ij}$ for all operations O_{ij}. The total processing time p_j of job j was set to the sum of the base processing times of its operations. For each of the combinations of n and m we generated five instances. Additionally, for each combination n, m and these base processing times we generated several instances with different α- and β-values. In the following, we discuss results for three different parameter sets: $(\alpha = 0, \beta = 2)$, i.e., a configuration in which we are allowed to deviate a lot from the base processing times and two more restrictive parameter sets, $(\alpha = 0.5, \beta = 1.5)$ and $(\alpha = 0.8, \beta = 1.2)$. All

computational evaluations were performed on a computer with an Intel Core i3-370M 2.4 GHz processor and 4 GB RAM.

		$\alpha = 0, \beta = 2$			$\alpha = 0.5, \beta = 1.5$			$\alpha = 0.8, \beta = 1.2$		
		Initial	Tabu		Initial	Tabu		Initial	Tabu	
m	n	Gap	Gap	Time	Gap	Gap	Time	Gap	Gap	Time
2	10	3.90	0.85	0	6.17	2.36	0	9.93	6.09	0
2	50	0.90	0.02	2	2.01	0.25	2	4.10	1.59	2
2	100	0.65	0.01	11	1.61	0.16	18	3.37	0.72	14
2	150	0.58	0.04	49	1.41	0.21	45	2.84	0.81	80
3	10	11.42	5.40	0	13.31	7.01	0	12.35	10.03	0
3	50	5.20	0.51	3	8.51	2.56	3	13.23	6.67	4
3	100	4.33	0.19	32	7.80	1.47	26	11.37	4.56	31
3	150	3.92	0.26	154	6.06	0.98	115	11.15	3.75	159
5	10	20.38	12.98	0	19.36	16.99	0	27.91	22.20	0
5	50	12.23	3.84	7	19.54	9.25	7	26.16	16.74	7
5	100	10.84	2.16	81	14.84	6.12	46	25.27	13.27	69
5	150	10.56	2.33	383	15.21	5.49	291	24.61	12.31	238

Table 1. Results for problem $F|synmv, plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$

Table 1 shows the results of the two-stage approach for $F|synmv, plbl(\underline{p}_{ij}, \overline{p}_{ij})|C_{\max}$. The gaps of the initial solutions as well as the results of the tabu search are reported relative to a lower bound obtained by an LP relaxation. The computation times for the tabu search are given in seconds, the time required to obtain an initial solution was below one second for all instances.

Overall, it can be seen that the two-stage approach leads to a large improvement of the initial solutions. Especially, in situations with high flexibility we can reach near-optimal solutions even for larger instances in a reasonable amount of time.

References

Baptiste, P., J. Carlier, A. Kononov, M. Queyranne, S. Sevastyanov, and M. Sviridenko, 2011, "Properties of optimal schedules in preemptive shop scheduling", *Discr. Appl. Math.* 159, pp. 272–280.

Huang, K.-L., 2008, "Flow shop scheduling with synchronous and asynchronous transportation times", PhD thesis, The Pennsylvania State University.

Kouvelis, P. and S. Karabati, 1999, "Cyclic scheduling in synchronous production lines", *IIE Transact.* 31, pp. 709–719.

Nawaz, M. E.E. Enscore, and I. Ham, 1983, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", *Omega* 11, pp. 91–95.

Weiß, C., S. Knust, N.V. Shakhlevich, and S. Waldherr, 2016, "Flow shop and open shop scheduling with job splitting" Proceedings of the 15th International Conference on Project Management and Scheduling, Valencia, Spain, pp. 26–29.

Computation of the project completion time distribution in Markovian PERT networks

Jeroen Burgelman[1], Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
jeroen.burgelman@ugent.be, mario.vanhoucke@ugent.be
[2] Technology and Operations Management Area, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, UK

Keywords: project scheduling, PERT, Linear Algebra.

1 Introduction

Since the introduction of PERT networks (Malcolm *et al.* 1959) uncertainty in activity durations has been increasingly modelled using the PERT methodology (Adlakha and Kulkarni 1989). Recently, uncertainty in activity durations has modelled using increasingly complex probability distributions (Colin and Vanhoucke 2015). Nevertheless, computing the exact project makespan distribution for project networks is infeasible in general (Hagstrom 1988). The special case where activity durations are modelled using independently distributed exponential random variables has received moderate attention in the literature (Kulkarni and Adlakha 1986), (Azaron *et al.* 2006). Moreover, this special case has frequently been used as a basis to study more involved project scheduling problems (Azaron *et al.* 2011, Gutin *et al.* 2015). Therefore the accurate computation of the resulting project makespan distribution is of vital importance.

This abstract proposes an integrated approach to validate the applicability, accuracy and robustness of project completion time distribution computations in Markovian PERT networks. The applicability of methods in the literature hinges on the theoretical assumptions underlying these methods. Given the size of the Markov chain, small rounding errors in the calculation of the project makespan distribution can propagate and result in inaccurate distribution functions. Furthermore, the ability of different methods to cope with changes in the input data is assessed.

Section 2 discusses different approaches to compute the project makespan distribution in Markovian PERT networks. Section 3 discusses the research and preliminary results.

2 Problem description

In this research, a project is represented using an acyclic directed graph $G = (N, A)$ where N is the set of nodes representing the project activities and A is the set of arcs representing the precedence relations of the project network. The objective is to compute the cumulative probability distribution of the project makespan:

$$F(t) = P(S_{n+1} \leq t) \tag{1}$$

Where S_{n+1} is the random variable representing the project makespan and $P(S_{n+1} \leq t)$ denotes the probability of the project completing before or at time t. Under the assumption of exponentially distributed activity durations, the project makespan distribution can be derived by computing the solution to a set of differential equations resulting from an underlying Continuous Time Markov chain (Kulkarni and Adlakha 1986, Azaron *et al.*

2006). A solution of the linear system of differential equations is given by:

$$F(t) = \mathbf{e_1}^T \cdot e^{\mathbf{Q}t} \cdot \mathbf{e_n} \tag{2}$$

Here, $\mathbf{e_1}$ and $\mathbf{e_n}$ are respectively the first and last column of the $n \pm \times n$ identity matrix $\mathbf{I_n}$ and $e^{\mathbf{Q}t}$ is the matrix exponential of the infinitissimal generator of the CTMC defined by (Kulkarni and Adlakha 1986) and (Azaron *et al.* 2006).

While several approaches exist to compute the matrix exponential (Moler and Van Loan 1978), the performance of most approaches is inadequate for our purpose. The three main approaches to compute the matrix exponential are matrix decompositions, approximation methods, scaling and squaring and Krylov methods.

First, matrix decomposition methods can be used to simplify the computation of $e^{\mathbf{Q}t}$ by decomposing $\mathbf{Q}$ into a matrix product form $Z \cdot \mathbf{D} \cdot Z^{-1}$. This approach has been advocated in the project scheduling literature (Azaron *et al.* 2006). Although matrix decomposition methods have a good performance on small to medium sized problems, the size of the state space encountered in project makespan distribution computations can be prohibitive for these methods. Furthermore, the performance of these methods has only been demonstrated on numerical examples in the literature (Azaron *et al.* 2006, Azaron *et al.* 2011).

Second, the scaling and squaring method relies on padé approximants r_m of order m in combination with a scaling parameter s to produce approximations for $e^{\mathbf{Q}t}$.

$$e^{\mathbf{Q}} \approx r_m \left(2^{-s}\mathbf{Q}\right)^{2^s} \tag{3}$$

Therefore the resulting solutions are obtained numerically, in contrast to the exact expressions obtained by the matrix decomposition methods. This approach is more stable than the matrix decomposition methods and has less theoretical limitations (Moler and Van Loan 1978).

Finally, Krylov methods (Moler and Van Loan 2003) do not compute the entire matrix exponential, but approximate the product $e^{\mathbf{Q}t} \cdot \mathbf{e_n}$ without computing $e^{\mathbf{Q}t}$ explicitly. The approximation is achieved by the Arnoldi process, to compute a matrix Q_k with orthonormal columns and the resulting approximation is given by

$$e^{\mathbf{Q}} \cdot \mathbf{e_n} \approx Q_k e^{H_k} Q_k \cdot \mathbf{e_n} \tag{4}$$

Where the matrix exponential of the upper Heisenberg matrix H_k is easier to compute and the large state space dimensions of the CTMC are reduced to dimensions $k \times k$. This method is especially suited to compute project makespan distribution functions for large scale problems but can suffer from loss of accuracy if the computation of the matrix Q_k is unstable.

The scaling and squaring and Krylov methods have never been assessed in a project scheduling setting. Therefore, the performance of algorithms to compute project makespan distribution functions in Markovian PERT networks has never been assessed with regard to applicability, accuracy and robustness.

3 Research and preliminary results

In this paper, the advantages and limitations of existing approaches to compute project makespan distributions are compared. Furthermore, we assess the ability of specialised techniques from linear algebra to overcome the existing limitations. Based on the resulting analysis, we provide theoretical and managerial insights in the performance of the different algorithms and the extent to which existing limitations can be resolved by adapting traditional project data generation schemes used in the project scheduling literature.

The presented approaches are assessed on three key metrics, applicability, accuracy and robustness. The assessment based on applicability comprises three parts. First the theoretical limitation of the methods are assessed for several standard datasets from the project literature (Vanhoucke *et al.* 2016) in terms of the invertibility of Z. Second, since the goal of the research is to compute the cumulative distribution of the project makespan, computational results that do not adhere to the properties of cumulative distribution functions, i.e. $\inf_t F(t) = 0$, $\sup_t F(t) = 1$ and monotonicity, essentially make the corresponding method inapplicable for our purposes. Finally, the computation of the matrix exponential requires matrix multiplication operations on matrices of vast dimensions, thus potentially causing memory problems and an incomplete computation of the project makespan distribution. The accuracy of an approach is measured in the number of significant digits lost during the computation of the makespan distribution function. High loss of accuracy can make the computation of probabilities very inaccurate. The robustness of the approaches is gauged by perturbing the input data of the infinitissimal generator matrix of the CTMC with a small factor $0.01 \leq \epsilon \leq 0.02$ and measuring the errors in the computation of the project makespan distribution function by the euclidean norm at the decile values of the computed distribution function.

Preliminary experiments show that the matrix decomposition methods advocated in the project scheduling literature exhibit very limited performance in terms of applicability, accuracy and robustness, regardless of the project data set on which they were assessed. Moreover, the scaling and squaring algorithm is more robust to small alternations in the input data, whereas Krylov methods fail to find stable solutions for project networks with more than 10 activities. To mitigate the limitations inherent in matrix decomposition methods, a new dataset is constructed building on the fundamental assumption underlying the method of (Azaron *et al.* 2006), namely the existence of a set of $|S|$ independent eigenvectors, where $|S|$ is the size of the state space of the CTMC. The performance of all methods on the new dataset was tested and the decomposition method proposed in the project scheduling literature has comparable performance to the scaling and squaring algorithm, albeit at a higher computational cost.

References

Adlakha V., V. Kulkarni, 1989, "Classified bibliography of research on stochastic PERT networks: 1966 –1988", *INFOR: Information Systems and Operational Research*, Vol. 27 (3), pp. 272-296.

Azaron A., B. Fynes and M. Modarres, 2011, "Due date assignment in repetitive projects", *International Journal of Production Economics*, Vol. 129, pp. 79-85.

Azaron A., H. Katagiri, K. Kato and M. Sakawa, 2006, "Longest path analysis in networks of queues: Dynamic scheduling problems", *European Journal of Operational Research*, Vol. 174 (1), pp. 132-149.

Colin J., M. Vanhoucke, 2015, "Empirical Perspective on Activity Durations for Project Management Simulation Studies", *Journal of Construction Engineering and Management*, Vol. 142 (1), pp. 04015047.

Gutin E., D. Kuhn, and W. Wiesemann, 2015, "Interdiction games on Markovian PERT networks", *Management Science*, Vol. 61 (5), pp. 999-1017.

Hagstrom J., 1988, "Computational Complexity of PERT problems", *Networks*, Vol. 18 (2), pp. 139-147.

Hartmann S., D. Briskorn, 2010, "A Survey of Variants and Extension of the Resource-Constrained Project Scheduling Problem", *European Journal of Operations Research*, Vol. 207, pp. 1-14.

Kulkarni V., V. Adlakha, 1986, "Markov and Markov-regenerative PERT networks", *Operations Research*, Vol. 34 (5), pp. 769-781.

Malcolm D., J. Roseboom, C. Clark and W. Fazar, 1959, "Application of a technique for a research and development program evaluation", *Operations Research*, Vol. 7 (5), pp. 646-669.

Moler C.B., C. F. Van Loan, 1978, “Nineteen dubious ways to compute the exponential of a matrix", *SIAM Review*, Vol. 20(4), pp. 801-836.

Moler C.B., C. F. Van Loan, 2003, “Nineteen dubious ways to compute the exponential of a matrix: Twenty-five years later", *SIAM Review*, Vol. 45(1), pp. 1-47.

Vanhoucke M., J. Coelho and J. Batselier, 2016, “An overview of project data for integrated project management and control", *Journal of Modern Project Management*, Vol. 3 (2), pp. 6-21.

Comparing event-node graphs with nonrenewable resources and activity-node graphs with renewable resources

Jacques Carlier

Université de Technologie de Compiègne
jacques.carlier@uds.utc.fr

1 Abstract

At the end of the fifties, two main approaches were proposed to manage a large project: the PERT method and the MPM method. In both approaches the project is modelled by a graph and one has to compute critical paths. In the PERT graph, an activity is represented by an arc whenever nodes represent events. In the MPM graph, an activity is represented by a node whenever arcs represent precedence constraints. The drawback of both methods is that they do not take into account resources. The specific drawback of the event-node graph is its large size. The scheduling literature is essentially devoted to problems with renewable resources and precedence constraints, modelled by an activity-node graph. Renewable resources are allocated to activities at their starting times and released at their completion times. A machine is an example of a renewable resource. The basic problem is the Resource Constrained Project scheduling Problem (RCPSP). The aim of this talk is to rehabilitate event-node graph and nonrenewable resources. A nonrenewable resource is produced or consumed by an activity at its occurrence time. The money is an example of a nonrenewable resource. Our basic problem is the Extended Resource Constrained Project Scheduling Problem (ERCPSP). We will present a brief review of literature on ERCPSP. We will explain that several approaches built for RCPSP can be adapted to ERCPSP. We will also report some polynomial algorithms. Next we will introduce several lower bounds and some linear programming models inspired from RCPSP ones. Finally we will report some computational results and explain why it is useful to study ERCPSP.

Synchronizing Heterogeneous Vehicles in a Routing and Scheduling Context

Marc-Antoine Coindreau[1], Olivier Gallay[1] and Nicolas Zufferey[2]

[1] Department of Operations, HEC – University of Lausanne, Switzerland
marc-antoine.coindreau, olivier.gallay@unil.ch
[2] Geneva School of Economics and Management, GSEM – University of Geneva, Switzerland
n.zufferey@unige.ch

Keywords: Vehicle Routing and Scheduling, Synchronization, Carpooling.

1 Introduction

The *Vehicle Routing Problem* (VRP) aims at defining optimal vehicle routes that visit a set of jobs spread on a given territory. Depending on the context, a job can be a delivery of goods, a pick up of components, or a service provided on-site. When scheduling the workers' routes for on-site services, the systematic use of cars (which is the main hypothesis in the VRP literature) can be inefficient when only light equipments are transported and when distances between some jobs could allow light transportation modes (e.g., bikes). Moreover, using independently light transportation modes, as done in the VRP with heterogeneous fleet (Baldacci et al. 2008), might not be always envisioned because of the limited range of such transportation modes (i.e., the maximum allowed distance to travel). Indeed, some jobs might be too distant from the depot (i.e., exceeding the allowed range or the total allowed duration of a tour). In such contexts, synchronizing light and heavy transportation modes could be a promising answer.

We focus here (see Section 2) on formulations involving, jointly, light and heavy resources to serve jobs, where both transportation modes can move independently, and where the light resources can be embedded in the heavy ones on some parts of their routes. The heavy resource can be a car, a truck or a van. The light resource can be workers on foot, on bike, equipped with an electric kick scooter, or whatever light transportation mean that can be easily embedded into the heavy resource.

A rather scarce literature addresses the problem of synchronizing light and heavy resources. In the home health-care context, a recent contribution considers synchronization of walking and driving (Fikar and Hirsch 2015). It shows that the number of vehicles can be reduced by up to 90% when an external company picks up and drops off nurses (who are also allowed to walk). This reduction comes however with an increase on the total number of workers employed. In the context of light-goods delivery, where foot couriers can be coupled with vans, Lin (2011) shows that both the average cost and the number of used cars can be reduced in comparison with the approach where vans are treated as independent transportation modes. This gain on both dimensions is observed even if that study only considers coordination during the van outbound or return leg. In parcel delivery, coupling a drone (the light resource) with a single van (the heavy resource) could lead to a gain up to 20% on the truck use (Murray and Chu 2015).

The above-mentioned papers successfully show the relevance of synchronizing heterogeneous vehicles with different characteristics. In this work, we consider the situation where the workers have the choice between traveling by car, by using electric kick scooters, or simply walking. Moreover, carpooling is enabled. The potential gain offered by the synchronization of such transportation modes is measured and discussed (see Section 3).

2 Synchronizing workers and vehicles with carpooling

We propose a new formulation that allows the synchronization of cars (heavy resource) and workers (light resource) potentially equipped with an electric kick scooter. On the one hand, the light resource is cheaper but limited by its speed and range. On the other hand, the heavy resource is faster and it can transport multiple workers, but at a larger cost and pollution impact. If not coupled with the heavy resource, the light resource is restricted to the exploration of the jobs located close to the depot. As a consequence, the heavy resources would have to make great detours to visit distant jobs in the same tour.

As electric kick scooters can be easily embedded into a car, coordinating and synchronizing these two types of resources turn out to be a promising approach to overcome the individual drawbacks of each of these two transportation modes. More precisely, we consider the case where carpooling is enabled, meaning that heavy resources can transport multiple light resources (with a maximum number of $Q = 2$ workers equipped with electric kick scooters per car). Workers are split into two categories, the car drivers and the passengers. Light and heavy resources can couple and uncouple as many times as required. Drivers are allowed to serve jobs and to use an electric kick scooter to reach jobs, but the return path to the car is mandatory. Passengers can be picked up elsewhere than at the drop-off location, after they have been using the electric kick scooter to travel between jobs that are located nearby. The considered problem is an extension of the classical VRP with time windows, in which workers with an individually assigned car must leave and come back to the depot within the working day, after having served the jobs within their assigned time windows. We focus here on analyzing the impact offered by the introduction of electric kick scooters, regarding their speed and range.

We have developed a metaheuristic (MH) based on the ruin and recreate principle (Pisinger and Ropke 2011). MH aims at improving a solution by sequentially removing and reinserting jobs. Depending on the search phase, MH can remove up to 30% of the inserted jobs. In general, the more time the search is trapped in the same local optimum, the more jobs are removed and reinserted afterwards. A typical output solution is given in Figure 1. Each worker has a color code: light gray for worker w_1, gray for w_2, double line for w_3, and black for w_4. Heavy (resp. light) resource paths are represented with plain (resp. dashed) lines. w_1 and w_4 leave the depot in the same car. w_4 is dropped off at job j_{30} and uses a light resource to travel to j_1, where s/he is picked up by w_1. Some drivers are traveling some sub-routes with a light resource, like w_2 on path $j_4 - j_8 - j_{37} - j_{19}$. All workers and vehicles start and end their working day at the central depot located in the middle of the grid.

3 Computational experiments

To validate the efficiency of MH, its results are compared to optimal VRP solutions where only heavy resources are used. The optimal VRP solutions are obtained with the *Branch-And-Price* algorithm (BP) proposed in Desaulniers et al. (2008). Allowing the workers to move without a car while enabling carpooling is expected to help managers reduce both the number of cars used (f_{car}) and the total driving distance (f_{dist}). f^*_{car} and f^*_{dist} refer to the optimal values of f_{car} and f_{dist} found by BP, respectively. Depending on the instance configuration, replacing a heavy resource by a light one can either reduce or increase the driving distance. The reduction occurs when detours to carry the light resources are overcompensated by the pooling of the heavy resources, whereas the augmentation occurs when too many detours are required to carry the light resources. We consider the case where managers want to reduce f^*_{car} without increasing f^*_{dist}.

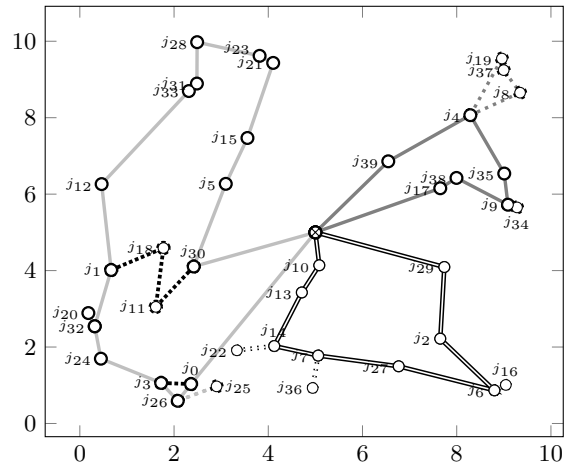


Fig. 1. Solution exhibiting coordination between light and heavy resources.

We consider 60 instances derived from real data of a large energy provider. The car speed is set to 30 km/h, whereas the light-resource speed is either 4 km/h (for walking) or 15 km/h (for the electric kick scooter). A 10-km square grid is considered, representing an urban configuration. The depot is located at the center of the grid, and Euclidean distances are considered between two job locations. Instances with $n \in \{20, 30, 40, 50\}$ jobs were generated. Indeed, lower instance sizes do not exhibit enough potential for carpooling, whereas BP is not able to provide optimal VRP solutions for larger sizes. The job characteristics (i.e., location, duration, time window) are randomly generated, based on the uniform distribution. The duration of each job belongs to [15, 34] minutes. There are three types of instances. First, for the 20 *All-Day* instances, each job has the same time window [8:00, 15:00], corresponding to the full planning horizon (i.e., the working day). Second, for the 20 *Half-Day* instances, each job has either time window [8:00, 11:30] or [11:30, 15:00]. Finally, for the 20 *Quarter-Day* instances, the possible time windows are [8:00, 9:45], [9:45, 11:30], [11:30, 13:15] and [13:15, 15:00]. These three types of instances represent three service levels that can be offered to the involved clients. Indeed, the shorter is the time window, the better it is from the client perspective, as s/he has to block a shorter time period within which s/he can be served.

Table 1 shows the percentage improvements obtained on f_{dist} and f_{car} where the following features are modified: light resource type (i.e., walking vs electric kick scooter), range (i.e., 5 km vs 10 km), service level (i.e., *All-Day* vs *Half-Day* vs *Quarter-Day*). Average results (over the 60 instances) are given in the last line. One can observe that the gain of only allowing walking and carpooling can help decreasing the driving distance by 5.57% and the number of cars by 5.76% (see the left double column labeled with "5 km"). The results highlight the importance of increasing the speed and range parameters to magnify the gain offered by the synchronization of the light and heavy resources. Indeed, both f_{dist} and f_{car} can be improved by 9.18% and 14.14%, respectively. Note that additional experiments on these instances have shown that without limiting the driving distance to f^*_{dist}, the f_{car}-gain can be up to 19.90%. Last but not least, it is important to have in mind that conservative assumptions were considered for generating the instances. Indeed, there are less than 0.5 job per km^2 and the average distance between jobs is around 5.5 km, and hence only 3% of the edges are eligible to be traveled with a light resource (i.e., when the distance between two jobs is below 1 km). One can reasonably assume that more favorable

cases would occur in other practical situations (especially in urban contexts), which would lead to the amplification of the gains.

Table 1. Potential gain when workers can move without cars (allowing carpooling).

Light resource	Walking worker				Electric kick scooter			
Range	5 km		10 km		5 km		10 km	
Objective	f_{car}	f_{dist}	f_{car}	f_{dist}	f_{car}	f_{dist}	f_{car}	f_{dist}
All-Day	11.48%	8.56%	14.75%	10.88%	18.03%	9.87%	22.95%	16.92%
Half-Day	6.25%	5.79%	6.25%	8.36%	10.94%	6.29%	12.50%	9.61%
Quarter-Day	0%	3.58%	1.52%	3.42%	6.06%	3.23%	7.58%	4.12%
Average	5.76 %	5.57 %	7.33%	6.93%	7.91 %	5.92 %	14.14 %	9.18 %

4 Conclusion

In this paper, we highlight the relevance of synchronizing heterogeneous vehicles that vary in their characteristics, more precisely light and heavy resources that differ in their speed, range and operational cost. Such a coordinated scheduling helps reducing both the number of heavy resources needed and the total driving distance. Increasing the speed of the light resource and its range leads to higher gains, and ultimately the obtained solutions would be close to those which can be achieved by coordinating truck and drones. Indeed, in the context of delivery, the next step, after having improved the situation by replacing walking by electric kick scooters, would be to consider drones as light resources. Interestingly, drones could even be faster than trucks, but additional constraints such as capacity and landing eligibility would have to be considered.

Acknowledgements

We would like to thank Prof. Guy Desaulniers for providing the optimal VRP results.

References

Baldacci R., Battarra M., and Vigo D., 2008, “Routing a heterogeneous fleet of vehicles", *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 3–27.

Desaulniers G., Lessard F., and Hadjar A., 2008, “Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows", *Transportation Science*, 42(3):387–404.

Murray C. and Chu G., 2015, “The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery", *Transportation Research Part C: Emerging Technologies*, 54:86–109.

Fikar C. and Hirsch P., 2015, “A matheuristic for routing real-world home service transport systems facilitating walking", *Journal of Cleaner Production*, 105:300–310.

Lin C. K. Y., 2011, “A vehicle routing problem with pickup and delivery time windows, and coordination of transportable resources", *Computers & Operations Research*, 38(11):1596–1609.

Pisinger D. and Ropke S., 2007, “A general heuristic for vehicle routing problems", *Computers & Operations Research*, 34(8):2403–2435.

On the construction of optimal policies for the RCPSP with stochastic activity durations

Erik Demeulemeester

KU Leuven, Faculty of Economics and Business, Department of Decision Sciences and Information Management, Leuven (Belgium)
Erik.Demeulemeester@kuleuven.be

Keywords: RCPSP, stochastic activity durations, optimal policies.

1 Abstract

In this paper, we will research for what types of resource-constrained project scheduling problems (RCPSPs) with stochastic durations an optimal policy can be constructed, which incorporates an optimal baseline schedule as well as optimal continuations whenever the realized durations of an activity render the baseline schedule or an already adapted version of it infeasible.

2 State of the art

Every single day millions of small, medium and large projects are being executed. The planning of these projects is not a simple endeavor. One often hears about the failure to complete a project within time, within budget and according to specifications (see Flyvbjerg, 2005, for a nice overview). Perfect examples thereof are the building of the international airport in Denver (200% overrun of the costs), the building of the Chunnel (80% overrun of the costs) and the organization of the Olympic Games in Athens (a billion Euro above budget). It might be obvious that project planning didn't live up to its promise in these cases (as in many others). Fundamental research in the field of project planning is therefore of utmost importance.

The vast majority of the project scheduling efforts over the last forty years have concentrated on the development of a workable baseline schedule with the goal of obtaining a project duration that is as short as possible. One traditionally makes the assumption that the durations of the activities are known and deterministic and that the resources are fully available. A realistic project, however, will always be subject to disruptions. Many types of disruptions have been studied in the literature (Yu and Qi, 2004, Wang, 2005, and Zhu *et al.*, 2005): activities can take longer than primarily expected, resource requirements or availabilities may vary, due dates may change during the execution of the project, new activities may have to be inserted (Artigues and Roubellat, 2000), etc. Research in project scheduling has focused on the one hand on proactive and reactive procedures to counteract the effects of these disruptions as much as possible: proactive planning attempts to build a stable project plan that takes the possible disruptions as much as possible into account, while the reactive planning procedures are called every time the disruption changes the baseline schedule such that it cannot be executed anymore as planned.

A typical objective function for the proactive project planning phase is the weighted sum of the deviations between the planned and the realized starting times of the different activities in the project. Quite some research (e.g., Leus and Herroelen, 2004, Van de Vonder *et al.*, 2006, 2007) focused itself on the construction of stable project plans and this mainly under the assumption of uncertain durations. Typically, the solution procedure consisted of two phases. In the first phase, a baseline plan is built that is feasible with respect

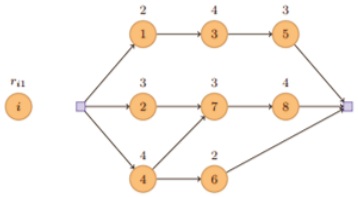
to the precedence relations as well as to the resource constraints and that is based on previously determined durations and resource requirements for every activity. In a second phase, this plan is made more stable through the introduction of time buffers before the activities (even if their predecessors take longer than expected, this doesn't automatically lead to a postponement of the corresponding activity) and through the determination of how the resources are passed along from activity to activity. A disadvantage of such a two-step procedure obviously lies in the fact that the ultimate results depend heavily upon the plan that was chosen in the first step (typically an optimal plan for the deterministic version of the RCPSP). However, very recently Davari and Demeulemeester (2016) have introduced an integrated proactive and reactive project scheduling problem for the RCPSP with uncertain durations and developed different Markov Decision Process (MDP) models to solve this problem. This means that not only a good baseline schedule is determined, but also all good continuations in case certain combinations of the activity durations occur that prohibit the baseline schedule or an already adapted schedule from being executed as planned.

A second strand of literature that solves the underlying problem in a totally different way is referred to as the stochastic RCPSP (SRCPSP). Methodologies for stochastic project scheduling view the scheduling problem as a multi-stage decision process. So-called *scheduling policies* are used to decide at each of the stages of a multi-stage decision process, that occur serially through time at random decision points, which activities selected from the set of precedence and resource feasible activities (the so-called admissibility constraints) have to be started (Ashtiani *et al.*, 2011, Möhring *et al.*, 1984, 1985, and Stork, 2001). The so-called non-anticipativity constraint requires that scheduling decisions can only be based on the observed past and a priori knowledge about processing time distributions. The objective is to minimize the expected project duration. Scheduling policies do not construct a complete schedule before the initiation of the project, but gradually build a schedule during the project's implementation. Because of this characteristic, stochastic scheduling policies are often referred to as purely reactive or on-line procedures. This also implies that no baseline schedule is constructed, which is considered as one of the more important drawbacks of this approach. In this SRCPSP, the duration D_i of each non-dummy activity i is a random variable. The random vector $(D_2, D_3, \ldots, D_{n-1})$ is written as $\mathbf{D}$. According to the definitions given in Igelmund and Radermacher (1983ab) and Möhring *et al.* (1984, 1985), a scheduling policy Π makes decisions at the decision points $t = 0$ (the start of the project) and at the completion times of activities. A decision at time t is to start at time t a precedence and resource feasible set of activities, $S(t)$, exploiting only information that has become available up to time t. As soon as the activities have been finished, the activity durations are known, yielding a *realization* (*sample*, *scenario*) $\mathbf{d}$ of the random vector $\mathbf{D}$. For a given scenario $\mathbf{d}$ and a policy Π, the project duration $C^{\Pi}_{\max}(\mathbf{d})$ is the resulting schedule makespan. The objective of the SRCPSP is to select a policy Π^* that minimizes $E(C^{\Pi}_{\max}(\mathbf{d}))$ within a specific class of scheduling policies.

Various *classes of scheduling policies* have been proposed in the literature. Stork (2001) reports promising computational results using so-called preselective policies that have been introduced by Igelmund and Radermacher (1983a) and three important subclasses of the class of preselective policies: early-start policies (ES-policies), linear preselective policies (LIN-policies) and activity-based policies (AB-policies). Ashtiani *et al.* (2011) introduce pre-processor policies (PP-policies) which make a number of a-priori sequencing decisions in a pre-processing phase while the remaining decisions are made dynamically during project execution. Quite some interesting research has been performed on determining the quality of the different scheduling policies.

3 Methodology

The goal of the research in this paper is to find optimal policies for particular versions of the RCPSP with uncertain activity durations. We will clarify this goal first by a small example instance for a standard case of the RCPSP with uncertain activity durations.



	$\pi(\tilde{p}_i = p)$									$w_{i,1}$
	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$	$p=7$	$p=8$	
$\tilde{p}_0$	1	-	-	-	-	-	-	-	-	-
$\tilde{p}_1$	-	0.4	**0.4**	0.2	-	-	-	-	-	4
$\tilde{p}_2$	-	-	-	-	-	-	0.3	0.5	**0.2**	4
$\tilde{p}_3$	-	-	-	**0.6**	0.4	-	-	-	-	7
$\tilde{p}_4$	-	-	-	0.1	0.5	**0.4**	-	-	-	1
$\tilde{p}_5$	-	-	-	-	-	-	-	**0.2**	0.8	4
$\tilde{p}_6$	-	-	-	-	-	**0.4**	0.6	-	-	1
$\tilde{p}_7$	-	-	-	0.5	**0.5**	-	-	-	-	1
$\tilde{p}_8$	-	-	**0.7**	0.3	-	-	-	-	-	1
$\tilde{p}_9$	1	-	-	-	-	-	-	-	-	38

Fig. 1. Representation of small project network.

Figure 1 represents a small project network of 8 real activities and 2 dummy activities (representing the start and end of the project), where the distribution of the activity durations is shown in the table on the right of the figure and the resource requirements for each activity are shown above the nodes that indicate the activities (the resource availability is determined to be 8 units per time unit).

Table 1. The starting times for ten feasible schedules

	S^k									
	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8	S^9	S^{10}
S_0^k	0	0	0	0	0	0	0	0	0	0
S_1^k	0	0	0	0	0	0	0	0	0	0
S_2^k	1	1	0	1	5	0	7	4	2	7
S_3^k	3	3	4	4	3	3	3	3	5	5
S_4^k	0	0	4	0	0	7	0	0	0	9
S_5^k	6	6	7	7	7	7	7	7	9	14
S_6^k	6	6	7	7	7	12	5	7	9	14
S_7^k	7	8	7	8	12	12	14	12	11	15
S_8^k	11	13	13	12	15	15	17	15	15	20
S_9^k	13	15	15	15	17	18	19	18	18	23

Table 1 represents the starting times for each activity of 10 schedules that are somehow created and that are feasible for at least one of the realizations of the durations of the activities. The optimal policy over these 10 schedules for this problem is then as follows: the optimal baseline schedule is schedule S^9, for which the planned starting times can be found in the last but one column of Table 1. However, if at time 2 it becomes clear that activity 1 takes longer than 2 time periods (a 20% chance, see Figure 1), schedule S^9 is no longer feasible (see upper left schedule of Figure 2 where activities 1, 2 and 4 are scheduled at the same time, needing $2 + 3 + 4 = 9$ resource units whereas only 8 are available). At that time, the optimal policy indicates that one should switch to schedule S^8, which is represented in the upper right corner of Figure 2. However, if at time 4 it turns out that

activity 4 requires a duration of 5 time units, the current schedule becomes infeasible again (see lower left corner of Figure 2): at that time the optimal policy indicates that one should switch to schedule S^5 (see lower right corner of Figure 2). Obviously, if more and better schedules could be generated, the resulting proactive/reactive policy will turn out to be better. This surely is a very interesting topic for further research. This paper, however, will analyze for which restricted versions of the RCPSP with stochastic durations true optimal policies can be constructed that do not depend on the generation of a restricted set of feasible schedules.

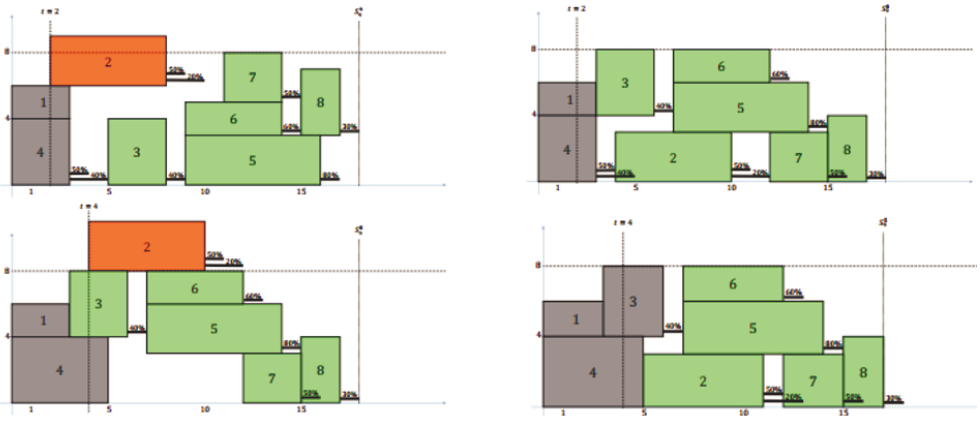


Fig. 2. Representation of the different schedules in the optimal policy.

References

B. Ashtiani, R. Leus and M.-B. Aryanezhad, "New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits of preprocessing", *Journal of Scheduling*, 14(2), 157–171, 2011.

C. Artigues and F. Roubellat, "A polynomial activity insertion algorithm in a multiresource schedule with cumulative constraints and multiple modes", *European Journal of Operational Research*, vol. 127, pp. 297–316, 2000.

M. Davari and E. Demeulemeester, "The proactive and reactive resource-constrained project scheduling problem", Revise and resubmit for *European Journal of Operational Research*, 2016.

B. Flyvbjerg, "Design by deception - The politics of megaproject approval", *Harvard Design Magazine*, Spring/Summer, pp. 50–59, 2005.

G. Igelmund and F. Radermacher, "Preselective policies for the optimization of stochastic project networks under resource constraints", *Networks*, vol. 13, pp. 1–28, 1983a.

G. Igelmund and F. Radermacher, "Algorithmic approaches to preselective strategies for stochastic scheduling problems", *Networks*, vol. 13, no. 1, pp. 29–48, 1983b.

R. Leus and W. Herroelen, "Stability and resource allocation in project planning", *IIE Transactions*, vol. 36, no. 7, pp. 667–682, 2004.

R.H. Möhring, F.J. Radermacher and G. Weiss, "Stochastic scheduling problems 1 - General strategies", ZOR - *Zeitschrift für Operations Research*, vol. 28, pp. 193–260, 1984.

R.H. Möhring, F.J. Radermacher and G. Weiss, "Stochastic scheduling problems II - Set strategies", ZOR - *Zeitschrift für Operations Research*, vol. 29, pp. 65–104, 1985.

F. Stork, Stochastic resource-constrained project scheduling, Ph.D. Thesis, Techische Universität Berlin, Germany, 2001.

S. Van de Vonder, E. Demeulemeester, W. Herroelen and R. Leus, "The trade-off between stability and makespan in resource-constrained project scheduling", *International Journal of Production Research*, vol. 44, no. 2, pp. 215–236, 2006.

S. Van de Vonder, F. Ballestin, E. Demeulemeester and W. Herroelen, "Heuristic procedures for reactive project scheduling", *Computers & Industrial Engineering*, vol. 52, no. 1, pp. 11–28, 2007.

G. Yu and X. Qi, *Disruption management - Framework, models and applications*, New Jersey: World Scientific, 2004.

J. Wang, "Constraint-based schedule repair for product development projects with time-limited Constraints", *International Journal of Production Economics*, 95, 399–414, 2005.

G. Zhu, J.F. Bard and G. Yu, "Disruption management for resource-constrained project scheduling", *Journal of the Operational Research Society*, vol. 56, pp. 365–381, 2005.

A B&B Approach to Schedule a No-wait Flow Shop to Minimize the Residual Work Content Under Uncertainty

Simone Dolceamore[1], Marcello Urgo[1]

Mechanical Engineering Department, Politecnico di Milano, Italy
simone.dolceamore@mail.polimi.it, marcello.urgo@polimi.it

Keywords: Stochastic Scheduling, Conditional Value-at-Risk, Aircraft Assembly.

1 Introduction and problem statement

In recent years, approaches providing robust schedules have been increasing their importance in the production scheduling research area. The pursued objective is to obtain schedules being insensitive - as much as possible - to disturbing factors, protecting the decision-maker against the impact of unfavorable uncertain events. In this paper we address the scheduling of a set of jobs $\mathcal{J}$ in a paced assembly line in presence of uncertainty affecting the availability of production resources. The proposed approach takes inspiration from the assembly process in the aircraft manufacturing industry. Each job j has to be processed in the assembly line made up of M stations. Being paced, the line is characterized by a cycle time, i.e., at a given time, all the parts move to the next station simultaneously. Hence, within the cycle time, a given deterministic amount of work has to be accomplished in each station. The availability of production resources, i.e., the available working hours of the workers during each cycle time, is modeled as a stochastic variable. The manufacturing system described is a permutation flow-shop with no-wait property (Emmons and Vairaktarakis (2013)). The proposed approach address the definition of a robust scheduling for the assembly line aiming at minimizing the conditional value-at-risk ($CVaR$) of the *residual work content*, i.e. the amount of workload that cannot be completed during the cycle time in the stations, due to a lack of available resources. A branch & bound approach is developed to solve the described problem to optimality. The objective function used, the $CVaR$ is a measure of risk widely used in the financial research, e.g. in portfolio optimization (Rockafellar and Uryasev (1999), Rockafellar and Uryasev (2002)). This class of risk measure has been already taken into consideration for scheduling approaches (Tolio, T. *et al.* (2011), Sarin, S. C. *et al.* (2014)). Specifically, the permutation flow-shop scheduling problem (with or without no-wait property) has been addressed in a considerably large number of papers, e.g., a branch & bound approach is developed by (Kim (1995)) with the objective of minimizing total tardiness, whereas several mixed integer formulations and an implicit enumeration approach are proposed in (Samarghandi and Behroozi 2017) and (Samarghandi and Behroozi (2016)). Nevertheless, the proposed scheduling problem has not been addressed in previous researches.

2 Description of the approach

The proposed branch & bound framework relies on a sequential definition of the schedule. At each level l of the associated tree $l \in \mathcal{J}$, a partial solution provides the sequence of the first l jobs scheduled, while the remaining $J - l \in \mathcal{J} \setminus S$ jobs are the candidates to be scheduled next in the sequence. Hence, each node of the tree has as many child nodes as the jobs to schedule, each of them representing a partial solution where a different jobs is

added to the partial sequence. The solution tree is explored adopting a depth-first strategy selecting the most promising branches in terms of the best lower bound. At each node, a lower and an upper bound on the target performance (the residual work content) are calculated to determine the most promising branches and prune the dominated ones. The contribution to the objective function of already scheduled jobs is easily calculated. Being the system a permutation flow-shop, once a job is scheduled in the first station, the cycle times where it will be processed by the following stations are automatically determined. Then, considering a single resource with availability A_c for each cycle time period c, the sequencing of that job j also entails a resource consumption R_{jc}. If a job j is scheduled to enter the first station of the line in period p, its contribution to the objective function is:

$$RWC_{S\cup\{j\}} = \underset{c}{\circledast}(A_c * R_{j,c}), \quad \forall \quad j \in \mathcal{J} \setminus S, \quad c = p, \dots p + M - 1 \tag{1}$$

where $*$ is the convolution operator.

The lower bound distribution of the residual work content caused by an unscheduled job $i \in \mathcal{J} \setminus S + \{j\}$ can be estimated through the scheduling of a dummy job $\tilde{i_1}$, having the lowest resource request among the ones of the $J - l$ unscheduled jobs. This contribution can be estimated according to Eq. 2.

$$RWC^{LB}_{S+\{j\}\cup i} = \underset{c}{\circledast}(A_c * R_{\tilde{i_1},c}), \quad \forall \quad i \in S + \{j\} \setminus \mathcal{J}, \quad c = p, \dots, p + M - 1 \tag{2}$$

In an dual way, the upper bound distribution of the residual work content caused by an unscheduled job $i \in \mathcal{J} \setminus S + \{j\}$ can be estimated scheduling a dummy job $\tilde{i_2}$ having highest among the resource request of the $J - l$ unscheduled jobs (Eq. 3).

$$RWC^{UB}_{S+\{j\}\cup i} = \underset{c}{\circledast}(A_c * R_{\tilde{i_2},c}), \quad \forall \quad i \in S + \{j\} \setminus \mathcal{J}, \quad c = p, \dots, p + M - 1 \tag{3}$$

Finally, the lower and upper bounds of the considered node can be calculated as:

$$RWC^{LB} = \underset{i}{\circledast} RWC^{LB}_{S+\{j\}\cup i} * \underset{j}{\circledast} RWC_{S\cup\{j\}}, \quad \forall j \in S \setminus \mathcal{J}, i \in S + \{j\} \setminus \mathcal{J} \tag{4}$$

$$RWC^{UB} = \underset{i}{\circledast} RWC^{UB}_{S+\{j\}\cup i} * \underset{j}{\circledast} RWC_{S\cup\{j\}}, \quad \forall j \in S \setminus \mathcal{J}, i \in S + \{j\} \setminus \mathcal{J} \tag{5}$$

Grounding on these calculations, the lower and upper bounding distributions for the residual work content can be calculated in each node. Furthermore, these distributions can also support the calculation of the lower and upper bound of a function of the risk associated to the resource consumption, e.g., the $CVaR$, with the aim at assessing the robustness of the solution. Notice that, Eq. 4 and 5 provides effective bounds for the $CVaR$ only in case the resource requirements of the jobs are deterministic. In this particular case, the convolution operator merely shifts the availability distributions without re-shaping it. This ensures the conditional value-at-risk of residual work content being a regular objective function.

Figure 1 further depicts the branching scheme adopted, as well as the computation of the bounds for the $CVaR$. Blue and black cumulative distribution functions represent the lower and upper bound distributions respectively. Nodes with a lower bound of the $CVaR$ higher than the incumbent $CVaR$ are pruned.

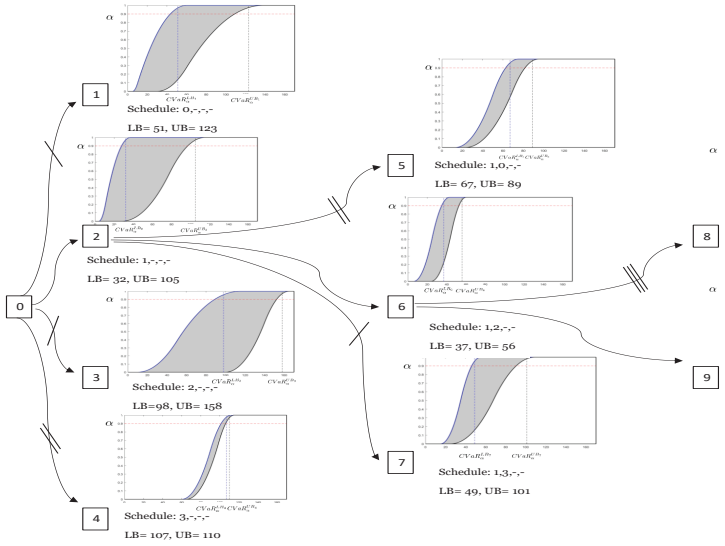


Fig. 1. Branching scheme and bounds computation

3 Testing and Industrial Application

The developed branch-and-bound approach has been implemented in C++ using the BoB++ library. Computational experiments have been performed on 8 parallel threads on an Intel Four-Core i7 Processor 7700-HQ@3.4GHz and 16 GB of DDR4 SDRAM. The performance of the algorithm has been analyzed in terms of the time to find an optimal solution and the fraction of nodes explored solving 9-jobs instances sampled from a pool of 68 real orders. The testing instances have been constructed as follows:

1. the resource requirement of a job j in station m is deterministic. In fact, at the time the assembling of an aircraft is scheduled, order specifications are known and fixed;
2. the resource availability in station m in time cycle c is a discrete triangular distribution, whose maximum value matches the planned ideal amount of workforce while minimum and the mode model the variability caused by absenteeism or other lacks of personnel;
3. the risk level used for the $CVaR$ is set to 10%, this value depends on the risk aversion of the planner, since it defines the quantile of the tail whose expected value must be minimized.

The algorithm was able to find the optimal solution in 8264.15 seconds on average, ranging from a minimum of 7803.20 to a maximum of 8819.61. The average number of evaluated nodes was 280721 over a total of 623547, with an average pruning efficiency of about 55%. The main cause of the relatively long computational times is due to the modest variability in terms of workload requirements among the considered orders, because their assembly process is composed of more or less 90% of mounting and testing operations for structural components that are common to all the orders, while customization activities have a lower impact in terms of equivalent man hours. Nevertheless this is partially due to the oversimplification of the assembly process to a single type of resource and, hence, reducing the impact of the uncertainty affecting the availability of specific resources. More-

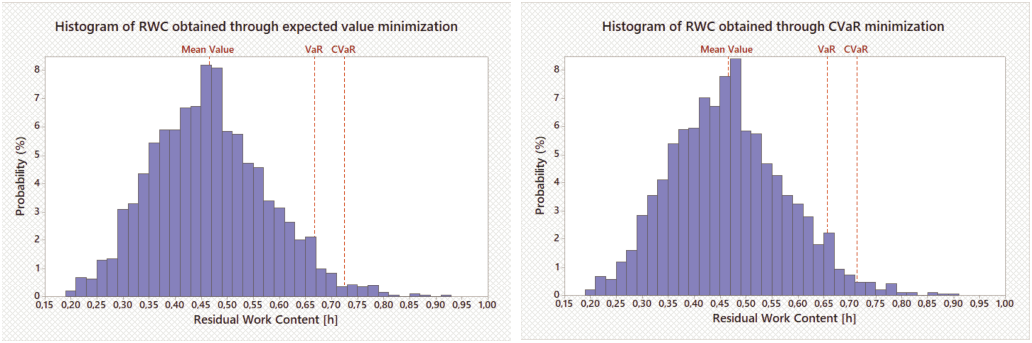


Fig. 2. Distribution of the residual work content obtained with the minimization of the $CVaR$ (right) and the expected value (left).

over, due to the convolution operations, the amplitude of the support of the distributions has a strong influence on the time needed to accomplish calculations within a single node.

An additional analysis was carried out to compare the proposed approach against scheduling to minimize the expected value fo the residual work content (RWC). An example is provided in Figure 2 showing the histogram of the RWC in the case of the minimization of the expected value (left) and the $CVaR$ (right). Although the expected value in both the cases is almost identical, the $CVaR$ is rather different (0.73 against 0.70) clearly showing that minimizing the $CVaR$ actually reduce its value in the optimal solution. Moreover the distribution on the right shows a low occurrence probability for the highest values of the RWC, thus demonstrating the capability of the approach to protect the schedule against the worst cases.

4 Acknowledgments

This research was supported by the EU projects ProRegio (Grant agreement No. 636966) and ReCaM (Grant agreement No: 680759) funded by the European Commission in the Horizon 2020 programme.

References

Emmons, H. and Vairaktarakis, G., 2013, Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications, *International Series in Operations Research & Management Science*, Vol. 182, Springer.

Rockafellar, R. T. and Uryasev, S. 2002, Conditional value-at-risk for general loss distributions, *Journal of Banking & Finance*, Vol. 26, pp. 1443-1471.

Rockafellar, R. T. and Uryasev, S. 1999, Optimization of Conditional Value-at-Risk, *Journal of Risk*, Vol. 2, pp. 21-41.

Tolio, T., Urgo, M., and Váncza, J., 2011, Robust production control against propagation of disruptions, *CIRP Annals - Manufacturing Technology*, Vol. 60, pp. 489-492.

Sarin, S. C., Sherali, H. D. and Liao, L., 2014, Minimizing conditional-value-at-risk for stochastic scheduling problems, *Journal of Scheuling*, Vol. 17, pp. 5-15.

Kim, Y.-D., 1995, Minimizing total tardiness in permutation flowshops, *European Journal of Operational Research*, Vol. 85, pp. 541-555.

Samarghandi, H and Behroozi, M., 2016, An Enumeration Algorithm for the No-Wait Flow Shop Problem with Due Date Constraints, *IFAC - PapersOnLine*, Vol. 49-12, pp. 1803-1808.

Samarghandi, H and Behroozi, M., 2017, On the exact solution of the no-wait flow shop problem with due date constraints, *Computers and Operations Research*, Vol. 81, pp. 141-159.

On Index Policies in Stochastic Scheduling

Franziska Eberle[1], Felix Fischer[2], Jannik Matuschke[3] and Nicole Megow[1]

[1] University of Bremen, Germany. {feberle,nicole.megow}@uni-bremen.de
[2] Queen Mary University of London, UK. felix.fischer@qmul.ac.uk
[3] Technical University of Munich, Germany. jannik.matuschke@tum.de

Keywords: stochastic scheduling, total completion time, approximation algorithm.

1 Introduction

We investigate a fundamental stochastic scheduling problem where jobs with uncertain processing times must be scheduled non-preemptively on m identical parallel machines. We are given a set J of n jobs, where each job $j \in J$ is modeled by a random variable P_j with known distribution. The actual realization of a processing time becomes known only by executing a job. More precisely, a job notifies the scheduler when it completes. The goal is to find a policy Π that decides for any point in time which jobs to schedule such as to minimize the expected total completion time, $\sum_{j\in J} \mathbb{E}[\, C_j^{\Pi}\,]$. Here C_j^{Π} denotes the completion time of job j under policy Π, and we drop the superscript whenever it is clear from the context. The scheduling problem can be stated in the standard three-field notation as $\mathrm{P}||\mathbb{E}[\sum_j C_j\,]$.

The deterministic version of the problem is well-known to be solved optimally by the *Shortest Processing Time* (SPT) rule (Rothkopf 1966). A natural generalization of this rule to the stochastic setting, the *Shortest Expected Processing Time* (SEPT) rule, is optimal when processing times follow exponential distributions (Bruno *et al.* 1981). For arbitrary distributions no optimal policy is known, and in the past decade research has focused on approximative policies. A stochastic scheduling policy Π is an α*–approximation*, for $\alpha \geq 1$, if for all instances I of the problem at hand it holds that $\sum_{j\in J_I} \mathbb{E}[\, C_j^{\Pi}\,] \leq \alpha \sum_{j\in J_I} \mathbb{E}[\, C_j^{*}\,]$. Here, C_j^* denotes the completion time under an optimal stochastic scheduling policy on the given instance I, assuming a priori knowledge of the set of jobs J_I and their processing time distributions P_j, but not their actual realizations. In particular, the optimal policy also does not know the realizations, i.e., it is non-clairvoyant.

Several approximation algorithms have been developed with approximation guarantees that depend either on the parameters m and n (Im *et al.* 2015) or on the probability distributions of the processing times (Möhring *et al.* (1999), Megow *et al.* (2006), Schulz (2008), and Skutella *et al.* (2016)). In the latter case, the approximation guarantee is of order $O(\Delta)$ where Δ is an upper bound on the squared coefficients of variation of the processing time distributions P_j, that is, $\mathrm{Var}[P_j]/\mathbb{E}[\,P_j\,]^2 \leq \Delta$ for all jobs j. Interestingly, there is a 2-approximation algorithm for the *preemptive* (weighted) variant of our stochastic scheduling problem $\mathrm{P}|pmtn|\mathbb{E}[\sum_j C_j\,]$ independently of the distributions (Megow and Vredeveld 2014).

In this note we rule out distribution-independent approximation factors for simple list scheduling policies in non-preemptive stochastic scheduling. More precisely we consider so-called *index policies* that assign the same priority to jobs with the same probability distribution and schedule jobs one after the other on the first machine that becomes available. *Job-based* index policies do not consider the number of jobs or the number of machines. We give a lower bound of $\Omega(\Delta^{1/4})$ for job-based index policies. Somewhat surprisingly this lower bound is obtained already for very simple instances with only two types of jobs, identical deterministic jobs and a set of stochastic jobs that all follow the same Bernoulli distribution. For this class of instances we also give a policy that is an $\mathcal{O}(m)$-approximation.

2 Lower bound for index policies

Theorem 1. *Any job-based index policy has approximation factor $\Omega(\Delta^{1/4})$ for* $\mathrm{P}||\mathbb{E}[\sum_j C_j]$.

To prove this lower bound we consider a simple class of instances that we call *Bernoulli-type instances*. This class consists of two types of jobs, deterministic jobs J_d and stochastic jobs J_s, with jobs of each type following the same distribution. A deterministic job $j \in J_d$ has processing time $P_j = p$, and a stochastic job $j \in J_s$ has processing time $P_j = 0$ with probability $q \in (0,1)$ and $P_j = l$ with probability $1-q$.

Proof. We define two families of Bernoulli-type instances, $I_1(\Delta, m)$ and $I_2(\Delta, m)$, for the problem $\mathrm{P}||\mathbb{E}[\sum_j C_j]$ where Δ is the upper bound on $\mathrm{Var}[P_j]/\mathbb{E}[P_j]^2$. The instances differ only in the number of deterministic and stochastic jobs, n_d and n_s, but not in the processing time distributions. We define the processing time for deterministic jobs in J_d to be $p = 1$, and for stochastic jobs $j \in J_s$ we define

$$P_j = \begin{cases} 0 & \text{with probability } 1 - 1/\Delta \\ \Delta^{3/2} & \text{with probability } 1/\Delta. \end{cases}$$

Note that the squared coefficients of variation are bounded from above by Δ.

For such Bernoulli-type instances there are only two job-based index policies, one where the deterministic jobs have higher priority, denoted by $J_d \prec J_s$, and one where the stochastic jobs have higher priority, denoted by $J_s \prec J_d$. We show that for any fixed $\Delta > 1$, there exists a value of m such that the cost of the schedule produced by $J_d \prec J_s$ on instance $I_1(\Delta, m)$ is greater by a factor of $\Omega(\Delta^{1/4})$ than the cost of the schedule produced by $J_s \prec J_d$, and vice versa for instance $I_2(\Delta, m)$. As the instances $I_1(\Delta, m)$ and $I_2(\Delta, m)$ are indistinguishable to a job-based index policy, this result implies the lower bound.

The First Instance. Instance $I_1(\Delta, m)$ is defined by $n_d = \Delta^{3/4} m$ and $n_s = \frac{1}{2}\Delta m$. We distinguish both priority orders.

- $\boldsymbol{J_d \prec J_s}$**:** When jobs in J_d are scheduled first, then no job in J_s starts before n_d/m (assuming w.l.o.g. that $n_d/m \in \mathbb{Z}$). Thus,
$$\mathbb{E}\Big[\sum_{j\in J} C_j\Big] \geq \frac{n_d}{m} n_s = \frac{1}{2}\Delta^{7/4} m.$$
- $\boldsymbol{J_s \prec J_d}$**:** Let X be a random variable denoting the number of jobs in J_s that turn out to be long. Note that $X \sim \mathrm{Bin}(n_s, 1/\Delta)$ and $\mathbb{E}[X] = m/2$. We distinguish two cases.
  - $\boldsymbol{X < \frac{3}{4}m}$**:** Every stochastic job starts at time 0. Thus, $\mathbb{E}\Big[\sum_{j\in J_s} C_j \,|\, X < \frac{3}{4}m\Big] \leq \frac{3}{4}\Delta^{3/2} m$. Furthermore, at least $\frac{1}{4}m$ machines are free for scheduling deterministic Jobs, J_d, at total cost bounded by $\mathbb{E}\Big[\sum_{j\in J_d} C_j \,|\, X < \frac{3}{4}m\Big] \leq \frac{n_d(n_d+1)}{\frac{1}{4}m} \leq 8\Delta^{3/2} m$.
  - $\boldsymbol{X \geq \frac{3}{4}m}$: we get a (very crude) upper bound on the expected cost by assuming all jobs have processing time $\Delta^{3/2}$ and then scheduling them on a single machine: $\mathbb{E}\Big[\sum_{j\in J} C_j \,|\, X \geq \frac{3}{4}m\Big] < \frac{1}{2}(n_d + n_s)(n_d + n_s + 1)\Delta^{3/2} \leq 3\Delta^{7/2} m^2$.

  To combine both cases and determine the total expected cost, we use the Chernoff-Hoeffding bound, which gives $\mathbb{P}\big[X \geq \frac{3}{4}m\big] \leq exp(-\frac{m}{24})$, and we conclude

$$\begin{aligned}\mathbb{E}\Big[\sum_{j\in J} C_j\Big] &\leq \mathbb{P}\Big[X < \frac{3}{4}m\Big]\mathbb{E}\Big[\sum_{j\in J} C_j \,\Big|\, X < \frac{3}{4}m\Big] + \mathbb{P}\Big[X \geq \frac{3}{4}m\Big]\mathbb{E}\Big[\sum_{j\in J} C_j \,\Big|\, X \geq \frac{3}{4}m\Big] \\ &\leq \frac{3}{4}\Delta^{3/2} m + 8\Delta^{3/2} m + exp\left(-\frac{m}{24}\right) \cdot 3\Delta^{7/2} m^2 = \mathcal{O}(\Delta^{3/2} m),\end{aligned}$$

  for sufficiently large m.

Thus, on sufficiently many machines, the index policy $J_d \prec J_s$ has total cost greater by a factor of $\Omega(\Delta^{1/4})$ than the cost of policy $J_s \prec J_d$.

The Second Instance. Instance $I_2(\Delta, m)$ is defined by $n_d = \Delta^{5/4}m$ and $n_s = 2\Delta m$. Using similar arguments as in the previous case, we can show that the index policy $J_s \prec J_d$ yields expected cost that are worse by a factor $\Omega(\Delta^{1/4})$ than the cost of policy $J_d \prec J_s$. □

3 Upper bound for Bernoulli-type instances

For the class of Bernoulli-type instances introduced above, we show that taking the number of machines and jobs into account yields an index policy that is $O(m)$-approximate. W.l.o.g. let $j \in J_d$ have processing time $P_j = p$, and $j \in J_s$ have processing time $P_j = 0$ with probability $1 - \frac{1}{l}$ and $P_j = l$ with probability $\frac{1}{l}$ for $l > 1$. Observe that the cost caused by individually scheduling J_d or J_s starting at time 0 gives a lower bound on the cost of an optimal policy. We denote these job set-individual scheduling cost by $\sum_{j\in J_t} \mathbb{E}[C_j^0]$ where $t \in \{s, d\}$. Obviously, the sum of both also is a lower bound on the optimum cost.

Firstly, note that in case of few deterministic jobs, $J_s \prec J_d$ is an $O(1)$-approximation.

Lemma 1. *$J_s \prec J_d$ is a 2-approximation for Bernoulli-type instances with $n_d \leq m$.*

Proof. The cost of scheduling $J_s \prec J_d$ is at most the cost of J_s and the cost of one deterministic job per machine starting at the completion of the last stochastic job on that machine. Then, by linearity of expectation,

$$\sum_{j\in J} \mathbb{E}[C_j] = \sum_{j\in J_s} \mathbb{E}[C_j^0] + \sum_{j\in J_d} \mathbb{E}[S_j + p] \leq 2 \sum_{j\in J_s} \mathbb{E}[C_j^0] + n_d p \leq 2 \sum_{j\in J} \mathbb{E}[C_j^*].$$ □

Moreover, if there are less stochastic jobs than deterministic ones, $J_d \prec J_s$ is $O(1)$-approximate.

Lemma 2. *$J_d \prec J_s$ is a 5-approximation for Bernoulli-type instances with $n_d > m$ and $n_s \leq 2n_d$.*

Proof. When scheduling in order $J_d \prec J_s$, machines start processing jobs in J_s no later than $\lceil \frac{n_d}{m} \rceil p \leq 2\frac{n_d}{m} p$, when all jobs in J_d have completed. Thus, the total cost of J_s is

$$\sum_{j\in J_s} \mathbb{E}[C_j^0] + n_s \cdot 2\frac{n_d}{m} p \leq \sum_{j\in J_s} \mathbb{E}[C_j^0] + 4 \sum_{j\in J_d} \mathbb{E}[C_j^0],$$

which follows from the well-known deterministic lower bound by Eastman *et al.* (1964). Adding the total cost of the deterministic jobs J_d implies the 5-approximation. □

To handle the remaining instances, recall X, the random variable counting the number of actual long stochastic jobs. Formally, $X := \sum_{j\in J_s} X_j$ with $X_j := \mathbf{1}_{\{P_j = l\}}$ indicating if $j \in J_s$ is long. Furthermore, fix a sequence of the stochastic jobs J_s and let Π_i denote the position of the ith long job in that sequence.

Lemma 3. *For X and Π_i defined as before and $1 \leq i \leq \lambda m \leq n_s$ for $\lambda \in \{1, \ldots, \lfloor \frac{n_s}{m} \rfloor\}$, the following holds:*

(i) $\mathbb{E}[\Pi_i \mid X = \lambda m] = \frac{i}{\lambda m+1}(n_s + 1)$ *and* $\mathbb{E}[\Pi_i \mid \lambda m \leq X < (\lambda + 1)m] \leq \frac{i}{\lambda m+1}(n_s + 1)$.
(ii) $\mathbb{E}[n_s - \Pi_m \mid m \leq X < 2m] \geq \frac{n_s}{4m}$.

Lemma 4. *$J_s \prec J_d$ is an $O(m)$-approximation for Bernoulli-type instances with $n_s > 2n_d > 2m$.*

Sketch of proof. We analyze the performance of $J_s \prec J_d$ by conditioning on the number X of long jobs.

- $\mathbf{0 \leq X < m}$: There is at least one machine available for scheduling the deterministic jobs. Hence, we loose at most a factor m w.r.t. an optimal solution using at most m machines.
- $\boldsymbol{\lambda m \leq X < (\lambda+1)m}$ for $\boldsymbol{\lambda \in \{1, \ldots, \lfloor \frac{n_s}{m} \rfloor\}}$ **:** All stochastic jobs are finished at the latest by $(\lambda+1)l$. Beginning at time $(\lambda+1)l$, all machines process deterministic jobs only. Hence,

$$\sum_{j \in J} \mathbb{E}[C_j \mid \lambda m \leq X < (\lambda+1)m] \leq \sum_{j \in J} \mathbb{E}[C_j^0 \mid \lambda m \leq X < (\lambda+1)m] + (\lambda+1)ln_d. \quad (1)$$

  Note that a non-clairvoyant policy does not know the positions of the long jobs. Thus, such a policy cannot start any of the stochastic jobs coming after the $(k \cdot m)$th long one before time $k \cdot l$ for $1 \leq k \leq \lambda$. Thus, $n_s - \Pi_{km}$ stochastic jobs are delayed by $k \cdot l$. For $\lambda = 1$, Lemma 3 (ii) implies that scheduling only J_s costs at least $l\frac{n_s}{4m}$, i.e., $\sum_{j \in J_s} \mathbb{E}[C_j^0 \mid m \leq X < 2m] \geq l\,\frac{n_s}{4m}$. For $\lambda \geq 2$, we can show with Lemma 3 (i) that $\sum_{j \in J_s} \mathbb{E}[C_j^0 \mid \lambda m \leq X < (\lambda+1)m] \geq \lambda l \frac{n_s}{4}$. This bounds the extra term $(\lambda+1)ln_d$ in Equation (1) in terms of the optimum cost.

Combining the results for the different values of X, we obtain

$$\sum_{j \in J} \mathbb{E}[C_j] \leq (8m+1) \sum_{j \in J} \mathbb{E}[C_j^*].$$

□

The lemmas above imply an $O(m)$-approximation algorithm based on index policies taking the number of jobs and machines into account. This result for Bernoulli-type instances can be slightly generalized to arbitrary deterministic jobs, i.e., $P_j = p_j$ for $j \in J_d$.

Theorem 2. *There exists an $O(m)$-approximate index policy for Bernoulli-type instances of $P||\sum_j \mathbb{E}[C_j]$, where the deterministic jobs may vary in size.*

References

Bruno, J.L., P.J. Downey, and G.N. Frederickson, 1981, "Sequencing tasks with exponential service times to minimize the expected flowtime or makespan", *J. ACM*, Vol. 28, pp. 100-113.

Eastman, W.L., S. Even, and I.M. Isaacs, 1964, "Bounds for the optimal scheduling of n jobs on m processors", *Management Science*, Vol. 11, pp. 268-279.

Im, S., B. Moseley, and K. Pruhs, 2015, "Stochastic Scheduling of Heavy-Tailed Jobs", *Proc. of STACS*, Vol. 30, pp. 474-486.

Megow, N., M. Uetz, and T. Vredeveld, 2006, "Models and algorithms for stochastic online scheduling", *Math. Oper. Res.* Vol. 31.3, pp. 513-525.

Megow, N. and T. Vredeveld, 2014, "A tight 2-approximation or preemptive stochastic scheduling", 2014, *Math. Oper. Res.* Vol. 39.4, pp. 1297-1310.

Möhring, R.H., A.S. Schulz, and M. Uetz, 1999, "Approximation in stochastic scheduling: the power of LP-based priority policies", *J. ACM* Vol. 46, pp. 924-942.

Rothkopf, M.H., 1966, "Scheduling with random service times", *Management Science*, Vol. 12, pp. 703-713.

Schulz, A.S., 2008 "Stochastic online scheduling", *Proc. of COCOA*, Vol. 5165, pp. 448-457.

Skutella, M., M. Sviridenko, and M. Uetz, 2016, "Unrelated machine scheduling with stochastic processing times", *Math. Oper. Res.*, Vol. 41.3, pp. 851-864.

Unrelated Parallel Machine Scheduling at a TV Manufacturer

Merve Burcu Sarıkaya, Okan Örsan Özener and Ali Ekici

Department of Industrial Engineering, Ozyegin University, Istanbul, Turkey
burcu.cakiroglu@ozu.edu.tr, orsan.ozener@ozyegin.edu.tr, ali.ekici@ozyegin.edu.tr

Keywords: parallel machine scheduling, unrelated machines, sequence-dependent setups.

1 Introduction

In this study, we analyze the scheduling problem faced by a TV manufacturer. TV manufacturing is planned based on a make-to-order strategy and mass customization due to diversified customer demand. The manufacturer utilizes multiple heterogeneous assembly/production lines that are specialized to produce TVs with different features. Each customer order is considered as a separate job, and these jobs are completed on one of the compatible assembly lines. For a given job, only a subset of assembly lines (called *compatible* assembly lines) can be used to complete the job, and the total processing time of a job depends on the assembly line used for that job. A job can only be started after all the materials (especially cell and cardboard box) are available. Before starting a new job on an assembly line, a setup time (depending on the previous job processed and the new job to be processed) is required to make the assembly line ready for production.

Our goal is to determine a production schedule with minimum total tardiness and earliness while considering the job-assembly line compatibility, cell and cardboard box availability, the sequence-dependent setup times between jobs and the workload balance among the assembly lines. We propose a sequential heuristic approach to address the problem.

The problem analyzed in this study is a variant of the unrelated parallel machine scheduling problem which is extensively studied in the literature. Logendran *et. al.* (2007) study the unrelated parallel machine scheduling problem with sequence- and machine-dependent setups and unequal release times for the jobs. They further assume that each machine has a availability constraint which sets the earliest time a machine can be used for processing jobs. They look for a minimum weighted tardiness solution. Six different search algorithms based on tabu search are developed to identify the best schedule. Lee *et. al.* (2013) also study the unrelated parallel machine setting where jobs have sequence- and machine-dependent setups. Different from Logendran *et. al.* (2007), they assume that all the jobs are available at the beginning, and the objective is to minimize total tardiness. The authors propose a tabu search algorithm that incorporates various neighborhood generation methods. Similarly, Zhu and Heady (2000) and Akyol and Bayhan (2008) consider unrelated parallel machine scheduling problem with sequence-dependent setups and equal release times. Different from the studies above, their objective is to minimize the total weighted earliness and tardiness. Zhu and Heady (2000) propose a mixed integer programming formulation, and Akyol and Bayhan (2008) develop a neural network approach to address the problem. The main differences between the above mentioned studies and the current study are machine-job compatibility restrictions and the workload balance requirement. Finally, Zhang *et. al.* (2007) consider the unrelated parallel machine setting with sequence-dependent setup times, unequal release times and machine-job compatibility restrictions. Their objective is to minimize the total weighted tardiness. They convert the problem into reinforcement learning problems by constructing a semi-Markov decision process and then apply the Q-Learning algorithm to find a solution. Different from our

setting, they do not consider the workload balance among machines and the earliness in the objective function.

2 Problem Definition

We have n assembly lines and m jobs to be processed on one of these assembly lines. We use L ($:= \{1, 2, \ldots, n\}$) to denote the set of assembly lines and I ($:= \{1, 2, \ldots, m\}$) to denote the set of jobs. Job i can only be processed on a subset of assembly lines. We use L_i to denote the set of assembly lines job i can be assigned to and I_l to denote the set of jobs that can be assigned to assembly line l. Processing time (in days) of a job depends on the assembly line it is assigned to. We denote the processing time of job i on assembly line l by p_{il}. When job j is processed immediately after job i on the same assembly line, then a sequence-dependent setup time t_{ij} is required to make the assembly line ready for processing job j. Each job has a certain due date d_i by which the job has to be finished. Job i can be started on an assembly line after its release date, and preemption is not allowed. Moreover, the two critical materials (cells and cardboard boxes) specific to each job have to be ready before a job can be started. Hence, the earliest time a job can be started is the maximum of the release time of the job, the available time of the cells and the available time of the cardboard boxes required for that job. We denote the earliest start time of job i by r_i. Finally, in order to maintain a balance between the workload of the assembly lines, the manufacturer imposes lower and upper limits on the number of jobs that can be assigned to an assembly line. We use C_1 and C_2 to denote these lower and upper limits, respectively. Our goal is to find an assignment of the jobs to the assembly lines and the processing order of the jobs on each assembly line with the objective of minimizing the total tardiness and earliness.

3 Sequential Heuristic Approach

In the proposed approach, called the *Sequential Heuristic Approach* (SHA), we decompose the set of decisions to be made into two and make one set of decisions at each stage. More specifically, in the first phase we assign the jobs to the assembly lines. Then, for each assembly line we determine processing order of the jobs assigned to it. In each phase, we make the decisions by solving mathematical models.

In the first phase, we determine which job is assigned to which assembly line. Our objective in this phase is to minimize the total processing time of the jobs. We also impose the lower and upper limits on the number of jobs that can be assigned to an assembly line. We use the following decision variable:

$$z_{il} = \begin{cases} 1, & \text{if job } i \text{ is assigned to assembly line } l \\ 0, & \text{otherwise.} \end{cases} \qquad i \in I, l \in L$$

The mathematical model solved in the first phase is as follows:

$$\text{MIP-A: Min} \quad \sum_{l \in L} \sum_{i \in I_l} p_{il} z_{il} \tag{1}$$

$$\text{s.t.} \quad \sum_{l \in L_i} z_{il} = 1 \qquad \forall i \in I \tag{2}$$

$$\sum_{i \in I_l} z_{il} \geq C_1 \qquad \forall l \in L \tag{3}$$

$$\sum_{i \in I_l} z_{il} \leq C_2 \qquad \forall l \in L \tag{4}$$

$$z_{il} \in \{0,1\} \qquad \forall i \in I, l \in L \qquad (5)$$

In this model, the objective function minimizes the total processing times of the jobs. Constraints (2) make sure that each job is assigned to an assembly line. Constraints (3) and (4) are the lower and upper limits on the number of jobs that can be assigned to an assembly line. Constraints (5) are the sign restrictions. By solving this model, we determine a feasible assignment of jobs to the assembly lines. Then, in the second phase we decide on the order jobs are processed on each assembly line. Let A_l be the set of jobs assigned to assembly line l. For assembly line l, we define the following decision variables:

$$y_{ik} = \begin{cases} 1, & \text{if job } i \text{ is processed at the } k\text{th order} \\ 0, & \text{otherwise.} \end{cases} \qquad i \in A_l, k \in \{1,2,\ldots,|A_l|\}$$

$$x_{ij} = \begin{cases} 1, & \text{if job } i \text{ is the immediate predecessor of job } j \\ 0, & \text{otherwise.} \end{cases} \qquad i,j \in A_l$$

s_i = start time of job i $\quad i \in A_l$

f_i = completion time of job i $\quad i \in A_l$

u_i = amount of tardiness for job i $\quad i \in A_l$

e_i = amount of earliness for job i $\quad i \in A_l$

We determine the order of jobs for assembly line l by solving the following model:

$$\text{MIP-S: Min} \quad \sum_{i \in A_l} (u_i + e_i) \qquad (6)$$

$$\text{s.t.} \quad \sum_{k \in \{1,2,\ldots,|A_l|\}} y_{ik} = 1 \qquad \forall i \in A_l \qquad (7)$$

$$\sum_{i \in A_l} y_{ik} \leq 1 \qquad \forall k \in \{1,2,\ldots,|A_l|\} \qquad (8)$$

$$y_{jk} + y_{i,k-1} - x_{ij} \leq 1 \qquad \forall i,j \in A_l, k \in \{2,\ldots,|A_l|\} \qquad (9)$$

$$s_j - f_i + M(1 - x_{ij}) - t_{ij}x_{ij} \geq 0 \qquad \forall i,j \in A_l \qquad (10)$$

$$s_i \geq r_i \qquad \forall i \in A_l \qquad (11)$$

$$f_i - s_i - p_{il} \geq 0 \qquad \forall i \in A_l \qquad (12)$$

$$f_i - u_i \leq d_i \qquad \forall i \in A_l \qquad (13)$$

$$e_i + f_i \geq d_i \qquad \forall i \in A_l \qquad (14)$$

$$y_{ik} \in \{0,1\} \qquad \forall i \in A_l, k \in \{1,2,\ldots,|A_l|\} \qquad (15)$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in A_l \qquad (16)$$

$$s_i, f_i, u_i, e_i \geq 0 \qquad \forall i \in A_l \qquad (17)$$

In this model, the objective is to minimize the total tardiness and earliness of the jobs. Constraints (7) make sure that each job is assigned to one of the assembly lines. Constraints (8) guarantee that no two jobs can be assigned to the same order of an assembly line. Constraints (9)-(10) enforce the setup times between consecutive jobs. Constraints (11) impose the earliest start time restriction. The completion time of a job is determined by Constraints (12). Constraints (13)-(14) determine the tardiness and earliness of each job. Finally, Constraints (15-17) impose the nonnegativity and binary restrictions.

4 Computational Results

We test the effectiveness of the proposed solution approach on real-life instances. In the real-life instances, we have 15 assembly lines dedicated for TV manufacturing and 150 jobs

to be processed on one of these assembly lines. Processing times of the jobs (depending on the assembly line used) vary between 12 minutes and 3 days. In terms of assembly line-job compatibility, depending on the type of the job, it can be processed on 1 up to 14 assembly lines. On average, a job can be processed on around 8 out of 15 assembly lines. Finally, in order to balance the workload between the assembly lines the minimum and maximum number of jobs that can be assigned to an assembly line are set to 2 and 13, respectively.

Currently, the manufacturer uses an advanced planning and scheduling module integrated with Enterprise Resource Planning (ERP) used at the company. After taking orders from ERP software, this module provides a visual display of the orders, release dates, due dates, etc. Then, the user assigns the jobs to the lines manually considering the setup times and earliness and tardiness. Experience of the user is significantly important in the current practice.

We test the proposed approach on these real life instances and compare the solutions found against the current practice in Table 1. In this table, under "SHA" column we present the percentage improvements in the total tardiness and earliness taking the solution found in the current practice as the reference point. We observe that the *Sequential Heuristic Approach* (SHA) provides significant improvements over the current practice. Total tardiness and earliness is decreased by 77.89% on average.

Table 1. Comparison between the solutions found by SHA and the current practice

Instance	SHA
1	98.27%
2	55.88%
3	91.70%
4	83.39%
5	65.75%
6	75.45%
7	77.95%
8	71.92%
9	78.37%
10	80.25%
Average	**77.89%**

References

Akyol D.E., G.M. Bayhan, 2008, "Multi-machine earliness and tardiness scheduling problem: an interconnected neural network approach", *International Journal of Advanced Manufacturing Technology*, Vol. 37, pp. 576-588.

Lee J.-H., J.-M. Yu, D.-H. Lee, 2013, "A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: minimizing total tardiness", *International Journal of Advanced Manufacturing Technology*, Vol. 69, pp. 2081-2089.

Logendran R., B. McDonell, B. Smucker, 2007, "Scheduling unrelated parallel machines with sequence-dependent setups", *Computers & Operations Research*, Vol. 34, pp. 420-438.

Zhang Z., L. Zheng, M.X. Weng, 2007, "Dynamic parallel machine scheduling with mean weighted tardiness objective by Q-Learning", *International Journal of Advanced Manufacturing Technology*, Vol. 34, pp. 968-980.

Zhu Z., R.B. Heady, 2000, "Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach", *Computers & Industrial Engineering*, Vol. 38, pp. 297-305.

A new set of benchmark instances for the Multi-Mode Resource Investment Problem

Patrick Gerhards[1]

Helmut Schmidt University Hamburg, Germany
patrick.gerhards@hsu-hh.de

Keywords: Project Scheduling, Multi-Mode Resource Investment Problem, Benchmark Instances

1 Introduction

In this paper we introduce a new set of benchmark instances for the multi-mode resource investment problem (MRIP). The MRIP is a project scheduling problem which has many practical applications such as construction projects or software development. It is an extension of the resource investment problem (RIP) also known as the resource availability cost problem (RACP) where the duration and resource requests of the activities are fixed and no mode choice is available (Möhring 1984). The goal is to find a schedule minimizing the resource costs while maintaining precedence and resource constraints as well as adhering to a given deadline. It shares some similarities with the multi-mode resource-constrained project scheduling problem (MRCPSP) where the available resources are fixed and the shortest possible makespan is to be determined.

Most of the existing work in the literature tackled the single-mode variant of the problem (RIP). For a good overview on heuristic and exact procedures we refer to Van Peteghem and Vanhoucke (2015) and Rodrigues and Yamashita (2015), respectively. For the MRIP, various heuristic approaches have been provided to tackle the problem. The problem was introduced by Hsu and Kim (2005) who developed a heuristic that combines two priority rules to schedule the activities. In Qi et al. (2015) apply modified particle swarm optimization to the MRIP. Both use problem instances from the PSPLIB (Kolisch and Sprecher 1997) which were originally designed for the MRCPSP and adapt them to get MRIP instances.

When considering the single mode case of the problem, i.e. the RIP, most of the existing work uses benchmark instances for the resource-constrained project scheduling problem (RCSPSP) such as the PSPLIB. For problems with only one mode it works just fine to adapt those RCPSP instances but when the multi-mode case is considered adapting MRCPSP instances has a major shortcoming: when the due date is set too small it can occur that many modes of the activities become not executable (further explained in Section 3). Hence, the instances lose some of their complexity since these modes can be omitted with simple preprocessing techniques. Another reason for proposing a benchmark dataset for the MRIP is that all of existing approaches use different problem instances in their computational studies which makes a comparison hard. Hence, we propose a new set of benchmark instances such that future contributions to this problem can be easily compared to one another (available at https://riplib.hsu-hh.de).

2 Problem description

The MRIP is defined by the following properties: A set of nonpreemptable activities $A = \{0, ..., n+1\}$, precedence constraints E, a set $\mathcal{R}$ of renewable resources and a set $\mathcal{R}^n$ of nonrenewable resources. For each activity i there is a set M_i of modes that can be chosen

for the execution of activity i. If mode $m \in M_i$ is chosen, activity i has duration $d_{i,m} \in \mathbb{Z}^+$ and it has a resource consumption $r_{i,m,k} \in \mathbb{Z}^+$ for each resource $k \in \mathcal{R} \cup \mathcal{R}^n$. A due date $D \in \mathbb{Z}^+$ for the makespan of the project is given. For each resource $k \in \mathcal{R} \cup \mathcal{R}^n$ the available capacity of the resource has to be chosen and resource cost factors $c_k \in \mathbb{Z}^+$ are given. The objective is to find a precedence and resource feasible schedule that minimizes the sum of resource costs.

$$\min \sum_{k \in \mathcal{R} \cup \mathcal{R}^n} c_k \cdot a_k \tag{1}$$

$$s.t. \sum_{m \in M_i} \sum_{t=ES_i}^{LS_i} x_{i,m,t} = 1 \quad \forall i \in A \tag{2}$$

$$\sum_{m \in M_i} \sum_{t=ES_i}^{LS_i} x_{i,m,t} \cdot (t + d_{i,m}) \leq \sum_{m \in M_j} \sum_{t=ES_j}^{LS_j} x_{j,m,t} \cdot t \quad \forall (i,j) \in E \tag{3}$$

$$\sum_{i \in A} \sum_{m \in M_i} \sum_{t=ES_i}^{LS_i} x_{i,m,t} \cdot r_{i,m,k} \leq a_k \quad \forall k \in \mathcal{R}^n \tag{4}$$

$$\sum_{i \in A} \sum_{m \in M_i} \sum_{q=\max(ES_i, t-d_{i,m}+1)}^{\min(t, LS_i)} x_{i,m,q} \cdot r_{i,m,k} \leq a_k \quad \forall k \in \mathcal{R}, \forall t \in T \tag{5}$$

$$a_k \geq 0 \quad \forall k \in \mathcal{R} \cup \mathcal{R}^n \tag{6}$$

$$x_{i,m,t} \in \{0,1\} \quad \forall i \in A, \forall m \in M_i, t = ES_i, \ldots, LS_i \tag{7}$$

The mathematical model presented in (1)–(7) is an adaptation of a model for the MRCPSP proposed by Talbot (1982). We define binary decision variables $x_{i,m,t}$ which are set to 1 if and only if activity i starts in mode m in period t (see (7)) and real-valued decision variables a_k which represent the available capacity of resource k (see (6)). For each activity i we calculate a lower bound ES_i and an upper bound LS_i for its possible starting period using forward and backward calculation (FBC) (Kelley 1963).

The objective function (1) minimizes the sum resource costs. Equation (2) makes sure that for every activity i exactly one mode and one starting time is assigned. With constraint (3) we ensure the precedence constraints. Constraints (4) and (5) model the nonrenewable and renewable resource requirements, respectively. The renewable resource can represent machines or workers as their available amount replenishes every time period. We also consider nonrenewable resources. They are a powerful tool for the decision maker to model the budget of the project or the use of external work force.

3 Instance Generation

We group the benchmark instances in three datasets with instances sharing the same number of activities, namely MRIP30, MRIP50 and MRIP100. The generated instances have the following characteristics: number of activities $|A|$, number of modes per activity $|M|$, number of renewable resources $|R|$, due date factor θ, order strength OS and resource factor RF. Here, order strength measures the fraction of precedence relations in E compared to the total number of possible relations and, hence, is an indicator if the precedence structure of the project is more parallel or more serial (Mastor 1970). The resource factor value is the average of how many resources are actually consumed for every mode of all the activities. Table 1 displays the values that are used. For every parameter combination we

Table 1. Parameter values

Parameter	Values
$\|A\|$	$\{30, 50, 100\}$
$\|M\|$	$\{3, 6, 9\}$
$\|R\|$	$\{2, 4, 8\}$
θ	$\{1.2, 1.4, 1.6, 1.8, 2\}$
OS	$\{0.25, 0.5, 0.75\}$
RF	$\{0.5, 1\}$

generated 5 instances, giving us in total a number of 4,050 instances. As done in the literature, the parameter θ is used to compute the due date of the project as in the following equation (activity $n+1$ is the dummy end activity that marks the end of the project and has a duration of 0):

$$D = \texttt{Round}(\theta \cdot EST_{n+1}) \tag{8}$$

For smaller values of θ many modes can be infeasible. That means that their earliest finish time (earliest start plus duration of the mode) is larger than their latest finish time (w.r.t to the latest start of their successors in order to not violate the due date constraint). This can happen when the due date is relatively small compared to the earliest start time of the dummy end activity end the fact that the minimal durations of the activities are used when calculating the earliest and latest start times with FBC. When, for example, we use $\theta = 1$ then the durations of modes can not differ for activities on the critical path or all modes with a duration higher than the minimum duration are infeasible (w.r.t to the due date constraint). Hence, we use only values greater or equal than 1.2 for θ and apply a repair mechanism when infeasible modes are encountered.

For every instance we have only one nonrenewable resource since it can be shown that an instance with multiple nonrenewable resources can be transformed in polynomial time into an instance with just one nonrenewable resource. An optimal solution for the transformed instance can be translated into a feasible optimal solution of the original instance and vice versa (the concept of a polynomial-time reduction will be given in the presentation due to space limitations).

Next, we describe how we actually computed an instance with the desired properties. We used the network generator RanGen (Demeulemeester et al. 2003) to generate an activity-on-the-node network with the desired number of activities and the desired order strength value. Next, we draw for every activity i and all its modes $m \in M_i$ the duration $d_{i,m}$ as a discrete uniform distributed random number $\mathcal{U}\{1, 10\}$. The resource requirements $r_{i,m,k}$ for every resource $k \in \mathcal{R} \cup \mathcal{R}^n$ are also drawn from $\mathcal{U}\{1, 10\}$. If the value of $RF = 0.5$, then we set arbitrarily half of the renewable resource requirements of each mode to 0. After all the resource requirements and the duration for an activity is determined, we check if there are dominated modes. A mode is dominated if there is another mode with shorter or equal duration and lower or equal resource requirements for all resources. If a dominated mode occurs, the duration and resource requirement values of the dominated mode as well as the other mode that is responsible for the domination get redrawn. This is repeated until each activity has no dominated modes. Then, we calculate the earliest start times (EST) with the forward pass technique and the due date D of the project as in (8). With D as an upper bound for the completion of the project we can use a backward pass to compute latest finish times (LFT) for every activity. We use the EST and LFT to check for infeasible modes. A mode m of activity i is infeasible if the following inequality does not hold:

$$EST_i + d_{i,m} \leq LFT_i \tag{9}$$

If an infeasible mode is encountered, the values for this mode get redrawn. Since the minimal durations can change, we compute EST, D and LFT again and repeat this procedure until no mode is infeasible.

We choose to set the cost factors c_k to be 1 for all resources in this benchmark set. Setting them to another random number or multiplying the resource requirements for the respective resource by that random number would basically result in the same outcome. In this benchmark set we get the randomness for the resource allocation by the resource consumption and the duration of the modes. For future work it could be interesting to analyse different cost structures or distributions (e.g., cheap resource types versus expensive resource types which are also considered in the design of the modes).

4 Computational Experiments

We tested the new instances with a relatively simple iterated local search (ILS) and implemented the mathematical model displayed in (1)-(7) as a integer program (IP) in Gurobi. Results are presented at the conference due to space limitations but show that the proposed instances are quite challenging and need further investigation by means of more advanced metaheuristic procedures.

5 Conclusion

In this work we argue why benchmark datasets for the multi-mode resource investment problem are needed and which specific features need to be considered regarding the multi-mode case. We introduce a procedure to obtain instances with no dominated or infeasible modes and provide those instances such that future research is easier to compare. First experiments show that the instances at hand are challenging and need further investigation by exact and heuristic approaches.

References

Demeulemeester, E., M. Vanhoucke, W. Herroelen, "RanGen: A random network generator for activity-on-the-node networks", *Journal of Scheduling*, Vol 6, No. 1, pp. 17-38.

Hsu, C. C., D. S. Kim, 2005, "A new heuristic for the multi-mode resource investment problem", *Journal of the Operational Research Society*, Vol. 56 No. 4, pp. 406-413.

Kelley, J. E., 1963, "The critical-path method: Resources planning and scheduling", *Industrial Scheduling*, Vol. 13, no. 1, pp. 347-365.

Kolisch, R., A. Sprecher, 1997, "PSPLIB - A project scheduling problem library", *European Journal of Operational Research*, Vol. 96, No. 1, pp. 205-216.

Mastor, A., 1970, "An experimental and comparative evaluation of production line balancing techniques", *Management Science*, Vol. 16, No. 11, pp. 728-746.

Möhring, R. H., 1984, "Minimizing costs of resource requirements in project networks subject to a fixed completion time", *Operations Research*, Vol. 32, No. 1, pp. 89-120.

Rodrigues, S. B., D. S. Yamashita, "Exact methods for the resource availability cost problem", In: Handbook on Project Management and Scheduling, editors C. Schwindt, J. Zimmermann, Vol. 1, pp. 319-338, Springer International Publishing, Cham.

Talbot, F. B., 1982, "Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case", *Management Science*, Vol. 28, No. 10, pp. 1197-1210.

Van Peteghem, V., Mario Vanhoucke, "Heuristic methods for the resource availability cost problem", In: Handbook on Project Management and Scheduling, editors C. Schwindt, J. Zimmermann, Vol. 1, pp. 339-359, Springer International Publishing, Cham.

Qi, J. J., Y. J. Liu, P. Jiang, B. Guo, 2015, "Schedule generation scheme for solving multi-mode resource availability cost problem by modified particle swarm optimization", *Journal of Scheduling*, Vol. 18, No. 3 pp. 285-298.

A simheuristic for stochastic permutation flow shop problem considering quantitative and qualitative decision criteria

Eliana Maria González-Neira[1,2], Jairo R. Montoya-Torres[2]

[1] Departamento de Ingeniería Industrial, Pontificia Universidad Javeriana, Colombia
eliana.gonzalez@javeriana.edu.co
[2] Facultad de Ingeniería, Universidad de La Sabana, Colombia
elianagone@unisabana.edu.co, jairo.montoya@unisabana.edu.co

Keywords: stochastic permutation flow shop, robustness, earliness, tardiness, qualitative criteria.

1 Introduction

The flow shop problem (FSP) has been one of the most well studied problems in scheduling literature (Vallada *et al.*, 2015). Nevertheless, less research has been conducted for the stochastic case (González-Neira *et al.*, 2017; Gourgand *et al.*, 2000). Li and Ierapetritou (2008) have mentioned that the fact of designing systematic ways to take into account stochasticity is as important as the model itself.

Regarding the objective function, four aspects must be mentioned: two related with quantitative decision criteria, one with qualitative decision criteria and one with robustness of the solutions obtained. Firstly, the makespan for the deterministic case, and the expected makespan for the stochastic counterpart, have been the most studied measures (Gourgand *et al.*, 2000; Vallada *et al.*, 2015). However, other criteria that consider due date related measures are key objectives in today competitive markets, as they may be measures of customer service level. Moreover, the adoption of just-in-time (JIT) measures such as earliness/tardiness has been object of interest in the past two decades, because earliness may cause obsolescence, more inventory holding costs, requirement for more storage space (Chandra *et al.*, 2009) among others.

Secondly, the fact of consider various objectives simultaneously is natural in real-life problems (Yenisey and Yagmahan, 2014). Still, most of the literature considers only single objective problems.

Thirdly, in the scheduling literature, almost all researches, with very few exceptions, have considered only quantitative decision criteria but, as in other optimization problems, qualitative criteria are important and can reduce the gap between theory and practice. Chang and Lo (2001) and Chang *et al.* (2008) studied a multi-criteria job shop in which strategic importance of customers was considered as qualitative criteria. The former hybridized a genetic algorithm, tabu search, analytic hierarchy process (AHP) and fuzzy theory to solve the problem. The later used a hybridization of ant colony algorithm and AHP. González-Neira *et al.* (2016) minimized expected costs of tardiness as quantitative criteria and strategic customer importance as qualitative criteria in a stochastic hybrid FSP. This last study employed a method based on stochastic multicriteria acceptability analysis, hybridized with a GRASP and a Monte Carlo simulation to deal with both type of criteria.

Fourthly, research on scheduling under uncertainties has taken mainly two approaches: the stochastic approach, in which parameters are modelled with probability distributions with the goal of minimizing the expected value of a selected measure, and the robust

approach, in which uncertain parameters are modelled with intervals and the schedule obtained is more stable and suffer less variations under uncertainty. Nonetheless, the combination of both approaches has not been addressed. It is known that if an enterprise collects all data of their production, in short time, it may have sufficient data to estimate accurately distribution probability of uncertain parameters. By having this probability distribution, the robust schedule obtained can be more adjusted than other schedules in which uncertainties are modelled with intervals.

To the best of our knowledge, there is not a work that includes simultaneously the analysis of a JIT environment with stochastic parameters, and quantitative and qualitative criteria to obtain robust solutions. Hence, the current work proposes a multicriteria optimization approach to solve a stochastic PFSP that includes both, quantitative and qualitative decision criteria. As quantitative objectives, the expected earliness/tardiness E[E/T] and the standard deviation of earliness/tardiness SD(E/T) are addressed; the latter to obtain more robust schedules. As qualitative measure, the expected fulfilment of customer importance (E[CI]) of jobs, that gives priority to the most important jobs for the company, is considered.

2 Proposed solution approach

The proposed methodology consists of a simheuristic that integrates Monte Carlo simulation into an GRASP metaheuristic (Resende and Ribeiro, 2010), hybridized with pareto archive strategic evolution algorithm (PAES) (Knowles and Corne, 2000) to deal with multiple objectives. Additionally, the AHP methodology is integrated to qualify all Pareto solutions under different weight vectors for the selected criteria.

Special variations of GRASP have been proposed to solve multi-objective problems. Those are combinations of pure and combined strategies for both, construction and local search phases (Martí *et al.* 2015). Pure strategies are those in which only one objective function guides each construction and the entire local search. In this paper, a GRASP with a pure strategy for construction stage is used. Local search does not need a strategy because it integrates the PAES algorithm to construct the Pareto Archive.

Two greedy functions were considered for the construction phase. EDD rule to deal with earliness/tardiness objective, and a penalization assigned to each job depending on its customer importance and position in the sequence, to deal with qualitative objective (see Table 1). The reason for penalizing the accomplishment of customer importance in relation with the position in the sequence, is because it is desired that a job of a very strategic and important customer for the company be processed in the first positions rather than in the final positions of the sequence. Likewise, it is undesirable to schedule a job of a not very much important customer in the first positions, because it would be stolen a position that should be taken by a job of a customer of greater importance. Obviously if the job is not tardy it doesn't matter which position of the sequence it occupies. Considering these aspects, the penalization scores were defined with the following criteria: i) a job that is not tardy has a score of zero; ii) if a job is tardy its penalization is greater if the customer importance is high, and lower if its customer importance is low; iii) a job penalization increases if job is taking the place of a job that has greater or lower customer importance. Table 1 presents an example for an instance of 10 jobs. For our experiments we supposed that there are 5 degrees of customer importance, where 1 is assigned to the most important clients and 5 to the worst ones. For the instances tested in this project, a random assignment of the customer importance for each job was done following the probabilities indicated in Table 2. Of course this scale from 1 to 5 for importance customer, the and probability of the importance level were established just for the purpose of testing the methodology. In real

cases, the assignment of customer importance will not be probabilistic but deterministic according with the decision maker.

Table 1. Penalizations for position in the sequence depending on customer importance

	Job	**5**	**4**	**10**	**1**	**3**	**7**	**2**	**6**	**9**	**8**
/	**Customer importance**	**1**	**2**	**2**	**2**	**3**	**3**	**4**	**4**	**4**	**5**
Job position in sequence	**1**	1	5	5	5	7	7	7	7	7	5
	2	6	1	1	1	4	4	5	5	5	4
	3	6	1	1	1	4	4	5	5	5	4
	4	6	1	1	1	4	4	5	5	5	4
	5	11	5	5	5	1	1	3	3	3	3
	6	11	5	5	5	1	1	3	3	3	3
	7	16	9	9	9	4	4	1	1	1	2
	8	16	9	9	9	4	4	1	1	1	2
	9	16	9	9	9	4	4	1	1	1	2
	10	21	13	13	13	7	7	3	3	3	1

Table 2. Probabilities of customer importance occurrence

Customer importance	**1**	**2**	**3**	**4**	**5**
Probability of occurrence	8%	12%	20%	28%	32%

The main idea of the construction procedure is alternating the two different greedy functions at each iteration of GRASP. Therefore, suppose that the procedure begins with EDD for the first iteration. Next it uses customer importance for iteration 2, and repeats EDD for iteration 3, and so on. The RCL set is defined as the subset of jobs for which greedy function values are in the first 10% of the total range of greedy function values. Then, a job is randomly selected from RCL to form part of the partial solution. The procedure continues until all jobs have been scheduled and then, the local search begins. Local search phase consists of 2-optimal interchanges between jobs.

To deal with the stochastic nature of the problem, a Monte-Carlo Simulation is embedded into GRASP. Each sequence obtained in both, construction and local search phases, is simulated with the required number of runs to give an accurate confidence interval of at least $\pm 1\%$ around each of the three objective functions, following the procedure proposed by Framinan and Perez-Gonzalez (2015).

Once these three measures are obtained for each solution, the solution is evaluated to decide if it should enter in the Pareto Archive or not. If it enters, the other solutions already saved in the Pareto Archive are evaluated to determine if they remain in the Archive or not. If the solution does not enter in the Pareto Archive, it is discarded. This is done according PAES method. A GRASP iteration ends when no interchanges can enter to the Pareto Archive and then, a new iteration begins. The simheuristic stop time is established as: $number\,of\,jobs \times number\,of\,machines \times 1s$. Once each Pareto frontier is obtained, we scored all Pareto sequences with the usage of AHP methodology. We used six different vectors of criteria weights (Table 3) for the three selected measures. These criteria weights resulted from an AHP qualification process in which we scored an objective function versus another, in the scale from 1 to 9, as indicated by AHP procedure. One example of a vector

of criteria weights is shown Table 4. Then, from each weight vector we could select the best solution among the Pareto frontier solutions. In order to compute the matrix of option scores, for each pair of sequences s_1 and s_2, we divided the expected earliness/tardiness of s_1 by the expected earliness/tardiness of s_2, so if the division was > 1 the earliness/tardiness of s_1 was worse than the earliness/tardiness of s_2 and vice versa. Similar divisions were done for the other two objective functions (standard deviation of earliness/tardiness and customer importance).

Table 3. Vectors of criteria weights used for qualification of Pareto Solutions

	Weights vector					
Objective Function	**1**	**2**	**3**	**4**	**5**	**6**
E[E/T]	66.67%	22.22%	66.67%	22.22%	11.11%	11.11%
SD(E/T)	22.22%	66.67%	11.11%	11.11%	66.67%	22.22%
CI	11.11%	11.11%	22.22%	66.67%	22.22%	66.67%

Table 4. Example of priority vector

AHP qualification				
Objective Function	**E[E/T]**	**SD(E/T)**	**CI**	**Resultant weight vector**
E[E/T]	1	3	6	66.67%
SD(E/T)	1/3	1	2	22.22%
CI	1/6	1/2	1	11.11%

3 Analysis of Results

Two probability distributions and two coefficients of variation were selected to model both, the stochastic processing and setup times. The first 60 Taillard' benchmark instances were taken to test the methodology; this corresponds to 960 Pareto frontiers. With the application of the AHP method, we selected the best solution for each one of the 6 different vectors of criteria weights, from each Pareto frontier. That means a total of 5760 solutions, each of which exhibits an AHP score and a value for the three objectives. Three ANOVAs were executed to analyse jointly the effect of seven factors in the three selected objective functions (E[E/T], SD(E/T) and E[CI]). The factors and their levels were: probability distribution of processing times (PDPT) (lognormal -lgn- and uniform -unf-), coefficient of variation of processing times (CVPT) (0.25 and 0.50), probability distribution of setup times (PDST) (lgn and unf), coefficient of variation of setup times (CVST) (lgn and unf), vectors of criteria weights of AHP (WV) (1 to 6), number of jobs (20 and 50) and number of machines (5, 10 and 20).

According to the results, all main effects are statistically significant in the three measures, and at least for one objective function the double interaction effects are also significant (P-values < 0.05). The Main effects plots can be seen in Figure 1. It shows that the WV discriminates the Pareto solutions, facilitating to the decision maker the selection of a solution from the Pareto Frontier. Also, it can be seen that for E[E/T] and SD[E/T],

the coefficients of variation of both, setup and processing times, affect the response substantially by incrementing the three objectives as the coefficients of variation increase. The same happens with E[CI] but not in the same degree. Additionally, the measures tend to be greater for lognormal probability distribution than for the uniform distribution. This shows the importance of making an accurate fitting of probability distribution to obtain adjusted robust measures.

Future work could be directed to analyze another probability distributions and coefficient of variations. In fact, it should be evaluated the case when the processing time probability distribution of each job has a different variation coefficient, which is normal in real cases. Finally, another qualitative criteria should be incorporated in the analysis.

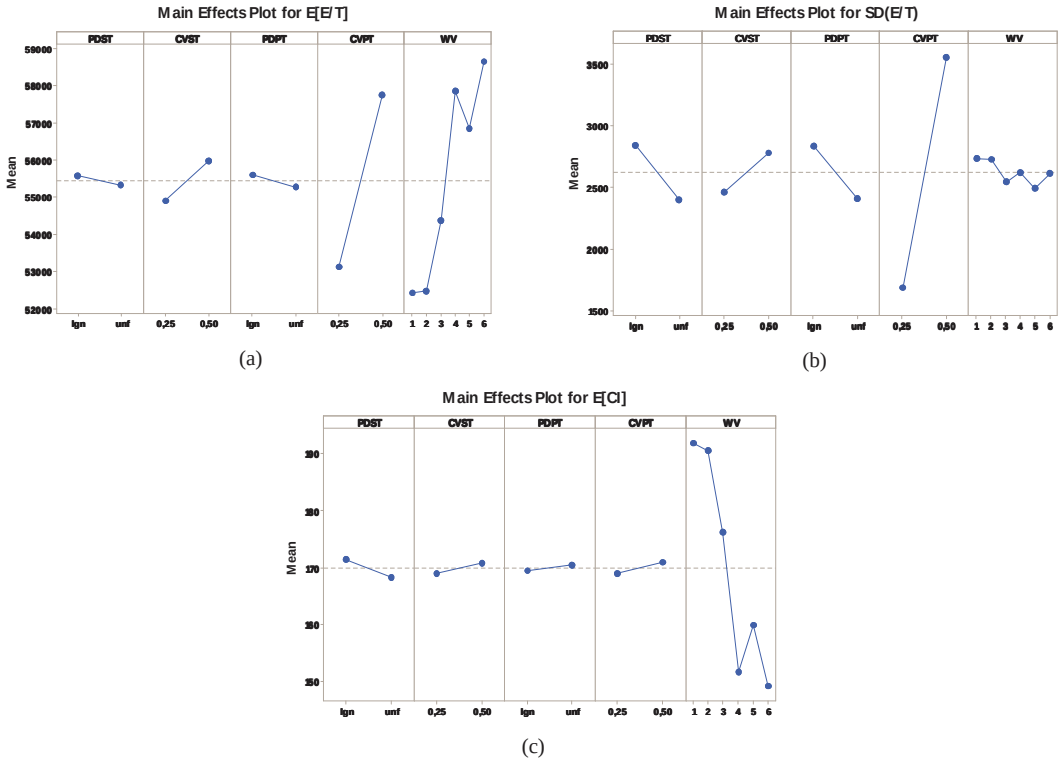


Fig. 1. Main effect plots: (a) for E[E/T], (b) for SD(E/T) and (c) for E[CI].

References

Chandra P., P. Mehta, D. Tirupati, 2009, "Permutation flow shop scheduling with earliness and tardiness penalties", *International Journal of Production Research*, Vol. 47, pp. 5591–5610.

Chang P.-T., K.-P. Lin, P.-F. Pai, C.-Z. Zhong, C.-H. Lin, L.-T. Hung, 2008, "Ant colony optimization system for a multi-quantitative and qualitative objective job-shop parallel-machine-scheduling problem", *International Journal of Production Research*, Vol. 46, pp. 5719–5759.

Chang P.-T., Y.-T. Lo, 2001, "Modelling of job-shop scheduling with multiple quantitative and qualitative objectives and a GA/TS mixture approach", *International Journal of Computer Integrated Manufacturing*, Vol. 14, pp. 367–384.

Framinan J. M., P. Perez-Gonzalez, 2015, "On heuristic solutions for the stochastic flowshop scheduling problem", *European Journal of Operational Research*, Vol. 246, pp. 413–420.

González-Neira E.M., R.G. García-Cáceres, J.P. Caballero-Villalobos, L.P. Molina-Sánchez, J.R. Montoya-Torres, 2016, "Stochastic flexible flow shop scheduling problem under quantitative and qualitative decision criteria", *Computers & Industrial Engineering*, Vol. 101, pp. 128–144.

González-Neira E.M., J.R. Montoya-Torres, D. Barrera, 2017, "Flow-shop scheduling problem under uncertainties: Review and trends", *International Journal of Industrial Engineering Computations*, Vol. 8, pp. 399–426.

Gourgand M., N. Grangeon, S. Norre, 2000, "A review of the static stochastic flow-shop scheduling problem", *Journal of Decision Systems*, Vol. 9, pp. 1–31.

Knowles J.D., D.W. Corne, 2000, "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy", *Evolutionary Computation*, Vol. 8, pp. 149–172.

Li Z., M. Ierapetritou, 2008, "Process scheduling under uncertainty: Review and challenges", *Computers & Chemical Engineering*, Vol. 32, pp. 715–727.

Martí R., V. Campos, M. Resende, A. Duarte, 2015, "Multiobjective GRASP with Path Relinking", *European Journal of Operational Research*, Vol. 240, pp. 54–71.

Resende M., C. Ribeiro, 2010, "Greedy randomized adaptive search procedures: Advances, hybridizations, and applications"; In *Handbook of Metaheuristics*; Gendreau M., Potvin, J.-Y.; pp. 283–319; Springer US.

Vallada E., R. Ruiz, J.M. Framinan, 2015, "New hard benchmark for flowshop scheduling problems minimising makespan", *European Journal of Operational Research*, Vol. 240, pp. 666–677.

Yenisey M.M., B. Yagmahan, 2014, "Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends", *Omega*, Vol. 45, pp. 119–135.

An Algorithm for Schedule Delay Analysis

Guida P.L.[1], Sacco G.[2]

[1] PMS Project Management Systems, Rome, Italy
pl.guida@alice.it
[2] Engineering Ingegneria Informatica, Rome, Italy
giovanni.sacco@eng.it

Keywords: project management, schedule, critical path method, delay analysis, claim management.

1 Schedule Delay Analysis

The analysis of the schedule delays is a permanent problem of practical application in project management. From delays can depend the final outcome and success of projects, particularly where large sums are at stake and time and cost are very sensitive variables, often dependent on each other. In client-supplier or owner-contractor relations, the schedule delays usually represent a very important issue, often undermining the commercial outcome of the whole project. In fact delay claims are a very well-known issue which is often to be managed in order to settle lengthy negotiations and even juridical cases, tracing to responsibilities and monetary compensations.

The present work tackles the problem of schedule delays, linking to literature already present on the topic, and contributing to solutions already published. We finally implement an algorithmic method, referenced to as "float banking" which can be of interest to practitioners and stakeholders in the field of project management.

2 Methods and literature review

Literature on schedule delay analysis is plenty of attempts to produce rational ways to cope with the problem of sharing delay responsibilities between the project actors, namely Owner and Contractor(s). Literature on the subject has become to appear in the 90s – e.g. Alkass (1996) – and is still flourishing.

A conventional way to allocate delaying events encountered on a project is to classify them according to their origin by the responsible party or event – either Owner, Contractor or Force Majeure – and whether the same events are excusable, compensable or not; which also should take into account the so-called "snow ball" effect of delay perturbations. In summary one finds basic types of delays so defined:

EC: owner-caused excusable compensable
EN: owner-caused excusable but not compensable
NN: contractor-caused neither excusable nor compensable.

In particular such effects and originating events can be the subject of extensive treatment in contractual clauses as well as their "correct" analysis and allocation give often rise to hard claims and juridical cases between Owner and Contractor, due to their financial and contractual impacts. The subject among others has originated a specific discipline known as Forensic Schedule Delay, where the so-called *Schedule Delay Analysis* (*SDA*) plays a major role, in the hands of scheduling (e.g. critical path analysis) arbitration and project management experts, e.g. Pickavance (2010).

In this framework a number of delay analysis methods have been proposed, such as:

- As-planned vs as-built
- Impacted as-planned
- As-planned But for
- Collapsed as built,
- "Windows" analysis, and
- Time Impact Analysis

for which the reader is invited to refer to respective literature, e.g. Davison and Mullen (2009), Keane and Caletka (2015). Furthermore professional associations like Society of Construction Law (SCL, 2002) and Association for the Advancement of Cost Engineering International (AACEI, 2007) have published seminal references on the subject. For the present discussion we particularly refer to the paper by Braimah (2013), which also provided us the case study here developed. The topic remains relatively complex to tackle when confronted with real case problems, though foundation theory has progressed and can be helpful to assist claimants and defendants in the courts. Among more specific issues arising on this subject one may recall delay concurrency, float ownership, acceleration and "pacing" (i.e. the slowing down work activities dependent on another party's lateness).

The most relevant approaches of analysing schedule delays appear the last two in the above cited list – Windows and Time Impact analysis – which have most inspired this paper. The aim is to improve methods which can be used in real-time and are efficient and convincing in providing logical solutions, also aligned to legal practices.

2.1 Time Impact Analysis

Time Impact Analysis (TIA) method applies re-scheduling at each specific delay or delaying event, the schedule being updated to a possibly new completion date, including a new or more critical paths. A picture of the project is developed each time it experiences a disturbing event, imputing delay responsibilities as soon as they occur, which can also steer management to undertake timely control actions.

In traditional literature authors report that the method may not be practical due to the large number of delay events/causes and the laborious re-planning work required. However the writing authors believe that the technique is now useable thanks to modern scheduling tools and project management architectures which are taking place in the construction field applications and yard control offices, where relatively complex projects should not be dispensed any more. Moreover real-time delay assessment can consolidate project information and respective performance either by Owner, Contractor or impacts caused by Force Majeure (so-called acts of God).

Finally TIA can become standard method for the problem in question, its "algorithmic" results being less prone to questioning vs. other methods that can provide more approximate solutions and different results on the same problem, as exemplified in Braimah (2013). In particular when a TIA analysis is performed following a delaying event, this can impact a large number, theoretically all other project activities, changing their floats and/or determining a new critical path to the forecasted completion date. In this application we investigate the effects of changing the single float values of other activities, while tracking responsibilities of the parties concerned – Owner, Contractor and Force Majeure – and improving the attention so far dedicated to float management by previous literature.

In our conceptual model an activity float is like an economic reserve or resource which can be impacted by another activity behaviour, up to being nullified or forced to become negative, where a project delay is to occur[1]. To this aim a model from economics is borrowed. Assuming each activity has an elementary account, where the float plays like reserve

[1] Assessment of *negative* floats and trends is a customary way to analyse schedule delays phenomena, as discussed for instance by Keane and Caletka (2015).

funds, these may be decreased or increased by other activities behaviors, representing credits and debts, alike in accounting practice. In particular assuming that floats are owned by the responsible party, Owner or Contractor, any values changed by the "same" party are of no charge, while e.g. a decreased float by the other party can represent a future credit. Total time budget is eventually synthesized on the project account, say completion date or total delay.

2.2 Proposed algorithm and method

The management of floats for each activity means recording of their evolution during the project and re-assigning pro-quota their reduction/gain to the respective party. In particular during the execution of a project, float possessed by an activity may increase or decrease due to other activities behaviour or external events. For example, some activity may become critical, so being penalized in future progress without having any direct responsibility.

The present implementation manages this accounting by introducing an appropriate data structure, defined as float bank, which is updated during the project dynamics and trace delays back to their original causes and responsible parties. Any time an activity duration and hence its float is changed, float banking updates the relevant information, such as event causing the float change and its responsible party. Besides zeroing, floats can increase, decrease and become negative. Moreover, following each event, one updates the history of all floats and activities concerned, can trace the changes of the critical path(s) and record whether the activities become critical or hypercritical (negative total float), with additional project delay.

In practice one can evaluate the impacts due to: - Owner (e.g. impoverishing his or some contractor safety margins); - Contractor (e.g. reducing project efficiency and escalating costs or liquidated damages); - Force majeure, with no direct responsibility on the project performance, but accepted as act of God. Therefore at any moment one can have an account of all integral float values and originating causes, like a bank statement.

2.3 Implementation method

The general logic of the method here implemented can be outlined as follows:

From project start
For each detected delaying event in chronological order:
compute the impact on the activity and all other activities possibly impacted.
determine the event responsibility (Owner, Contractor, FM) of all changes.
These steps apply the CPM scheduling algorithm, recomputing the critical path and updating the float bank.
Make available the new information to project and contractual management rules.
(Recycle for new delay event until the project end).

The specific algorithmic procedures and required data administration cannot be fully described here due to space limitation. A more complete paper will be made available online [see ScheDA in References] and is planned to be submitted to a project management journal.

As already mentioned we only report here the results obtained with the case study by Braimah (2013) while other cases from other literature on the subject have also positively been tested for validation.

3 Case study

From the referenced case study, where a planning network of 12 activities is defined and 10 delaying events of various timing impact are injected during the project course, one obtains the results in Table.1. In this exercise the project original duration of 40 days was delayed by 11 days, with delays justified (EC, NN) according to classification already reported.

Table 1. Summary of delay analysis results for the case study according to different methodologies

Delay analysis methodology	**Delay**	
	EC	**NN**
As-planned vs As Built	9	2
Impacted As-planned	6	8
As-planned But for:		
a) Contractor's point of view	4	7
b) Owner's point of view	9	2
Collapsed As-built	6	5
Window Analysis	7	4
Time Impact Analysis	***6***	***5***

One can see the summary of delay responsibility allocation, shared between EC and NN, as produced by the different methods, taken from the cited reference, Braimah (2013), with the additional and last row (bolded italics) obtained by our application. As already said, different and more heuristics based methods may provide different results; in particular one method (Impacted as Planned) gives a total delay greater than the actual one.

Besides aligning to the various approaches present in the literature, the method here developed provides additional focus on the float dynamics and float management modelling, which, according to our knowledge, is not so explicitly developed in previous papers. The more recent paper on the subject appears to be Yang and Kao (2012) whose algorithmic mechanics is however different from ours. Here and by previous authors – particularly Hehazy and Zhang (2005) – the question of how selecting the rescheduling window is discussed, in order to be efficient and not losing information. While rescheduling "every day" may seem more correct and safe, other considerations may induce selecting different window intervals, e.g. for taking into account "complete" activity influences and better considering the acceleration and slowing down effects of some activities.

4 Conclusions and future development

In case of project and contract claims, the computer-assisted methods can provide more efficient, transparent and rational mean to settle disputes than clumsy and difficult ways to reconstruct the work history from yard journal, records etc. The algorithmic method here developed can be the core of a more general approach for evaluating the schedule delays on field project applications. The implementation of the proposed model using examples form literature and other published cases is providing positive results. More difficult is to get access, or authorization for publication of real life applications, which often are related to legal cases and therefore are protected by privacy or difficult to disclose.

In principle *floats* is a resource that should be given due consideration in contractual arrangements. Among the relationships that the method of float banking may have with

other project management fields of interest, we recall the Critical Chain Method (CCM) where the concept of buffer management is introduced, see e.g. Leach (2000); in this regard the float accounting can be considered to improve the concept of buffer control introduced in CCM.

Once implemented, the system can better support ways to settle claims, arbitration and other procedures between claimant and defendant, resorting to court as the last chance.

Specifically, we are developing the method as a web-based tool that can be made available and demonstrated for gaining feed-back from prospective users. This project is nicknamed *ScheDA* (*Scheduling Delay Analysis*), which means in Italian language a recording or reporting sheet, based on some template or standard format. The same term originates from late Latin "*schedula*" or strip of paper, and later meaning a "note" or something to use as reference. Before *scheda* meant one of the strips forming a papyrus sheet, also literally in Greek "*skhida*" ($\sigma\chi\acute{\varepsilon}\delta\eta$), that is piece of wood, table, paper or small notebook.

Acknowledgements

We acknowledge Engineering Ingegneria Informatica Company and particularly Salvatore Di Rienzo, who supported a preliminary implementation of the method during the graduate internship of Giovanni Sacco.

References

Alkass S. *et al.*, 1996, "Construction delay analysis techniques", *J. Constr. Manag. Econ.*, Vol. 14, pp. 375–394.

Association for the Advancement of Cost Engineering International (AACEI), 2007, "Recommended Practice No. 29R-03, Forensic Schedule Analysis".

Braimah N., 2013, "Construction Delay Analysis Techniques", *Buildings*, Vol. 3, pp. 506–531.

Davison R.P., Mullen J., 2009, *Evaluating Contract Claims*, 2nd ed., Wiley-Blackwell.

Hegazy T., Zhang K., 2005, "Daily window delay analysis", *J. Constr. Eng. Manag.*, ASCE 2005, Vol. 131, pp. 505–512.

Keane P.J., Caletka A.F., 2015, *Delay Analysis in Construction Contracts*, 2nd ed., Wiley-Blackwell.

Leach L.P., 2000, *Critical Chain Project Management*, Artech House.

Pickavance K., 2010, *Delay and Disruption in Construction Contracts*, 4th ed., Sweet & Maxwell.

ScheDA (Schedule Delay Analysis), *www.pmscheda.it*.

Society of Construction Law (SCL), 2002, "Protocol for determining extensions of Time and Compensations for delay and disruption", SCL, Burbage, UK.

Yang J-B., Kao C-K, 2012, "Critical path effect based delay analysis method for contruction projects", *Int. Journal of Project Management*, Vol. 30, pp. 385–397.

Minimizing the total weighted completion time in single machine scheduling with non-renewable resource constraints

Péter Györgyi[1], Tamás Kis[1]

Institute for Computer Science and Control, Budapest, Hungary
gyorgyi.peter@sztaki.mta.hu, kis.tamas@sztaki.mta.hu

Keywords: single machine scheduling, non-renewable resources, total weighted completion time.

1 Introduction

In a machine scheduling problem with non-renewable resources, besides the machine(s), there are non-renewable resources, like raw materials, energy, or money, consumed by the jobs. The non-renewable resources have some initial stock, and they are replenished over time in given quantities. The objective function can be any of the widely-used optimization criteria in machine scheduling problems, see e.g., Carlier (1984) or Györgyi and Kis (2017).

Now, we consider a single machine variant with a single non-renewable resource. Formally, there is a single machine, a set of n jobs $\mathcal{J}$, and a non-renewable resource. Each job j has a processing time $p_j > 0$, a weight $w_j > 0$, and resource requirement $a_j \geq 0$. The non-renewable resource has an initial stock $\tilde{b}_1 \geq 0$ at time $u_1 = 0$, and it is replenished at $q-1$ distinct supply dates $0 < u_2 < \cdots < u_q$ in quantities $\tilde{b}_\ell \geq 0$ for $\ell = 2, \ldots, q$. However, the total demand does not exceed the total supply, i.e., $\sum_{j \in \mathcal{J}} a_j \leq \sum_{\ell=1}^{q} \tilde{b}_\ell$. The cumulative supply up to supply date u_ℓ is $b_\ell = \sum_{k=1}^{\ell} \tilde{b}_k$. A schedule specifies the starting time S_j of each job $j \in \mathcal{J}$; it is feasible if (i) no pair jobs overlap in time, i.e., $S_{j_1} + p_{j_1} \leq S_{j_2}$ or $S_{j_2} + p_{j_2} \leq S_{j_1}$ for each pair of distinct jobs j_1 and j_2, and (ii) for each time point t, the total supply until time t is not less than the total consumption of those jobs starting not later than t, i.e., if $u_\ell \leq t$ is the last supply date before t, then $\sum_{j \in J: S_j \leq t} a_j \leq b_\ell$.

An example problem along with a feasible schedule is depicted in Figure 1. There are 5 jobs represented by 5 rectangles. For each job j, the width of the corresponding rectangle indicates its processing time, while the resource requirement a_j is provided in the rectangle. Further on, there is an initial supply of $\tilde{b}_1 = 3$ at time $u_1 = 0$, and two more supplies at u_2 and u_3 with supplied quantities $\tilde{b}_2 = 4$ and $\tilde{b}_3 = 6$, respectively. In the depicted schedule, job j_1 cannot start earlier, since it requires 2 units from the resource, but there is only $\tilde{b}_1 + \tilde{b}_2 - a_2 - a_5 - a_3 = 1$ unit on stock before the supply arrives at u_3.

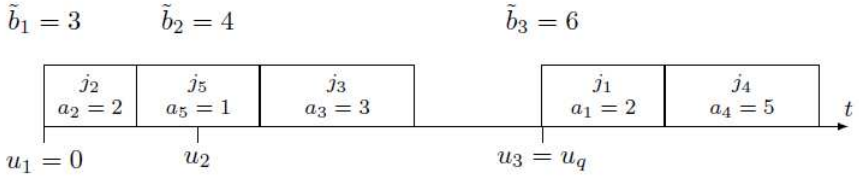


Fig. 1. A feasible schedule ($n = 5$, $q = 3$)

We aim at finding a feasible schedule S minimizing the total weighted completion time $\sum_{j\in\mathcal{J}} w_j C_j$, where $C_j := S_j + p_j$ denotes the completion time of job j. Using the standard $\alpha|\beta|\gamma$ notation, we denote our problem by $1|nr = 1|\sum w_j C_j$, where '$nr = 1$' indicates that we have only one type of non-renewable resource.

1.1 Previous results

The first results of the area are from the 1980s. Carlier (1984) presented several complexity results for variants where the makespan, the maximum lateness, or the total completion time have to be minimized in single and parallel machine environments. Slowinski (1984) examined a preemptive version of the problem for parallel machines. Toker *et. al.* (1991) and Xie (1997) applied reductions to the two-machine flow shop problem for variants where the supplies arrive uniformly over time. Grigoriev *et. al.* (2005) presented easy approximation algorithms for the makespan and the lateness objective. Gafarov *et. al.* (2011) proved several complexity results for various objective functions. Györgyi and Kis (2014) presented approximation schemes for the makespan objective in case of one resource. This was extended for a constant number of resources by Györgyi and Kis (2015b) and for parallel machines by Györgyi and Kis (2017) and by Györgyi (2017). Györgyi and Kis (2015a) proved reductions between the makespan minimization problem with two supply dates and variants of the Knapsack Problem. The most relevant antecedent of this research is Kis (2015), which considered the same objective function and presented an FPTAS for the problem with $q = 2$.

1.2 Preliminaries

This paper examines variants with more supplies, where we can state job independent connections among the processing times, the resource requirements and the weights. If these connections are strong enough we can find easy ordering rules that yield optimal schedules, see Table 1. In the next sections we deal with two other variants.

Table 1. Easy variants of $1|nr = 1|\sum w_j C_j$.

Variant	Optimal schedule
$p_j = a_j = \bar{a}$	non-increasing w_j order
$p_j = w_j = 1$	non-decreasing a_j order
$a_j = w_j = 1$	SPT order
$w_j = \bar{w},\ p_j = a_j$	SPT order
$a_j = \bar{a},\ p_j = w_j$	LPT order

Notice that SPT and LPT means that jobs are ordered in increasing, respectively, decreasing processing time order. In the corresponding algorithm, jobs are simply scheduled in increasing (SPT) / decreasing (LPT) processing time order. If the resource level is below the requirement of the next job, we simply wait until enough supply arrives.

While the SPT order gives the optimal schedule for the problem $1||\sum C_j$ (all job weights are 1), the LPT order is originally used in a list scheduling algorithm for the parallel machine problem $P||C_{\max}$ where it yields a 4/3-approximation algorithm.

2 The problem $1|nr = 1, p_j = a_j = w_j|\sum w_j C_j$

Surprisingly, this very restrictive case is already NP-hard:

Theorem 1. *The problem* $1|nr = 1, q = 2, p_j = a_j = w_j|\sum w_jC_j$ *is weakly NP-hard, and* $1|nr = 1, p_j = a_j = w_j|\sum w_jC_j$ *is strongly NP-hard.*

These complexity results are new, formerly only the NP-hardness of the variant $1|nr = 1, q = 2|\sum C_j$ (see Kis (2015)) and that of $1|nr = 1|\sum C_j$ (Carlier (1984), Kis (2015)) were known.

However, we could derive a 2-approximation algorithm for it.

Theorem 2. *Scheduling the jobs in LPT order is a 2-approximation algorithm for* $1|nr = 1, p_j = a_j = w_j|\sum w_jC_j$.

3 A PTAS for $1|nr = 1, p_j = w_j, q = const|\sum w_jC_j$

In this section we describe an PTAS (polynomial time approximation scheme) for $1|nr = 1, p_j = w_j, q = const|\sum w_jC_j$. Notice that the resource consumption of the jobs is job-dependent, but the number of supplies is a constant, not part of the input. A PTAS is a family of algorithms $\{A_\varepsilon\}_{\varepsilon>0}$, such that for each $\varepsilon > 0$, A_ε is an $(1+\varepsilon)$-approximation algorithm for the problem with a complexity polynomially bounded in the size of the input.

Let $P_{\text{sum}} := \sum_j p_j$ be the total processing time of the jobs. Let $\Delta := 1 + (\varepsilon/q^2)$. We will guess the total processing time of those jobs scheduled after u_ℓ for $\ell = 2, \ldots, q$, where a guess is a $q-1$ dimensional vector of non-increasing numbers $P_2^g, \ldots, P_q^g$, i.e., $P_\ell^g \geq P_{\ell+1}^g \geq 1$ for $\ell = 2, \ldots, q-1$, and each P_ℓ^g is of the form Δ^t for some integer $t \geq 0$ with $\Delta^t \leq P_{\text{sum}}$. Also fix $P_1^g := P_{\text{sum}}$. For any guess, define the set of *medium size jobs* $\mathcal{M}_\ell := \{j \mid p_j \geq (\Delta-1)P_\ell^g\}$. Note that $\mathcal{M}_q \supseteq \mathcal{M}_{q-1} \supseteq \cdots \supseteq \mathcal{M}_1$, since $P_q^g \leq P_{q-1}^g \leq \cdots \leq P_1^g$. Let $\mathcal{S}_\ell$ be the complement of $\mathcal{M}_\ell$, i.e., $\mathcal{S}_\ell := \{j \mid p_j < (\Delta-1)P_\ell^g\}$. Clearly, $\mathcal{S}_q \subseteq \mathcal{S}_{q-1} \subseteq \cdots \subseteq \mathcal{S}_1$. After these preliminaries, the PTAS for $1|nr = 1, p_j = w_j, q = const|\sum w_jC_j$ consists of the following steps:

1. Consider each possible guess $(P_2^g, \ldots, P_q^g)$ of the total processing time of those jobs starting after the supply dates $u_2, \ldots, u_q$, respectively. For each possible guess define the sets of jobs $\mathcal{M}_\ell$ and $\mathcal{S}_\ell$ (see above), and perform the steps 2-5. After processing all the guesses, go to Step 6.
2. For each $\ell = 1, \ldots, q$, choose at most $1/(\Delta-1)$ medium size jobs from $\mathcal{M}_\ell$ (since the sets $\mathcal{M}_\ell$ are not disjoint, care must be taken to choose each job at most once). For each possible choice $(T_1, \ldots, T_q)$ of the medium size jobs (where $T_\ell \subseteq \mathcal{M}_\ell$), perform steps 3-5. After evaluating all choices, continue with the next guess in Step 1.
3. Determine a schedule of the medium jobs. That is, for $\ell = q, \ldots, 2$, schedule the jobs in T_ℓ in any order after u_ℓ contiguously, and if necessary, push to the right the jobs in $\bigcup_{\ell'=\ell+1}^{q} T_{\ell'}$.
4. Let $\mathcal{J}_0^u$ be the set of unscheduled jobs. For $\ell = q, q-1, \ldots, 1$, repeat the following. In a general step with $\ell \geq 2$, pick jobs from $\mathcal{J}_{q-\ell}^u \cap \mathcal{S}_\ell$ in non-increasing a_j/p_j order until the selected subset K_ℓ satisfies $p(K_\ell) + p(T_\ell) \geq P_\ell^g - (1/\Delta)P_{\ell+1}^g$, or if no more jobs left, $K_\ell = \mathcal{J}_{q-\ell}^u \cap \mathcal{S}_\ell$. In either case, insert the jobs of K_ℓ in any order after u_ℓ and after all the jobs in $T_1 \cup \cdots \cup T_{\ell-1}$, and before all the jobs in $T_\ell \cup \bigcup_{\ell'=\ell+1}^{q}(K_{\ell'} \cup T_{\ell'})$ (pushing some of them to the right if necessary). Let $\mathcal{J}_{q-\ell+1}^u := \mathcal{J}_{q-\ell}^u \setminus K_\ell$ and continue with $\ell - 1$ until $\ell = 1$ or no more unscheduled jobs are left. For $\ell = 1$ just schedule all the remaining jobs from time $u_1 = 0$ on (pushing the already scheduled jobs to the right, if necessary). If the complete schedule obtained satisfies the resource constraints, then continue with Step 5, otherwise with the next choice of medium size jobs in Step 2.
5. Compute the objective function value of the complete schedule obtained in step (4), and store this schedule as the best schedule if it is the first feasible schedule or if it is better than the best feasible schedule found so far. Continue with next choice of medium size jobs in Step 2.

6. Output the best schedule found in the previous steps.

Theorem 3. *The proposed algorithm is an PTAS for* $1|nr = 1, p_j = w_j, q = const| \sum w_j C_j$.

Acknowledgements

This work has been supported by the National Research, Development and Innovation Office - NKFIH grant K112881, and by the GINOP-2.3.2-15-2016-00002 grant of the Ministry of National Economy of Hungary.

References

Carlier J., 1984, "Problèmes d'ordonnancement à contraintes de ressources: algorithmes et complexité", *Université Paris VI-Pierre et Marie Curie, Institut de programmation*

Gafarov E.R., A.A. Lazarev and F. Werner, 2011, "Single machine scheduling problems with financial resource constraints: Some complexity results and properties", *Math. Social Sci.*, Vol. 62, pp. 7-13.

Grigoriev A., M. Holthuijsen and J. van de Klundert, 2005, "Basic scheduling problems with raw material constraints", *Naval Res. Logist.*, Vol. 52, pp. 527-553.

Györgyi P., 2017, "A PTAS for a resource scheduling problem with arbitrary number of parallel machines", *Oper. Res. Lett.*, Vol. 45, pp. 604-609.

Györgyi P., T. Kis, 2014, "Approximation schemes for single machine scheduling with non-renewable resource constraints", *J. Sched.*, Vol. 17, pp. 135-144.

Györgyi P., T. Kis, 2015a, "Reductions between scheduling problems with non-renewable resources and knapsack problems", *Theoret. Comput. Sci.*, Vol. 565, pp. 63-76.

Györgyi P., T. Kis, 2015b, "Approximability of scheduling problems with resource consuming jobs", *Ann. Oper. Res.*, Vol. 235, pp. 319-336.

Györgyi P., T. Kis, 2017, "Approximation schemes for parallel machine scheduling with non-renewable resources", *European J. Oper. Res.*, Vol. 258, pp. 113-123.

Kis T., 2015, "Approximability of total weighted completion time with resource consuming jobs", *Oper. Res. Lett.*, Vol. 43, pp. 595-598.

Slowinski R., 1984, "Preemptive scheduling of independent jobs on parallel machines subject to financial constraints", *European J. Oper. Res.*, Vol. 15, pp. 366-373.

Toker A., S. Kondakci and N. Erkip, 1991, "Scheduling under a non-renewable resource constraint", *J. Oper. Res. Soc.*, Vol. 42, pp. 811-814.

Xie J., 1997, "Polynomial algorithms for single machine scheduling problems with financial constraints", *Oper. Res. Lett.*, Vol. 21, pp. 39-42.

The Cyclic Job Shop Problem with uncertain processing times

Idir Hamaz[1], Laurent Houssin[1] and Sonia Cafieri[2]

[1] LAAS-CNRS, Universite de Toulouse, CNRS, UPS, Toulouse, France
{ihamaz, lhoussin}@laas.fr
[2] ENAC, Universite de Toulouse, F-31055 Toulouse, France
sonia.cafieri@enac.fr

Keywords: Cyclic scheduling, budgeted uncertainty set, robust optimization.

1 Introduction

Most models for scheduling problems assume deterministic parameters. In contrast, real world scheduling problems are often subject to many sources of uncertainty, for example activities duration can decrease or increase, machines can break down, new activities can be incorporated, *etc.* In this paper, we focus on scheduling problems that are cyclic and where activity durations are affected by uncertainty. Indeed, the best solution for a deterministic problem can quickly become the worst one in the presence of uncertainties.

In this paper, we consider the *Cyclic Job Shop Problem* (CJSP) where processing times are affected by uncertainty. Several studies were conducted on the deterministic CJSP. The CJSP with identical parts is studied in (Roundy, R. 1992). The author shows that the problem is NP-hard and designs a branch and bound algorithm to solve the problem. Hanen (1994) investigates the general CJSP and presents a branch and bound procedure to tackle the problem. A general framework for modeling and solving cyclic scheduling problems is presented in (Brucker, P. and Kampmeyer, T. 2008). The authors present different models for cyclic versions of the job shop problem. However, a few works consider cyclic scheduling problems under uncertainty. Che, A. *et. al.* (2015) investigate the cyclic hoist scheduling problem with processing time window constraints where the hoist transportation times are uncertain. The authors define a robustness measure for cyclic hoist schedule and a bi-objective mixed integer linear program to optimize the cycle time and the robustness.

In order to deal with uncertainty, we use a robust optimization approach. We model the uncertain parameters by using the idea of uncertainty set proposed by Bertsimas and Sim (2004). Each task duration belongs to an interval, and the number of parameters that can deviate from their nominal values is bounded by a positive number called *budget of uncertainty*. This parameter allows us to control the degree of conservatism of the resulting schedule. Finally, we propose a branch and bound procedure that computes the minimum cycle time for the robust CJSP such that, for each scenario in the uncertainty set, there exists a feasible cyclic schedule.

2 Problems description

2.1 Basic Cyclic Scheduling Problem (BCSP)

We are given a set of n generic operations $\mathcal{T} = \{1, ..., n\}$. Each operation $i \in \mathcal{T}$ is characterized by a non-negative processing time p_i and has to be performed infinitely often without preemption. We denote $< i, k >$ the k^{th} occurrence of the generic operation i and $t(i, k)$ the starting time of k^{th} occurrence of the operation i.

The operations are subjected to a set of *precedence constraints* (uniform constraints). The constraints between the occurrences $< i, k >$ and $< j, k + H_{ij} >$ are given by

$$t(i,k) + p_i \leqslant t(j, k + H_{ij}), \quad \forall\, i \in \mathcal{T},\ \forall\, k \geq 1 \tag{1}$$

where H_{ij} is an integer that represents the depth of the occurrence shift, usually referred to as *height*. The H_{ij} parameter is an occurrence shift between the operations i and j. For instance, for each execution of the occurrence $< i, k >$, the next execution of j is the occurrence $< j, k + H_{ij} >$.

A schedule S is an assignment of starting time $t(i, k)$ for each occurrence $< i, k >$ of task $i \in \mathcal{T}$. Such schedule is called *periodic* with cycle time α if it satisfies

$$t(i,k) = t(i,0) + \alpha k, \quad \forall\, i \in \mathcal{T},\ \forall\, k \geq 1 \tag{2}$$

where α is the cycle time and represents the difference between the stating times of two successive occurrences of the same task.

Therefore, a schedule S can be entirely defined by the staring times $t_i = t(i, 0)$ of the first occurrences and the cycle time.

In this study, the objective is to minimize the cycle time α while satisfying the precedence constraints between operations. Notice that different objective functions exist for cyclic scheduling problems, such as work in progress minimization or both cycle time and work in progress minimization.

A bi-valued directed graph $G = (\mathcal{T}, U)$ can be associated with any instance of BCSP. In this graph, a node (resp. an arc) of G corresponds to a generic operation (resp. constraints) in the BCSP. Each arc (i, j) of G has two valuations, the length $L_{ij} = p_i$ and the height H_{ij}. These arcs are called uniform arcs and are built by considering the precedence constraints. For instance, a precedence constraint between task i and task j leads to an arc (i, j) of G labeled with $L_{ij} = p_i$ and H_{ij}. We denote $H(c)$ (resp. $L(c)$) the height (resp. length) of a circuit c in graph G the sum of heights (resp. lengths) of the arcs composing the circuit c.

The minimum cycle time is given by the maximum circuit ratio of the graph which is defined by

$$\alpha = \max_{c \in \mathcal{C}} \frac{\sum_{(i,j) \in c} L_{ij}}{\sum_{(i,j) \in c} H_{ij}}$$

where $\mathcal{C}$ is the set of all circuits in G.

We call *critical circuit* the circuit c realizing the maximum circuit ratio. Several algorithms have been proposed for the computation of critical circuits. An experimental study about maximum circuit ratio algorithms was published in (Dasdan, A. 2004). The author remarks that, among the several tested algorithms, the most efficient one is the Howard's algorithm. Although the algorithm has a pseudo-polynomial complexity, it shows noteworthy practical results.

Once the cycle time is determined, the starting times $(t_i)_{i \in \mathcal{T}}$ can be determined by computing the longest path in the graph G where each arc $(i, j) \in U$ is valued with $p_i - \alpha H_{ij}$.

2.2 Cyclic Job Shop Problem (CJSP)

In the present work, we focus on the cyclic job shop problem (CJSP). The difference with the problem defined above is that for CJSP the number of machines is lower than the number of tasks to perform. As a result, the same resource must be shared between different operations. A CJSP can be considered as a BCSP equipped with resource constraints.

Each occurrence of an operation $i \in \mathcal{T}$ has to be executed, without preemption, on the machine $M_{(i)} \in \mathcal{M} = \{1, ..., m\}$. Operations are grouped on a set of jobs $\mathcal{J}$, where a job j represents a sequence of elementary operations that must be executed in order. To avoid overlapping between the tasks executed on the same machine, for each pair of operations i and j where $M_{(i)} = M_{(j)}$, the following *disjunctive constraint* holds

$$\forall\, i, j \ s.t.\ M_{(i)} = M_{(j)},\ \forall k, l \in \mathbb{N} : t(i,k) \leq t(j,l) \Rightarrow t(i,k) + p_i \leq t(j,l) \tag{3}$$

In summary, a cyclic job shop problem is defined by

- a set $\mathcal{T}$ of elementary tasks,
- a set $\mathcal{M}$ of machines,
- for each task $i \in \mathcal{T}$, a processing time p_i and a machine $M_{(i)} \in \mathcal{M}$ on which the task has to be performed,
- a set $\mathcal{P}$ of precedence constraints,
- a set $\mathcal{D}$ of disjunctive constraints that occur when two tasks are mapped on the same machine,
- a set $\mathcal{J}$ of jobs corresponding to a production sequence of generic operations. More precisely, a job J_1 defines a sequence $J_1 = t_{1,1} \dots t_{1,k}$ to be executed in that order.

The CJSP can be represented by directed graph $G = (V, \mathcal{P} \cup \mathcal{D})$, called *disjunctive graph*. The sequence of operations that belongs to the same job are linked by uniform arcs in $\mathcal{P}$ where the heights are equal to 0. Additionally, for each pair of generic operations i and j executed on the same machine, a disjunctive pair of arcs (i, j) and (j, i) occurs. These arcs are labeled respectively with $L_{ij} = p_i$ and $H_{ij} = K_{ij}$, and $L_{ji} = p_j$ and $H_{ji} = K_{ji}$ where K_{ij} is an occurrence shift variable that satisfies $K_{ij} + K_{ji} = 1$ (Hanen C 1994).

The following bounds on occurrence shift variables K_{ij} have been proposed in (Hanen C 1994):

$$K_{ij}^{-} \leq K_{ij} \leq 1 - K_{ij}^{-}. \tag{4}$$

with

$$K_{ij}^{-} = 1 - min\{H(\mu) \,|\, \mu \ from\ j\ to\ i\ in\ G\}. \tag{5}$$

A schedule is an assignment of all the occurrence shifts, i.e., determine precedence relations on the operation occurrences mapped to the same machine. Note that once the occurrence shifts are determined the problem is equivalent to the BCSP, therefore, the minimum cycle time can be obtained by the cited algorithms.

Previous studies have shown that the problem is NP-Hard (Hanen C 1994) for cycle time minimization.

2.3 Robust Cyclic Job Shop Problem (RCJSP)

In this paper, we investigate the robust version of the CJSP. More precisely, we are interested in the CJSP where processing times are affected by uncertainty and belong to a finite uncertainty set $\mathcal{U}$. Based on the *budget of uncertainty* concept introduced in (Bertsimas, D. and Sim, M. 2004), the processing time deviations can be modeled trough the following uncertainty set:

$$\mathcal{U}^{\Gamma} = \left\{ (p_i)_{i \in \mathcal{T}} \in \mathbb{R}^n : p_i = \bar{p}_i + \hat{p}_i \xi_i,\ \forall\, i \in \mathcal{T};\ \xi_i \in \{0,1\};\ \sum_{i \in \mathcal{T}} \xi_i \leq \Gamma \right\}$$

where $\bar{p}_i$ represents the nominal processing time of operation i and $\hat{p}_i$ its deviation. The parameter Γ is a positive integer and represents an upper bound on the number of processing times deviating from their nominal value.

The objective of the problem is to find, for a given budget of uncertainty Γ, the minimum cycle time such that, for each $p \in \mathcal{U}^{\Gamma}$, there exists a vector $(t(p)_i)_{i\in\mathcal{T}}$ satisfying both the precedence and disjunctive constraints.

3 Branch and bound procedure for the RCJSP

Recently, an Howard's algorithm adaptation taking into account the uncertainty set $\mathcal{U}^{\Gamma}$ has been presented in (Hamaz, I. *et. al.* 2017). The computational experiments on the algorithm show small execution times for robust BCSP instances.

To take into account the uncertainty on the processing times for the RCJSP, we develop a branch and bound procedure that uses the robust version of the Howard's algorithm. The procedure starts by initializing the upper bound on the cycle time to $\sum_{i\in\mathcal{T}} p_i + p_f$ where p_f is the sum of the first Γ greatest deviations and the lower bound to the optimal cycle time of $G = (\mathcal{T}, U)$ computed by the Howard's algorithm adaptation.

We use the same branching scheme as in (Fink, M. *et. al.* 2012). The search tree is initialized with a node (the root) where the graph $G = (\mathcal{T}, U)$ contains only the uniform arcs U and no fixed disjunctions. Then, the branching is performed on unfixed disjunctions K_{ij}. For this purpose, a successor node is created for each value on the interval $[K_{ij}^{-}, 1-K_{ij}^{-}]$. The value of the node is then computed by running the robust version of the Howard's algorithm with $G = (\mathcal{T}, U^{'} \cup \{(i,j),(j,i)\})$, where $U^{'}$ contains the uniform arcs and a precedent fixed disjunctive arcs. When all the occurrence shifts are fixed, a feasible schedule is obtained, then the upper bound can be updated.

Preliminary numerical results show that the branch and bound procedure (implemented in C++ and executed on an Intel Xeon E5-2695 processor running at 2.30GHz CPU) delivers promising results. Besides, the algorithm is insensitive regarding the value of the budget of uncertainty.

Once the optimal cycle time computed by the branch and bound procedure, a periodic schedule $S^{\Gamma} = (\alpha, ((t(p)_i)_{i\in\mathcal{T}})$ can be determined for each $p \in \mathcal{U}^{\Gamma}$.

4 Conclusion

The RCJSP with budgeted uncertainty set is addressed in this paper. We present a branch and bound procedure that uses a Howard's algorithm adaptation. Further investigation will address dominance rules to speed up the branch and bound procedure.

References

Bertsimas, D., Sim, M., 2004, "The price of robustness", *Operations research*, Vol. 52(1), pp. 35-53.

Brucker, P., Kampmeyer, T., 2008, "A general model for cyclic machine scheduling problems", *Discrete Applied Mathematics*, Vol. 156(13), pp. 2561-2572.

Che, A., Feng, J., Chen, H., and Chu, C., 2015, "Robust optimization for the cyclic hoist scheduling problem", *European Journal of Operational Research*, Vol. 240(3), pp. 627-636.

Dasdan, A., 2004, "Experimental analysis of the fastest optimum cycle ratio and mean algorithms", *ACM Transactions on Design Automation of Electronic Systems*, Vol. 9(4), pp. 385-418.

Fink, M., Rahhou, T. B. and Houssin, L., 2012, "A new procedure for the cyclic job shop problem", *IFAC Proceedings Volumes*, Vol. 45(6), pp. 69-74.

Hamaz, I., Houssin, L. and Cafieri, S., 2017, "Robust Basic Cyclic Scheduling Problem", *Manuscript submitted for publication.*

Hanen, C., 1994, "Study of a NP-hard cyclic scheduling problem: The recurrent job-shop", *European journal of operational research*, Vol. 72(1), pp. 82-101.

Roundy, R., 1992, "Cyclic schedules for job shops with identical jobs", *Mathematics of operations research*, Vol. 17(4), pp. 842-865.

Modeling techniques for the *e*S-graph

Hegyháti M.

Department of IT, Széchenyi István University, Hungary
hegyhati@sze.hu

Keywords: scheduling, modeling, *e*S-graph.

1 Motivation

The scheduling of batch processes has been addressed with a variety of techniques in the literature: Mixed-Integer Linear Programming models (Floudas and Lin, 2004), S-graph Framework (Sanmartí *et al.*, 2002), Linear-Priced Timed Automata (LPTA, Panek *et al.*, 2008), just to name the most frequent ones. Many of the approaches use an internal representation of the scheduling problem, e.g. a State Task Network (STN, Kondili *et al.*, 1993), Resource Task Network (RTN, Pantelides, 1993), State Sequence Network (SSN, Majozi and Zhu, 2001), that is used as a basis for formulating the mathematical model for the optimization algorithms. An important advantage of the S-graph framework is that the initial representation and the mathematical model for the algorithms are the same, the so-called S-graph which is a directed acyclic graph. This transparency makes the approach easier to understand, and modeling issues (Hegyháti *et al.*, 2009) are easily avoided. This simple model, however, carries limitations too. To apply it for industrial examples, minor extensions or alterations are needed in some cases. The recently proposed *e*S-graph model (Hegyháti, 2014) aims to extend its modeling power to avoid the need for further extensions, while keeping the simplicity and transparency of the S-graph. In this work, the modeling power of the *e*S-graph framework is presented through several industrial examples.

2 Short introduction of the S-graph framework

The S-graph framework was originally proposed for the short-term scheduling of chemical batch processes. The framework is based on two major components:

- The mathematical model, the S-graph
- The branch-and-bound algorithm to find the optimal schedule

The S-graph model is a directed graph with weighted arcs. The nodes represent tasks and products, which are illustrated by circles in the graphical representation. The circles have labels attached that are the name of the product for the product nodes, and the name of the task above the name of the suitable unit in case of task nodes. These nodes are connected with weighted arcs, which express the production order of the tasks belonging to the same product. Moreover, the weight of an arc is a lower bound on the necessary timing difference between the two connected nodes. This graph, called recipe-graph, is extended by the algorithm with arcs representing scheduling decisions. The algorithms report the S-graph model of the optimal solution that can be easily and unambiguously converted to a common graphical representation, such as a Gantt-chart. An example S-graph is shown in Figure 1 with 9 tasks and 2 products. The scheduling decisions are expressed by the gray arcs, e.g., the execution of $t9$ can not start earlier than the that of $t4$. The arcs representing scheduling decisions are zero-weighted by default, which is often omitted in the graphical representation as well.

Since its introduction, the S-graph framework has been applied to numerous case studies which often required small extensions, alterations of the model to fit and address the problem-specific constraints. As an example, solving wet-etch scheduling problems required an extension of the original S-graph model to implement zero-wait policy (Hegyháti *et al.*, 2014), or a new algorithm has been proposed to solve problems with the objective of throughput maximization (Majozi and Friedler, 2006). While the individual difficulty of these separate extensions vary, keeping them compatible is a challenging task on both theoretical and software implementation levels. To overcome this issue, the *e*S-graph framework was proposed.

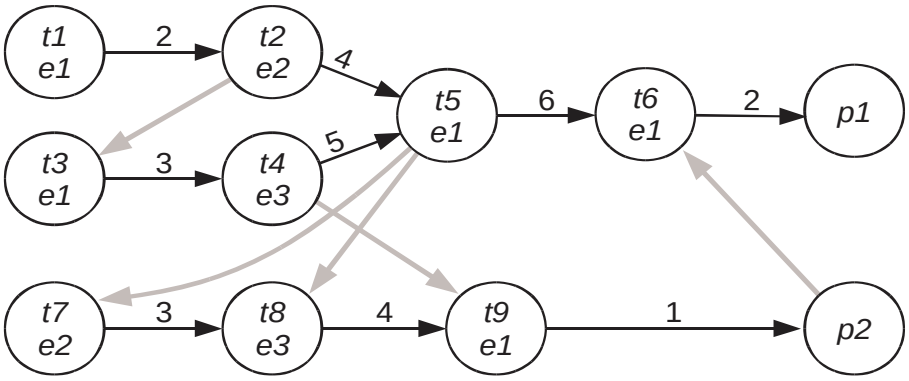


Fig. 1. S-graph representing a schedule for two products.

3 The *e*S-graph model

The *e*S-graph model can be seen as a generalization of the S-graph model. In the S-graph framework, there is a one-to-one relation between the nodes of the graph and the tasks that have to be carried out using the available resources. The *e*S-graph model relaxes this connection, and has the following basic building blocks:

- **Events** – are the atomic building blocks, that represent a single transition of the states of the system. In the model, events are represented with nodes.
- **Subprocesses** – are the generalizations of tasks. Any activity that spans over some events and requires the presence of some resources continuously can be considered as a subprocess. Formally, any subset of the events can be a subprocess. Subprocesses may also overlap each other.

Each subprocess may be carried out by a set of resources simultaneously, however, several of such sets may exist, and it may influences the weights of the arcs within the subprocess.

The original S-graph model can be seen as a special class of *e*S-graphs, where

1. The events considered are the starting of the tasks and removal of the products from the last processing step.
2. Each subprocess spans over the event representing the start of a task, and its out-neighbors in the recipe graph.
3. The resource sets of each subprocess are singletons.

The generalized definition, however, enables implementation of a wider set of practical considerations without the need for further extensions to the model or the algorithms. The *e*S-graph model can be solved to optimality by a slightly modified version of the original S-graph algorithms, or with a precedence based MILP model.

4 An example *e*S-graph model

To illustrate the expressive power of the *e*S-graph framework, parts of the model for scheduling job cells with automated guided vehicles is presented here. In this practical study, there are several workstations where the jobs must be processed. The intermediates between the stations are transported via automated guided vehicles (AGVs) (Zeng *et al.*, 2014).

The problem entails several specific constraints which cannot be modeled in a straightforward way in the S-graph framework, or the STN, RTN, or SSN representations. Two such constraints are selected here, and used as an illustration for the simple modeling techniques with the *e*S-graph model:

1. For each transportation, an arbitrary AGV is needed, which traverses through specific line segments. While the AGV is transporting something, no other AGV can use the same line segments. However, each station has its own loading area, thus, the path segments become free when the AGV arrives to its destination.
2. Loading and unloading of intermediates takes a specific time, for which both the station and the assigned AGV is required. It is not mandatory however, that the same AGV is used for subsequent transportations of the same product.

Part of the proposed model is presented in Figure 2., where the events and subprocesses are the following:

$e1$: job leaving machine $m1$.
$e2$: job arriving to machine $m2$.
$e3$: job starting to be processed on machine $m2$.
$e4$: job finishing to be processed on machine $m2$.
$e5$: job leaving machine $m2$.
$e6$: job arriving to machine $m3$.
$sp1$: transportation processes between machines $m1$ and $m2$.
$sp2$: transportation processes between machines $m2$ and $m3$.
$sp3$: traversing between machines $m1$ and $m2$.
$sp4$: traversing between machines $m2$ and $m3$.
$sp5$: manufacturing step of the job on machine $m2$.

As both events $e2$ and $e3$ are covered by both $sp1$ and $sp5$, both the AGV assigned to the transportation and the machine $m2$ are needed for the unloading of the intermediate, as required by the second statement above. $sp2$ however, is a completely different subprocess from $sp1$, thus, a different AGV may be selected for it.

As for the first statement, the subprocess $sp3$ is a subset of $sp1$, and requires the segments of one of the suitable paths between machines $m1$ and $m2$. While the AGV is moving between the machines, the segments are unavailable for other AGVs, however, they become free when it arrives to the loading area of $m2$. The same holds for $sp4$ and $sp2$, however, there is only one possible path between $m2$ and $m3$.

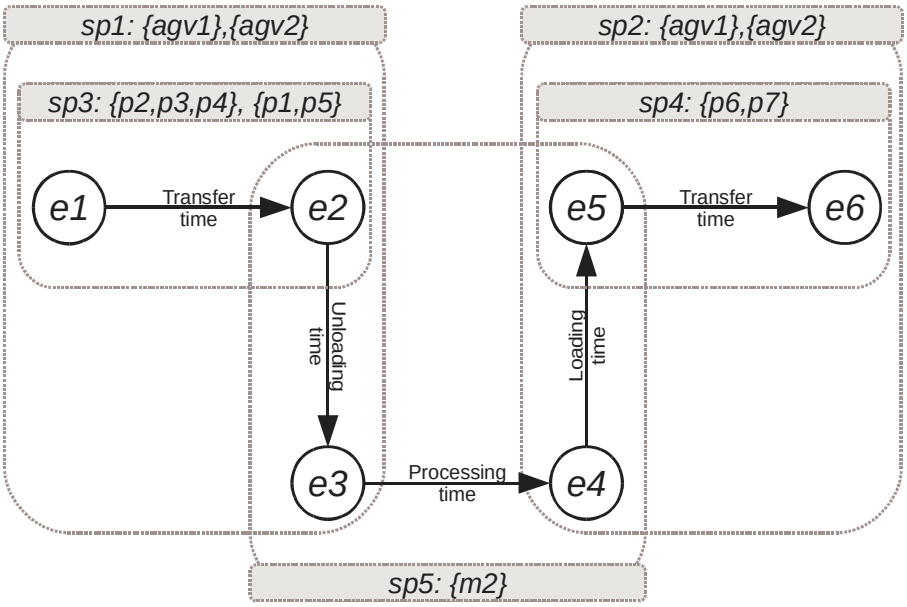


Fig. 2. Part of an *e*S-graph for the illustrative example.

5 Results

The modeling power of the *e*S-graph framework was examined on several industrial case studies from the literature. The strong and weak points of this modeling technique were identified, and the modified S-graph algorithms were compared with a proposed precedence based MILP model on these examples.

6 Conclusions

The *e*S-graph model allows to model a much wider set of practical constraints arising in industrial scheduling problems without the need for any extension on the model or the applied algorithms. There are limitations to this model as well, however, it holds a great potential. *e*S-graph can be seen as a middle ground between problem-specific scheduling models, such as the S-graph, and the very general models like MILP and LPTA. While *e*S-graphs are still scheduling specific, they are general enough so that the algorithms working on them can be developed independently from the problems that are modeled with it. From each acceleration, however, all of the modeled scheduling problem classes can benefit.

7 Acknowledgements

This research was supported by the National Research, Development and Innovation Office (NKFIH) K108405 and by the EFOP-3.6.2-16-2017-00015 “HU-MATHS-IN; Intensification of the activity of the Hungarian Industrial Innovation Service Network” grant. Supported by the ÚNKP-17-4 New National Excellence Program of the Ministry of Human Capacities.

References

Floudas C. A., X. Lin, 2004, "Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review", *Computers & Chemical Engineering*, Vol. 28, pp. 2109–2129.

Hegyháti M., 2014, "A combinatorial modeling tool for event-based batch process scheduling: the eS-graph", presented at ASCONIKK & VOCAL 2014, Veszprem, Hungary.

Hegyháti M., T. Majozi, T. Holczinger, F. Friedler, 2009, "Practical infeasibility of cross-transfer in batch plants with complex recipes: S-graph vs MILP methods", *Chemical Engineering Science*, Vol. 64, pp. 605–610.

Hegyháti M., O. Osz, B. Kovács, F. Friedler, 2014, "Scheduling of automated wet-etch stations", *Chemical Engineering Transactions*, Vol. 39, pp. 433–438.

Kondili E., C. Pantelides, R. Sargent, 1993, "A general algorithm for short-term scheduling of batch operations–I. MILP formulation", *Computers & Chemical Engineering*, Vol. 17, pp. 211–227.

Majozi T, F. Friedler, 2006, "Maximization of throughput in a multipurpose batch plant under fixed time horizon: S-graph approach", *Industrial & Engineering Chemistry Research*, Vol. 45, pp. 6713–6720.

Majozi T, X. X. Zhu, 2001, "A novel continuous-time MILP formulation for multipurpose batch plants. 1. Short-term scheduling", *Industrial & Engineering Chemistry Research*, Vol. 40(25), pp. 5935–5949.

Panek S., S. Engell, S. Subbiah, O. Stursberg, 2008, "Scheduling of multi-product batch plants based upon timed automata models", *Computers & Chemical Engineering*, Process Systems Engineering: Contributions on the State-of-the-Art - Selected extended Papers from ESCAPE '16/PSE 2006., Vol. 32, pp. 275–291.

Pantelides C., 1993, "Unified frameworks for optimal process planning and scheduling", *Proceedings of the second international conference on foundations of computer-aided process operations*, pp. 253–274.

Sanmarti E., T. Holczinger, L. Puigjaner, F. Friedler, 2002, "Combinatorial framework for effective scheduling of multipurpose batch plants", *AIChE Journal*, Vol. 48(11), pp. 2557–2570.

Zeng C., J. Tang, C. Yan, 2014, "Scheduling of no buffer job shop cells with blocking constraints and automated guided vehicles", *Applied Soft Computing*, Vol. 24, pp. 1033–1046.

Scheduling Multiple Flexible Projects with Different Variants of Genetic Algorithms

Luise-Sophie Hoffmann and Carolin Kellenbrink

Institute of Production Management, Leibniz Universität Hannover, Germany
luise-sophie.hoffmann, carolin.kellenbrink@prod.uni-hannover.de

Keywords: Multi-Project Scheduling, RCMPSP-PS, Flexible Project, Genetic Algorithm.

1 Introduction

In Kellenbrink and Helber (2015), the resource-constrained project scheduling problem with a flexible project structure (RCPSP-PS) is introduced. For such flexible projects, the project structure is not known in advance. Instead, it depends on model-endogenous decisions in which exactly one activity out of a decision set has to be selected. Those activities can trigger further decisions or cause activities. Therefore, in addition to scheduling the activities, the project structure has to be chosen.

The regeneration of complex capital goods, like aircraft engines, is an example for such a flexible project due to different technical repair options. The regeneration is usually conducted by an external service provider. These service providers mostly handle several different projects at the same time which results in a problem setting similar to the scheduling of multiple projects, cf., e.g., Pritsker *et al.* (1969). Therefore, the RCPSP-PS is extended to the resource-constrained multi-project scheduling problem with flexible project structures (RCMPSP-PS). Furthermore, different variants of genetic algorithms regarding the representation are presented and evaluated.

2 Problem Setting

A flexible project $l \in \mathcal{L}$ comprises a set of different activities. These activities can be divided into sets of mandatory activities $j \in \mathcal{V}_l$ and optional activities. In each decision $e \in \mathcal{E}_l$ triggered by an activity $a(l,e)$, exactly one of the optional activities $j \in \mathcal{W}_{le}$ has to be chosen for implementation. A decision is triggered, if the triggering activity $a(l,e)$ is implemented. While a mandatory decision is assumed to be triggered by the start-dummy job, a non-mandatory decision is triggered by an optional job. Additionally, the decision for the implementation of an optional activity $j \in \mathcal{W}_{le}$ may cause the implementation of further activities $i \in \mathcal{B}_{lj}$.

Due to the different possible project structures, not only renewable resources $r \in \mathcal{R}$ but also nonrenewable resources $n \in \mathcal{N}$ have to be considered. This may lead to infeasible combinations of project structures. Furthermore, we consider specific due date δ_l for all projects. In case this due date is not met, specific delay costs c_l for each delayed period occur. Overall, the aim is to minimize the total delay cost.

Figure 1 shows an example for the given problem setting with two different flexible projects I and II. These projects consist of eight non-dummy activities each. The project's decisions and caused activities, the resource consumption k_{ljr} and k_{ljn}, as well as the duration d_{lj} of the activities are given in the figure. While scheduling those two projects, the capacities of one renewable and one nonrenewable resource are considered.

Project I contains two decisions. The first decision is mandatory and thus triggered by the start-dummy activity I-1. Hence, the decision between the implementation of activity I-4 and of activity I-5 has to be made. If activity I-4 is selected in the first decision, it

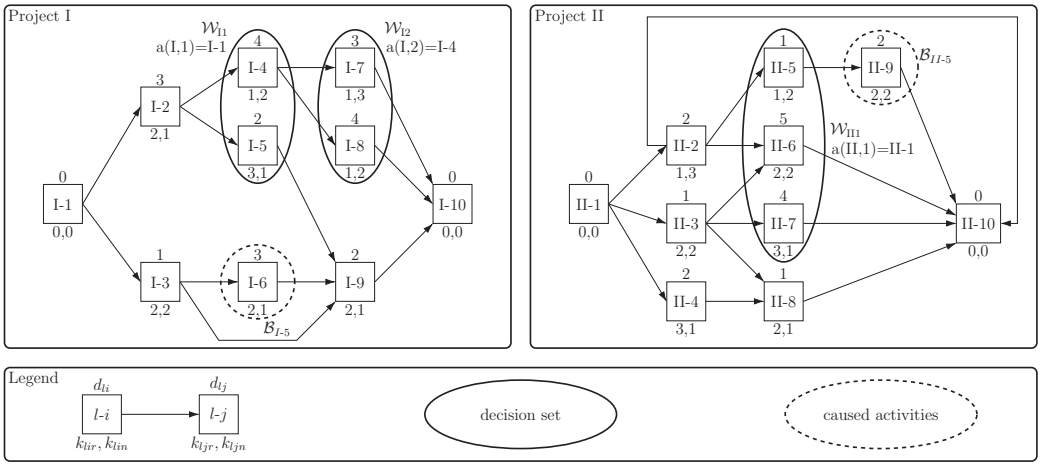


Fig. 1. Example of multiple flexible projects

triggers the second decision on activities I-7 and I-8. In case activity I-5 is selected in the first decision, activity I-6 is caused. For scheduling project I, the capacity of the renewable resource is $K_{Ir} = 2$ and the capacity of the nonrenewable resource is $K_{In} = 9$. With the given capacity of the nonrenewable resource, all three possible project structures are feasible. The due date for project I is at the end of period seven. The delay costs for each period equal seven units.

Project II has only one mandatory decision on the three activities II-5, II-6 and II-7. While activity II-6 and II-7 neither trigger a decision nor cause an activity, activity II-5 causes activity II-9. The resource capacities are $K_{IIr} = 3$ and $K_{IIn} = 10$. The capacity of the nonrenewable resource leads to an infeasible solution in case activity II-5, which causes activity II-9, is chosen. The due date of project II is at the end of period eight with delay costs of three units per period.

When scheduling both projects separately, in project I the optional activities I-4 and I-7 are implemented and we get a six period delay resulting in total delay cost of $6 \cdot 7 = 42$ units. The optimal schedule of project II shows no delay. In this project the optional activity II-7 is selected and scheduled.

For scheduling both projects simultaneously, we assume capacities of $K_r = 2 + 3 = 5$ and $K_n = 9 + 10 = 19$. In the optimal schedule, project I is finished without a delay but project II gets a one period delay. This leads to total delay cost of three units, which is lower than the result of 42 units for considering separate schedules. Furthermore, the selected project structures of both projects have changed. In project I activities I-5 and I-6 are implemented. In project II activities II-5 and II-6 are scheduled. This shows that for multiple flexible projects not only the scheduling but also the chosen project structures influence each other.

3 Genetic Algorithms

Many approaches for scheduling multiple projects make use of priority rules, cf. Browning and Yassine (2010) for an overview. Therefore, in Hoffmann *et al.* (2017) different priority rules to solve the RCMPSP-PS were evaluated. However, the numerical results have not been overly satisfying. Presumably, the interaction of the differently prioritized

projects, the project structures and the scheduling is too complex to be represented by a priority rule. Therefore, the use of a genetic algorithm seems to be more promising. In the following, we sketch different options to represent an individual and a solution, respectively.

To evaluate the different approaches, we present first numerical results. The test set of our numerical study contains 1728 PSPLIB-based two-project instances with 15 non-dummy activities each. However, we excluded 112 instances that could not been solved to proven optimality by a standard solver as well as 15 infeasible instances and one instance with total delay cost of zero.

3.1 Selection of the project structure

In addition to the scheduling of activities, for flexible projects the solution representation has to include the decision on the project structure. Therefore, Kellenbrink and Helber (2015) use a choice list to indicate the selected activities. Due to the effectiveness of the random-key representation for scheduling projects, cf., e.g., Gonçalves *et al.* (2008), we additionally developed a random-key based representation for the decision on the project structure. According to our numerical study, both approaches obtain comparable results.

3.2 Scheduling the activities

There are different variants of genetic algorithms known in the literature to schedule single projects with a fixed project structure. Two important aspects of a genetic algorithm are the representation of a schedule and its decoding, including the schedule generation scheme (SGS) applied.

Hartmann (1998) introduces a genetic algorithm to solve the resource-constrained project scheduling problem using an activity list. Gonçalves *et al.* (2008) achieve good results by using a random-key representation to schedule the activities of multiple projects with a fixed project structure. Our numerical results show that the random-key representation leads to better results for our problem setting.

For each representation, the fitness can be determined by applying the serial SGS. Hartmann (2002) introduces a self-adapting genetic algorithm where the solution representation includes the decision on using the parallel or the serial SGS. We have refined this approach by giving information on the SGS used in *each* scheduling step. In Figure 2, the relative

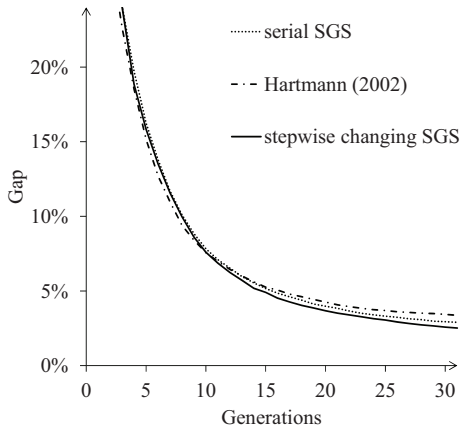


Fig. 2. Consideration of different SGS

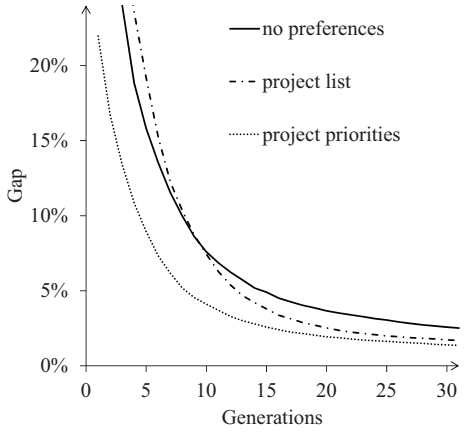


Fig. 3. Consideration of project preferences

deviations from the total delay cost of the considered instances are given. After only eight generations, the stepwise changing SGS outperforms the other approaches.

When computing the numerical results in Figure 2, we did not take the different importance of the projects into account. However, Hoffmann *et al.* (2017) show that the inclusion of project preferences is important for scheduling multiple projects with flexible project structures.

Thus, we define representations where project priorities are added. The project priorities are represented as random-keys which are then multiplied by the specific delay costs to itensify the effect. Those project preferences determine for each scheduling step which project is prioritized. As another option, we use a project list which directly gives the information, which project is scheduled next. Figure 3 shows our results for the representation of project preferences. The usage of project priorities enhances the solutions found. After 30 generations, the mean deviation from the optimum amounts to 1.39%.

4 Outlook

As described above, there are many different possibilities to define the representation for the different aspects of the given problem setting. Our numerical results show efficient combinations of representations working together best regarding performance and computational time. We will focus on applying our approach to larger instances containing more than two projects in future research. Moreover, we will determine the potential of using forward-backward-improvement while scheduling the activities. In addition, we will alternate the evolutionary process and evaluate resulting effects.

Acknowledgements

The authors thank the German Research Foundation (DFG) for the financial support of this research project in the CRC 871 "Regeneration of Complex Capital Goods".

References

Browning T. R., A. A. Yassine, 2010, "Resource-constrained multi-project scheduling: Priority rule performance revisited", *International Journal of Production Economics*, Vol. 126 (2), pp. 212-228.

Gonçalves J. F., J. J. M. Mendes, M. G. C. Resende, 2008, "A genetic algorithm for the resource constrained multi-project scheduling problem", *European Journal of Operational Research*, Vol. 189 (3), pp. 1171-1190.

Hartmann S., 1998, "A competitive genetic algorithm for resource-constrained project scheduling", *Naval Research Logistics*, Vol. 45 (7), pp. 733-750.

Hartmann S., 2002, "A self-adapting genetic algorithm for project scheduling under resource constraints", *Naval Research Logistics*, Vol. 49 (5), pp. 433-448.

Hoffmann L.-S., T. Kuprat, C. Kellenbrink, M. Schmidt, P. Nyhuis, 2017, "Priority Rule-based Planning Approaches for Regeneration Processes", *Procedia CIRP*, Vol. 59, pp. 89-94.

Kellenbrink C., S. Helber, 2015, "Scheduling resource-constrained projects with a flexible project structure", *European Journal of Operational Research*, Vol. 246 (2), pp. 379-391.

Pritsker A. A. B., L. J. Watters, P. M. Wolfe, 1969, "Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach", *Procedia CIRP*, Vol. 59, pp. 89-94.

A comparison of neighborhoods for the blocking job-shop problem with total tardiness minimization

Julia Lange[1]

Otto-von-Guericke-University Magdeburg, Germany
julia.lange@ovgu.de

Keywords: job-shop scheduling, blocking, total tardiness, heuristics.

1 Introduction

The job-shop scheduling problem is one of the well-studied issues in scheduling research. The integration of blocking constraints is motivated by real-world applications like the scheduling of trains in a network and the production of huge items. It refers to the absence of buffers in the planning system, so that a job blocks a machine until its subsequent machine is idle. As a customer-oriented optimization criterion, the minimization of the total tardiness of all jobs with regard to given due dates is considered.

Different mathematical programming formulations of the blocking job-shop problem (BJSP) with total tardiness minimization are tested and discussed in Lange and Werner (2017). The results provide an indication to the necessity of heuristic methods for the BJSP. In line with this idea, several authors present heuristic approaches to tackle related types of job-shop scheduling problems. Minimizing the makespan, the BJSP is solved by a genetic algorithm in Brizuela *et. al.* (2001) and by a tabu search heuristic in Gröflin and Klinkert (2009). Different neighborhoods adapted to a total weighted tardiness objective are presented by Kuhpfahl and Bierwirth (2016) for the job-shop problem without blocking constraints. Furthermore, Bürgy (2017) applies a graph-based tabu search approach to the BJSP considering different regular optimization criteria. Based on a generalized graph formulation for the BJSP, a hybrid branch-and-bound-method is applied to a train scheduling problem with a tardiness-based objective in D'Ariano *et. al.* (2007).

In contrast to the techniques given in the literature, two permutation-based heuristic approaches are presented and compared in this paper. Since total tardiness minimization corresponds to a regular optimization criterion, a solution to the problem is a schedule defined by the operation sequences on the machines. Here, these sequences are given by a list or permutation of all operations. Two well-known strategies are implemented to set up a neighborhood. First, a neighbor is determined by an adjacent pairwise interchange (API) of two operations on a machine. Second, a neighbor is defined by a random leftward shift of all operations of a job in the permutation.

While these neighborhoods are successfully applied to job-shop problems without additional constraints and characteristics like connectedness can be shown easily, there are significant feasibility issues occurring in the BJSP. Since a given permutation does not necessarily correspond to a feasible schedule, complex construction and repair procedures have to be used to define feasible neighbors. Therefore, performance and characteristics of these neighborhood structures need in-depth investigation for the BJSP .

In this paper, special emphasis is given to the adjustment of the neighborhood to the optimization criterion. In line with the idea of observing a critical path for a makespan objective, neighbors are defined based on choices of interchanges and shifts made from the set of tardy jobs. General and tardiness-based neighborhoods are described and implemented in a simulated annealing (SA) metaheuristic. Computational experiments are done on train scheduling-inspired instances as well as on benchmark instances from Lawrence (1984).

Conclusions are drawn regarding the solution quality of the heuristic methods compared to the results obtained by solving the corresponding MIP formulations.

2 Problem description

A set of jobs $\mathcal{J} = \{J_i \mid i = 1, ..., n\}$ is given, where each job consists of a set of operations and $O_{i,j}$ denotes the j-th operation of job J_i. The technological route of a job J_i is defined by the requirement of a certain machine $M_k \in \mathcal{M}$ by each operation, where $\mathcal{M}$ describes the set of machines. Additionally, release dates r_i and due dates d_i are given for $J_i \in \mathcal{J}$ and recirculation is allowed. Among all schedules, which are feasible with regard to technological route and blocking constraints, a schedule with minimal total tardiness is to be found. The considered BJSP is characterized by $J \mid r_i, d_i, block, recr \mid \sum T_i$.

A schedule can be expressed by an operation-based representation s^{op} corresponding to the permutation of operations and by a machine-based representation s^{ma} corresponding to the operation sequences on the machines. Thus, every operation $O_{i,j}$ is assigned to a list index $lidx(O_{i,j}) \in \{1, 2, \ldots, n_{op}\}$ in s^{op}, where n_{op} denotes the number of operations, and to a machine index $midx(O_{i,j}) \in \{1, 2, \ldots, R_k\}$ in s^{ma}, where R_k denotes the number of operations on machine M_k. These indexes can also be referred to as positions in the permutation and on the machine, respectively.

3 Permutation-based neighborhoods for the blocking job-shop problem with total tardiness minimization

Defining neighbors by APIs is a well-known strategy in job-shop scheduling. W.l.o.g., a pair of operations will only be interchanged, if there is no idle time on the machine between these operations. In this paper, a general *API neighborhood* is set up by choosing the neighbor-defining API from the set of all possible pairs of operations. Furthermore, the *TAPI neighborhood* is described by choosing the neighbor-defining API from the set of possible pairs of operations for which the second (leftward shifted) operation belongs to a tardy job. Both neighborhoods are set up using the machine-based representation of the solution.

In order to involve some randomness in the optimization process, the *TJ neighborhood* is defined and operated on the operation-based representation. Here, a job is randomly chosen from the set of tardy jobs and all its operations are shifted to arbitrary positions with lower list indexes in the permutation.

All three neighborhoods are exemplary illustrated for an instance with 3 machines and 4 jobs in Figure 3. A feasible schedule is given, where job J_4 is tardy and job J_2 is finished on time. A neighbor in the TJ neighborhood is generated by shifting all operation of the tardy job J_4 to positions with lower list indexes. As an example of a TAPI neighbor, the pair $O_{3,1}$ and $O_{4,2}$ on machine M_1 is chosen, since operation $O_{4,2}$ belongs to the tardy job J_4. In the more general API neighborhood, one possible neighbor-defining API reverses the order of the operations $O_{3,2}$ and $O_{2,2}$ on machine M_2.

In operating these three neighborhoods, the resulting operations-based representations are infeasible with regard to blocking constraints for most of the neighbors. A complex repair procedure is applied to construct feasible neighbors while taking the neighbor-defining API as given. By doing so, necessary changes in the schedule are made to regain feasibility. Gröflin and Klinkert (2009) present a connected neighborhood for the BJSP based on a job-insertion technique with an underlying disjunctive graph. The neighborhoods considered in this paper are more general and operate on a simple list structure. Since the connectivity is not yet shown, computational experiments are a good index of performance

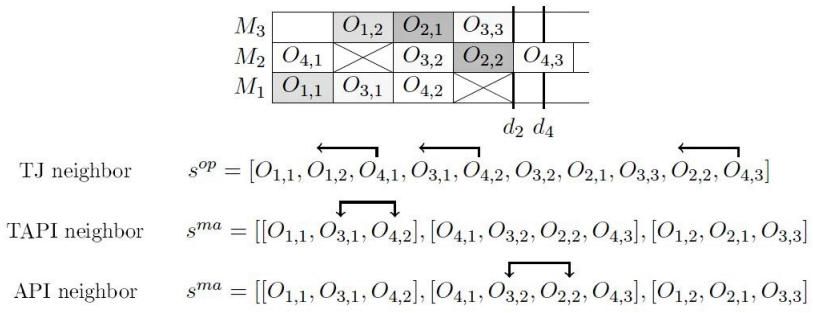


Fig. 1. Illustration of three different neighbors of a schedule

and potential. Furthermore, it is not clear whether a restriction of the API neighborhood based on the optimization criterion is reasonable for the minimization of total tardiness.

4 Computational Results

The computational experiments are done on randomly generated train scheduling-inspired instances (TS) as well as on Lawrence instances (LA) adapted for the BJSP. The release dates r_i of the jobs are generated so that jobs are forced to overlap in time and the due dates are determined by $d_i = \delta \cdot \sum p_{i,j}$ with a tight due date factor of $\delta = 1.2$. The size of the instances is denoted by (m, n), where m corresponds to the number of machines and n indicates the number of jobs. There are five different instances for each instance size.

A simulated annealing (SA) is used to solve the given problems, where the TJ neighborhood is applied with a probability of 0.1 and the API and TAPI neighborhoods are complementary applied with a probability of 0.9, respectively. A geometric cooling scheme $t_{i+1} = k \cdot t_i$ is used with $k \in \{0.99, 0.995, 0.999\}$. The initial and terminal temperature, t_0 and T, are chosen in accordance to the range of the objective function values with $(t_0, T) \in \{(20, 10), (200, 50), (1000, 100)\}$. The total number of iterations done by the SA ranges dependent on the instance size between 11000 and 64000.

The best out of the five runs for each instance (w.r.t. the objective function value) is compared to the results obtained by solving a MIP formulation for the BJSP with IBM ILOG CPLEX 12.6.1. as given in Lange and Werner (2017). The computational experiments involving the MIP solver, the SA with tardiness-based neighborhoods (TJ, TAPI) and the SA mainly relying on the general neighborhood (TJ, API) are summarized in Table 1. For each instance size, the number of instances solved to optimality (opt) by the MIP solver and the number of instances, for which a feasible solution is found, are given. Below, the number of instances for which the variants of the SA obtained the optimal solution or improved the best known feasible solution (opt/im) is stated. Additionally, the number of instances for which the SA approaches reached a solution within a 10% gap compared to the best known solution found by the MIP solver is given.

The comparison of the performance of the neighborhoods is done firstly based on the total number of instances solved with (near-)optimal solutions and secondly regarding the number of instances solved to optimality or improvement for each instance size. The superior setting is emphasized in bold face. It can be observed that the SA with the API neighborhood obtains better or equivalent solutions for eight of the ten given instance sizes. This indicates that due to the tardiness-based optimization criterion and a high number of interdependencies caused by blocking constraints the idea of an adaption to the objective

Table 1. Computational results of applying an SA to the BJSP

	TS inst.		Lawrence instances							
(m,n)	$(11,10)$	$(11,15)$	$(5,10)$	$(5,15)$	$(5,20)$	$(10,10)$	$(10,15)$	$(10,20)$	$(10,30)$	$(15,15)$
total	5	5	5	5	5	5	5	5	5	5
MIP										
opt	5	3	5	1	-	5	1	-	-	2
feasible	-	2	-	4	5	-	4	5	1	3
SA - (TJ, TAPI)										
opt/im	4	**1**	3	2	2	3	1	**2**	5	-
$< 10\%$	1	**3**	2	1	3	1	1	**1**	-	-
SA - (TJ, API)										
opt/im	**5**	-	**4**	**1**	**3**	3	**2**	1	5	-
$< 10\%$	-	2	**1**	**4**	**2**	1	-	2	-	-

function does not improve the API neighborhood for the BJSP. Statistical issues, which are not presented in detail here, show that the larger number of possible neighbors causes a higher deviation in the objective function values of the best solutions found but leads to better results on average.

5 Conclusion

In this paper, general and tardiness-based neighborhood structures for the BJSP are embedded in a SA and tested with regard to their performance compared to MIP solving techniques. Computational experiments are done on train-scheduling inspired and benchmark instances. The results give evidence to the fact that an adaption of the API neighborhood to the objective by exclusively choosing leftward interchanged operations of tardy jobs does not improve the solution quality. Significant feasibility issues involved by the blocking constraints seem to necessitate a larger number of possible neighbors to obtain (near-)optimal solutions with higher frequency.

References

Brizuela, C., Y. Zhao and N. Sannomiya, 2001, 'No-wait and Blocking Job-Shops: Challenging problems for GAs', in *IEEE International Conference on Systems, Man and Cybernetics*, pp. 2349–2354.

Bürgy, R., 2017, 'A neighborhood for complex job shop scheduling problems with regular objectives', *Journal of Scheduling*, Vol. 20, pp. 391–422.

D'Ariano, A., D. Pacciarelli and M. Pranzo, 2007, 'A branch and bound algorithm for scheduling trains in a railway network', *European Journal of Operational Research*, Vol. 183, pp. 643–657.

Gröflin H., A. Klinkert, 2009, 'A new neighborhood and tabu search for the blocking job shop', *Discrete Applied Mathematics*, Vol. 157, pp. 3643–3655.

Kuhpfahl, J., C. Bierwirth, 2016, 'A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective', *Computers & Operations Research*, Vol. 66, pp. 44–57.

Lange, J., F. Werner, 2017, 'Approaches to modeling train scheduling problems as job-shop problems with blocking constraints', *Journal of Scheduling*, published online, DOI 10.1007/s10951-017-0526-0.

Lawrence, S., 1984, 'Supplement to resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques', GSIA, Carnegie Mellon University, Pittsburgh.

Oddi, A., R. Rasconi, A. Cesta and S. Smith, 2012, 'Iterative improvement algorithms for the blocking job shop', in *ICAPS*.

A parallel machine scheduling problem with equal processing time jobs, release dates and eligibility constraints to minimize total completion time

Kangbok Lee and Juntaek Hong

Pohang University of Science and Technology, Korea
{kblee,hongjt3000}@postech.ac.kr

Keywords: equal processing time jobs, release dates, eligibility constraints, total completion time.

1 Introduction

We consider a problem of scheduling n jobs on m identical parallel machines to minimize the total completion time. Let $J = \{1, \ldots, n\}$ be the set of jobs and $M = \{1, \ldots, m\}$ be the set of machines. Job j has a given release date r_j and a set of machines that can process job j, which is called the *eligible set* of job j and denoted by M_j for $j \in J$. All jobs have an equal processing time, denoted by p. This problem is denoted by $P|r_j, M_j, p_j = p|\sum C_j$. When the number of machines is considered a fixed constant, the problem is denoted by $Pm|r_j, M_j, p_j = p|\sum C_j$.

There are a lot of papers on the parallel machine scheduling to minimize the total completion time. It is known that $R||\sum C_j$ can be solved in $O(n^3)$ time by assignment formulation (Bruno *et al.* 1974, Horn 1973). Note that $P|M_j|\sum C_j$ is a special case of $R||\sum C_j$ because we can consider that $p_{ij} = p_j$ for $i \in M_j$ and $p_{ij} = \infty$ for $i \notin M_j$. Thus, $P|M_j|\sum C_j$ can be solved in $O(n^3)$ time as well.

If release dates are involved, the complexity of the problem changes. While Lenstra *et al.* (1977) showed that $1|r_j|\sum C_j$ is strongly NP-hard, $1|r_j, prmt|\sum C_j$ can be solved by Shortest Remaining Processing Time (SRPT) rule optimally. Brucker and Kravchenko (2004) proved that $P|r_j, prmt|\sum C_j$ is strongly NP-hard.

As for equal processing time jobs cases, the problem $P|r_j, p_j = p, D_j|\sum C_j$ where job j has a deadline D_j can be solved in $O(mn^2)$ time (Simons and Warmuth 1989). $Qm|r_j, p_j = p|\sum C_j$ is solvable in $O(mn^{2m+1})$ time (Dessouky *et al.* 1990). However, the complexity for $Q|r_j, p_j = p|\sum C_j$ with an arbitrary number of machines is still open. For preemptive case, Brucker and Kravchenko (2005) showed that $P|r_j, p_j = p, prmt|\sum C_j$ can be solved in polynomial time by providing a linear programming formulation. Kravchenko and Werner (2009) generalized the previous result for the problem $Q|r_j, p_j = p, prmt|\sum C_j$ and provided a more complicated linear programming formulation with $O(mn^3)$ variables and constraints to solve the problem in polynomial time.

In Section 2, we show that the problem with a fixed m, $Pm|r_j, M_j, p_j = p|\sum C_j$, can be solved in polynomial time. For an arbitrary m, it is unknown whether the problem is polynomial solvable or not. Thus, we present an approximation algorithm for the problem with an arbitrary m along with worst case analysis in Section 3. Section 4 shows the experimental results with a modified algorithm and Section 5 concludes the paper.

2 Dynamic programming algorithm with fixed m

2.1 Preliminary

Let $\mathcal{M} = \{M_j | j \in J\}$, which is the set of all distinct eligible sets of all jobs. Then, $\mathcal{M} \subset 2^M \backslash \emptyset$. Let $k = |\mathcal{M}|$ and, without loss of generality, $\mathcal{M} = \{M^1, \dots M^k\}$. Note that $k < 2^m$.

Proposition 1. *There exists an optimal schedule in which jobs having $M^e \in \mathcal{M}$ as the eligible set scheduled at each machine are scheduled in Earliest Release Date first (ERD) order.*

From Proposition 1, we define the partition of the set of jobs as $J^e = \{j | M_j = M^e\}$ for $e = 1, \dots, k$. Let $n^e = |J^e|$. We assume that jobs in J^e are sorted in ERD order. Let $r^e(j)$ denote the release date of j-th job in J^e for $j = 1, \dots, n^e$.

Proposition 2. *There exists an optimal schedule in which the completion time of job j scheduled at a machine has a form of $r_j + ap$ for some $j \in J$ and $a \in {}_,\{1, \dots, n\}$.*

From Proposition 2, we define the set of candidates for completion times of jobs as $\Lambda = \{t | t = r_j + ap, j \in J, a \in \{1, \dots, n\}\}$. Thus, $|\Lambda| = n^2$.

2.2 Dynamic programming algorithm

We consider a partial schedule with first b^e jobs from J^e for $e = 1, \dots, k$ in which the latest completion time of machine i is t_i for $i \in \Lambda$. Let $\alpha = (t_i : b_i^1, b_i^2, \dots, b_i^e, \dots, b_i^k)$ be the state of machine i where b_i^e set of jobs among first b^e jobs from J^e are scheduled at machine i for $e = 1, \dots, k$, where $\sum_{i=1}^m b_i^e = b^e$. Note that if $i \notin M^e$, then $b_i^e = 0$.

The state of a partial schedule is a collection of states of all machines and is denoted by $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_m)$ where α_i denotes the state of machine i and α_i has the latest completion time t_i and the collection of b_i^e's, the numbers of scheduled jobs from M^e for $e = 1, \dots, k$, i.e., $\alpha_i = (t_i : b_i^1, \dots, b_i^k)$, for $i \in M$.

Let $V(\alpha)$ be the total completion time of the current partial schedule.

Restriction

- $t_i \in \Lambda$ for $i \in M$ and $b_i^e \in \{0, 1, \dots, n^e\}$ for $e \in \{1, \dots, k\}$

Boundary conditions

- $V(\alpha^0) = 0$ where $\alpha^0 = (\alpha_1^0, \alpha_2^0, \dots, \alpha_m^0)$ and $\alpha_i^0 = (0 : 0, \dots, 0)$ for $i \in \{1, \dots, m\}$
- $V(\alpha) = \infty$ if there exists i such that $t_i < 0$ or there exist e and i such that $b_i^e < 0$

Recursive relationship

- $V(\alpha) = \min\{V(\hat{\alpha}(h, f)) + t_h | h \in \{1, \dots, m\}, f \in \{1, \dots, k\}, r^f(b^f) \le t_h - p, t_h \in \Lambda\}$ where
  - $\hat{\alpha}(h, f) = (\hat{\alpha}_1, \hat{\alpha}_2, \dots, \hat{\alpha}_m)$
    where $\hat{\alpha}_i = \alpha_i$ for $i \neq h$ and $\hat{\alpha}_h = (\hat{t}_h : \hat{b}_h^1, \dots, \hat{b}_h^k)$
  - $b^f = \sum_{i=1}^m b_i^f$
    where $\hat{b}_h^e = b_h^e$ for $e \neq f$ and $\hat{b}_h^f = b_h^f - 1$
  - $\hat{t}_h = t_h - p$ and $\hat{t}_h \in \Lambda$
- If there does not exist a pair of (h, f), then $V(\alpha) = \infty$.

Optimal condition

- $\min\{V(\alpha)|b^e = n^e \text{ for } e \in \{1, \ldots, k\}\}$

The time complexity for an arbitrary eligibility case is $O(m2^{m-1}n^{2+(1+2^m)m})$, so the proposed Dynamic Programming (DP) algorithm is polynomial for a fixed m. However, for an arbitrary m, the complexity of the problem is still unknown. Thus, we propose an approximation algorithm for it in the next section.

3 Approximation algorithm

Let I be an instance of the problem, $P|r_j, M_j, p_j = p|\sum C_j$, and $z(I)$ be the optimal objective function value of I. Then, we consider two problem instances that can be defined from I:

- I_L: r_j is redefined ad $\lfloor \frac{r_j}{p} \rfloor \times p$
- I_U: r_j is redefined ad $\lceil \frac{r_j}{p} \rceil \times p$

Since the release dates of I_L and I_U are integer multiples of p, by scaling, these can be regarded as problem instances of $P|r_j, M_j, p_j = 1|\sum C_j$ where r_j is a non-negative integer. $P|r_j, M_j, p_j = 1|\sum C_j$ can be solved by assignment formulation in polynomial time.

The optimal solution of I_U is feasible to I, and is a 2-approximation solution of I because $z(I_L) \leq z(I) \leq z(I_U)$ and $z(I_U) - z(I_L) \leq np$. This approximation ratio is tight from the following example. Consider a problem instance with one job with processing time p and infinitesimal release date Δ. Since $\Delta > 0$, release date of the job in I_U is p. The optimal objective value of I_U is $2p$, while optimal objective value of I is $p + \Delta$. As $\Delta \to 0$, the ratio between optimal objective values of I_U and I approaches 2.

4 Experiments and Result

For the practical purpose, we can elaborate the algorithm. After solving with I_U, we can keep machine assignment and the job sequence at each machine while we schedule jobs as early as possible. From this procedure, we can reduce the objective function value from $z(I_U)$. For more effect, jobs assigned to the same machine m should be ERD-ordered according to the original release date $r_j \in I$. For this to happen, we propose the following time-indexed MIP formulation.

Parameters

- m: number of machines
- n: number of jobs
- p: processing time of all jobs
- r'_j: modified release date of job j, ($r'_j = \lfloor \frac{r_j}{p} \rfloor p$ for I_L, $r'_j = \lceil \frac{r_j}{p} \rceil p$ for I_U)
- $\delta_j = r'_j - r_j$
- T: the set of possible starting times, $T = \{0, p, 2p, \ldots, \max\{r'_j\} + np\}$
- P_{ijt}: big number P if $i \notin M_j$ or $t < r'_j$ for $i \in M$, $j \in J$, $t \in T$, and 0 otherwise

Variables

- x_{ijt}: 1 if job j starts at time t by machine i, 0 otherwise

Minimize

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{t \in T} t x_{ijt}(1 + \epsilon\delta_i) + np + P_{ijt}x_{ijt}$$

Subject to

$$\sum_{i=1}^{m}\sum_{t\in T} x_{ijt} = 1, \quad \text{for } j \in J$$
$$\sum_{j=1}^{n} x_{ijt} \leq 1, \quad \text{for } i \in M,\ t \in T$$
$$x_{ijt} \in \{0,1\}, \quad \text{for } i \in M,\ j \in J,\ t \in T.$$

The term δ_j in objective function enables jobs with earliest original release dates r_j to be processed earlier than those who have the same r'_j but greater r_j. The term $P_{ijt}x_{ijt}$ in objective function eliminates the schedules in which any job violates machine eligibility and release date constraints.

We can apply this idea to I_L as well by keeping machine assignment and the job sequence at each machine while scheduling jobs as early as possible. The total completion times from the solutions I_L and I_U obtained this way will be denoted as $z'(I_L)$ and $z'(I_U)$, respectively.

We also consider a simple priority-rule based algorithm, denoted as Greedy, as follows:

- First, choose the job with earliest release date(ERD) rule. If more than one job have the same ERD, choose the one with smallest eligibility $|M_j|$.
- Second, choose the machine $i \in M_j$ with earliest available time. If a tie happens, choose the one with smallest job eligibility among remaining jobs.
- Assign chosen job to chosen machine's job queue, iterate until all jobs are assigned.

The total completion time obtained by Greedy algorithm will be denoted as $grd(I)$.

So far we propose three simple algorithms to obtain a feasible schedule. For given problem instance, the algorithm's objective value (ALG) is defined as a minimum of $z'(I_L)$, $z'(I_U)$, and $grd(I)$. Optimal objective value (OPT) can be calculated by an exact MIP formulation or dynamic programming algorithm. In order to evaluate the performance of the proposed algorithm, we conduct an experiment with randomly generated test instances. Total 30,000 instances are created with following parameters.

- $m \in \{2, 3, 4, 8, 16\}$
- $n \in \{2m, 3m, 4m, 5m, 6m\}$
- $p \in \{2, 3, 4, 8, 16\}$
- $r_j \sim U\left(0, \frac{np}{m} \cdot dr\right)$, $dr \in \{0.5, 1, 1.5, 2\}$
- $E(|M_j|) = m \cdot dM$, $dM \in \{0.4, 0.6, 0.8\}$ subject to $|M_j| \geq 1\ \forall j \in J$
- Replication per each setting: 20

The longest computation time among 30,000 instances is less than 10 seconds. The result says 28,604 instances (95.3% of all instances) show $ALG = OPT$ and in only 1,396 instances (4.7% of all), ALG was strictly greater than OPT. The worst case ratio of the ALG/OPT was 19/18 (≈ 1.056). It shows that the proposed algorithm effectively works in the worst perspective.

The average ALG/OPT trend according to each parameter is shown in graphs below.

The mean of ALG/OPT increased as number of machines m increases until $m = 8$, and decreased when $m = 16$. The algorithm is expected to perform well when number of machines is large, and the result is as expected. The ratio between number of jobs n and m seems to have no effect on ALG/OPT because it neither makes the problem easier nor harder. The algorithm performs better as processing time p increases. It is considered reasonable because using the sequence of $I_L(I_U)$ on original instance I have the same effect as pushing jobs backward(forward) in the range of p. The algorithm performs better when

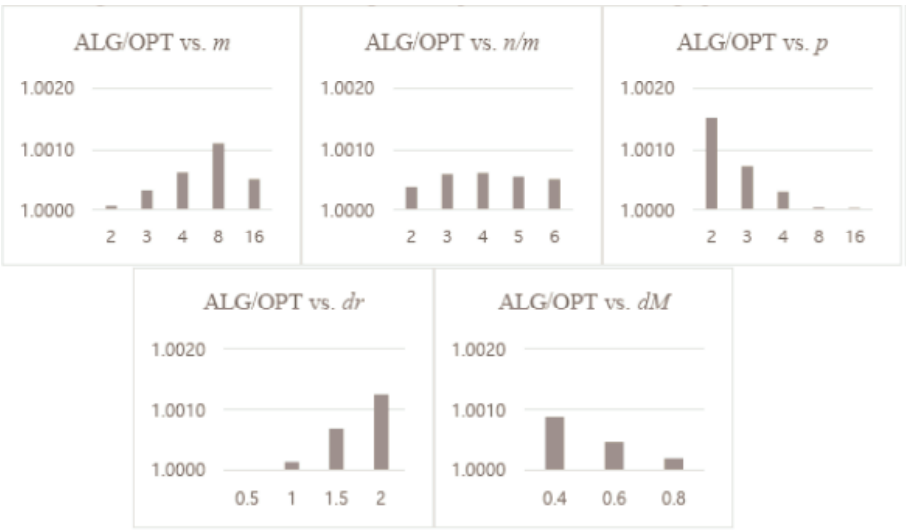


Fig. 1. Trend graphs of ALG/OPT.

the mean of release dates is smaller, which is similar to problem with no release dates. The algorithm performs better when the mean of $|M_j|/|M|$ is larger, which is closer to problem with no eligibility constraints.

5 Conclusions

We propose a polynomial time DP algorithm for $Pm|r_j, M_j, p_j = p|\sum C_j$ with fixed m. With a slight modification, this result can be extended to $Qm|r_j, M_j, p_j = p|\sum C_j$ without increasing the time complexity by defining $\Lambda_i = \{t|t = r_j + ap/v_i, j \in J, a \in \{1, \ldots, n\}\}$ with v_i being a speed of machine i as a set of candidate completion times at machine i for $i \in M$. The proposed approximation algorithm is a 2-approximation algorithm and its practical modification works very well experimentally in both worst and average perspectives.

References

Brucker, P. and Kravchenko, S.A., 2004, "Complexity of mean flow time scheduling problems with release dates", Universität Osnabrück, Fachbereich Mathematik/Informatik, OSM Reihe P, Heft 251.

Bruno, J., Coffman, E.G. Jr, and Sethi, R., 1974, "Scheduling independent tasks to reduce mean finishing time", *Communications of the ACM*, Vol. 17(7), pp. 382-–387.

Horn, W.A., 1973, "Technical note –minimizing average flow time with parallel machines", *Operations Research*, Vol. 21(3), pp. 846—47.

Kravchenko, S.A., Werner, F., 2009, "Preemptive scheduling on uniform machines to minimize mean flow time", *Computers & Operations Research*, Vol. 36(10), pp. 2816—2821.

Lenstra, J.K., Kan, A.R., Brucker, P., 1977, "Complexity of machine scheduling problems", *Annals of Discrete Mathematics*, Vol. 1, pp. 343—362.

Simons, B.B., and Warmuth, M.K., 1989, "A fast algorithm for multiprocessor scheduling of unit-length jobs", *SIAM Journal on Computing*, Vol. 18, pp. 690—710.

A new grey-box approach to solve challenging workforce planning and activities scheduling problems

Stefano Lucidi[1,2] and Ludovica Maccarrone[1,2]

[1] Department of Computer, Control, and Management Engineering,
University of Rome "La Sapienza ", Italy
lucidi, maccarrone@dis.uniroma1.it
[2] ACTOR s.r.l., via Nizza 45, 00198 Rome, Italy
stefano.lucidi, ludovica.maccarrone@act-OperationsResearch.com

Keywords: Workforce management, Project scheduling, Grey-box optimization, MILP.

1 Introduction and motivations

Nowadays, a key success factor for many large enterprises is the ability of properly managing labor cost and timetables. This is the reason why workforce planning and scheduling tools are now getting more and more developed.

Two are the typical issues arising in such applications: the first is related to the medium and long-term goal of estimating the amount of workers that the company will require in future periods. The second, mostly linked to short-term operations, involves the assignment of human resources to activities in order to meet deadlines and industrial plans.

In practice, to conduct a complete analysis and evaluate the effectiveness of a solution it is important to take into account both time and financial objectives, considering not only the need of reducing durations and delays but also the ability to do so within reasonable budgets. The result is a trade-off problem looking at the same time at avoiding resource underutilization and incapacity to comply with due dates.

In the following, we present a new approach to solve the workforce scheduling problem in complex applicative contexts such as manufacturing and logistics, characterized by the simultaneous processing of several activities, the occupation of wide areas, the coexistence of independent workloads, the use of advanced machineries and, above all, the employment of different types of operators, having various abilities and experience levels.

Standard approaches usually address this issue by defining distinct planning, scheduling and allocation problems. However, within the considered context, the problem of providing the right number of workers with the right skills at the right time is inherently linked to the schedule of the activities. For this reason, we rather propose a strategy to tackle all these aspects at the same time, taking into account a reasonable time horizon. As a result we obtain a large problem requiring not only a proper representation of processes complexity, but also a feasible assignment of operators to tasks and an optimized activities scheduling.

In what follows, the structure of the problem is formalized and a specialized simulation-based decomposition framework is proposed.

2 Problem description

Hereinafter, we will consider systems where one or more processes are executed. Roughly, a process can be described in terms of two basic definitions: the skills employed and the component activities. A skill represents the ability of an operator to perform certain tasks, thus identifying a worker type. An activity can be any non-interruptible elementary time-consuming operation requiring skilled operators to be completed.

Activities may be linked by some precedence constraints, but have variable starting times that can be modified in order to create optimal schedules satisfying logical and strategical restrictions. Indeed, activities may be subject to release and deadline constraints, and are affected by workforce availability limitations.

A basic assumption of our approach is that the number of skills required by each activity is not fixed and therefore there exist many feasible combinations of operators guaranteeing tasks completion. In particular, allowing to vary the workforce assignments between a lower and an upper limit, we evidently admit variability to operations processing times. Such aspect heavily characterizes our procedure. Assuming it is not possible to derive analytic functions expressing the link between allocated skills and time to complete the activities, we have based our solution method on the use of a set of ad hoc simulators, having as input a vector of worker availabilities and as output a duration estimate.

The result is a simulation-optimization problem facing a typical trade-off between different objectives. On the one hand it aims at reducing the employment cost, minimizing the number of necessary skilled operators, on the other, it encourages an optimal activities scheduling, trying to parallelize the tasks and decrease the overall completion time.

2.1 Mathematical formulation

In order to introduce the general mathematical formulation we have developed for this problem, we first need to list some basic definitions. We will consider m activities and n different skills. Let $A = \{1, ..., m\}$ be the set of indexes for activities, and $S = \{1, ..., n\}$ be the set of indexes for skills. Each activity is non-preemptable and is characterized by a variable processing time, a release date r_j and a due date d_j. Both r_j and d_j are real parameters and can be set to zero and infinity to nullify the associated constraints.

Precedence relations are given by the set Q of ordered index pairs, such that $(j_1, j_2) \in Q$ means that the execution of activity j_2 must start after the end of activity j_1. The same concept can be expressed by an *activity-on-node* graph whose nodes correspond to activities, and arcs represent sequence constraints. From this perspective, a necessary condition to guarantee consistent precedence relations is that the graph contains no cycles.

Our problem formulation involves three main types of decision variables. First, the total number of operators made available for each skill is represented by a vector $y \in \mathbb{N}^n$, such that y_i denotes the availability of resource i. Second, integer variables x_{ij} are required to indicate the number of workers with skill i assigned to activity j. Finally, the starting-time continuous variables t_j are introduced for each activity j, thus making possible the scheduling.

Alongside these definitions, two auxiliary variables γ_{jc} and θ_{jc} are used in our mathematical model, where j and c both belong to A; their meaning will be soon clear.

Ultimately, minding our assumption on the dependence among operators assignments and time to complete the activities, we can identify the output of the j-th simulator with the symbol $\tau_j = \phi_j(x_{1j}, \ldots, x_{ij}, \ldots, x_{nj})$, so expressing the processing time of activity j as an unknown function of the variable skill allocations.

We can therefore formalize the problem in a bilevel programming formulation having as upper-level and lower-level objectives two generic functions. Their global effect can be thought of as the combination of two conflicting components: the first accounting for the workforce cost, the second expressing a time objective. As an example of this trade-off, we can consider a situation where variables y_i are, at the same time, pushed down to lower salaries expenses, and pushed up to relax resource constraints and obtain better results in activities scheduling, improving, for example, the overall *makespan*, the sum of projects completion times or the average finish time of activities.

We propose the following formulation:

$$\min_{x,y,\tau,t,\gamma,\theta} \; f_1(x,y,\tau,t,\gamma,\theta) \tag{1}$$

$$s.t. \quad l_{ij} \le x_{ij} \le u_{ij} \qquad i \in S, \;\; j \in A \tag{2}$$

$$x_{ij} \le y_i \qquad i \in S, \;\; j \in A \tag{3}$$

$$y_i \le \sum_{j \in A} x_{ij} \qquad i \in S \tag{4}$$

$$\tau_j = \phi_j(x_{1j}, .., x_{nj}) \qquad j \in A \tag{5}$$

$$y_i \in \mathbb{N} \qquad i \in S \tag{6}$$

$$x_{ij} \in \mathbb{N} \qquad i \in S, \;\; j \in A \tag{7}$$

$$\tau_j \in \mathbb{R}^+ \qquad j \in A \tag{8}$$

$$(t,\gamma,\theta) \;\in\; \arg\min_{t,\gamma,\theta} \quad f_2(t,\gamma,\theta) \tag{9}$$

$$s.t. \quad t_j \ge r_j \qquad j \in A \tag{10}$$

$$t_j \le d_j - \tau_j \qquad j \in A \tag{11}$$

$$t_{\tilde{j}} \ge t_{\hat{j}} + \tau_{\hat{j}} \qquad (\hat{j},\tilde{j}) \in Q \tag{12}$$

$$\sum_{j \in A} x_{ij}\gamma_{jc} \le y_i \qquad i \in S, \;\; c \in A \tag{13}$$

$$t_c - t_j \ge M(\gamma_{jc} - 1) \qquad j \in A, \;\; c \in A \tag{14}$$

$$t_c - t_j \le \tau_j + M(1 - \gamma_{jc}) - \varepsilon \qquad j \in A, \;\; c \in A \tag{15}$$

$$t_c - t_j \ge -M\theta_{jc} + \tau_j - \frac{\tau_j}{2}\gamma_{jc} \qquad j \in A, \;\; c \in A \tag{16}$$

$$t_c - t_j \le M(1 - \theta_{jc}) + \frac{\tau_j}{2}\gamma_{jc} - \varepsilon \qquad j \in A, \;\; c \in A \tag{17}$$

$$t_j \in \mathbb{R}^+ \qquad j \in A \tag{18}$$

$$\gamma_{jc} \in \{0,1\} \qquad j \in A, \;\; c \in A \tag{19}$$

$$\theta_{jc} \in \{0,1\} \qquad j \in A, \;\; c \in A \tag{20}$$

The upper and lower level objective functions are respectively contained in (1) and (9). In (2) are the bounds for variables x_{ij}. Constraints (3) and (4) express two concepts: the availability of operators with skill i must be (i) enough to guarantee that each activity can be independently executed (e.g. if scheduled in sequence with the others), and (ii) not more than the total amount of resources that would be needed if all the activities were parallelized. Relation (5) brings processing time simulations into the problem. Constraints (10) and (11) give release date and deadline limits, while inequalities (12) describe the precedence relations between activities.

In order to understand the meaning of constraints from (13) to (17), it is first necessary to clarify the role of binary variable γ. For each couple of activities (j,c), we have that γ_{jc} is equal to 1 if j is in progress when c is starting, 0 otherwise. Thus, we make use of the following double implication, which is guaranteed by inequalities (14)–(17) where ε and M are two appropriate small and large constants:

$$\gamma_{jc} = 1 \Leftrightarrow t_j \le t_c < t_j + \tau_j$$

Then, constraints (13) indicate the relation between available and allocated operators, i.e. the sum of resources simultaneously occupied cannot exceed the total number of workers, for each skill i.

Finally, (6)–(8) and (18)–(20) define variables domains. Notice that activities durations τ are black-box values varying on the positive side of the real axis. This assumes a particular meaning when the structure of lower-level formulation is analyzed: indeed, if we consider each τ_j to be externally calculated (once a value for every x_{ij} and y_i is fixed by the upper-level decision-maker) and f_2 to be the overall *makespan*, we can prove our problem to fall under the standard definition of Resource Constrained Project Scheduling Problem (see Artigues, Demassey and Néron (2008)), with additional due date constraints.

However, due to the a priori unknown values of processing times, an appropriate comparison of our formulation with existing ones makes sense only by considering analogous approaches, as those proposed by Artigues, Michelon and Reusser (2003) and Koné, Artigues, Lopez and Mongeau (2011), that admit continuous starting time variables and do not recourse to time horizon discretization. In this respect, it is worth making two observations: the first is that, similarly to authors just cited, we have developed a MILP formulation of the problem (that is evident when looking at upper-level variables as constants). The second, instead, captures the difference between our and previous approaches. In particular, by exploiting the relations between pairs of activities, we are able to formulate the same problem in a new way which differs and in some cases outperforms existing methods in terms of total amount of variables and constraints.

Anyway, the solution of the RCPSP constituting our lower-level optimization is not the only source of complexity in our procedure. The presence of black-box values calculated by simulators is an important issue to be addressed. For this reason, we propose a decomposition approach modeling the problem from a new grey-box optimization perspective.

3 Simulation-Optimization framework

Our solution framework is composed of three main nested blocks as shown in Figure 1. The most external one is a *black-box* optimization formulation working on variables y_i and x_{ij}, subject to constraints (2)–(4) and (6)–(7). Its objective function, denoted by $\tilde{f}$, has the structure of (1) and is calculated every time from the results of inner blocks.

In turn, the second module, represented by the resource constrained scheduling formulation described above, is (approximately) solved at every iteration, immediately after the execution of the third block, that takes the x_{ij} as inputs, runs a parallel simulation for each activity j, and returns the processing times τ_j.

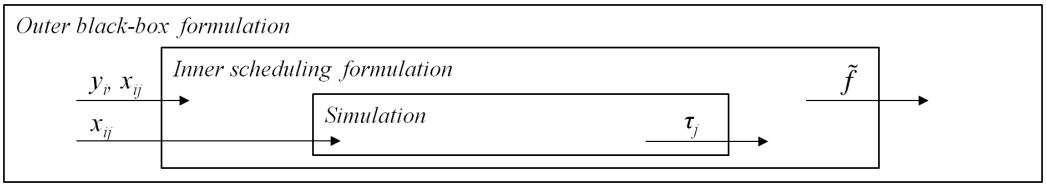


Fig. 1. Framework structure

References

Artigues C., Demassey S., Néron E., 2008, Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications, ISTE, London, UK.

Artigues C.,Michelon P.,Reusser S., 2003, "Insertion techniques for static and dynamic resource-constrained project scheduling", *European Journal of Operational Research*, Vol.149 pp.249-67.

Koné O., Artigues C., Lopez P., Mongeau M., 2011, "Event-based MILP models for resource-constrained project scheduling problems", *Computers & Operations Research*, Vol.38 pp.3-13.

Scheduling Identical Parallel Machines with Delivery Times to Minimize Total Weighted Tardiness

Söhnke Maecker[1], Liji Shen[2]

[1] Faculty of Business and Economics, Technische Universität Dresden, 01062 Dresden, Germany
soehnke.maecker@tu-dresden.de
[2] Chair of Operations Management, WHU - Otto Beisheim School of Management, 56179 Vallendar, Germany
liji.shen@whu.edu

Keywords: scheduling, parallel machines, delivery times, memetic algorithm.

1 Introduction

In most manufacturing and distribution systems, semi-finished jobs are transferred from one facility to another for further processing or finished jobs are delivered to the customer. In the latter case, the job completion time is defined as the time by which the job arrives at the customer. Different operations must be carefully coordinated to achieve ideal overall system performance (Lee and Chen 2001).

Multiple definitions of delivery times exist in the machine scheduling literature. Maggu and Das (1980) first consider job transportation in a two-machine flow shop makespan problem where job-dependant transportation times occur between the processing stages and transportation capacity is unlimited. Potts (1980) studies the problem of scheduling jobs on a single machine with release dates and job-dependent delivery times to minimize the time by which all jobs are delivered. A similar problem with identical release dates is studied by Woeginger (1994) for the parallel machine case. Lee and Chen (2001) define two types of scheduling problems with job delivery where the transportation capacity is limited in terms of both available number of vehicles and vehicle capacity. *Type-1* transportation considers job transportation inside a manufacturing facility between processing stages and *type-2* transportation takes place between the facility and a customer area. In both cases, jobs share a common delivery time and the objective is to minimize the makespan. A complexity analysis is presented for single machine, parallel machine, and flow shop environments. Chang and Lee (2004) study *type-2* transportation for the single and parallel machine chase with jobs, that require different amounts of space on the transportation vehicle. Furthermore, two separate customer areas exist. Koulamas and Kyparisis (2010) present a single machine problem where jobs have *past-sequence-dependent* delivery times proportional to their waiting time before processing. Polynomial-time algorithms are presented for multiple optimization criteria. Another definition of job delivery times is presented by Chen *et. al.* (2016) who investigate a parallel machine problem where a set of delivery times are given and each delivery time needs to be assigned to an individual job.

In this paper, we consider the delivery aspect when scheduling jobs on identical parallel machines to minimize the total weighted tardiness (TWT) and delivery times are machine-dependent. We first formally describe the problem and present a mixed integer linear programming formulation (MILP). Afterwards, a memetic algorithm (MA) is developed and compared to the MILP as well as multiple well-known scheduling heuristics on a large set of randomly generated test problem instances. In addition, the impact of the instance parameter setting, especially the delivery times, on the algorithm performance is investigated.

2 Problem Formulation

The problem can be described as follows: Given are a set of n jobs $j = 1, \ldots, n$ and m identical machines $h = 1, \ldots, m$. Each job needs to be processed by one and only one machine without interruption while each machine can handle exactly one job at a time. All jobs are available at time zero. Each job j has a specific processing time p_j, due date d_j, and weight w_j. A machine-dependent delivery time q_h occurs immediately upon completing a job on the respective machine. While a job is being transferred, the machine may already start processing the next job in line. The transportation capacity is assumed to be unlimited in terms of both vehicle availability and vehicle capacity. The problem is to determine a schedule π to minimize the TWT ($\sum w_j T_j$). The tardiness of a job T_j is defined as $T_j = \max\{C_j - d_j, 0\}$ with C_j being the time job j reaches the customer. Following the conventional three-field-notation by Graham *et. al.* (1979), the problem can be expressed as $Pm|q_h|\sum w_j T_j$. Since it is well known that problem $1||\sum w_j T_j$ is $\mathcal{NP}$-hard in the strong sense (Lenstra *et. al.* 1977), the problem considered here is $\mathcal{NP}$-hard in the strong sense as well.

To formulate the problem as an MILP, we introduce two binary decision variables x_{ij} with

$$x_{ij} = \begin{cases} 1, \text{ if job } j \text{ is sequenced immediately after job } i, \\ 0, \text{ otherwise,} \end{cases} \tag{1}$$

and y_{jh} with

$$y_{jh} = \begin{cases} 1, \text{ if job } j \text{ is the first sequenced job on machine } h, \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

Furthermore, a dummy job $j = n + 1$ is introduced with $p_{n+1} = 0$ that marks the end of the schedule on each machine. The problem can now be formulated as follows:

$$\min \sum_{i=1}^{n} w_j T_j \tag{3}$$

subject to

$$\sum_{j=1}^{n} y_{jh} \leq 1 \quad h = 1, \ldots, m; \tag{4}$$

$$\sum_{h=1}^{m} y_{jh} \leq 1 \quad j = 1, \ldots, n; \tag{5}$$

$$y_{jh} + \sum_{i=1, i \neq j}^{n} x_{ij} = 1 \quad j = 1, \ldots, n; \quad h = 1, \ldots, m; \tag{6}$$

$$\sum_{i=1, i \neq j}^{n+1} x_{ji} = 1 \quad j = 1, \ldots, n; \tag{7}$$

$$C_j \geq (p_j + q_h) y_{jh} \quad j = 1, \ldots, n; \quad h = 1, \ldots m; \tag{8}$$

$$C_j \geq C_i + p_j - H(1 - x_{ij}) \quad i, j = 1, \ldots, n; i \neq j; \tag{9}$$

$$T_j \geq C_j - d_j \quad j = 1, \ldots, n; \tag{10}$$

The objective (3) is to minimize the TWT. Constraints (4) and (5) ensure that at most one job is assigned to the first position on each machine. Constraints (6) and (7) determine job sequences. The completion time of the first jobs on machines is calculated by inequality (8) while (9) defines the completion time for the remaining jobs, where H is a sufficient large number. The tardiness of each job is calculated by inequality (10).

3 The Memetic Algorithm

Our MA adopts the permutation-like representation scheme by Cheng *et. al.* (1995) where the genotype consists of job- and partitioning symbols. In the reproduction phase, the offspring is generated through a *subschedule preservation* crossover operator (Cheng *et. al.* 1995) and a mutation operator that uses an *insertion* strategy. The mutation operator alters individuals by removing a random element from the chromosome and reinserting it at another random position to facilitate diversification. The neighbourhood structure of the mutation operator includes changing the position of jobs on a machine, reinserting jobs on other machines, and changing the overall partitioning.

More importantly, the MA incorporates a local search (LS) to improve all offspring solutions after reproduction. In the subsequent hill-climbing phase, the LS is performed on each generated offspring solution, that systematically examines all possible, non-trivial exchanges of elements on the chromosome while the maximum length of the search is limited. Consistent with the mutation operator, the neighbourhood structure of the LS includes swapping jobs on one machine, swapping jobs on different machines, and changing the overall partitioning. Note that the selection of diverging neighbourhoods for mutation and hill-climbing is crucial to the success of our MA in order to explore the solution space efficiently. The new generation is then selected based on fitness of the former population and the offspring. The MA terminates after a predefined number of iterations without improvement.

4 Computational Results

In our preliminary tests, we use the proposed MILP and several existing heuristics as reference for comparison. The heuristics include the *apparent tardiness cost* (ATC) rule (Vepsalainen and Morton 1987), the *modified due date* (MDD) algorithm by Alidaee and Rosa (1997), and the KPM heuristic (Koulamas 1994). We implemented the MILP in IBM ILOG CPLEX Optimization Studio 12.6 with a time limit of 30 minutes and a maximum of 8 threads. The heuristics and the MA were implemented in C++. Experiments were conducted on a personal computer with an AMD Opteron 6282 SE processing unit with 2.6 GHz and 128 GB RAM. Results for the MA were obtained by keeping the best objective value out of five independent runs.

The problem instance data including n, m, p_j, w_j, and d_j were generated based on the method proposed by Potts and Van Wassenhove (1982). Moreover, machine delivery times are uniformly distributed with $q_h \sim [1, 50]$ for small delivery times and with $q_h \sim [101, 300]$ for large delivery times. In total, we have 40 configurations. For each of them, five problem instances are generated, which results in 200 instances.

Table 1 shows the average relative percentage deviation (ARPD) achieved by each approach. The ARPD is defined as $\frac{Z-Z^*}{Z^*} \times 100$, where Z is the objective value obtained by the respective solution approach and Z^* is the best found solution among all approaches. Note that the results for all individual due date settings were summarized to stress the impact of delivery times. Additionally, the average CPU time required by the MA is given.

It can be seen that the MA, in general, outperforms the other approaches. On the other hand, the computing time increases significantly with the problem size. For $n = 20$, the MILP found for 3 out of 40 instances slightly better results than the MA. For $n = 100$, the MILP solutions are not competitive and no feasible solutions were generated for $n = 200$ within the 30 minutes. The delivery times appear to have a strong impact on the performance of the solution approaches since the heuristics perform, compared to the MA, considerably worse for instances with small delivery times. This effect deserves further investigation.

Table 1. Computational Results

Problem Setting			ARPD					CPU Time
n	m	q_h	ATC	KPM	MDD	MILP	MA	[sec]
20	5	[1,50]	14.17	11.17	43.31	0.32	0.02	0.41
20	5	[101,300]	0.68	1.21	3.91	0.38	0.00	0.38
100	5	[1,50]	23.13	28.67	109.06	1021.94	0.00	68.23
100	5	[101,300]	6.98	27.03	69.82	302.93	0.00	102.46
100	20	[1,50]	27.38	13.98	48.86	84.36	0.00	68.76
100	20	[101,300]	0.78	0.35	3.34	34.96	0.00	100.04
200	5	[1,50]	21.29	55.12	159.58	–	0.00	587.86
200	5	[101,300]	14.13	40.37	111.47	–	0.00	699.98
200	20	[1,50]	34.80	26.60	72.74	–	0.00	595.77
200	20	[101,300]	3.59	2.83	18.10	–	0.00	685.49
		Avg.	14.69	20.73	64.02	240.81	0.00	

To conclude, the MA shows promising results in our preliminary tests. Nonetheless, extensive tests and comparison with other metaheuristics are desirable.

References

Alidaee B., D. Rosa, 1997, "Scheduling parallel machines to minimize total weighted and unweighted tardiness", *Computers & Operations Research*, Vol. 24, pp. 775-788.

Chang Y. C., C. Y. Lee 2004, "Machine scheduling with job delivery coordination", *European Journal of Operational Research*, Vol. 158, pp. 470-487.

Chen Y., L. Lu and J. Yuan, 2016, "Two-stage scheduling on identical machines with assignable delivery times to minimize the maximum delivery completion time", *Theoretical Computer Science*, Vol. 622, pp. 45-65.

Cheng R., M. Gen and T. Tozawa, 1995, "Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithms", *Computers & Industrial Engineering*, Vol. 29, pp. 513-517.

Graham R. L., E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan, 1979, "Optimization and approximation in deterministic sequencing and scheduling: a survey", *Annals of Discrete Mathematics*, Vol. 5, pp. 287-326.

Koulamas C., 1994, "The total tardiness problem: review and extensions", *Operations Research*, Vol. 42, pp. 1025-1041.

Koulamas C., G. J. Kyparisis, 2010, "Single-machine scheduling problems with past-sequence-dependent delivery times", *International Journal of Production Economics*, Vol. 126, pp. 264-266.

Lee C. Y., Z. L. Chen, 2001, "Machine scheduling with transportation considerations", *Journal of Scheduling*, Vol. 4, pp. 3-24.

Lenstra J. K., A. H. G. Rinnooy Kan and P. Brucker, 1977, "Complexity of machine scheduling problems", *Annals of Discrete Mathematics*, Vol. 1, pp. 343-362.

Maggu P. L., G. Das, 1980, "On 2 $\times$ n sequencing problem with transportation times of jobs", *Pure and Applied Mathematika Sciences*, Vol. 12, pp. 1-6.

Potts C. N., 1980, "Analysis of a heuristic for one machine sequencing with release dates and delivery times", *Operations Research*, Vol. 28, pp. 1436-1441.

Potts C. N., L. N. Van Wassenhove, 1982, "A decomposition algorithm for the single machine total tardiness problem", *Operations Research Letters*, Vol. 1, pp. 177-181.

Vepsalainen A. P., T. E. Morton, 1987, "Priority rules for job shops with weighted tardiness costs", *Management Science*, Vol. 33, pp. 1035-1047.

Woeginger G. J., 1994, "Heuristics for parallel machine scheduling with delivery times", *Acta Informatica*, Vol. 31, pp. 503-512.

Modelling and Solving the Hotspot Problem in Air Traffic Control

Patrick Schittekat, Carlo Mannino, and Giorgio Sartor

SINTEF Digital, Norway
carlo.mannino@sintef.no

1 Introduction

Air traffic management involves the coordination of flights in a particular region of the world with the objective of guaranteeing their safety while possibly reducing delays. There usually exist different levels of responsibility.

For example, the Air Traffic Control (ATC) provided by an airport handles airplanes on the ground and in the controlled airspace in the proximity of the airport while the Air Navigation Service Providers (ANSP) manage the air traffic of a bigger region or an entire country. For a discussion of the different issues arising in air traffic management see (Allignol, 2012). One of the critical tasks of an ANSP is to provide the Air Traffic Flow Management that consists of preventing overcrowded portions of air space while trying to exploit their maximum capacity. In fact, the air space within a country or a region is subdivided in sectors which are assigned to specific controllers. Each controller can handle no more than a given number of airplanes at a time. Consequently, each sector has its own capacity, that is the maximum number of airplanes that can occupy the sector in a given time. Note that not only capacity, but even the shape and number of sectors varies from time to time. For instance, at peak hours the number of sector increases.

The number of airplanes that will occupy each sector in a given time can be forecast taking into account the timetable and the planned route of the airplanes. A hotspot is a sector in which the forecast number of airplanes is greater than its maximum capacity in at least one point in time. The task of an air traffic flow manager is to prevent hotspots while guaranteeing an efficient utilization of the air space.

The official flight plans of airplanes may already give rise to hotspots. In addition, the timetable changes when one or more airplanes are delayed and, as a result, more hotspots might appear.

When this happens, the air traffic flow manager has to modify the flight plans of many airplanes in order to avoid hotspots and reduce delays. More specifically, the manager can delay some take-offs, or reduce speeds on certain trip segments for airborne aircraft. This procedure is usually carried out in a heuristic way and with little software support, leading most of the time to suboptimal solutions.

2 A MILP model for the hotspot problem

The MILP model for the Hotspot problem resembles very closely the classic job-shop scheduling problem with blocking and no-wait constraints (Mascis and Pacciarelli, 2002) exploited in several papers for different transportation problems.

When airborne an airplane f will traverse an ordered sequence of sectors $(s_1, s_2, \ldots, s_q)$. We define a route as an ordered sequence of pairs aircraft-sector, say $O(f) = ((f, s_1), (f, s_2), \ldots, (f, s_q)) = (v_1, v_2, \ldots, v_q)$. An element $v \in O(f)$ is then a pair (f, s), where f is a flight and s a sector. With each element v we associate a minimum traversing time λ_v and a maximum traversing time Λ_v. To simplify the notation, if the route of an airplane f starts

at an airport a, then we also consider a as a special sector and we have the special pair (f, a) in $O(f)$. Similarly, the destination airport will be represented as a special sector.

Consider now the set of all flights F, and let O be the set of sector occupations by all flights in F. With every element $v = (f, s) \in O$ we associate a variable t_v, representing the time airplane f enters sector s. Now let $u, v \in O$ correspond to the occupation of two successive sectors in the route of flight f. Then the following constraints must hold:

$$\lambda_u \leq t_v - t_u \leq \Lambda_u. \tag{1}$$

Now, consider a set of distinct flights $\bar{F} = \{f_1, \ldots, f_q\}$ traversing a sector s. For each flight f_i, let a_i be the time the flight enters s and d_i the time the flight exits s (that is, it enters the next sector). Now, assume that the capacity c_s of the sector is not enough to accommodate all flights in $\bar{F}$, namely $c_s < |\bar{F}|$. Then, at least for a pair f, g of flights in $\bar{F}$, f and g do not meet in s, namely either f exits s before g enters s or vice-versa. This can be expressed by the following disjunctive constraint:

$$(a_g - d_f \geq 0) \vee (a_f - d_g \geq 0). \tag{2}$$

The above disjunctive constraint can be linearized in standard fashion by introducing a binary variable y_{fg} for each ordered pair of flights $(f, g) \in \bar{F} \times \bar{F}$, such that $y_{fg} = 1$ if f exits s before g enters and $y_{fg} = 0$ otherwise. Then, for all pairs in $\bar{F}$, constraint (2) can be replaced by the following conjunction of linear constraints:

$$\begin{array}{ll} (i) \;\; y_{fg} + y_{gf} = 1, & \{f, g\} \subseteq \bar{F} \\ (ii) \; a_g - d_f \geq -M(1 - y_{fg}) & (f, g) \in \bar{F} \times \bar{F} \end{array} \tag{3}$$

where M is a suitably large positive constant.

Now, for $\{f, g\} \subseteq \bar{F}$, we introduce a binary variable x_{fg} and we let $x_{fg} = 1$ if f and g meet in s, and $x_{fg} = 0$ otherwise. Then, if $c_s < |\bar{F}|$, we must have:

$$\sum_{\{f,g\} \subseteq \bar{F}} x_{fg} \leq \binom{|\bar{F}|}{2} - 1 \tag{4}$$

so that at least a pair of flights in $\bar{F}$ do not meet in s. Because constraint (2) actually holds only if f and g do not meet in s, which in turn depends on the value of x_{fg}, we can suitably modify $(3.i)$ to take this into account:

$$y_{fg} + y_{gf} = 1 - x_{fg}. \tag{5}$$

So, a complete formulation for a set of flights F with their routes traversing a set S of sectors can be obtained by considering now constraints (1) for all routes, and constraints $(3.ii)$, (4) and (5) for all sectors $s \in S$ and all set $F(s) \subseteq F$ of flights exceeding the capacity c_s of s. Let $P \subset \mathbb{R}^p$ be the set of points (x, y, t) satisfying all such inequalities, including the integer stipulation on variables x, y: then our problem reduces to $\{\min f(t) : (x, y, t) \in P\}$. The objective $f(t)$ may vary form instance to instance, but for our first set of experiments it will simply be the (weighted) delay at destination.

3 Solution approach: sketch

In principle, problem $\{\min f(t) : (x, y, t) \in P\}$ could be solved by simply resorting to some general purpose commercial solver. However, formulation P has two major sources of complexity which do not allow such a simple approach, already for small-medium size realistic instances. First, the family of constraints (4) can grow exponentially with F. This is tackled in a standard fashion by resorting to the so called "lazy constraints" trick. Namely, constraints are generated dynamically during the search as lazy constraints - i.e. only if they are violated by the current integer feasible solution.

Next, in order to make the constraint redundant for certain values of the binary variables, in $(3.ii)$ we make use of a second, infamous trick, namely we include the large coefficient M (the "big-M trick"). In turn, this makes the formulation very weak and prone to return poor bounds - and thus often intractable search trees.

Our approach to tackle this problem and solve $\{\min f(t) : (x, y, t) \in P\}$ extends the methodology first developed in (Lamorgese and Mannino, 2016). In particular, we exploit Benders' decomposition to obtain a (master) problem only in the binary variables - plus a few continuous variables to represent the objective function. The decomposition allows us to get rid of big-M coefficients (at the cost of an increased number of linear constraints). Moreover, the constraints of the reformulated master correspond to basic graph structures, such as paths, cycles and trees. The new formulation is obtained by strengthening and lifting the constraints of a classical Benders' reformulation[1].

Computational experiments. In preparation.

References

Allignol, C., Barnier, N., Flener, P., and Pearson, J., 2012, "Constraint programming for air traffic management: a survey 1: In memory of pascal brisset", *The Knowledge Engineering Review*, Vol. 27(3), pp. 361–392.

Lamorgese, L. and Mannino, C., 2016, "A non-compact formulation for job-shop scheduling problems in transportation" (Dagstuhl Seminar 16171), *Dagstuhl Reports*, Vol. 6(4), p. 151, 2016, submited to *Operations Research*, under revision.

Mascis, A. and Pacciarelli, D., 2002, "Job-shop scheduling with blocking and no-wait constraints", *European Journal of Operational Research,*, Vol. 143(3), pp. 498–517.

[1] Strengthening and lifting are non-straightforward and both necessary: the master of the classic Benders' reformulation would still contain many copies of the despised big-M.

A proactive-reactive approach to schedule an automotive assembly line

Massimo Manzini[1], Erik Demeulemeester[2] and Marcello Urgo[1]

[1] Politecnico di Milano, Mechanical Engineering Department, Milano, Italy
massimo.manzini, marcello.urgo@polimi.it
[2] KU Leuven, Faculty of Business and Economics, Department of Decision Sciences and Information Management, Leuven, Belgium
erik.demeulemeester@kuleuven.be

Keywords: flow-shop, proactive-reactive scheduling, assembly.

1 Introduction and problem statement

The assembly of bodywork parts for the automotive sector is operated in dedicated assembly lines implementing the sequence of assembling operations through specific joining technologies (e.g., spot welding, clinching, hemming, etc.). These assembly lines are organized as a set of stations executing assembly operations, input/output stations to load components and unload final parts, and a transportation device moving parts within the line. The latter is usually a *7-axis* robot shared among the stations. In this paper we consider an assembly line where a batch of parts has to be processed. Assembly operations are executed by automatic devices while load/unload operations are executed manually. The line has a single transportation robot to be shared among the stations and the proposed approach aims at scheduling its missions. Due to the manual execution of load/unload operations, uncertain process times must be considered, thus, the problem under study is a *Stochastic Resource-Constrained Flow-Shop Scheduling Problem* to minimize the time needed to complete a batch of products, i.e., the makespan. In the need to address uncertainty, specific approaches must be adopted. Examples are the ones optimizing the expected value of the makespan (Fernandez 1995, Igelmund and Radermacher 1983). Nevertheless, the minimization of the expected value does not protect against rare but very extreme scenarios, as discussed in (Alfieri *et al.* 2012) and (Manzini and Urgo 2015) for *Make-to-Order* processes. To this aim, we propose a *proactive-reactive* approach providing a *baseline schedule* and looking for the optimal sequence of the robot considering the actual duration of operations during the execution of the assembly process. Differently from other approaches of this class, e.g., (Davari and Demeulemeester 2016), the proposed approach identifies *disjunctive constraints* without explicitly deciding the starting times of operations.

2 Solution approach

Consider an *Activity-on-Node* (*AoN*) representation of a flow-shop where $V = \{0, 1, ..., n\}$ is the set of nodes representing operations and $E = (i, j),\ i, j \in V$ the set of arcs modeling precedence constraints. Operation durations are modeled through general and independent random distributions $\tilde{\mathbf{p}} = \tilde{p}_0, \ldots, \tilde{p}_n$, p_i being a realization of distribution $\tilde{p}_i$ and $\mathbf{p} = p_0, \ldots, p_n$ a realization of the entire set $\tilde{\mathbf{p}}$. Notice that, if an operation is deterministic, the described formulation still applies with a single value as support. The flow-shop under study has a limited availability of the transporter and hence we consider a single resource with unary availability. We address the scheduling of shared transporter's misisons through the decisions over a set of *disjunctive constraints* named E_{DC} (additional to the ones in E), resolving resource utilization conflicts.The uncertainty embedded in the

problem is addressed by adopting a *proactive-reactive* approach made up of two steps. The first step provides the *baseline schedule* as the optimal sequence of the robot considering a given duration of the uncertain operations (e.g., a quantile can be used). The second one is supposed to operate while the *baseline schedule* is being operated, every time an incongruity between the fixed operation duration and the one experienced in the execution of the schedule occurs. It checks whether the *baseline schedule* is supposed to remain optimal and, if needed, reacts by inverting some of the *disjunctive constraints* previously selected. The two steps are described in detail in the following.

2.1 Proactive step

The *proactive* step hypothesizes that the duration of operations is fixed. In case of uncertain durations this value can be decided by fixing a quantile q obtaining $\mathbf{p^q} = p_0^q, \ldots, p_n^q$, without considering any anticipation of associated uncertainty. The scheduling problem is solved using the deterministic approach presented in (Demeulemeester and Herroelen 1992). The *baseline schedule* obtained provides the set of additional constraints E_{DC}. In addition to this, a sensitivity analysis on the solution is also executed. For each precedence constraint in E_{DC}, the range of variability of operation durations is calculated such that, if the durations go outside this range, then the decision taken for the considered *disjunctive constraint* is not optimal anymore, and thus the opposite constraint should be considered. Consider the constraint $(i,j) \in E_{DC}$ assuming durations $\mathbf{p^q}$, and the eligible times of operations i and j, $Q_i^{\mathbf{p^q}}$ and $Q_j^{\mathbf{p^q}}$, defined as the instants on which each operation can start in terms of all the precedence constraints in E, without considering any of those in E_{DC}. Define $\Delta_{i,j}^{\mathbf{p^q}} = Q_i^{\mathbf{p^q}} - Q_j^{\mathbf{p^q}}$ as the difference between the eligible times of two operations linked with a *disjunctive constraint* (i,j). If the decision on this *disjunctive constraint* is optimal, the associated makespan is shorter than the one considering the opposite direction, i.e., $S_n^{(i,j)} \leq S_n^{(j,i)}$, where $S_n^{(i,j)}$ is the starting time of operation n, considering *disjunctive constraint* (i,j). Clearly, this depends on the duration of the operations in $\mathbf{p^q}$. The makespan takes advantage of an inversion of the *disjunctive constraint* if and only if the lateness of i, compared to $Q_i^{\mathbf{p^q}}$, is enough to cause a delay of the makespan that is longer than the delay caused by an inversion without any lateness of i. More formally, the inversion is effective if there is a difference between the eligible times that is greater than $\Delta_{i,j}^T = \Delta_{i,j}^{\mathbf{p^q}} - (S_n^{(i,j)} - S_n^{(j,i)})$. The threshold $\Delta_{i,j}^T$ will be used in the *reactive* step for evaluating the optimality of the *disjunctive constraint* (i,j) during the process execution.

2.2 Reactive step

The *reactive* step considers a vector of realizations $\mathbf{p}$ for the durations of the operation and grounds on the definition of a state space Ω modeling the execution of the operations in the flow-shop. The execution of the operations can be modeled through a sequence of states over time t, $\omega(\mathbf{p},t) = (O,F,S,d_O) \in \Omega$. Each state is fully described by the set of operations in execution O, their starting times S and their durations $d_O(i), \forall i \in O$, as well as the set of completed ones F. Algorithm 1 models the execution of operations starting from $t=0$ with initial state $\omega(\mathbf{p},0) = (0,\emptyset,0,0)$ and finishes when all the operations are completed, i.e., $F = V$ (steps 1-2). Every time an operation is completed, the set F is updated (step 4) and, if there is an operation i that can start because all its predecessors are completed (step 6), it is put into execution and added to the set of ongoing operations O (step 11). On the contrary, if its execution is constrained by the completion of another operation k through a decision on one *disjunctive constraint* $(k,i) \in E_{DC}$ (step 7), then the algorithm checks whether (k,i) remains optimal in relation to the realizations in $\mathbf{p}$. This evaluation is done through the estimation of the probability that the actual difference between the eligible

REACTIVE-PROCEDURE

1 $\omega(\mathbf{p}, 0) = (0, \emptyset, 0, 0)$
2 **While** $F \,!= V$
3 $t = t + 1$
4 **If** $d_O(i) - S(i) = p_i, \forall i \in O \rightarrow F = F + i$
5 **Else** $d_O(i) = d_O(i) + 1$
6 **If** $i \notin O, i \notin F$ and $j \in F, \forall j \in (j, i)$
7 **If** $(k, i) \in E_{DC}$ and $\mathbb{P}(\Delta^{\mathbf{P}}_{k,i}(t) > \Delta^{T}_{k,i}) > T$
8 $E_{DC} = E_{DC} - (k, i) + (i, k)$
9 $O = O + i$, $S(i) = t$
10 **Else**
11 $O = O + i$, $S(i) = t$

Algorithm 1: Reactive step algorithm.

Operation	Mode	Min	Max
I	6	5	29
T_1	13	–	–
A	10	–	–
T_2	9	–	–
O	5	4	21

Table 1: Operation duration in seconds.

times exceeds the threshold previously identified: $\mathbb{P}[\Delta^{\mathbf{P}}_{k,i}(t) > \Delta^{T}_{k,i}]$. If this probability exceeds a threshold T, the reaction is applied by inverting the constraint (k, i) (steps 8-9). The $\mathbb{P}[\Delta^{\mathbf{P}}_{k,i}(t) > \Delta^{T}_{k,i}]$ is estimated considering the duration of the operations in O preceding k and their distributions $\tilde{p}$. The probability that $\Delta^{\mathbf{P}}_{k,i}(t)$ is greater than $\Delta^{T}_{k,i}$ is equal to the probability that the difference between the finish time of the last preceding operation of k and the eligible time of i is greater than $\Delta^{T}_{k,i}$, conditioned on the ongoing durations in d_O. We are looking at the residual duration probability of the operations preceding k: $\mathbb{P}[\Delta^{\mathbf{P}}_{k,i}(t) > \Delta^{T}_{k,i}] = \mathbb{P}[max_{l \in prec(k)}(d_F(l)) - Q_i > \Delta^{T}_{k,i} \mid d_O(l)] = \mathbb{P}[max_{l \in prec(k)}(d_F(l) - d_O(l)) > \Delta^{T}_{k,i} - Q_i]$, where $prec(k)$ indicates an operation preceding k.

3 Application

The proposed approach is applied on a single product flow-shop assembling a hood bodywork. The execution of the process is modeled using the *AoN* representation in Figure 1. The process consists of five operations, the first and the last ones model the loading (I) and unloading (O) of the parts, executed manually. In the third operation (A), a reinforcement bar is added through a spot welding process, while the second and fourth operations are handling tasks (T_1 and T_2 respectively) operated by the *7-axis* robot moving the hood in the line. The two manual operations follow a triangular distribution, while the others are deterministic (Table 1). The triangular distributions consider an average execution duration as the *mode*, very close to the minimum value, and a worst-case duration as the maximum value, modeling the occurrence of a problem or a delay. The approach addresses the conflicts between transport operations in the production of a whole batch. These conflicts are depicted with dotted arcs in Figure 1 for a single transport of the first job, only (T_{21}), but are repeated for the whole batch. In addition, we set the threshold T to 0.5, but let the quantile q, used for fixing the duration in the *proactive* step, vary between 0.1 and 0.9. We evaluate the performances of the approach in terms of the *mean square error* compared to the minimum makespan solution obtained with complete knowledge of the durations of operations using 10000 runs. In addition, we estimate the approach's performances without the *reactive* step and compare the results. Aggregated performances for different lengths of the batch (from 5 to 50 jobs) are included in Table 2. Grounding on these results, the *proactive-reactive* approach always performs as good or better than the *proactive* schedule without reaction (*PR* and *P-only* in Table 2). Indeed, if the *reactive* step

	P-only			*PR*		
Quantile	0.1	0.5	0.9	0.1	0.5	0.9
5	5.473	5.473	0.917	0.917	0.917	0.917
# jobs 10	4.963	4.963	0.980	0.980	0.980	0.980
20	7.445	7.445	1.347	1.347	1.347	1.347
50	8.456	8.456	1.865	1.865	1.865	1.865

Table 2: Aggregated results of the application.

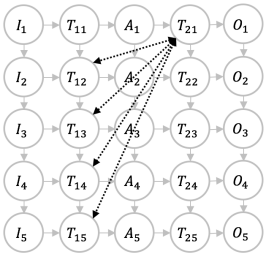


Fig. 1: *AoN* process representation.

does not apply any modification, the *baseline solution* is automatically applied, as depicted for the 5 jobs and 90*th* percentile case. The impact of the number of jobs and the percentile is also analyzed: the percentile impacts on results of the *only-P* approach, with better performances for high values. On the other hand, this parameter does not affect the reaction's performance due to the uncertainty source being limited to the first and last operations. The performances get worse as the number of jobs increases for both approaches. As a conclusion, the *proactive* approach provides a good *baseline schedule*, nevertheless, the *reaction* step improves the performances when used to manage the occurrence of unexpected events, providing a good support in the line's real-time management.

4 Conclusions

In this article we propose a *proactive-reactive* approach to schedule a semi-automatic assembly system, with a specific focus on the definition of the reaction policy. The approach has been tested on a five-operation process with good results, demonstrating that the application of the *reactive* step significantly improves the performances of the *baseline* one. Future developments will address the investigation of (i) completely manual processes or (ii) tuning the threshold for the reactive step to match user's aversion to risk and (iii) the application of additional *disjunctive constraints* modeling the schedule of machines besides handling operations.

Acknowledgments

This research has been supported by *ReCaM EU project*, grant agreement No: 680759.

References

Alfieri, A., Tolio, T. and Urgo, M., 2012, "A two-stage stochastic programming project scheduling approach to production planning", *Int J Adv Man Technol*, Vol. 62, pp. 279-290.

Davari, M. and Demeulemeester, E., 2016, "The proactive and reactive resource-constrained project scheduling problem", Working paper.

Demeulemeester, E. L. and Herroelen, W. S., 1992, "A Branch-and-Bound Procedure for the Multiple RCPSP", *Man Sci*, Vol. 38, pp. 1803-1818.

Fernandez, A. A., 1995, "The Optimal Solution to the Resource-Constrained Project Scheduling Problem with Stochastic Task Durations", Unpublished Doctoral Dissertation.

Igelmund, G. and Radermacher, F. J., 1983, "Preselective Strategies for the Optimization of Stochastic Project Networks under Resource Constraints", *Networks*, Vol. 13, pp. 1-28.

Manzini, M. and Urgo, M., 2015, "Makespan estimation of a production process affected by uncertainty: Application on MTO production of NC machine tools", *J Man Syst*, Vol. 37, No. 1, pp. 1-16.

Applying a cost, resource or risk perspective to improve tolerance limits for project control: an empirical validation

Annelies Martens[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
annelies.martens@ugent.be, mario.vanhoucke@ugent.be
[2] Technology and Operations Management Area, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, UK

Keywords: project control, buffer monitoring, analytical tolerance limits

1 Introduction

While timely completion is an important factor of project success, projects often exceed their predefined deadline. In order to protect this deadline, a project buffer can be placed at the end of the project. Further, during the project control process, the project progress can be evaluated using tolerance limits that generate warning signals when the project deadline is expected to be exceeded.

In this study, four methods that use different perspectives to construct tolerance limits for the schedule progress of projects are empirically validated on the large and diverse dataset of Batselier & Vanhoucke (2015). Each of the used perspectives, namely the time, cost, resource and risk perspective, consider project-specific information to determine the allowable buffer consumption during project execution. Based on this allowable buffer consumption, threshold values for the schedule performance can be set for each project phase. These threshold values generate warning signals when the project deadline is expected to be exceeded, such that the project manager can take corrective actions to get the project back on track. The limits using a time, cost and resource perspective have been proposed in recent literature. Their performance has been evaluated using artificial data. First, the time perspective to determine the allowable buffer consumption has been introduced by Colin & Vanhoucke (2015). Since this is the most straightforward approach that requires the least project-specific information, the resulting limits, which are referred to as *linear limits*, act as a benchmark for the other perspectives. Second, Martens & Vanhoucke (2017a) proposed a cost perspective by setting the allowable buffer consumption based on the cost accrue of the project and compare the resulting *cost limits* to the linear benchmark limits. Further, Martens & Vanhoucke (2017b) use the resource availability and requirements information to determine the allowable buffer consumption using a resource perspective to construct *resource limits*. Finally, in this study, we propose a novel approach that employs a risk perspective to set the allowable buffer consumption and to construct *risk limits*. For each type of limits, we evaluate the ease of implementation and performance for real-life projects. These limits are discussed in greater detail in section 2. In the remainder of this section, a brief introduction to project control is given.

Since uncertainty and variation during project execution inevitably result in deviations from the plan, projects often do not finish on time or within budget. In order to protect the project deadline from these deviations, a project buffer can be placed at the end of the project. Moreover, the project control phase is an important component of Integrated Project Management and Control that focuses on detecting problems and/or opportunities during project execution such that corrective actions can be taken to get the project back

on track (Vanhoucke 2014). The project control process consists of three parts, namely monitoring the project progress, evaluating this progress, and taking corrective actions when necessary. A well-known technique to monitor the cost and time progress of projects is Earned Value Management (EVM, Fleming & Koppelman (2010)). This methodology provides a birds-eye view on the project progress by aggregating the activity progress information on a higher work breakdown structure (WBS) level. Since both the schedule and cost performance metrics provided by EVM are cost-based metrics, Earned Schedule (ES, Lipke (2003)) has been developed as an extension that focuses on the time aspect of projects. In this study, EVM/ES schedule performance metrics are used to monitor the project progress. Further, project control tolerance limits are a tool to evaluate the project progress and to decide whether corrective actions are required. For each project phase, threshold values for the schedule performance are set. When the measured progress is below this threshold, the project is expected to exceed its deadline and a warning signal is generated. When a signal is generated by the tolerance limits, the project manager should take corrective actions to get the project back on track. In section 2, the different types of tolerance limits are briefly discussed. Further, results of the empirical experiment are described in section 3.

2 Tolerance limits for project schedule control

The tolerance limits that have been proposed in recent literature can be classified in three groups, namely static, statistical and analytical tolerance limits. First, *static tolerance limits* are constant throughout the entire project life cycle and do not consider any project-specific or historical information. These limits are determined by applying rules of thumb and are introduced by Goldratt (1997) and Leach (2005). Further, *statistical tolerance limits* apply concepts of Statistical Process Control (SPC, Shewhart (1931)) and require historical information or Monte Carlo simulations to define the desired state of the progress at each project phase. The statistical tolerance limits introduced in literature have been validated using simulation studies (Colin & Vanhoucke 2014, Colin & Vanhoucke 2015, Colin, Martens, Vanhoucke & Wauters 2015) or empirical data (Aliverdi, Moslemi Naeni & Salehipour 2013, Bauch & Chung 2001, Leu & Lin 2008, Lipke & Vaughn 2000, Wang, Jiang, Gou, Che & Zhang 2006). Finally, *analytical tolerance limits* require project-specific information that is readily available during the scheduling phase to determine the threshold values for each project phase. Since these limits do not require historical data or Monte Carlo simulations, they are easier to implement than statistical tolerance limits. Moreover, by including project-specific information, they are more accurate than static tolerance limits. This type of tolerance limits has been proposed by Colin & Vanhoucke (2015), Hu, Cui, Demeulemeester & Bie (2015), Martens & Vanhoucke (2017a) and Martens & Vanhoucke (2017b).

The tolerance limits reviewed in this study are analytical tolerance limits, and follow the same general procedure to be constructed. First, for each project phase, the allowable buffer consumption is determined. This reflects the amount of buffer that can be consumed at each project phase during execution without endangering the project deadline. Second, the buffered planned progress (BPP) curve is determined. This curve represents the project progress when, at each project phase, the allowable buffer consumption is entirely consumed. The construction process for the BPP-curve is illustrated in Figure 1. Finally, the threshold values are constructed by comparing the BPP to the planned progress. Consequently, when the actual progress is below the BPP, the project is expected to exceed its deadline and a warning signal is generated. For a more detailed discussion on the con-

struction of this type of tolerance limits, the reader is referred to Martens & Vanhoucke (2017a).

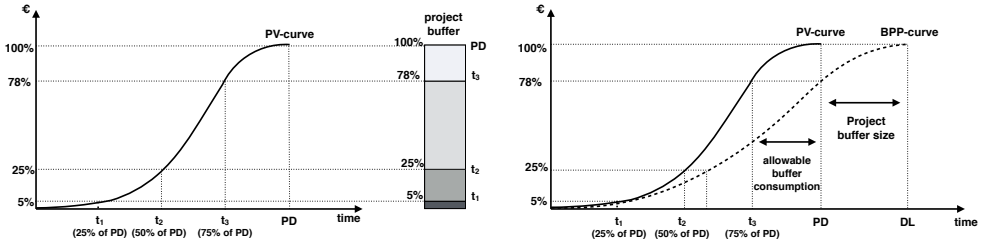


Fig. 1. Determining the BPP-curve.

The four different perspectives all propose a different approach to determine the allowable buffer consumption. The construction of the BPP and the calculation of the threshold values, on the contrary, do not differ. First, the *linear limits* assume that the project buffer can be consumed proportionally with the time, e.g. at x% of the project makespan, x% of the buffer can be consumed. Since these limits do not consider the amount of work that has to be completed during each project phase, *cost limits* have been introduced. These limits determine the allowable buffer consumption proportionally with the cost of each phase. Further, resource limits have been proposed to account for the impact of resource conflicts on project delays. Finally, we introduce the *risk limits*, which consider the risk of each project phase to determine the allowable buffer consumption. Two steps have been implemented to determine the aggregate risk of each project phase. First, a risk value is assigned to each project activity. This risk value is defined as the product of the activity duration variability (σ) as estimated by the project manager and, since activity delays may affect the actual start of successors, the number of succeeding activities ($\#succ$). Second, the risk of each project phase is determined by aggregating the risk values of the scheduled activities at each phase. The allowable buffer consumption at each project phase is determined by the risk limits proportionally with this aggregated risk.

3 Research study and preliminary results

In this study, we discuss the merits and pitfalls of using artificial and empirical data to evaluate the performance of project control tolerance limits. Further, a new perspective, e.g. a *risk perspective*, is introduced to assign portions of the project buffer to each project phase based on the risk level of these phases. We determine the aggregated risk level of each project phase by considering the estimated activity duration variance and the position of these activities in the baseline schedule. Finally, we compare the performance of the different perspectives and evaluate their ease of implementation.

The artificial data used in the simulation studies consists of 900 project networks with varying topological network structures, generated using the project network generator RanGen (Demeulemeester, Vanhoucke & Herroelen 2003). Risk and variability is added using generalised beta distributions for the activity durations. Further, the empirical data consists of a wide variety of real-life projects in different industries from the database of Batselier & Vanhoucke (2015). In this database, the baseline schedule, risk analysis and project control data of the real-life projects are listed.

The empirical experiment conducted in this study confirms the result of previous simulation studies performed by Martens & Vanhoucke (2017a) and Martens & Vanhoucke

(2017b), e.g. that including project-specific information improves the efficiency of tolerance limits for project control. However, deploying the cost perspective improves the efficiency only slightly in our empirical experiment. Further, while deploying the resource perspective entails additional effort compared to the other perspectives, this effort enhances the efficiency substantially. Finally, the novel risk perspective improves the efficiency of the tolerance limits more than the cost perspective, and is hence an appropriate alternative when projects are not constrained by scarce resources.

In general, this experiment has shown that including project specific information is an effective approach to improve the project monitoring efficiency. Further, the results of this study can be used by project managers to determine which perspectives they can deploy to monitor their projects.

References

Aliverdi, R., Moslemi Naeni, L. & Salehipour, A. (2013). Monitoring project duration and cost in a construction project by applying statistical quality control charts, *International Journal of Project Management* **31**(3): 411–423.

Batselier, J. & Vanhoucke, M. (2015). Construction and evaluation framework for a real-life project database, *International Journal of Project Management* **33**: 697–710.

Bauch, G. T. & Chung, C. A. (2001). A statistical project control tool for engineering managers, *Project Management Journal* **32**: 37–44.

Colin, J., Martens, A., Vanhoucke, M. & Wauters, M. (2015). A multivariate approach for top-down project control using earned value management, *Decision Support Systems* **79**: 65–76.

Colin, J. & Vanhoucke, M. (2014). Setting tolerance limits for statistical project control using earned value management, *Omega The International Journal of Management Science* **49**: 107–122.

Colin, J. & Vanhoucke, M. (2015). A comparison of the performance of various project control methods using earned value management systems, *Expert Systems with Applications* **42**: 3159–3175.

Demeulemeester, E., Vanhoucke, M. & Herroelen, W. (2003). Rangen: A random network generator for activity-on-the-node networks, *Journal of Scheduling* **6**: 17–38.

Fleming, Q. & Koppelman, J. (2010). *Earned Value Project Management*, 3rd edition edn, Project Management Institute, Newton Square, Pennsylvania.

Goldratt, E. (1997). *Critical Chain*, North River Press, Great Barrington, MA.

Hu, X., Cui, N., Demeulemeester, E. & Bie, L. (2015). Incorporation of activity sensitivity measures into buffer management to manage project schedule risk, *European Journal of Operational Research* **249**: 717–727.

Leach, L. P. (2005). *Critical chain project management*, Vol. 2nd, Artech House.

Leu, S. S. & Lin, Y. C. (2008). Project performance evaluation based on statistical process control techniques, *Journal of Construction Engineering and Management* **134**: 813–819.

Lipke, W. (2003). Schedule is different, *The Measurable News Summer*, 31–34.

Lipke, W. & Vaughn, J. (2000). Statistical process control meets earned value, *CrossTalk: The Journal of Defense Software Engineering June*, 16–20,28–29.

Martens, A. & Vanhoucke, M. (2017a). A buffer control method for top-down project control, *European Journal Of Operational Research* **262**: 274–286.

Martens, A. & Vanhoucke, M. (2017b). The integration of constrained resources into top-down project control, *Computers & Industrial Engineering* **110**: 277–288.

Shewhart, W. A. (1931). *Economic control of quality of manufactured product*, Vol. 509, ASQ Quality Press.

Vanhoucke, M. (2014). *Integrated Project Management and Control: First come the theory, then the practice*, Management for Professionals, Springer.

Wang, Q., Jiang, N., Gou, L., Che, M. & Zhang, R. (2006). Practical experiences of cost/schedule measure through earned value management and statistical process control, *Lecture Notes in Computer Science* **3966**: 348–354.

A Metamodel Approach to Projects Risk Management: outcome of an empirical testing on a set of similar projects

F. Minelle[1], F. Stolfi[2], Di Gioacchino[2] and Santini[2]

[1] Computer Science Dept "Sapienza" University, Rome-Italy
minelle@di.uniroma1.it
[2] PRS Planning, Ricerche e Studi, Rome-Italy
stolfi@prsmonitor.it, digioacchino@prsmonitor.it, santini@prsmonitor.it

Keywords: project risk management, e-government, context-based risk analysis, multi-project experimental outcome.

1 Looking for a metamodel, context-based, approach to project risk management

This paper outlines a metamodel approach, context-based, to project risk management based on McFarlan model (McFarlan, 1981, M. Baldini *et al.*, 2002), built by the authors analyzing a set of information technology projects along the entire life cycle. The model considers each project as characterized by a specific risk level, depending on the following risk factors:

- Size (project/product volume);
- Innovation extent (technology, process, organization, and so on) of project products/solutions to be implemented;
- General complexity (impact of induced changes on stakeholders organizations and their relevant operating processes and/or impact on management of contractual constraints and clauses between customer/owner and supplier/contractor).

Evaluating the risk factors, the metamodel allows to identify:

- The **main project risk** (or structural risk);
- The **strategy of most suitable countermeasures** to be implemented in order to restrain negative effects on success criteria values and, as a consequence, on project/product performances;
- The **typical countermeasures**, more focused on the proper action, can be selected progressively depending on suitability level of its own management approach;
- The **specific countermeasures**; considering typical countermeasures generated by metamodel, project manager can identify specific actions to mitigate each project risk.

2 The e-government program launched by the Italian Public Administration and Innovation Department

This model was applied to the Italian e-government program, a process of innovation of Italian local public administrations (Regions, Provinces, Municipalities etc.) initiated and funded in the mid-2.000s. The Program promoted the implementation of projects from Local Public Administrations aimed to delivery e-government services and infrastructure for citizens and firms.

One of the main focus of the e-government program was the implementation of projects not only by individual local public administrations but mainly from a "group of administrations" with the possibility of direct or indirect participation to the program. Indirect participation was about the reuse of products and solutions implemented by other local public administrations.

The funding of the program was €120 million, covering 134 projects (selected out of approximately 400 submitted projects) with a total value of approximately €500 million. The program involved 20 Regions (100%), 93 Provinces (90%), more than 170 Mountain Communities and more than 4,000 Municipalities (49%).

3 The risk survey process on co-financed e-government projects

3.1 Context-based risk analysis and selection of the "most-likely" effective countermeasures

The model structure has the following components:

a. **Summary**: Summary report containing a dashboard of indicators whose values come from the risk analysis carried out at the project;
b. **Detection Model**: Form for classification of project risk factors (by importance). Most of the data concerning these factors are carefully extracted from the executive plan (mandatorily prepared by the proponent entity, according to a predefined standard) and minimally integrated with further data from the analysis and interpretation of the project (done by the authors, as program assessors);
c. **Countermeasures**: Form dedicated to point out the basic countermeasures, suggested by the authors and/or adopted by the project, according to the specific structural risks. In particular, the initial indication of suggested countermeasures was completed during the project implementation, with information concerning their implementation.

The risk analysis process involves the following steps:

- Filling the Detection Model worksheet;
- Filling and checking the Countermeasures Identification and Implementation worksheet;
- Detection of risk indicators through the summary report dashboard.

Detection Model

This form contains a checklist dedicated to detect the project risks; the description of such type of risks is listed in a worksheet table whose columns have the following meanings:

- **Risk factors**: list of the factors to be detected for the purposes of the risk analysis;
- **Drafting criteria**: they represent the evaluation of the correspondent risk factor, according to the project team leader point of view. Each risk factor was evaluated according a scale of 3 values (G=big; M=medium; P=small), with the value limit of each class defined by analyzing the statistical distribution of projects;
- **Source**: field used to indicate if data are extrapolated in objective way from the executive plan or, alternatively, submitted according to a specific interpretation by the project team.

Each element of the checklist contributes to define the criticalities of the projects in terms of Technological Complexity (TC), Organizational Complexity (OC) and Dimension (DIM). In particular:

- the risk factors evaluation such indicated in Detection Model allows to identify the most critical situations;

- the identification of specific strategies for the risk management (to prevent or to control them) allows to get information concerning the types of countermeasures more suitable according the characteristics of the specific project.

The identification of the types of countermeasures has the purpose:

I. To select the prevailing approach, devoted to:
   - contain both organizational issues and integration problems with other initiatives/projects (*IE-External Integration*);
   - mitigate both organizational issues and management problems which are internal to the project itself; such problems also include issues caused by the multiplicity of stakeholders involved in the project (*II-Internal Integration*);
   - ensure formal and rigorous management of the project, either at the initial state and during its execution (*PC-Formal Planning*);
   - control, in qualitative way, processes and products realized within the project (*QC-Formal Quality Assurance and Control*).

II. Selecting the countermeasures mix, to specifically adopt as the best strategy, belonging to the above mentioned types, for risk mitigation (higher results in equal effort). Such approach allow to correlate the assessment of risk factors with the structural risk of the whole project (risk level) and the prominent approach for risk mitigation. In particular:
   a. the **structural risk** of the whole project has been rated on a 5-level qualitative scale (Very Low, Low, Medium, High, Very High);
   b. The evaluation of **risk categories** was represented on an only 2-level scale (low and high) with the aim to reduce the potential combinations generated to identify the strategy for risk mitigation indicated on the risk mitigation approach table;
   c. The **strategy for risk mitigation** was focused taking into account the weighted configuration of the Risk Factor Assessment and it is expressed on a scale of 3-level values (Low, Medium, High) for each management approach (External Integration, Internal Integration, Formal Planning & Control).

Component "Countermeasures"

Each Project Manager uses checklist to self-assess project risk factor identifying main risk and suitability level of each one of the following management approach: IE-External integration, II-Internal integration, PC-Formal project management and QC-Quality assurance or control.

Based on suitability level of each management approach, metamodel allows to identify the typical most suitable countermeasures to each specific project; Project Manager may accept or modify or integrate the suggested typical countermeasures.

3.2 Risk analysis summary and countermeasures actual application rate

The last step of metamodel has to do with verifying the actual compliance of suggested countermeasures and their application rate. In addition to project manager evaluation, metamodel allows independent assessor evaluation, aimed to mitigate subjective evaluation of project manager.

Independent assessor evaluation aimed to understand the suitability of risk management actions identified by project manager. In order to perform that, assessors analyze project documentation and may modify or integrate selected countermeasures defined from project manager.

In order to perform an effective audit of applied countermeasures, metamodel provides a short description for each countermeasure in terms of: [i] *Countermeasure*, name of countermeasure; [ii] *Meaning*, short description of countermeasure; [iii] *Objective evidences*,

examples of objective evidences that should be found to prove countermeasure was actually applied.

For each selected typical countermeasure (defined from project manager or integrated from assessor) the metamodel allows to indicate actual level of applied countermeasure: *0* = not applied; *1* = partially applied, *2* = widely applied; *3* = totally applied.

Matching applied countermeasures versus planned ones allows to define application rate.

3.3 Outcome achieved from more than one hundred similar projects

The metamodel outlined in this paper has been implemented in more than 130 projects; though all of them were aimed to design e-government services to local public administration, they were all different in terms of dimension (volume), cost and duration.

From this experience we can infer two order of results: first result is methodological and it is about a large and coherent application of this model to a large and distributed set of projects; second result is about impact of model on project management performances.

About the first order, we tried the "easy for use" and applicability of model in all projects with different contexts and dimensions. The metamodel contributed to spread risk management culture in project management teams. Moreover, the metamodel countermeasures database has been enhanced by the results of the most common countermeasures applied in the projects.

About the second order, the results analysis, ongoing and final, of e-government program highlighted as projects which applied suitable countermeasures had a positive impact on time constraint (cost was "out of scope" of assessor control and quality was measured expost in terms of stakeholder benefits on about 45 projects), with less delay to achieve the intermediate milestones and to complete the entire project.

Picture below, taken from periodic report of e-government program, shows an example of correlation between suitability of applied countermeasures and projects delay. In that picture we can see as projects with "suitable" or "most suitable" countermeasures application rate have less delay than projects with "not suitable" countermeasure application rate.

For instance, projects which showed (the minority, at a certain time) a "most suitable" countermeasures application rate had, as an average, 82% progress and 12 months delay, while projects which showed (at a certain time) a "not suitable" countermeasure application rate had 60% progress and 16 months delay.

4 Conclusions and the way forward

The unusual case of a set of similar, contemporary and independent projects (more than one hundred), was likely to be an empirical proof of the consistent effectiveness of Risk Management in improving project patterns. While such "experiment" is not easy to be repeated on a so large number of projects, because projects have the characteristic to be a "single shot" items, the authors replicated a similar (and someway more sophisticated) approach while monitoring a large program (in a multi-years, multi-projects, multi-contracts environment) for the ICT reengineering of a main governmental Institution. Diagnosis was excellent, but unfortunately therapy (i.e. countermeasures) not always applied: program stakeholders (owner and contractors) did not "buy" the approach.

Anyway, the proof of evidence about benefit on projects by using the suggested risk management approach (or anyone in the literature) would encourage all the project managers and their sponsor to consider it a mandatory task in performing the job they were assigned to.

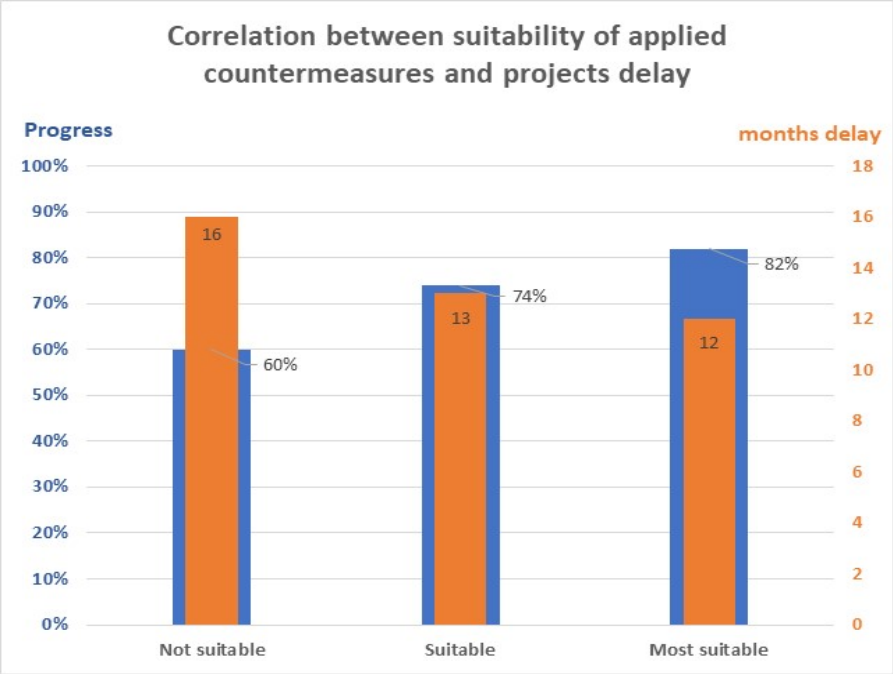


Fig. 1.

Future plans to improve the above-described risk management approach, would consider the paradigm shift for Project Management 2.0 (Kerzner, 2015), in order to insert in the model the evaluation of: (i) soft skill competence in the project team, mainly for the project manager, project team and "sponsor", (ii) communication plan and its contents for the various stakeholder clusters, (iii) consistency of expected benefits, both monetary and not monetary ones. In addition, final correlation between the applied strategy for risk countermeasures and project performance, including also final success (proven benefit for stakeholders) should be thoroughly exploited.

5 Acknowledgements

The authors acknowledge the support of CNIPA (now AgID - Italian Agency for Digitalization) which appointed this activity to PRS as a part of the monitoring engagement to the already mentioned e-government program.

References

McFarlan, 1981, "Porfolio approach to information systems." *Harvard Business Review*, pp. 142-150.

M. Baldini, A. Miola, A. Neri, 2002, *Project management e processi progettuali*, Franco Angeli (6th edition).

H. Kerzner, 2015, *Project Management 2.0: Leveraging Tools, Distributed Collaboration, and Metrics for Project Success*, Wiley (1st edition).

A column generation scheme for the Periodically Aggregated Resource-Constrained Project Scheduling Problem

Pierre-Antoine Morin[1,2], Christian Artigues[2] and Alain Haït[1,2]

[1] ISAE SUPAERO, University of Toulouse, Toulouse, France
pierre-antoine.morin@isae.fr, alain.hait@isae.fr
[2] LAAS CNRS, University of Toulouse, CNRS, Toulouse, France
artigues@laas.fr

Keywords: project, planning, scheduling, periodical aggregation, mixed integer linear programming, column generation.

This abstract is focused on the Periodically Aggregated Resource-Constrained Project Scheduling Problem (PARCPSP) (Morin *et. al.* 2017b), that can be seen as a continuous-time variant of a restricted Resource-Constrained Project Scheduling Problem with partially renewable resources (RCPSP/π) (Böttcher *et. al.* 1999). The purpose of this work is to compare an existing compact formulation with a new extended formulation.

The PARCPSP is defined as follows. A set $\mathcal{A}$ of activities, subject to end-to-start precedence relations $E \subset \mathcal{A} \times \mathcal{A}$, and a set $\mathcal{R}$ of renewable resources are given. During its processing (duration p_i), activity $i \in \mathcal{A}$ requires $r_{i,k}$ units of resource $k \in \mathcal{R}$ (capacity b_k). The scheduling horizon is divided uniformly into a set $\mathcal{L}$ of L periods of length Δ. The PARCPSP can be described by the following abstract model:

$$\text{Minimize} : \ S_{n+1} - S_0 \tag{1}$$

$$\text{s.t.} : \ S_j - S_i \geq p_i \qquad \forall (i,j) \in E \tag{2}$$

$$\sum_{i \in \mathcal{A}} r_{i,k} \frac{d_{i,\ell}(S_i)}{\Delta} \leq b_k \qquad \forall k \in \mathcal{R} \ , \ \forall \ell \in \mathcal{L} \tag{3}$$

Where S_i is the start date of activity i and $d_{i,\ell}(t)$ is the length of the intersection of the intervals $[(\ell-1)\Delta, \ell\Delta]$ and $[t, t+p_i]$. The objective (1) is to minimize the project duration (activities $0/n+1$ are the dummy beginning/end of the project) under precedence constraints (2) and periodically aggregated resource constraints (3): for every resource, in every period, the capacity should not be exceeded on average.

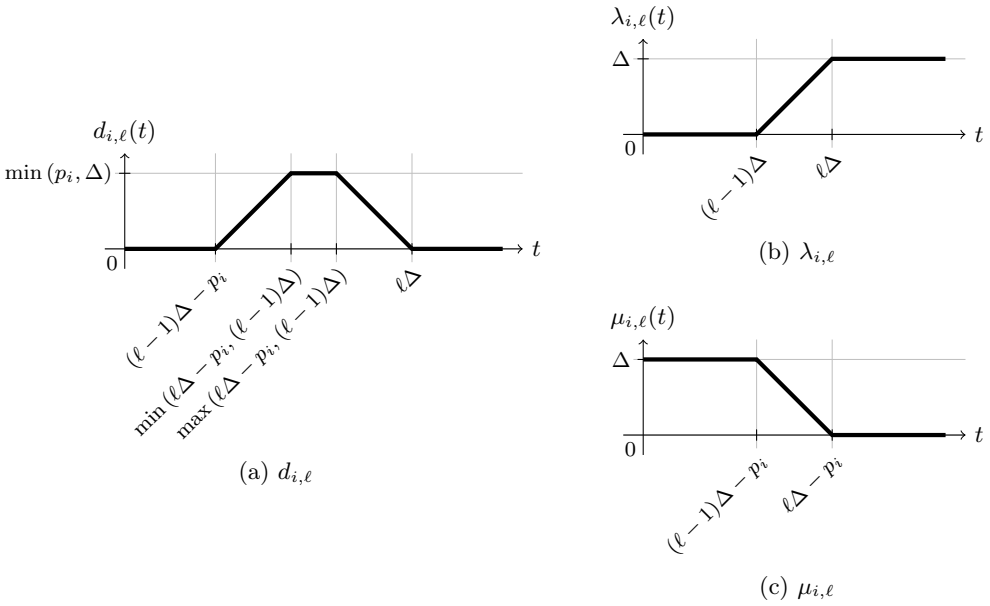
1 Compact model

Two formulations based on mixed (continuous and discrete) time frameworks have been proposed to model the PARCPSP. Although the computation of the values $d_{i,\ell}(S_i)$ can be done by introducing only step binary variables (Morin *et. al.* 2017b), we focus here on an alternative scheme based on period partitionning (Morin *et. al.* 2017a) that requires more continuous variables, but involves less constraints, all big-$\mathcal{M}$-free, thus yielding a better linear relaxation.

Two additional functions are considered. Let $\lambda_{i,\ell}(t)$ be the length of the intersection of the intervals $[(\ell-1)\Delta, \ell\Delta]$ and $(-\infty, t]$; let $\mu_{i,\ell}(t)$ be the length of the intersection of the intervals $[(\ell-1)\Delta, \ell\Delta]$ and $[t+p_i, +\infty)$ (cf. Figure 1).

Notice that it is easier to describe $\lambda_{i,\ell}$ and $\mu_{i,\ell}$ compared to $d_{i,\ell}$. Moreover, the intervals whose lengths are measured by these functions form a partition of period ℓ. Therefore:

$$\lambda_{i,\ell}(t) + d_{i,\ell}(t) + \mu_{i,\ell}(t) = \Delta \qquad \forall i \in \mathcal{A} \ , \ \forall \ell \in \mathcal{L} \ , \ \forall t \in \mathbb{R} \tag{4}$$



(a) $d_{i,\ell}$ (b) $\lambda_{i,\ell}$ (c) $\mu_{i,\ell}$

Fig. 1: Piecewise linear functions $d_{i,\ell}$, $\lambda_{i,\ell}$ and $\mu_{i,\ell}$

The values $d_{i,\ell}(S_i)$, $\lambda_{i,\ell}(S_i)$ and $\mu_{i,\ell}(S_i)$ are represented as continuous variables $D_{i,\ell}$, $\Lambda_{i,\ell}$ and $M_{i,\ell}$, respectively. To model the piecewise linear functions $\lambda_{i,\ell}$ and $\mu_{i,\ell}$, auxiliary binary variables are introduced; more precisely, to ensure a non-increasing (resp. non-decreasing) step behavior of the variables $\Lambda_{i,\ell}$ (resp. $M_{i,\ell}$), step binary variables $z^{\lambda}_{i,\ell}$ (resp. $z^{\mu}_{i,\ell}$) are required.

$$\text{Minimize} \ : \ S_{n+1} - S_0 \tag{5}$$

$$\text{s.t.} \ : \ S_j - S_i \geq p_i \qquad \forall (i,j) \in E \tag{6}$$

$$\sum_{i \in \mathcal{A}} r_{i,k} D_{i,\ell} \leq b_k \Delta \qquad \forall k \in \mathcal{R} \ , \ \forall \ell \in \mathcal{L} \tag{7}$$

$$S_i = \sum_{\ell \in \mathcal{L}} \Lambda_{i,\ell} \qquad \forall i \in \mathcal{A} \tag{8}$$

$$D_{i,\ell} = \Delta - \Lambda_{i,\ell} - M_{i,\ell} \qquad \forall i \in \mathcal{A} \ , \ \forall \ell \in \mathcal{L} \tag{9}$$

$$D_{i,\ell} \geq 0 \qquad \forall i \in \mathcal{A} \ , \ \forall \ell \in \mathcal{L} \tag{10}$$

$$\sum_{\ell \in \mathcal{L}} D_{i,\ell} = p_i \qquad \forall i \in \mathcal{A} \tag{11}$$

$$z^{\lambda}_{i,\ell+1} \leq \frac{\Lambda_{i,\ell}}{\Delta} \leq z^{\lambda}_{i,\ell} \qquad \forall i \in \mathcal{A} \ , \ \forall \ell \in \mathcal{L} \tag{12}$$

$$z^{\mu}_{i,\ell-1} \leq \frac{M_{i,\ell}}{\Delta} \leq z^{\mu}_{i,\ell} \qquad \forall i \in \mathcal{A} \ , \ \forall \ell \in \mathcal{L} \tag{13}$$

$$z^{\lambda}_{i,\ell} \in \{0,1\} \qquad \forall i \in \mathcal{A} \ , \ \forall \ell \in \mathcal{L} \tag{14}$$

$$z^{\mu}_{i,\ell} \in \{0,1\} \qquad \forall i \in \mathcal{A} \ , \ \forall \ell \in \mathcal{L} \tag{15}$$

The objective (5) is to minimize the project duration, under both precedence constraints (6) and periodically aggregated resource constraints (7). Constraints (8) enable the computation of start dates S_i directly from $\Lambda_{i,\ell}$ variables, while constraints (9), derived from

the partition relation (4), enable the computation of $D_{i,\ell}$ values that cannot be negative [constraints (10)]. Constraints (11) permit to balance the values of $\Lambda_{i,\ell_i^\lambda}$ and M_{i,ℓ_i^μ}, where ℓ_i^λ (resp. ℓ_i^μ) is the period activity i starts (resp. completes) in. Finally, constraints (12) [resp. (13)] enforce an interdependent non-increasing (resp. non-decreasing) step behavior of variables $\Lambda_{i,\ell}$ and $z_{i,\ell}^\lambda$ (resp. $M_{i,\ell}$ and $z_{i,\ell}^\mu$) using binary variables [constraints (14) and (15)]. Therefore, every variable $\Lambda_{i,\ell}$ (resp. $M_{i,\ell}$) with $\ell \neq \ell_i^\lambda$ (resp. $\ell \neq \ell_i^\mu$) is bound either to 0 or Δ, as shown in Figure 2.

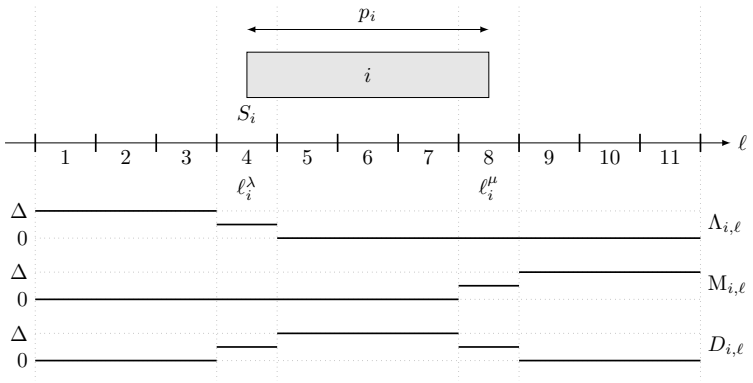


Fig. 2: Partition-based mixed time framework

2 Dantzig-Wolfe decomposition

We now introduce a new extended formulation that enhances and exploits the combinatorial structure of the PARCPSP. On the one hand, the (restricted) master problem consists in selecting start dates $t \in \mathcal{T}_i$ for every activity $i \in \mathcal{A}$ (binary decision variables $x_{i,t}$ such that $S_i = \sum_{t \in \mathcal{T}_i} t\, x_{i,t}$, $\forall i \in \mathcal{A}$) in such a way that all constraints are satisfied, while minimizing the project duration. On the other hand, the sub-problem consists in finding time points t to insert into sets $\mathcal{T}_i$. Notice that, although the start date of an activity can be any (real) time point in the (continuous) interval $[0, L\Delta - p_i]$, only a finite number of them need to be considered, since optimal solutions match extreme points of a polytope described by a finite number of constraints.

2.1 Master problem

$$\text{Minimize} \;:\; \sum_{t \in \mathcal{T}_{n+1}} t\, x_{n+1,t} - \sum_{t \in \mathcal{T}_0} t\, x_{0,t} \tag{16}$$

$$\alpha_i \;:\; \sum_{t \in \mathcal{T}_i} x_{i,t} = 1 \qquad \forall i \in \mathcal{A} \tag{17}$$

$$\beta_{i,j} \;:\; -\sum_{t \in \mathcal{T}_j} t\, x_{j,t} + \sum_{t \in \mathcal{T}_i} t\, x_{i,t} \leq -p_i \qquad \forall (i,j) \in E \tag{18}$$

$$\gamma_{k,\ell} \;:\; \sum_{i \in \mathcal{A}} \sum_{t \in \mathcal{T}_i} r_{i,k}\, d_{i,\ell}(t)\, x_{i,t} \leq b_k\, \Delta \qquad \forall k \in \mathcal{R}\ ,\ \forall \ell \in \mathcal{L} \tag{19}$$

$$x_{i,t} \in \{0,1\} \qquad \forall i \in \mathcal{A}\ ,\ \forall t \in \mathcal{T}_i \tag{20}$$

The objective (16) is to minimize the project duration, assigning a unique start date to each activity [constraints (17)], under both precedence constraints (18) and periodically aggregated resource constraints (19), using binary variables [constraints (20)].

Notice that dual variables $\beta_{i,j}$ and $\gamma_{k,\ell}$ are non-negative. The linear relaxation of the master problem is obtained by replacing constraints (20) with "$x_{i,t} \geq 0$"; notice that constraints α_i imply "$x_{i,t} \leq 1$".

2.2 Sub-problem

$$\text{Minimize} \ : \ \alpha_i + \sum_{j \in E_i^{\oplus}} \beta_{i,j}\, t - \sum_{j \in E_i^{\ominus}} \beta_{j,i}\, t + \sum_{k \in \mathcal{R}} \sum_{\ell \in \mathcal{L}} \gamma_{k,\ell}\, r_{i,k}\, d_{i,\ell}(t) \tag{21}$$

$$ES_i \leq t \leq LS_i \tag{22}$$

Where, for each activity $i \in \mathcal{A}$: $E_i^{\oplus} = \{j \in \mathcal{A} \ : \ (i,j) \in E\}$ (set of direct successors of i), $E_i^{\ominus} = \{j \in \mathcal{A} \ : \ (j,i) \in E\}$ (set of direct predecessors of i), ES_i and LS_i are respectively the earliest and latest starting time of i (those input values are typically obtained by computing longest paths in the activity precedence graph).

Given an activity $i \in \mathcal{A}$, the sub-problem returns a candidate start t within the horizon [constraint (22)] such that the new variable $x_{i,t}$ has the least reduced cost [objective (21)]. This returned date t will be inserted in $\mathcal{T}_i$ in the restricted master problem only if needed, i.e., if the reduced cost of $x_{i,t}$ is negative.

Notice that, after the partition relation (4), the reduced cost of $x_{i,t}$ can be transformed into a sum of continuous monotonic piecewise linear functions of t. Therefore, the sub-problem can be solved by a forward algorithm, linear in the number of breakpoints, hence linear in the number of periods.

Computational experiments will be provided by time of the conference. Depending on the results, it could be interesting to additionally separate either precedence or periodically aggregated resource constraints. For instance, the framework proposed by Mingozzi *et. al.* (1998) for the standard Resource-Constrained Project Scheduling Problem (RCPSP) could be adapted to the case of the PARCPSP. The precedence constraints are managed by the master problem, while the resource constraints are managed by the sub-problem. Instead of using vector columns with binary components indicating whether an activity is processed in a unit time period, these components should be replaced with real values in the interval $[0, \Delta]$ indicating how much each activity is processed in a period of length Δ.

References

Mingozzi A., Maniezzo V., Ricciardelli S. and Bianco L., 1998, "An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation", *Management Science*, Vol. 44, pp. 714–729.

Böttcher J., Drexl A., Kolish R. and Salewski F., 1999, "Project scheduling under partially renewable constraints", *Management Science*, Vol. 45, pp. 543–559.

Morin P.A., Artigues C. and Haït A., 2017a, "A new mixed time framework for the Periodically Aggregated Resource-Constrained Project Scheduling Problem", Proceedings of the *13th Workshop on Models and Algorithms for Project Scheduling Problems* (MAPSP 2017), Seeon Seebruck, Germany.

Morin P.A., Artigues C. and Haït A., 2017b, "Periodically Aggregated Resource-Constrained Project Scheduling Problem", *European Journal of Industrial Engineering*, Vol. 11, No. 6, pp. 792–817.

Development of a Schedule Cost Model for the Resource Constrained Project that incorporates Idleness

Babatunde Omoniyi Odedairo and Victor Oluwasina Oladokun

Department of Industrial and Production Engineering, University of Ibadan, Ibadan, Nigeria
{bo.odedairo,vo.oladokun}@ui.edu.ng

Keywords: project management, activities scheduling, resource-constrained, idleness cost.

1 Introduction

Scheduling project activities is a challenging decision-making process because such decisions must cope with physical, technological and resource availability constraints. In the classical resource-constrained project scheduling problem (RCPSP), the aim is to determine the start and finish times for all project activities within the specified precedence relationship and resource constraints such that maximum completion time can be minimised. Möhring (1984) termed this problem as 'problem of scarce resources'.

In practice, resource availability is often faced with conditions arising from the remoteness of project location, logistics cost of resource transportation, and costs associated with hiring and releasing renewable resources (Sears *et al.*, 2008). On the decision to release and rehire renewable resources, this is practicable with some resources (e.g. unskilled labour) and it is usually carried out to simultaneously meet daily manpower needs and eliminate idleness (or waste). On the other hand, for resources which comes at a high hiring rate (or are capital intensive) and are used from project start to finish not by a single activity but by several activities, the decision to release becomes more complicated as it may not fit in with activity resource requirements (Akpan, 1997; Vanhoucke, 2007; Odedairo, 2016). Therefore, in a resource constrained project management environment; a replacement strategy should be planned for resources that require uninterrupted usage and are jointly used by a group of activities. This is necessary in order to forestall the following: (i) some skilled workers (or machineries) released for another job may not return on time which can cause delay in job processing, (ii) there is no guarantee that the same set of resources will be hired, and (iii) the time to engage new resources might not be available. Hence, it becomes necessary to decide on the minimum level of additional renewable resources to hire and hold (with all costs implication) throughout a reasonable time period or for the entire project duration. Also, while being held, the usage of a resource will differ in one or more time intervals due to precedence constraints among project activities.

With this reality comes resource use-time and idle-time (and associated costs of usage and idleness). The cost implications of resource usage and idleness times in this research is assumed to have the same features as the time-dependent costs (TDC) introduced by Gong (1997) and further elaborated and explained by Goto *et al.* (2000) and Vanhoucke (2006) respectively. In this study, the objective is to characterise RCPSP within the context of idleness cost (IC) arising from the use of additional hired resources (with TDC features) held throughout the project makespan. Thereafter, a mathematical model that focuses on the minimisation of the total schedule cost for the resource constrained project scheduling problem with idleness cost (RCPSP-IC) will be developed to represent the essence of the decision problem. The remainder of this paper is structured as follows. In section 2, related work will be discussed. The mathematical models are presented in section 3 while in section 4, preliminary solution approach and results are discussed.

2 Related literature

Imreh and Noga (1999) investigated how scheduling problems change when machine (resource) costs are considered. They argued that resource usage has associated cost, and if the required resources are not available, then such can be procured or hired. Other studies have been carried out on the impact of machine/resource cost on scheduling decisions (Imreh, 2009; Ruiz-Torres *et al.*, 2010). In their study, Ruiz-Torres *et al.* (2010) identified two ways in which resource cost can be conceptualised and modeled as components of the scheduling process. These are (i) using the duration of time required to process an activity on a resource i.e. resource use-time and (ii) number of resources used.

As stated earlier, in projects; situations often arise when activity processing requires uninterrupted availability and usage of specialised resources. Such a resource could be said to be critical (or a bottleneck), in this context; a resource could be critical if it offers specialised skills/services and its availability is constrained because it is capital intensive. Furthermore, in their usage; inefficiencies such as resource idle-time may be encountered due to predefined precedence constraints between activities in the project. In literature, the problem of idle-time of resources due to processing of repetitive activities from unit to unit and within-unit has been researched (Harris and Ioannou, 1998; Vanhoucke, 2007).

El-Rayes and Moselhi (1998) as cited by Vanhoucke (2013) define the term "work continuity constraints" as a way to schedule repetitive units of a project to enable timely movement of resources from unit to unit to minimise total resource idle time. Vanhoucke (2007) concluded that the minimisation of resource idle time for a work continuity optimisation involves a trade-off between project completion and cost of idle time. Although, work continuity constraints is widely known with repetitive projects; Vanhoucke (2007) opined that in non-repetitive projects, uninterrupted usage of important resources e.g. specialized consultants, etc. can also pose problem of idle time minimisation.

Therefore, for a RCPSP-IC; the decision on the number of additional resources to hire and consequently hold to minimise total resource idle time should be considered during project planning phase. To the best of our knowledge, there is no study available in which the number of additional TDC resources in a RCPSP-IC is defined as a decision variable.

3 Problem abstraction and mathematical model

The RCPSP-IC can be stated as follows. Consider a set of activities, n, with index $j = 1, \ldots, n$ numbered from a dummy start and end node of 0 and $n+1$ respectively. Each activity j has the following information: an activity is to be processed on X renewable resources (with an index of $m = 1, \ldots, X$); once started, the processing cannot be interrupted. There is a finish-start precedence relationship with zero time-lag between activities which enforces each activity to be scheduled after all its predecessors are completed. The precedence relationship between activities is depicted by activity-on-node (AON) network. For each activity, its processing time is independent of the schedule and can only be executed in a single mode composed of a fixed duration and renewable resource requirements.

For renewable resources, each resource has the following characteristics: a resource cannot process more than one activity at a time; a pre-specified unit of resource $m = X$ is available for every period of the project horizon. It is assumed that the project will require additional hired renewable resources (K) with TDC features. Furthermore, the additional resources are assumed identical, held from project start to finish with service time equivalent to the project makespan. The identical nature of the TDC resources allows for the possibility of parallel processing.

3.1 Relationship between Cost of Project Schedule and Number of TDC resources

Since activity scheduling constitutes the core of cost minimisation in project management, any strategic plan to minimise cost must be centred on determining a good schedule. Therefore, for a resource, its time-dependent cost is equivalent to the product of cost of hiring per time (hours, days and weeks) unit and its service (usage) time. For a TDC resource hired and held throughout the project lead time, cost interpretations of such decision is shown in Figure 1.

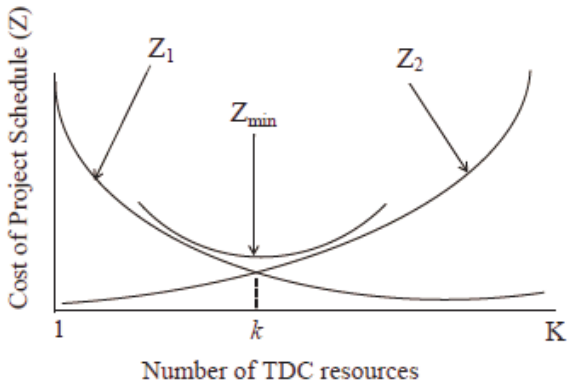


Fig. 1. Relationship between cost of project schedule and number of TDC resources.

In Figure 1, let Z_1, be the cost attributable to project completion time, and Z_2, the cost attributable to SRIT (now termed resource idleness cost). Two scenarios are possible, the first is the availability of one TDC resource, in this case, the sum of resource idleness time (SRIT) will be minimum (zero) because the single resource is assumed to be continuously busy; however, the project completion time ($C_{\max}$) will be maximum. The second scenario involves multiple TDC resources; it is obvious that some or all the resources will be idle during one or more time intervals of the project execution due to precedence constraints between activities. In this case, project completion time is minimised while SRIT ($k = 1, \ldots, K$) is assumed to be maximum.

Arising from the two scenarios, their cost implications can be depicted from Figure 1, the total cost of project schedule (Z) is assumed to be a combination of two independent components (Z_1 and Z_2). The relationship between the behaviour of Z_1 and Z_2 with respect to available number of TDC resources ($k = 1, \ldots, K$) can be further conceptualised to show that, Z_1 is a function $C_{\max}$ and invariably a function of schedule (σ) and, for Z_2, it is a function of $C_{\max}$ and k.

Therefore, for the RCPSP-IC, the minimum total schedule cost ($Z_{\min}$) can be expressed as shown in equations (1)–(3).

$$Z_1 = f(C_{\max}) \equiv f(\sigma) \tag{1}$$

$$Z_2 = f(C_{\max}, k) \equiv f(\sigma, k) \tag{2}$$

$$Z_{\min} = f(\sigma) + f(\sigma, k) \tag{3}$$

In equation (3), the minimum schedule cost for RCPSP-IC is a function of the schedule and number of TDC resources.

3.2 Mathematical model of Resource Idleness Cost (RIC)

To model resource idleness cost (RIC), the schedule (σ) and number of TDC resources (k) are defined as decision variables as described in equation (2). Before RIC model is presented, some notations used and their definitions will be explained. p_j is activity processing time (in days); the assignment variable y_{jk} (1 = activity j is being processed on resource k and 0 = otherwise); C_{day} is cost per time unit paid during each day of the project; $s = 1, \ldots, S$ is index for schedule; $t = 1, \ldots, T$ is index for time periods; $RS^k_{t(use-time)}$ is resource use-time for k; $RS^k_{t(idle-time)}$ is resource idle-time for k; R_k is per period availability of resource k; r_{jk} is the resource unit required by activity j being processed by k in each period; CR_k is the cost of using TDC resource k per time unit (i.e. hiring cost per day).

The cost associated with resource idleness time is presented in equation (4)–(8). As explained, each TDC resource is expected to be held throughout the project makespan (either used or idle); hence, each resource will have a duration equivalent to $C_{\max}$ (in days) as presented in equation (4). In equation (5), the idle time component of equation (4) is obtained.

$$C_{\max}(\sigma_s) = RS^k_{t(use-time)} + RS^k_{t(idle-time)} \tag{4}$$

$$RS^k_{t(idle-time)} = C_{\max}(\sigma_s) - RS^k_{t(use-time)} \tag{5}$$

If activity j ($j = 1, \ldots, n$) with processing time p_j can be processed by TDC resource k, then, $RS^k_{t(use-time)}$ for resource k is given by equation (6).

$$RS^k_{t(use-time)} = \sum_{j=1}^{n} p_j y_{jk} \tag{6}$$

Therefore for $k = 1, \ldots, K$, the sum of resource idle time (SRIT) in days can be mathematically expressed as shown in equation (7).

$$\text{SRIT} = \sum_{k=1}^{K} \left[C_{\max}(\sigma_s) - \sum_{j=1}^{n} p_j y_{jk} \right] \tag{7}$$

From equation (7), RIC can be expressed as shown in equation (8).

$$\text{RIC} = \left[CR_k \left(\sum_{k=1}^{K} \left[C_{\max}(\sigma_s) - \sum_{j=1}^{n} p_j y_{jk} \right] \right) \right] \tag{8}$$

3.3 RCPSP-IC model

The total schedule cost for RCPSP-IC is described in equation (9). The first and second components are the cost attributable to project completion time and to SRIT respectively.

$$TSC_{(\sigma,k)} = C_{\max}(\sigma_s) \cdot C_{day} + \left[CR_k \left(\sum_{k=1}^{K} \left[C_{\max}(\sigma_s) - \sum_{j=1}^{n} p_j y_{jk} \right] \right) \right] \tag{9}$$

subject to

$$FT_i \leq FT_j - p_j, j = 1, \ldots, n, \ \forall i \in IP_j, \ i \rightarrow j \tag{10}$$

$$\sum_{j \in A(t)} r_{jk} \leq R_k, \ k = 1, \ldots, K, \ t = 1, \ldots, T \tag{11}$$

Due to the usage of variable of the classical RCPSP (Pritsker *et al.*, 1969), RCPSP-IC is subjected to all of constraints already established in RCPSP. Two of these constraints are described in equation (10)–(11). In equation (10), the precedence relations between activities is enforced (where FT is finish time of activity; i for predecessor and j for successor). Equation (11) ensure that resource consumption by each activity $j = 1, \ldots, n$ does not exceed the limit per unit time.

4 Preliminary solution approach and results

A Serial Schedule Generation Scheme with latest finish time (LFT) as priority rule was preliminary used to generate good schedules. The LFT priority is logically feasible because an activity's predecessor must have an earlier late finish time and so appears earlier in the priority list. In addition, an idleness calculator (IDCalc) was incorporated into the procedure which computes idle-time for the TDC resources at every time interval of the project horizon. The computer implementation of the procedure was developed using MATLAB. Data from a real-life project management situation were collected and the associated problem solved as RCPSP-IC. For each TDC resource level k ($k = 8$ to 13), the best schedule (σ) was obtained, keeping σ ($\sigma = 1$ to 6) constant, resource level was varied to reflect levels of resource availability. Thirty-six (36) pairs of (σ, k) were formulated and for each pair, Total Schedule Cost (TSC), Cost attributable to project completion time (Z_1) and Resource Idleness Cost (Z_2) were calculated respectively. A conflicting relationship exists between Z_1 and Z_2. Z_1 decreased (increase) with increase (decrease) in resource level. Z_2 increased (decrease) with increase (decrease) in resource level. Hence, TSC for RCPSP-IC was influenced by both the schedule and number of TDC resources.

References

Akpan E.O.P., 1997, "Optimum resource determination for project scheduling", *Journal of Production Planning and Control*, Vol. 8, pp. 462–469.

El-Rayes, K., and Moselhi, O., 1998, "Resource-driven scheduling of repetitive activities", *Journal of Construction management and Economics*, Vol. 16, pp. 433–446.

Gong, D.,1997, "Optimization of float use in risk analysis-based network scheduling", *International Journal of Project Management*, Vol.15, pp. 187–192.

Goto E., Joko, T., Fujisawa, K., Katoh, N., and Furusaka, S., 2000, "Maximizing net present value for generalized resource constrained project scheduling problem", Nomura Research Institute, Tokyo, Japan.

Harris, R. B., and Ioannou, P. G., 1998, "Scheduling projects with repeating activities", *Journal of Construction Engineering and Management*, Vol. 124, pp. 269–278.

Imreh, Cs., 2009, "Online scheduling with general machine cost function", *Discrete Applied Mathematics*, Vol. 157, pp. 2070–2077.

Imreh, Cs. and Noga, J. 1999, "Scheduling with machine cost", in: *Proceedings of APPROX99*: pp. 168–176.

Möhring, R.H., 1984, "Minimizing costs of resource requirements in project networks subject to a fixed completion time", *Operations Research*, Vol. 32, pp. 89–120.

Odedairo B.O., 2016, "Development of Scheduling heuristic for the Resource Constrained Project Management Problem with Idleness Cost", Unpublished Ph.D Thesis, University of Ibadan.

Pritsker, A., Allan. B., Watters, L.J. and Wolfe, P.M. 1969, "Multiproject scheduling with limited resources: A zero-one programming approach", *Management Science*, Vol. 16. pp. 93–108.

Ruiz-Torres, A.J., López, F.J., Wojciechowski P.J. and Ho, J.C., 2010, "Parallel machine scheduling problems considering regular measures of performance and machine cost", *The Journal of the Operational Research Society*, Vol. 61(5), pp. 849–857.

Sears, S. K., Sears, G.A. and Clough, R. H., 2008, *Construction project management: a practical guide to field construction management*. 5th ed. New York: John Wiley & Sons.

Vanhoucke, M., 2006, "Work continuity constraints in project scheduling", *Journal of Construction Engineering and Management*, Vol. 132, pp. 14–25.

Vanhoucke, M., 2007, "Work continuity optimization for the Westerscheldetunnel project in the Netherlands", *Tijdschrift voor Economie en Management*, Vol. 52, pp. 435–449.

Vanhoucke, M., 2013, "Project baseline scheduling: An overview of past experiences", *Journal of Modern Project Management*, Vol. 1, pp. 18–27.

Optimization problems in intermodal transport

Erwin Pesch

Universität Siegen
erwin.pesch@uni-siegen.de

1 Abstract

In intermodal container transportation, where containers need to be transported between customers (shippers or receivers) and container terminals (rail or maritime) and vice versa, transshipment of containers is commonly arranged at the terminals. Attracting a higher share of freight traffic on rail requires freight handling in railway terminals that is more efficient, and which includes technical innovations as well as the development of suitable optimization approaches and decision-support systems. In this talk we will review some optimization problems of container processing in railway yards, and analyze basic decision problems and solution approaches for the two most important yard types: conventional rail-road and modern rail-rail transshipment yards. Furthermore, we review some of the relevant literature and identify open research challenges. Additionally we address a container dispatching and conflict-free gantry crane routing problem that arises at a storage container block in an automated, maritime container terminal. A container block serves as an intermediate buffer for inbound and outbound containers and exchanges of containers between water- and landside of a maritime terminal. The considered block is perpendicular to the waterside and employs two rail mounted gantry cranes. Cranes may have the same or different sizes and therefore either are based at the opposite sides of the container block or can cross each other. The question arises in which order and by which crane containers are transported in order to minimize the makespan and prevent crane conflicts.

The Stakeholder Perspective: how management of KPIs can support value generation to increase the success rate of complex projects

Massimo Pirozzi

Istituto Italiano di Project Management, Rome, Italy
pirozzi@isipm.org

Keywords: stakeholder, requirements, expectations, satisfaction, value, success, complexity, measures, KPI, CSF, business.

1 Abstract

In today's world, growing complexity demands that projects, in order to be successful, have to satisfy not only stakeholder requirements, which refer to cost, time, and delivered quality, but also stakeholder expectations, which refer directly to the capability of generating proper business value. Since business value can be measured only after project completion, there is the need, during project life cycle, to handle value indicators: management of proper Key Performance Indicators turns out to be a powerful Project Management tool, which can be effectively used to increase the success rate of complex projects.

2 Stakeholder, who is this?

The word "stakeholder" dates back to the beginning of the eighteenth century, meaning the person who was entrusted with the stakes of bettors, and, then, who was holding all the bets placed on a game or a race, and, moreover who was paying the money to the winners: therefore, the first stakeholder was a "holder of interests". In addition, it is believed that the first modern meaning of stakeholders, which has been attributed (Freeman, 1984) to an internal memorandum of Stanford University Research Center dated 1963, was "those groups without whose support the organization would cease to exist", while in the first text on the theory of stakeholders (Freeman, 1984), the definition of stakeholder was "a stakeholder in an organization is any group or individual who can affect or is affected by the achievement of the organization's objectives". Ten years later (Freeman, 1994), the concept of generated value was added too, and stakeholders were defined as "participants in the human process of joint value creation". Furthermore, starting from the second half of the eighties, the theory of stakeholder management, which was focused on corporate social responsibility, incorporated an important ethical component into the concept of stakeholder.

Definitively, a stakeholder, or an interested party, is a person, or a group of persons, or an organization, that: has some kind of interest in the project; may affect the project, or may be affected by the project; participates, or would like to participate, in the project; can bring a value, which could be either positive or negative, to the project; may have responsibilities towards the project, which, in turn, is supposed to satisfy stakeholders' requirements and expectations.

Each project could then include a large variety of stakeholders, as, for example, project manager, project team, sponsor, funders, partners or shareholders, customers, users, business partners, suppliers, authorities, regulatory bodies, central and local public administration, potential customers and users, participants and candidates to participate in the

project, local communities, web communities, associations, trade unions, media, competitors, and so forth.

3 The stakeholder perspective and the value of project stakeholder relations

All the project stakeholders are important, since all the stakeholders are central towards each project (Pirozzi, 2017): the stakeholders are both the actors, and the beneficiaries, of the project, and the stakeholders are the critical success factor of the project, since they are both the realizers of the results, and the validators, at various levels, of their satisfaction in terms of needs and expectations. In fact, stakeholders, including the project manager and the project team, are the doers of the project, as well as stakeholders, including customers, users, and funders, are the target groups of the project itself: business is the domain in which various stakeholders (project manager, project team, project management office, sponsor, board, shareholders, customers, users, suppliers, investors, central and local public administration, groups of opinion, local communities, and so forth) interact to create and exchange value. The relationships between the project stakeholders are, then, real and proper business relationships, which are associated with the generation, and the exchange, of both material and immaterial value: in general, this flow of value, among the stakeholders, courses through the project with a continuous exchange of resources and results.

In fact, organizations define strategies, which are based on their own mission and vision, then select opportunities in accordance with defined strategy, then set business cases up, and, finally, start projects up. The inputs of a project, and, specifically, to the project management initiating process group, include business case, contract, and Statement of Work (The International Organization for Standardization, 2012): generally, of course, there are different business cases for different stakeholders, as, for instance, providers and customers are. While business cases, which are the causes of project start-up, are based on stakeholder business expectations, whose satisfaction correspond to the achievement of project goals, contract and SOW, which are the references for project development and delivery, are based on stakeholder requirements, which are, in turn, the conversion of different stakeholder expectations in a commonly agreed (at least initially) project scope, and whose fulfilment correspond to the achievement of project objectives.

A project can be considered really successful when its goals are realized, then achieving those results that correspond to the stakeholder expectations, and which are characterized by a satisfactory perceived quality; on the other hand, in order to realize the expectations of stakeholders (project goals), each project must necessarily achieve its objectives, by realizing those deliverables that fulfill stakeholder requirements, and which are characterized by a proper delivered quality. Effective Stakeholder Management should target the satisfaction of both stakeholder requirements and expectations, which corresponds to the achievement of both project goals and objectives (Figure 1): stakeholder satisfaction, instead of being "a" critical success factor, proves to be "the" critical success factor; in fact, projects may not succeed their goals, or may fail at all, for various reasons, which could be technically very different, but, for sure, each project that was not successful had at least one key stakeholder whose expectations were not satisfied.

In Stakeholder Management, then, effective management of both the domain of "deliverable", which is based on delivered contents, and of the domain of "perceivable", which is based on relations, becomes essential: the realization of the expectations of the stakeholders, which, of course, implies also their acceptance of the deliverables, is therefore a primary goal of the project, and it coincides with the most important critical success factor (Pirozzi, 2017). In any case, the stakeholder relations are the core of the project value, since they

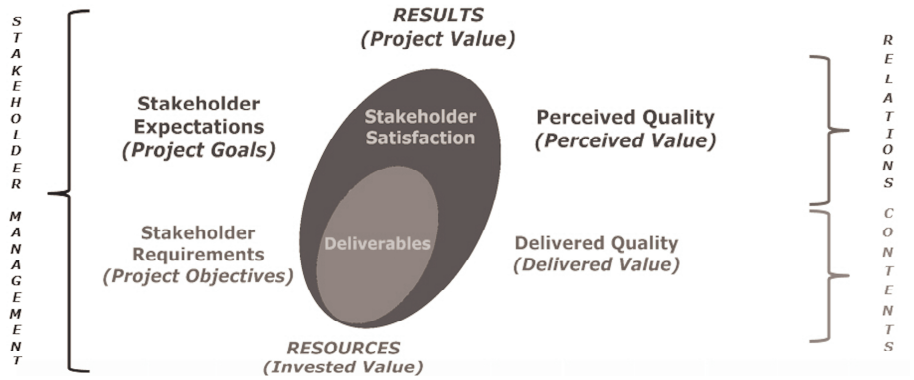


Fig. 1. The Stakeholder Perspective (Pirozzi, 2017).

are a value, which is fundamental to the existence of the project and to its definition, but also since they generate value, which is incorporated in the project, and because they allow the exchange of value, through the project results, among the stakeholders themselves: the results of a project are, in fact, the results of the relations among its stakeholders. Stakeholder perspective, ultimately, supports and determines project success: «The emphasis on Relationship Management is of special importance in today's world» (Archibald, 2017).

4 Achieving the planned business value: the success factor in complex projects

PMI's 2017 Global Project Management Survey (Project Management Institute, 2017) reported that more of 30% of the projects do not meet their original goals and business intent, i.e. they do not satisfy stakeholder expectations: therefore, the attention to the satisfaction of stakeholder expectations must be considered as a critical factor, rather than as a simple warning. In today's Project Management, Stakeholder Management becomes, then, the crucial process group, since it targets effectively the project success, by supporting the generation of that project value which could satisfy both stakeholder requirements and stakeholder expectations: if we use the perspective of project success, we can distinguish two cases, the "classical" projects, and the "complex" projects.

In "classical" projects: project is part of customer core business (as, e.g., in internal or in outsourcing projects), and/or project deliverables are product oriented, and/or are tangible (as, e.g., in infrastructure projects), and/or, in any case stakeholder requirements are either well defined (traditional contexts) or are evolutive, but all stakeholders cooperate effectively (agile contexts); triple constraints (time, cost, quality) are dominant; relations with stakeholders are important, and periodical. In classical projects, success is based on the satisfaction of stakeholder requirements: in fact, there is just a small gap between the satisfaction of requirements and the satisfaction of expectations, and, then, the measures of the value could be limited to the measures of costs and of consistency/state of progress of the deliverables, as usually happens in traditional/agile Project Management.

On the other hand, in "complex" projects: project is a support of customer core business (as, e.g., in the majority of external projects), and/or project deliverables are oriented to services, and/or are intangible (as, e.g., in software projects), and/or, in any case, stakeholder requirements are either not well defined or are evolutive, but not all stakeholders cooperate effectively; competing constraints are dominant, so that value and reputation overcome triple constraints (Kerzner, 2015); relations with stakeholders are primary, and can be continuous, fast, interactive (2.0), evolutionary (Kerzner, 2015). In complex projects,

success is based on the satisfaction of stakeholder expectations (Figure 1): since there is a significant gap between the satisfaction of requirements and the satisfaction of expectations, the measures of the value must include the measure of *business value*, too. Definitively, in each complex context, «Success is not necessarily achieved by completing the project within the triple constraint. Success is when the planned business value is achieved within the imposed constraints and assumptions.» (Kerzner and Saladis, 2009).

5 Managing effectively business value by use of Key Performance Indicators

Value management requires measures: during project life cycle, the measure of actual cost and the assessment of the state of progress of the deliverables are commonly used as indicators to estimate time and cost of the project completion, while the measures of the generated business value, which is "future" with respect to project life cycle, could be done, unfortunately, only after project completion. Therefore, since, during project life cycle, the measures of business value are not possible, there is the need of the support of indicators, which could be used to estimate both the current situation and the possible evolution of the business value: proper Key Performance Indicators (KPIs) are then required. In each complex project, an effective Stakeholder Management is thus based on measuring, monitoring, and sharing, value-driven specific Key Performance Indicators: KPIs have to be S.M.A.R.T (Specific, Measurable, Attainable, Realistic, Time Related), but also few, relevant, actionable, and predictive, and can be shared continuously, quickly, and effectively with stakeholders through dashboards, which can often replace efficiently traditional reports (Kerzner, 2015). Moreover, the use of dashboards can be effective also in several cases of reluctant, indifferent, and negative/hostile, stakeholders, because dashboards generally ask only for answers yes/no, and no-answers can be interpreted positively, too.

Since stakeholders are different, they have different behaviour, and they target different values: while "providers" (Project Manager, project team, etc.) target technical (delivered) values, which are typical of Project Management, as triple constraints, project objectives, and revenues, "investors" (top management, funders etc.) target economic values, as costs, revenues, and business prospects, and "purchasers" (customers, users etc.) target business values, as customer costs (that correspond to providers/investors revenues), project goals, and benefits achievement. Therefore, effective KPIs have to target different types of value, which refer to both Project Management, economic, and business domains. Examples of Project Management KPIs include Earned Value, Cost Performance Index, Schedule Performance Index, percentages of completed work packages compared to those which have been planned, percentages of critical work packages which are aligned to the budget and/or to the schedule, numbers and percentages relating to resources, risks, revisions, to requests for change and changes etc.; examples of economic KPIs include economic, financial, marketing, CRM, operational, HR, and sustainability indicators; examples of business value indicators include specific functional and/or quantitative measures, and relevant percentages of completion and/or of deviation from budget and/or schedule, but also measures and percentages of stakeholder satisfaction (in terms of both requirements and expectations), measures and percentages of stakeholder engagement, and, definitively, measures of *perceived value* (e.g. business value, social value, quality, reputation, business climate, innovation, sustainability). In any case, while, in value-driven projects, the use, and the sharing, of Project Management KPIs, of economic KPIs, and, only if they are considered precisely measurable, of customer satisfaction KPIs too, can be considered well present in the literature (Kerzner, 2017), in both value driven and complex projects the use, and the sharing, of the above mentioned "new" KPIs which are relevant to the perceived value, can be considered innovative. Furthermore, specific KPIs that are relevant to different busi-

ness sectors (e.g., referring to some cases that will be shown in the presentation at the conference, local public transportation, pharmaceutical industry, railway infrastructure, sustainable smart cities, web marketing, etc.), could be effectively used, as trend indicators, also in project management during project life cycle, and not only as performance indicators after project completion.

In Project Management, definitively, any measurable value can be effectively used as a KPI, and the use of an appropriate selection of KPIs is a powerful tool to target the success of the complex projects, by supporting both the value generation, and the project goals achievement.

6 Conclusions

Stakeholders, who are central towards both projects and Project Management, define success in terms of generation of their own business value: proper value indicators (KPIs) in the domains of project management, of economics, and of business value, can be, then, measured, shared with stakeholders, and used, in order to effectively confirm/redirect the action of the project team during life cycle of the complex projects. Stakeholder Perspective, in this way, allows targeting both project objectives and project goals, then supporting both the realisation of deliverables and the accomplishment of value generation, so as to achieve the overall result of a significant increase of the project success rate. During the presentation at the conference, a case study will be illustrated, in order to show both the possibility of having unsuccessful projects, in which objectives can be reached, but stakeholders expectations are not satisfied, and how the different Key Performance Indicators could be managed, in particular if a conflict among some of them occurs.

References

Archibald R.D., 2017, Foreword in "Istituto Italiano di Project Management - Guida ai Temi ed ai Processi di Project Management (Guide to Project Management Themes and Processes)", edited by Mastrofini E., texts by Introna V., Mastrofini E., Monassi M., Pirozzi M., Tramontana B, and Trasarti G., foreword by Archibald R.D., FrancoAngeli.

Freeman R.E., 1984, *Strategic Management: a Stakeholder Approach*, Pitman series in Business and Public Policy.

Freeman, R. E., 1994, "The politics of stakeholder theory: Some future directions", *Business Ethics Quarterly*, Vol. 4, No. 4.

International Organization for Standardization, 2012, "ISO 21500:2012 Guidance on Project Management", International Organization for Standardization.

Kerzner H., 2015, *Project Management 2.0 - Leveraging Tools, Distributed Collaboration, and Metrics for Project Success*, Wiley.

Kerzner H., 2017, *Project Management Metrics, KPIs, and Dashboards*, Third Edition, Wiley.

Kerzner H. and Saladis F.P., 2009, "Value-Driven Project Management", Wiley.

Pirozzi M., 2017, "The Stakeholder Perspective", Featured Paper, *PM World Journal*, Vol. VI, Issue VI, June 2017.

Project Management Institute, 2017, "PMI's Pulse of the Profession 2017", 9th Global Project Management Survey, Project Management Institute.

Multi-skill project scheduling in a nuclear research facility

Oliver Polo Mejia[1,2], Marie-Christine Anselmet[1], Christian Artigues[2] and Pierre Lopez[2]

[1] DEC/SETC - CEA Cadarache, St Paul lez Durance, France
[2] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
oliver.polomejia@cea.fr

Keywords: RCPSP, Preemptive scheduling, Multi-skill, Nuclear laboratory, Optimization

1 Introduction

This paper addresses the weekly scheduling of the activities within one of the research facilities of the French Alternative Energies and Atomic Energy Commission (CEA in short for French). After analyzing the operations and characteristics of the studied laboratory, we conclude that the problem under consideration amounts to an extension of the classical Resource-Constrained Project Scheduling Problem (RCPSP).

The RCPSP is a combinatorial optimization problem that covers a wide range of scheduling situations. The problem consists in scheduling non-preemptive tasks on limited renewable resources. These tasks are linked together by precedence relationships (task i cannot start while task l is in process, $\forall(l,i) \in E$). Usually, the objective is to find a solution that minimizes the makespan of the project, while complying both the precedence constraints and the resource constraints.

Even if the standard version of the RCPSP allows the modeling of a broad spectrum of scheduling problems, it may not cover all the situations that can be found in real-life problems. Extended versions of the RCPSP are then necessary. For a more exhaustive lecture about the variants and extensions of the resource-constrained project scheduling problem, we refer to the survey on this topic published by Orji and Wei (2013). Among all the existing extended versions, we distinguish two that are of great interest for the modeling of the studied problem: the Preemptive RCPSP and the Multi-Skill Project Scheduling Problem (MSPSP). A first attempt to combine these two models for scheduling research activities can be found in Polo Mejia et al. (2017), where a pure preemptive MSPSP with multi-skilled resources is proposed. However, an intensive analysis of the laboratory under study highlighted the need to develop a more extended version in order to have a better representation of the reality. That is why we propose in this paper a new extended variant of the RCPSP: MSPSP with partial preemption.

The remainder of the paper is as follows. In the next section, we briefly describe the problem under consideration. In Section 3, we present the mixed integer linear programming model representing the partially preemptive MSPSP and some computational experiments carried out. Finally, in the last section, we conclude and discuss future research.

2 Problem description

The classical version of the RCPSP is supposed to be non-preemptive, that means, once started an activity must run continuously until its completeness. However, in some practical applications as in the case of scheduling research or engineering activities, it may be interesting to allow the preemption. Allowing preemption may lead to a reduced makespan of the project, especially when resource availability is very limited. On the other

hand, it increases the number of possible solutions and consequently the computational complexity of the problem (Herroelen et al. 1998).

Traditionally in the preemptive RCPSP, the preemption is allowed for all the activities. However, due to some safety and operational constraints, proper to nuclear regulation, we must forbid the preemption of a subset of activities. Another hypothesis of this variant is the release of all resources during the preemption periods. When scheduling research activities, we may be interested in avoiding the release of some equipment or resource having an important setup time for some activities. That is why we propose to work with a variant allowing the partial release of resources according to the characteristics of the activities. We must indicate for each activity what resource can be released during the preemption periods.

Other assumption of the RCPSP is that each resource has specific functions, or in other words the resources are supposed mono-skilled. This hypothesis can become false when we are also studying the allocation of human resources working in the project. In our study case, some resources could perform several functions leading us to a multi-skill RCPSP (MSPSP). In the MSPSP, a resource is therefore characterized by the set of skills it possesses; and a task is no longer only defined by the quantities required of each resource, but also by the number of resources with a specific competence. This variant acquires great importance for scheduling activities in very specific fields, such as pharmaceutical, chemical and nuclear, where the regulation requires the presence of a group of technicians having a set of well-defined competences for the execution of an activity.

In the MSPSP, as defined by Montoya et al. (2014), technicians can only respond to one skill requirement per activity. However, in our practical case, technicians may respond to more than one skill requirement per activity. Additionally, due to operational and safety reasons, we need to guarantee a minimal number of technicians present during the execution of the activity.

Keeping in mind all the aforementioned characteristics, and looking for the most realistic model, we decided to develop an extended variant of RCPSP combining the characteristics of the MSPSP and the preemptive RCPSP. In the proposed variant, that we called MSPSP with partial preemption, the objective is to find the best schedule for a set of activities on renewable multi-skilled resources with limited capacity, being able to respond to more than one skill requirement per activity. An activity is now defined by its duration, precedence relationships and constant requirements of both resources and skills. Preemption is now handled in three levels according to the activities characteristics: 1) Non-preemption, for activities where none of the resources can be preempted; 2) Partial preemption, for activities where a subset of resources can be preempted; and 3) Full preemption, for activities where all resources can be preempted. In our practical case, activities may be subject to a release date and to a deadline (activities in the subset B) or due date (this is determined by the importance of the activity). Additionally, due to the durations of some activities (larger than technicians' work shifts), we need to relax the constraint stating that the same technician execute the totality of the activity.

For establishing the complexity of the MSPSP with partial preemption, we use as a starting point the classical RCPSP. For each instance of the RCPSP we can match an instance of the MSPSP with partial preemption, where all resources are mono-skilled and none of the resources can be preempted. So, we can see the RCPSP as a particular case of the MSPSP with partial preemption. The RCPSP has been proved to be strongly NP-hard (Blazewicz et al. 1983); we can therefore infer that the MSPSP with partial preemption is also strongly NP-hard. Once defined the characteristics and the complexity of the proposed problem, we proceed to formalize the problem using a mixed integer linear programming model that we discuss in the next section.

3 Modeling

The RCPSP can be modeled using different approaches: continuous time-based models based on flows, discrete-time mixed integer linear programming (MILP) formulations, or event-based MILP formulations. Among the discrete-time formulations, more precisely the time-indexed formulations, we find the so-called on/off formulation. This formulation uses binary variables $Y_{i,t}$, where $Y_{i,t} = 1$ if activity i is in progress at time t and $Y_{i,t} = 0$ otherwise. This formulation, which seems to be the most suitable for the preemptive case, has been the basic formulation for the construction of tested models.

In order to choose an effective model, we tested two models, that are similar in essence, constructed using the on/off formulation. In both models, most restrictions are modeled in the same way. The main difference lies in the way in which we handle the preemption periods. For testing these models, we generated a set of instances inspired by real data using the method proposed in Polo Mejia et al. (2017). After computational experiments, one of the models showed significantly better results, and it is presented below.

In the model $DO_{j,t}$ is the operator's availability over the time. $Br_{i,k}$ represents the resource requirements. $DR_{k,t}$ indicates the resource capacities. Parameter $PR_{i,k}$ indicates whether the resource k can be preempted ($PR_{i,k}$=0) or not ($PR_{i,k}$=1). Skill requirements are given in parameter $Bc_{i,c}$. $CO_{j,c}$ indicates the set of skills of technicians ($CO_{j,c} =$ 1 if technician j has the competence c, 0 otherwise). Parameter Pc_i indicates whether technicians can be preempted (Pc_i=0) or not (Pc_i=1). The minimal number of required technicians is given in Nt_i. D_i represents the duration of activities. Parameters dl_i and r_i are the deadlines and release dates.

- $Y_{i,t} \in \{0,1\}$, $Y_{i,t} = 1 \iff$ activity i is in progress at time t
- $O_{j,i,t} \in \{0,1\}$, $O_{j,i,t} = 1 \iff$ technician j is allocated to activity i at time t
- $Z_{i,t} \in \{0,1\}$, $Z_{i,t} = 1 \iff$ activity i starts at time t or before
- $W_{i,t} \in \{0,1\}$, $W_{i,t} = 1 \iff$ activity i ends at time t or after
- $Pp_{i,t} \in \{0,1\}$, $Pp_{i,t} = 1 \iff$ activity i is preempted at time t
- $Tard_i \in \mathbb{Z}_{\geq 0}$: Tardiness of activity i

$$\min \quad \sum_i Tard_i + \sum_i \sum_t t * Y_{i,t} \tag{1}$$

$$s.t. \quad \sum_i O_{j,i,t} \leq DO_{j,t} \qquad \forall j, \forall t \tag{2}$$

$$\sum_i ((Y_{i,t} + PR_{i,k} * Pp_{i,k}) * Br_{i,k}) \leq DR_{k,t} \qquad \forall t, \forall k \tag{3}$$

$$(Y_{i,t} + Pc_i * Pp_{i,t}) * Bc_{i,c} \leq \sum_j (O_{j,i,t} * CO_{j,c}) \qquad \forall i, \forall t, \forall c \tag{4}$$

$$\sum_j O_{j,i,t} \geq (Y_{i,t} + Pc_i * Pp_{i,t}) * Nt_i \qquad \forall t, \forall i \tag{5}$$

$$\sum_t Y_{i,t} \geq D_i \qquad \forall i \tag{6}$$

$$D_l * (1 - Y_{i,t}) \geq \sum_{t'=t}^{T} Y_{l,t'} \qquad \forall (l,i) \in E, \forall t \tag{7}$$

$$\sum_{t=dl_i+1}^{T} Y_{i,t} \leq 0 \qquad \forall i \in B \tag{8}$$

$$\sum_{t=1}^{r_i-1} Y_{i,t} \leq 0 \qquad \forall i \tag{9}$$

$$Pp_{i,t} \geq Z_{i,t} + W_{i,t} - Y_{i,t} - 1 \qquad \forall i, \forall t \tag{10}$$

$$Z_{i,t} \geq Y_{i,t'} \qquad \forall i, \forall t, \forall t' \leq t \tag{11}$$

$$W_{i,t} \geq Y_{i,t'} \qquad \forall i, \forall t, \forall t' \geq t \tag{12}$$

$$Z_{i,t} \leq \sum_{t'=1}^{t} Y_{i,t'} \qquad \forall i, \forall t \tag{13}$$

$$W_{i,t} \leq \sum_{t'=t}^{T} Y_{i,t'} \qquad \forall i, \forall t \tag{14}$$

$$Tard_i \geq t * Y_{i,t} - dl_i \qquad \forall i, \forall t \tag{15}$$

The objective in (1) represents the minimization of the tardiness and also ensures the scheduling of units of duration of each activity as soon as possible. Equations (2) ensure that operator's capacities are satisfied. In equations (3), we ensure that all resource requirements are satisfied respecting the resource capacities. Equations (4) ensure the respect of skill requirements taking into account the set of skills of technicians. The constraints given in (5) and (6) ensure the respect of the minimal number of technicians and duration of activities. Precedence constraints are given in (7). Inequalities (8) and (9) are the constraints for deadlines and release dates. Equations (10) determine whether an activity is preempted or not. Inequalities (11) to (14) are constraints for getting the values of variables $Z_{i,t}$ and $W_{i,t}$. Finally, inequalities (15) calculate the tardiness.

Using CPLEX, this model allows us to solve optimally a set of small instances (20 activities with duration between 1 and 10 units of time and a mean of 4 precedence relationships, 13 skills) within a mean time of 7.23 seconds. For a set of larger instances (20 activities with duration between 5 and 20 units of time and a mean of 6 precedence relationships, 13 skills), we were not able to solve them optimally after 2 hours of computing having final gap between 3-15%. By conference time, heuristic methods capable of obtaining good answers in reduced times for large instances will be presented.

4 Conclusions

In this paper we show how operations research techniques can be applied to schedule research activities within a nuclear facility. Reducing the scheduling horizon allows us to manage the inherent variability of research activities and hence to treat the scheduling problem as a traditional one. The application of operations research techniques to the scheduling process of research activities can reduce the time spent by researchers in the planning of activities, giving them more time to devote to research. Additionally, using these techniques in the nuclear field increase the safety on the facility by ensuring the respect of all technical constraints.

The RCPSP has been shown to be a very powerful model, being able to represent a huge amount of real-life problems. However, for some complex systems, the classical RCPSP may not take into consideration some very important aspects. We then proposed in this paper the multi-skill project scheduling problem with partial preemption and an MILP formulation for formalizing the problem.

As future work, we must study ways to improve the proposed model in terms of the quality of the linear relaxation and time solving. We also need to develop new heuristics allowing us to have good solutions in reasonable times. In order to develop algorithms for exact solving, approaches for calculating good lower bounds will be studied.

References

Blazewicz J., Lenstra J.K. and Rinnooy Kan A.H.G., 1983, "Scheduling projects subject to resource constraints: classification and complexity", *Discrete Applied Mathematics*, Vol. 5, pp. 11-24.

Herroelen W., De Reyck B. and Demeulemeester E., 1998, "Resource-constrained project scheduling: A survey of recent developments", *Computers & Operations Research*, Vol. 25, pp. 279-302.

Montoya C., Bellenguez-Morineau O., Pinson E. and Rivreau D., 2015, "Integrated column generation and lagrangian relaxation approach for the multi-skill project scheduling problem", *Handbook on Project Management and Scheduling*, Springer International, pp. 565-586.

Orji I.M.J. and Wei S., 2013, "Project scheduling under resource constraints: A recent Survey", *International Journal of Engineering Research and Technology*, Vol. 02, pp. 1-20.

Polo Mejia O., Anselmet M.-C., Artigues C., Lopez P., 2017, "A new RCPSP variant to schedule research activities in a nuclear laboratory", *International Conference on Computers & Industrial Engineering (CIE-47), October 11-13, Lisbon, Portugal.*

Scheduling Vehicles with spatial conflicts

Oddvar Kloster[1], Carlo Mannino[1], Atle Riise[1] and Patrick Schittekat[1]

SINTEF Digital, Norway
Carlo Mannino, carlo.mannino@sintef.no

Keywords: Job shop scheduling, Conflict resolution, Linear Programming.

1 Introduction

In several important real world applications we find the problem of scheduling the movement of objects on a network under spatial constraints on their relative position. Vehicles moving on a transportation network need to fulfil spatial constraints that prevents them from colliding, or getting too close to each other. Typically, the movement of a vehicle in a network is represented as a sequence of atomic movements (the *route*), each requiring a certain time and corresponding to the occupation of a specific network resource. For instance, in railway networks a resource correspond to a track segment, for aircraft a resource can be either an airborne sector or an airport segment, for boats may be channels regions etc. In this framework, the standard way of modelling spatial conflicts is to sequence vehicles on shared resources (by satisfying suitable disjunctive constraints). Examples of this approach in different contexts are in (Mascis and Pacciarelli 2002) for trains, (Boccia *et. al.* 2018) for airplanes, (Günther *et. al.* 2010) for ships, etc. However, in many cases this approach can be insufficient, both because the vehicles and the network resources may have complicated spatial shapes - giving rise to conflicts in non-shared resources, and because two vehicle actually *can* be on the same network resource if they are not too close to each other.

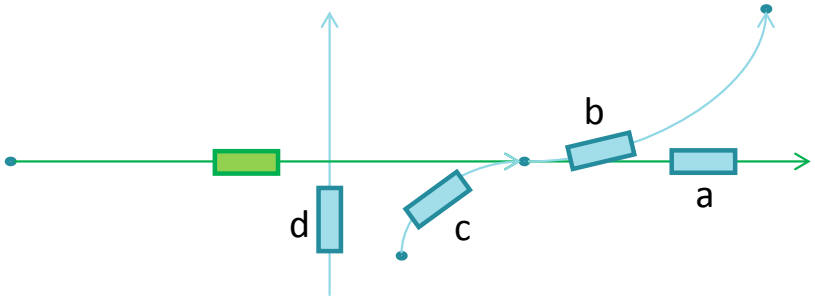


Fig. 1: Vehicles moving on a network with potential conflicts.

This work is motivated by an application in air traffic management, namely that of finding a conflict free trajectory solution for taxiing aircraft at an airport. Simply put, for each allocation of taxi routes to aircraft, the task is to determine a temporal movement of each aircraft along it route so that no aircraft collides.

In this work we present a mathematical construction, so called "Conflict Diagrams", and demonstrate how this concept is a powerful mechanism for presenting spatial conflicts

between two objects moving on a spatial graph. We discuss the conflict diagram's properties and show how they can be constructed and extended following simple rules. We then discuss how the conflict diagram relates to timing variables associated with each vehicle,and how conflict diagrams can be used to construct feasible schedules.

2 Conflict diagrams

We are considering movements of vehicles on a spatial graph, where each arc corresponds to some curve in space ($\mathbb{R}^n_+$ with $n = 1, 2, 3$). For instance, arcs may represent road segments, rail tracks, airborne sectors, etc. The *route* of a vehicle is an ordered sequence of arcs, with the tail of an arc starting at the head of the previous arc. Consider a pair of vehicles v_x and v_y with given routes R_x and R_y of length L_x and L_y, respectively. We can describe the movement of vehicle v_x and vehicle v_y along their routes as real function $x = x(t)$ and $y = y(t)$, respectively, with $x(t) : \mathbb{R}_+ \to [0, L_x]$, and $y(t) : \mathbb{R}_+ \to [0, L_y]$, denoting the position in the route as a distance from the origin at time t. Also, we assume that both $x(t)$ and $y(t)$ are piece-wise linear functions.

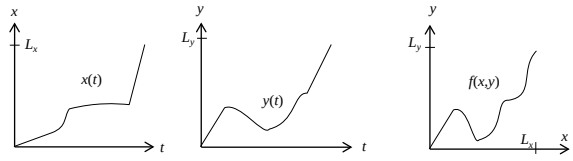


Fig. 2: Two trajectories and concurrent trajectory

If we now eliminate parameter t, we obtain a function $f(x, y) = 0$ (the *concurrent trajectory*) which describes the concurrent positions of the two vehicles along their respective routes. Namely, $f(\bar{x}, \bar{y}) = 0$ if and only if there is a time $\bar{t} \geq 0$, such that $\bar{x} = x(\bar{t})$ and $\bar{y} = y(\bar{t})$. Observe that the curve is defined only in the box $B_L = \{(x, y) \in \mathbb{R}^2 : 0 \leq x \leq L_x, 0, \leq y \leq L_y\}$ and that both point $(0, 0)$ and point (L_x, L_y) belong to the curve (and we call them *first* and *last* point).

Now, let $(\bar{x}, \bar{y}) \in B_L$ represents a point where the two vehicles are too close to each other. That is, there is some spatial constraint that says that vehicle v_x and vehicle v_y cannot be at these respective positions along their routes at the same time. Such point $(\bar{x}, \bar{y})$ cannot belong to any (feasible) concurrent trajectory $f(x, y)$. We call one such point a *conflict point*. We denote by C the set (of the unit box) of conflicting points. To simplify the discussion, from now on we assume C to be the region delimited by a polygon (see Figure 3). So, a concurrent trajectory $f(x, y) = 0$ is infeasible (or 'conflicted') if it intersects the conflict region C. In Figure 3), f_1 is infeasible, whereas f_2 is feasible.

The conflict region C must be constructed from a geometrical analysis of the movement of the two vehicles along their respective routes, resulting in the conflict regions C (illustrated as the dark grey region in Fig. 3).

By exploiting specific knowledge on how vehicles actually move, the conflict region C can be extended to the infeasible region $\bar{C}$, namely a set of points which cannot be intersected by any concurrent trajectory. For instance, we may assume that airplanes can only move forward in their trajectories. Points in B_L which would necessary lead the vehicles to a

point in C have to be prevented, even if strictly speaking they are not conflict points. Similarly, points in $B_L \setminus C$ which cannot be reached without crossing C can be neglected.

For forward trajectories, We have the following

Lemma 2.1 *Suppose vehicles can only move forward in their route, and let* $(x^*, y^*) \in B_L$. *If there exist non-negative quantities* $\delta_x \geq 0$ *and* $\delta_y \geq 0$ *such that both* $(x^* + \delta_x, y^*) \in C$ *and* $(x^*, y^* + \delta_y) \in C$, *then* (x^*, y^*) *is infeasible.*

Similarly, if there exist non-positive quantities $\delta_x \leq 0$ *and* $\delta_y \leq 0$ *such that both* $(x^* + \delta_x, y^*) \in C$ *and* $(x^*, y^* + \delta_y) \in C$, *then* (x^*, y^*) *is infeasible.*

The region $\bar{C} \subseteq C$ which contains C and all the additional infeasible points associated with C is called the *infeasible region* and one can show an effective polynomial algorithm which builds $\bar{C}$ from C. If C is the region delimited by a polygon, so is $\bar{C}$.

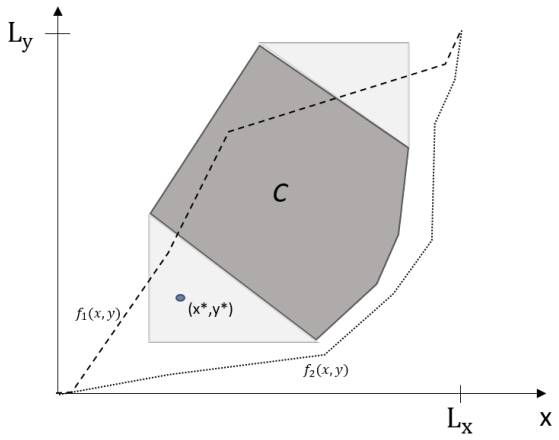


Fig. 3: A conflict diagram showing the conflict region and its infeasible region.

Observe that the infeasible region $\bar{C}$ has always a first vertex (the leftest) and a last vertex (the one most to the right). All other vertices can be classified into lower vertices (having now infeasible points below) and left vertices (having no infeasible point to the left).

3 Feasible concurrent trajectories

Lemma 3.1 *We can partition the family of feasible concurrent trajectories into two classes*

1. $v_y - wins$: *Any point on the trajectory lies above any infeasible point with same x-coordinate.*
2. $v_x - wins$: *Any point on the trajectory lies below any infeasible point with same x-coordinate.*

In Figure 3, f_2 is an $x - wins$ trajectory. We now focus on $v_x - win$ trajectories. A symmetric result applies to the other case. To simplify the notation, we assume that $f(x, y) = 0$ in B_L is the set of points satisfying $y = g(x)$, for $x \in L_x$. One can show the following important result

Lemma 3.2 *v_x wins if, for every vertex $(\bar{x}, \bar{y})$ of $\bar{C}$, the point $(\bar{x}, g(\bar{x}))$ lies in the area below C and, for any two adjacent lower vertices $(\bar{x}_1, \bar{y}_1), (\bar{x}_2, \bar{y}_2)$, $f(x, y)$ is linear between points $(\bar{x}_1, g(\bar{x}_1))$ and$(\bar{x}_1, g(\bar{x}_2))$.*

Let $\bar{x} = x(\bar{t}_x)$, i.e. $\bar{t}_x = t_x(\bar{x})$ is the time when x reaches $\bar{x}$ on its route. Similarly, let $\bar{y} = y(\bar{t}_y)$, with $\bar{t}_y = t_y(\bar{y})$. Then the above Lemma is equivalent to the following

Lemma 3.3 *v_x wins if, for every vertex $(\bar{x}, \bar{y})$ of C, we have $t_x(\bar{x}) \leq t_y(\bar{y})$ and $f(x, y)$ is linear between points associated with successive low vertices as in Lemma 3.2.*

4 Scheduling without conflicts.

Consider a vehicle v_x, with piecewise linear trajectory $x = x(t)$, and let $X = (x_1, \ldots, x_k = L_x)$ be the ordered set of breakpoints. A schedule for v_x is a vector $t^x \in \mathbb{R}_+^X$, where t_i^x specifies when v_x is at point x_i. The schedule must satisfy time precedence constraints associated with the breakpoints, i.e. $t_{x_i+1} - t_{x_i} \geq \lambda_{i,i+1}$, where $\lambda_{i,i+1}$ is the minimum time necessary to v_x to run from x_i to x_{i+1}. Precedence constraints may involve also variables associated with different vehicles, for instance $t_{y_k} - t_{x_j} \geq \lambda_{x_j y_k}$. Now, consider a second vehicle v_y, its trajectory $y = y(t)$ and list of breakpoints $Y = (y_1, \ldots, y_q = L_y)$. Suppose we are given for the pair of vehicles v_x, v_y a conflict diagram C and its infeasible region $\bar{C}$. Also, assume v_x wins. We assume that the break points X contain also the set of x coordinates of the lower vertices of $\bar{C}$, plus the x coordinates of the first and last vertex in $\bar{C}$, namely x_f and x_l. Denote by $\bar{X}$ the ordered subsets of X between x_f and x_l (included). Let $\bar{Y} = \{y \in [0, L_y] : y = g(\bar{x}), \bar{x} \in \bar{X}\}$. We now assume that $Y \supseteq \bar{Y}$.

Lemma 4.1 *Let $\overline{XY} = \{(\bar{x}, \bar{y}) \in \bar{X} \times \bar{Y} : \bar{y} = g(\bar{x})\}$. If $t_{\bar{x}} \leq t_{\bar{y}}$ for all $(\bar{x}, \bar{y}) \in \bar{XY}$, then $f(x, y)$ is feasible.*

In this extended abstract we are not focussing on the actual decisions of who wins, which requires the definition of a suitable disjunctive program (Mascis and Pacciarelli 2002).

We are currently implementing a system to schedule and route airplanes in an airport. The solution algorithm is based on the solution of large disjunctive programs, and makes use of conflict diagrams to represent and identify conflicts, and to associate suitable timing variables with trajectories. Indeed, the standard shared-resource conflict model would not suffice to represent the different conflicting situation that may occur. The system will be tested in April in an official test campaign (sponsored by the EU joint undertaking SESAR 2020). The test case will be Budapest airport (a medium size airport). The test campaign will last for two weeks, involving several airport ground traffic controllers, and will be carried out with the support of EUROCONTROL's simulation platform.

References

Avella P., M. Boccia, C. Mannino, I. Vasilev, 2018, "Time-indexed formulations for the Runway Scheduling Problem", *Transportation Science*, to appear.

Günther E., M.E. Lübbecke, R.H. Möhring, 2003, "Ship Traffic Optimization for the Kiel Canal", *Seventh Triennial Symposium on Transportation Analysis (TRISTAN 2010).*

Mascis A., D. Pacciarelli, 2002, "Job shop scheduling with blocking and no-wait constraints", *European Journal on Operational Research*, Vol. 143 (3), pp. 498-517.

On some approach to solve a scheduling problem with a continuous doubly-constrained resource

Różycki R, Waligóra G

Institute of Computing Science, Poznan University of Technology, Poznan, Poland
{rafal.rozycki, grzegorz.waligora}@cs.put.poznan.pl

Keywords: continuous resource, doubly-constrained resource, parallel machines, discrete-continuous scheduling.

1 Introduction

The nature of scheduling problems where execution time of a job is not set a priori and depends on the amount of allocated doubly-constrained resource has its characteristic specificity. Although the length of the optimal schedule depends in this case both on the available temporary amount of the doubly-constrained resource and its total amount, in practice for a specific instance of the problem, both of these restrictions are rarely active. This fact can be used in the methodology to solve such problems. In this work, we will demonstrate this approach on the example of a doubly-constrained resource with a continuous nature, exemplified by power/energy. We will use the well-known model of a job (Węglarz, 1981), in which its speed of execution depends on the temporary amount of power allocated to it. We consider the problem of scheduling independent preemptable jobs with the criterion of minimizing the makespan. The general methodology for solving such problems is known and has been presented in many papers (Józefowska and Węglarz, 1998, Różycki and Węglarz, 2014, Różycki and Węglarz, 2015). It involves solving a non-linear mathematical programming problem. Unfortunately, the practice shows that solving such problems with known numerical methods is extremely difficult. Below, we present an approach that in some situations allows to find the optimal solution with less computational effort.

2 Problem formulation

Let there be given a set of n independent, preemptable jobs and a set of m parallel identical machines. A job requires a certain amount of continuous doubly-constrained resource (power/energy) and a machine to be performed. The power usage of all jobs must not exceed the limit P at the moment. The consumption of energy by all jobs is limited by the amount E. The temporary speed of job i depends on the current allocation of power $p_i(t)$ and is described by a continuous increasing function (processing speed function), s_i, $s_i(0) = 0$. Before assigning the power to a jobs, its execution time is unknown. Instead of that, job i is characterized by size w_i. The problem is to find an assignment of jobs to machines, and simultaneously an allocation of power to jobs which lead to the schedule of minimal length. In the further part of the paper we will limit our considerations to concave processing speed functions, for which the considered problem is non-trivial.

3 General methodology

The general methodology of solving the problem is based on the theorem (known from e.g. Węglarz, 1981), which assumes that the execution of jobs is only limited by power and energy (there is no limit due to the available number of machines). This theorem defines

an allocation of power to jobs, which leads to a schedule of the shortest length. It can be shown that for the considered concave processing speed functions, it is desirable to perform the jobs (if possible) in parallel. Optimal constant power allocation p_i, $i = 1, 2, \ldots, n$, for jobs executed in parallel, may be calculated basing on the optimal length of their schedule T^*, found as a solution of one of the two nonlinear equations:

$$T \sum_{i=1}^{n} s_i^{-1}(w_i/T) = E \tag{1}$$

$$\sum_{i=1}^{n} s_i^{-1}(w_i/T) = P. \tag{2}$$

The first equation allows to calculate T^* from the constraint on energy, the second from the power limit. Of course, in most cases only one of these restrictions is active. Unfortunately, it is often difficult to evaluate a priori which one. Therefore, it is justified to utilize a rule where the easier equation is solved first and then it is checked whether the calculated power allocation does not violate the second limitation. If the second limit is violated, it means that for the given instance it is active and the optimal length of the schedule should be calculated from the second equation. In many practical cases, e.g. for processing speed functions of form:

$$s_i(p_i) = p_i^{1/\alpha_i}, \quad \alpha_i \in \{2, 3, 4\}, \;\; i = 1, 2, \ldots, n. \tag{3}$$

Equations (1) and (2) can be solved analytically since they are algebraic equations of an order not greater than 4. However, equation (1) is of an order less by 1, and thus should be solved first.

The above approach can be used in the general situation, where, due to the limited number of machines, all jobs must not be performed in parallel. A potentially optimal schedule (see Figure 1) of preemptable jobs can be represented by the sequence of $r, r = \binom{n}{m}$, m - element combinations. A single combination Z_k, $k = 1, 2, \ldots, m$, represents the m jobs performed in parallel in a given part of the schedule. Let us denote by w_{ik} the part of size w_i of job i performed in k-th part of a schedule (represented by Z_k) and by E_k the portion of energy allocated to Z_k. Set K_i contains the indices of combinations where job i belongs to. In order to find the optimal schedule, the following nonlinear mathematical problem has to be solved in the general case:

$$\text{NLP1: minimize} \quad T = \sum_{k=1}^{r} T_k^*(\{w_{ik}\}_{i \in Z_k}, E_k, P) \tag{4}$$

$$\text{subject to} \quad \sum_{k=1}^{r} E_k \leq E \tag{5}$$

$$\sum_{k \in K_i} w_{ik} = w_i, \quad i = 1, 2, \ldots, n \tag{6}$$

$$w_{ik} \geq 0, \quad i = 1, 2, \ldots, n, \;\; k \in K_i \tag{7}$$

$$E_k \geq 0, \quad k = 1, 2, \ldots, r \tag{8}$$

where T_k^* (optimal length of k-th part of schedule) are calculated as functions of E_k, P, and $\{w_{ik}\}$ for $i \in Z_k$ from (1) or (2). E_k and w_{ik} are variables in NLP1, i.e. an optimal distribution of energy as well as a optimal distribution of job sizes among combinations have to be found. NLP1 is non-convex and it is very difficult to solve by standard nonlinear solvers.

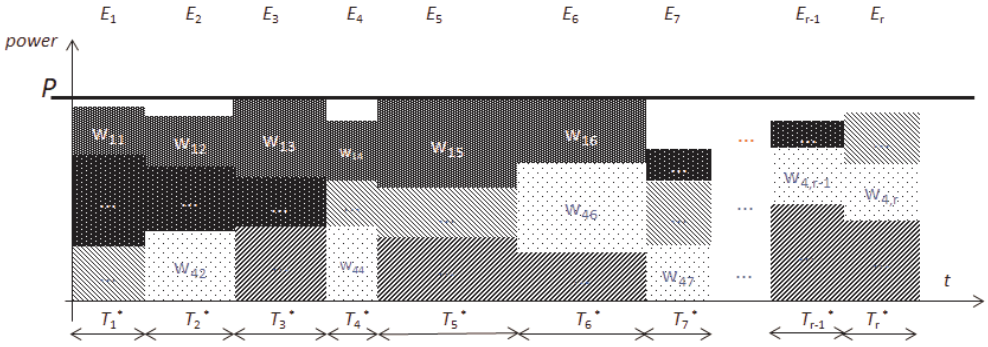


Fig. 1. Exemplary optimal schedule with active energy and power constraint.

3.1 Power constraint only

A schedule for the case of the problem, where limit on energy is inactive is illustrated Figure 2.

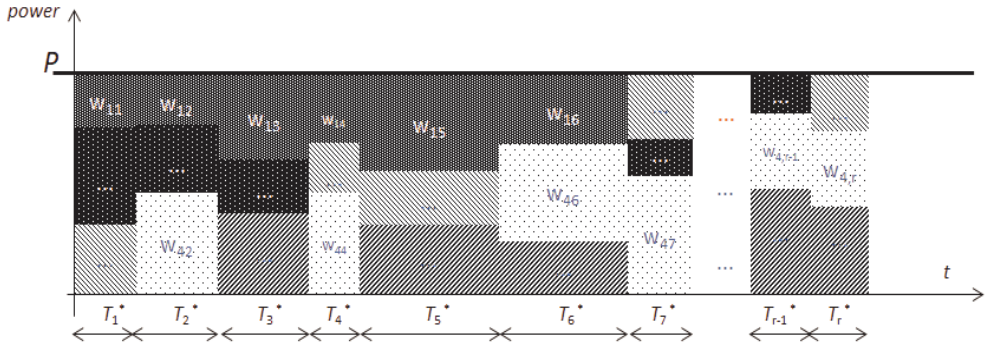


Fig. 2. Exemplary optimal schedule of the problem with power constraint only.

The length of k-th combination, T_k^*, depends on the assigned job sizes and the known amount of P power. In this case, solving the NLP1 problem can be replaced by solving a simpler problem:

$$\text{NLP2: minimize} \quad T = \sum_{k=1}^{r} T_k^*(\{w_{ik}\}_{i \in Z_k}, P) \tag{9}$$
$$\text{subject to} \quad (6)\text{–}(7)$$

where $T_k^*, k = 1, 2, \ldots, r,$
is the unique positive root of the equation:

$$\sum_{i \in Z_k} s_i^{-1}(w_{ik}/T_k) = P. \tag{10}$$

As you can see in the problem, it is not needed to distribute energy between combinations. NLP2 is convex, which greatly simplifies the search for an optimal solution by numerical methods.

3.2 Energy constraint only

A schedule for the case of inactive power constraint is shown in Figure 3.

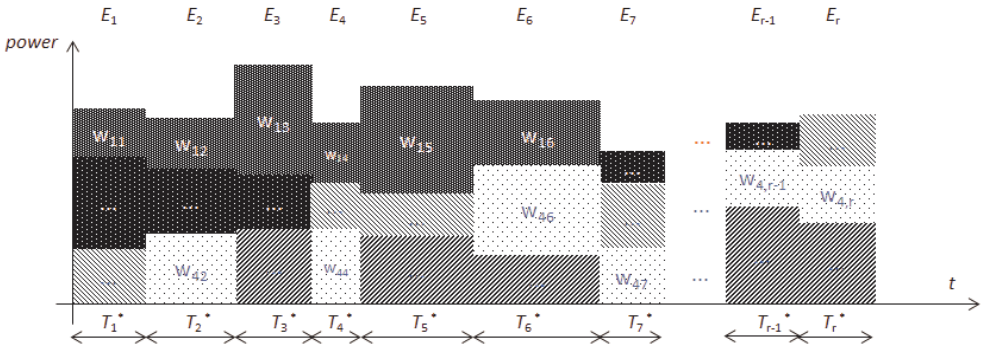


Fig. 3. Exemplary optimal schedule of the problem with energy constraint only.

In this case NLP3 can be solved instead of NLP1:

$$\text{NLP3: minimize} \quad T = \sum_{k=1}^{r} T_k^*(\{w_{ik}\}_{i \in Z_k}, E_k) \tag{11}$$

$$\text{subject to} \quad \sum_{k=1}^{r} E_k = E \tag{12}$$

(6)–(8)

where $T_k^*, k = 1, 2, \ldots, r,$
is the unique positive root of the equation:

$$T_k \cdot \sum_{i \in Z_k} s_i^{-1}(w_{ik}/T_k) = E_k. \tag{13}$$

Similarly to NLP1, the above problem is non-convex, as it also seeks for optimal distribution of energy among combinations. However, the calculation of T_k^* is easier than in the general case.

A two-layer specialized numerical method can be proposed here. In the lower layer, only the optimal division of job sizes among combinations is found. In the higher layer for a given size division, a dynamic programming algorithm is started, in which energy E is distributed optimally among combinations in order to obtain the minimal makespan.

4 Summary

On the basis of the above considerations, the following methodology of solving the problem is justified. Start by solving the convex problem NLP2. Next, check whether the obtained solution violates the restriction on the available amount of energy E. If this limitation has been violated, NLP3 should be solved. Then check whether the solution of NLP3 violates the restriction on P in any part of the schedule. Only when such a rare situation occurs, it is necessary to solve general $NLP1$. However, to solve $NLP1$ one can use the appropriately modified implementation of the two-layer method indicated in point 3.2.

References

Józefowska J., J. Węglarz, 1998, "On a methodology for discrete-continuous scheduling", *European Journal of Operational Research*, Vol. 107(2), pp. 338–353.

Różycki R., J. Węglarz, 2014, "Power-aware scheduling of preemptable jobs on identical parallel processors to minimize makespan", *Annals of Operations Research*, Vol. 213(1), pp. 235–252.

Różycki R., J. Węglarz, 2015, "Solving a power-aware scheduling problem by grouping jobs with the same processing characteristic", *Discrete Applied Mathematics*, Vol. 182, pp. 150–161.

Węglarz J., 1981, "Project scheduling with continuously-divisible doubly constrained resources", *Management Science*, Vol. 27(9), pp. 1040–1053.

Simple metaheuristics for Flowshop Scheduling: All you need is local search

Rubén Ruiz

Universitat Politècnica de València
rruiz@eio.upv.es

1 Abstract

Many scheduling problems are simply too hard to be solved exactly, especially for instances of medium or large size. As a result, the literature on heuristics and metaheuristics for scheduling is extensive. More often than not, metaheuristics are capable of generating solutions close optimality or to tight lower bounds for instances of realistic size in a matter of minutes. Metaheuristics have been refined over the years and there is literally hundreds of papers published every year with applications to most domains in many different journals. Most regrettably, some of these methods are complex in the sense that they have many parameters that affect performance and hence need careful calibration. Furthermore, many times published results are hard to reproduce due to specific speed-ups being used or complicated software constructs. These complex methods are difficult to transfer to industries in the case of scheduling problems. Another important concern is the recently recognized "tsunami" of novel metaheuristics that mimic the most bizarre natural or human processes, as for example intelligent water drops, harmony search, firefly algorithms and the like. See K. Sörensen "Metaheuristics - The Metaphor exposed" (2015), ITOR 22(1):3-18. In this presentation, we review many different flowshop related problems. From the basic flowshop problem with makespan minimization to other objectives like flowtime minimization, tardiness, flowshops with sequence-dependent setup times, no-idle flowshops or other variants and extensions, all the way up to complex hybrid flexible flowline problems. We will show how simple Iterated Greedy (IG) algorithms often outperform much more complex approaches. IG methods are inherently simple with very few parameters. They are easy to code and results are easy to reproduce. We will show that for all tested problems so far they show state-of-the-art performance despite their simplicity. As a result, we will defend the choice of simpler, yet good performing approaches over complicated metaphor-based algorithms.

Exact Methods for Large Unrelated Parallel Machine Scheduling Problems with Sequence Dependent Setup Times

Rubén Ruiz, Luis Fanjul-Peyró and Federico Perea

Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática,
Ciudad Politécnica de la Innovación, Edificio 8G, Acc. B.
Universitat Politècnica de València, Camino de Vera s/n, 46021, València, Spain.
rruiz@eio.upv.es, lfpeyro@gmail.com, perea@eio.upv.es

Keywords: Parallel machine, scheduling, setup times, exact methods.

1 Introduction

An important scheduling problem entails a set of n jobs that have to be assigned and scheduled to a set of m machines in parallel. Each job must be manufactured by exactly one machine. No machine can process more than one job at a time. In the most general case machines are said to be unrelated, meaning that the time needed to process a given job depends on the machine to which it is assigned. This time is denoted as p_{ij}, $i \in \{1, 2, \ldots, m\}$, $j \in \{1, 2, \ldots, n\}$. Unrelated Parallel Machines scheduling problems (UPM) model high output production shops or even central stages in certain production processes. C_j is the completion of job j. The most commonly studied objective is the minimization of makespan ($C_{\max}$) and the problem is denoted as $R//C_{\max}$. The UPM is known to be $\mathcal{NP}$-Hard. Practical industrial problems commonly include setup times. This paper considers the Unrelated Parallel Machine scheduling problem with sequence dependent Setup times (UPMS) or $R/s_{ijk}/C_{\max}$, where s_{ijk} denotes the amount of setup time needed at machine i after job j and before job k, $j \neq k$. The UPMS is significantly more difficult than the UPM. As a matter of fact, a special case of the UPMS with a single machine can be modeled as a particular Traveling Salesman Problem (TSP). While the literature on the UPM is extensive, the UPMS has been, comparatively speaking, much less studied. Furthermore, most existing literature deals with heuristics and metaheuristics and exact approaches are only valid for relatively small to medium instances. One of the contributions of this paper is a new mathematical reformulation for the UPMS. Another contribution of this paper is the design of an efficient mathematical programming based algorithm. Some existing MILP models can be found in the older literature where problems of up to 14 jobs could be solved to optimality. It was not until the last few years that much larger instances of the UPMS were solved to optimality. Avalos-Rosales *et al.* (2015) proposed a MILP that efficiently solved some instances of up to 60 jobs and 8 machines. A similar MILP previously served as a master problem in some iterative algorithms presented in Tran *et al.* (2016). The sizes of the problems solved in these last papers are much larger, albeit not all are solved to optimality.

2 Proposed models and algorithms

Avalos-Rosales *et al.* (2015) presented the following MILP, denoted as AAA in this paper. AAA uses the following variables: $X_{ijk} = 1$ if k is the successor of j on machine i, zero otherwise. $Y_{ij} = 1$ if j is processed on machine i, zero otherwise. $\tilde{C}_j \geq 0$ is the

completion time of job j. Finally, $C_{\max}$ is the maximum completion time (makespan).

$$\min C_{\max} \tag{1}$$

$$\text{s.t.} \sum_{j\in N_0, k\in N, k\neq j} s_{ijk}X_{ijk} + \sum_{j\in N} p_{ij}Y_{ij} \leq C_{\max},\ i \in M. \tag{2}$$

$$\sum_{k\in N} X_{i0k} \leq 1,\ i \in M \tag{3}$$

$$\sum_{i\in M} Y_{ij} = 1,\ j \in N \tag{4}$$

$$Y_{ij} = \sum_{k\in N_0, j\neq k} X_{ijk},\ i \in M, j \in N \tag{5}$$

$$Y_{ik} = \sum_{j\in N_0, j\neq k} X_{ijk},\ i \in M, k \in N \tag{6}$$

$$\tilde{C}_k - \tilde{C}_j + V(1 - X_{ijk}) \geq s_{ijk} + p_{ik}, j \in N_0, k \in N, j \neq k, i \in M \tag{7}$$

$$\tilde{C}_0 = 0 \tag{8}$$

$$C_{\max} \geq \tilde{C}_j,\ j \in N \tag{9}$$

$$X_{ijk} \in \{0,1\},\ Y_{ij} \geq 0, \tilde{C}_j \geq 0.$$

Sets N and M denote the jobs and machines. Set N_0 includes a dummy job. Full notation details are omitted due to space considerations. Constraints (2) define the makespan. Constraints (3) ensure that at most one job is scheduled as the first on each machine. Constraints (4) state that each job is to be processed on exactly one machine. Constraints (5) and (6) ensure that all jobs have one successor and one predecessor. Constraints (7) provide a right processing order and break subtours. Constraint (8) sets the completion time of the dummy job to zero. Constraints (9) are feasible cuts. This model was reported to efficiently solve some instances of up to 60 jobs.

We present a model, called MTZ-AM, based on the heterogeneous traveling salesman problem (TSP). The TSP obtains the minimum length route that visits all nodes/jobs of N exactly once. When one considers that more than one salesman is available, and that each city must be visited by exactly one salesman, we have a multiple traveling salesman problem (m-TSP). The UPMS is a heterogeneous m-TSP, in which the jobs correspond to cities, and the machines correspond to salesmen. If two cities j and k are visited one after the other by the same salesman i, in the corresponding UPMS we say that k is the successor of j on machine i. The cost for the salesman (machine) i of traversing the arc linking cities j and k (of processing job j and then k) is equal to $p_{ij} + s_{ijk}$. MTZ-AM shares the structure defined by equations (1) to (6). Note that constraints (7) are basically subtour elimination constraints (SEC). We substitute (7), (8) and (9) by the well known MTZ subtour elimination constraints:

$$U_j - U_k + n\sum_{i\in M} X_{ijk} \leq n - 1,\ j, k \in N, j \neq k. \tag{10}$$

This set of constraints ensures that, if k is the successor of j on any machine (and therefore $\sum_{i\in M} X_{ijk} = 1$), then $U_k \geq U_j + 1$. Otherwise ($\sum_{i\in M} X_{ijk} = 0$). Note that without constraints (7), (8) and (9) variables $\tilde{C}_j$ are no longer needed. Instead, we define the following set of variables: $U_j \in \mathbb{Z}_+$ is the number of jobs processed before j on the machine

where j is processed. We add a valid inequality adapted from m-TSP problems:

$$U_j + (n-1)\sum_{i\in M} X_{i0j} \leq n-1,\ j \in N. \tag{11}$$

$$U_j + \sum_{i\in M} X_{i0j} \geq 1,\ j \in N. \tag{12}$$

These constraints impose that if a job j is the first job on one machine then $U_j = 0$ regardless whether j is also the last job or not.

Tran *et al.* (2016) have published a branch and check decomposition algorithm. A master problem, basically consisting of constraints (1) to (6), is solved, obtaining a feasible assignment of jobs to machines. The cycles created in the solutions obtained by this master problem are broken by the Concorde TSP solver, yielding optimal schedules on each machine for the assignments given by the master problem. We present an algorithm which takes some of these ideas and combines them with our proposed MTZ-AM model. This algorithm works with a similar master problem:

$$\min C_{\max}, \qquad \text{s.t.:}(2),(3),(4),(5),(6), CUTS, X_{ijk} \in [0,1],\ Y_{ij} \in \{0,1\}.$$

First the master problem is solved with $CUTS = \varnothing$, allowing for a 2% gap, like in Tran *et al.* (2016). In subsequent iterations the next feasible solution to the master problem will be obtained. These solutions yield feasible job-machine assignments, given by the values of variable Y, denoted by y^M. However, no feasible sequence is guaranteed as the X variables are relaxed and no subtour elimination constraints are included. From the assignments y^M obtained in the master problem, a feasible sequence is built by solving the complete MILP model in which we minimize the sum of the machine completion times:

$$\min \sum_{i\in M}\sum_{j\in N_0, k\in N, k\neq j} s_{ijk}X_{ijk} + \sum_{j\in N} p_{ij}y^M_{ij}$$
$$\text{s.t.:}(3),(5),(6),(10),(11),(12), X_{ijk} \in \{0,1\}, U_j \geq 0.$$

Afterwards, cuts are added to the master problem. If N_i^h denotes the set of jobs assigned to machine i in the master problem of iteration h, the proposed cut at iteration h (denoted by $CUT(h)$) is: $CUT(h): \quad C_{\max} \geq C^{hi*}_{\max} - \sum_{j\in N_i^h}(1-Y_{ij})\theta_{hij}$. We denote this algorithm as MPA (Mathematical Programming based Algorithm).

3 Computational evaluation and conclusions

We show results for large instances from 200 to 1000 jobs. We run our MTZ-AM model, the model AAA of Avalos-Rosales *et al.* (2015), our proposed MPA and our own implementation of the branch-and-check algorithm of Tran *et al.* (2016) (B&C). In such an implementation we solve the master problem with Gurobi, instead of SCIP as the original authors did, and the TSP in each machine is also solved with Gurobi (instead of Concorde's TSP solver). This implementation proved far superior than the original one. The MILP models were run for a maximum CPU time of three hours. Gurobi 7.0.2 is used (CPLEX 12.7 proved to be inferior). We measure the Relative Percentage Deviation from the optimum or best lower bound (RPD). Table 1 shows results broken down by n values. MTZ-AM performs best in terms of average RPD. It is for $n = 400$ that our proposed model MTZ-AM clearly outperforms the AAA model. Table 2 shows the average results over the large instances for the best two algorithms tested: our implementation of the Branch-And-Check in (Tran *et al.* 2016) (B&C) and our Mathemathical-Programming-Based algorithm

	AAA		MTZ-AM	
n	*RPD*	Time	*RPD*	Time
200	1.56	7290.98	1.98	8596.62
400	595.69	10 418.89	174.26	9410.34
Average	298.63	8854.94	88.12	9003.48

Table 1. MILP models AAA and proposed MTZ-AM (times in seconds).

(MPA). "Best" corresponds to the time at which the best feasible solution returned by the algorithm was found. "Master" and "Sched" show the average CPU time in seconds spent solving the master problem and the sequencing problem respectively. We note that both

	B&C					MPA				
n	*RPD*	Time	Best	Master	Sched	*RPD*	Time	Best	Master	Sched
200	0.74	6303	2973	6145	158	0.93	6269	1497	6208	61
400	16.67	7488	4206	6290	1198	0.36	7385	2997	7221	164
600	217.78	8496	6698	5373	3122	0.30	7611	4620	7315	294
800						0.37	8623	5494	8017	598
1000						0.41	9256	6015	8026	1195
Average	78.40	7429	4626	5936	1493	0.47	7829	4125	7358	462

Table 2. Results for the large instances for the B&C reimplementation and the proposed MPA.

algorithms perform similarly for $n = 200$. However, for $n = 400, 600$ our MPA produces much lower average *RPD* than B&C in shorter CPU times. *RPD* are computed against the best lower bound. B&C was not able to cope with instances larger than $n = 600$. We tested the proposed MPA algorithm for instances of really large sizes ($n = 800, 1000$). As a general conclusion, our proposed MPA is able to generate average relative percentage deviations from lower bounds of 0.41% in the largest instances of 1000 jobs for the UMPS. This significantly improves upon the previous recent results in the literature by Tran *et al.* (2016) of about 2.48% *RPD* for $n = 120$.

Acknowledgments

The authors are partially supported by the Spanish Ministry of Economy and Competitiveness, under the project "SCHEYARD – Optimization of Scheduling Problems in Container Yards" (No. DPI2015-65895-R) financed by FEDER funds. Special thanks are due to Tony T. Tran and coauthors for all the help received during the reimplementation of their efficient methods.

References

Avalos-Rosales, O., Angel-Bello, F., and Alvarez, A. 2015, "Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times", *International Journal of Advanced Manufacturing Technology*, Vol. 76, No. 9-12, pp. 1705-1718.

Tran, T. T., Araujo, A., and Beck, J. C. 2016, "Decomposition methods for the parallel machine scheduling problem with setups", *INFORMS Journal on Computing*, Vol. 28, No. 1, pp. 83-95.

Power usage minimization in server problems of scheduling computational jobs on a single processor

Różycki R, Waligóra G, and Węglarz J

Poznan University of Technology, Poznan, Poland
{rafal.rozycki, grzegorz.waligora, jan.weglarz}@cs.put.poznan.pl

Keywords: scheduling, single processor, power, energy, server problem.

1 Problem formulation

In this work we deal with a scheduling problem from the field of green computing (Hurson and Memon, 2012, 2013), where the main idea is to find a good balance between computing performance and consumption of natural resources. Since during the execution of a computer program (a computational job), energy is consumed as a main resource, appropriate power management is a basic technique for applying the green computing principles. We consider a problem of scheduling n preemptable, independent jobs on a single processor, where each job requires for its execution the processor as well as an amount (unknown in advance) of power, and it consumes some amount of energy during the execution. Power/energy is a doubly constrained, continuous resource available in positive amounts P and E, respectively. Processing speed of a job depends on the amount of power allotted to this job at a time, according to the following relation (Węglarz, 1976):

$$\dot{x}_i(t) = \frac{dx_i(t)}{dt} = s_i(p_i(t)), \quad x_i(0) = 0, \quad x_i(C_i) = w_i \tag{1}$$

where

$x_i(t)$ is the state of job i at time t;
$s_i(\cdot)$ is the continuous, increasing processing speed function of job i, such that $s_i(0) = 0$;
$p_i(t)$ is the amount of power allotted to job i at time t;
C_i is the completion time (unknown in advance) of job i;
w_i is the size (final state) of job i.

Completion of job i requires that:

$$w_i = \int_0^{C_i} s_i(p_i(t))dt. \tag{2}$$

Thus, job i is characterized by both: processing speed function s_i and size w_i. We will assume that the processing speed function of each job is strictly concave, as an inverse of a strictly convex power usage function. Power/energy is a doubly constrained resource available in positive amounts P and E, respectively. As a consequence, a feasible schedule of length T has to meet the following constraints for any $t \in [0, T]$, where $T = \max_i\{C_i\}$ (Węglarz, 1981):

$$\sum_{i=1}^{n} p_i(t) \leq P. \tag{3}$$

$$\sum_{i=1}^{n} \int_0^T p_i(t)dt \leq E. \tag{4}$$

The problem is, in general, to find a vector function $\mathbf{p}(t) = [p_1(t), p_1(t), \ldots, p_n(t)]$, $p_i(t) \geq 0$, $i = 1, 2, \ldots, n$, which, under the constraints imposed, optimizes the chosen scheduling criterion.

The specificity of the model presented above is that it relates time, power, and energy. In consequence, three possible optimization problems may arise: minimization of a time-related criterion, power usage, or energy consumption under constraints imposed on the other two quantities from which only one or both can be active. In this work we consider a so-called server problem where the power usage is to be minimized assuming a given level of a computer system performance. The level of performance is expressed by an assumed deadline D for the completion of the given set of n jobs. Moreover, we analyze two cases of the problem: when the energy amount available is not limited, as well as when it is limited by E. In each case we formulate an appropriate nonlinear mathematical programming (NLP) problem that finds an optimal power allocation, as well as we discuss the methodology for solving the problem under consideration.

Let us first formulate two important properties of optimal schedules (Różycki and Węglarz, 2014).

Property 1. For each job i characterized by a strictly concave processing speed function s_i and for any energy level E, the following condition holds:

$$\lim_{T \to \infty} T s_i^{-1}(w_i/T) < E. \tag{5}$$

Relation (5) can be interpreted so that extending the execution time of a job results in decreasing the amount of energy consumed by this job. Therefore, from the energy minimization point of view, it is desirable to extend the schedule as long as possible.

Another property concerning sever problems follows directly from Property 1.

Property 2. In each type of server problem, if a feasible solution exists, the length of an optimal schedule is equal to D.

Property 2 is important for server problems since it is known that in order to minimize power usage or energy consumption the schedule has to be completed at the moment of deadline, under an obvious condition that the resource amounts are sufficient to do so.

In the next two sections power usage minimization server problems of scheduling computational jobs on a single processor will be discussed. Let us firstly notice that since we consider concave processing speed functions, scheduling on one processor need not be easier then scheduling on parallel processors. It follows from the fact that for such functions parallel schedules lead to optimal solutions, and they are impossible to construct on a single processor. As a result, some interesting analytical results for the case of one processor can be obtained. Secondly, it is worth mentioning that since only sequential schedules may be considered on a single processor, preemptability of jobs can be neglected. It is obvious that interrupting a job and resuming it later cannot improve the schedule. Thus, each job i, $i = 1, 2, \ldots, n$, is processed using a constant amount of power $p_i > 0$, $i = 1, 2, \ldots, n$, from its start to its completion. The p_i's are variables in the NLP formulations presented in the next two sections.

2 Power minimization under unlimited energy

In this server problem, the objective is to minimize the power usage under a given deadline D for the completion of the last job, and assuming that the amount of energy available for the execution of all jobs is not limited. Based on Property 2, the following NLP problem can be formulated:

$$\underline{\text{NLP1}}\text{: minimize} \quad P = \max_{1=1,\ldots,n}\{p_i\} \tag{6}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \frac{w_i}{s_i(p_i)} = D. \tag{7}$$

The objective function (6) represents the maximum power usage over the entire set of jobs, which is to be minimized. Constraint (7) assures that sum of execution times of all jobs is equal to deadline D, according to Property 2.

By analyzing problem NLP1, we can formulate an important proposition for the considered case of a server problem.

Proposition 1. *In an optimal schedule all jobs are processed using the same fixed power amount.*

Proof. Let us first remove the nonlinearity from the objective function in NLP1. The resulting problem is:

$$\underline{\text{NLP2}}\text{: minimize} \quad p \tag{8}$$

$$\text{subject to} \quad p_i - p \leq 0, \quad i = 1, 2, \ldots, n \tag{9}$$

$$\sum_{i=1}^{n} \frac{w_i}{s_i(p_i)} = D. \tag{10}$$

The Lagrange function for problem NLP2 is as follows:

$$L(p_i, \lambda_i, \rho) = p + \sum_{i=1}^{n} \lambda_i (p_i - p) + \rho \left(\sum_{i=1}^{n} \frac{w_i}{s_i(p_i)} - D \right). \tag{11}$$

Gradient conditions take the form:

$$\frac{\partial L}{\partial p} = 1 - \sum_{i=1}^{n} \lambda_i = 0 \tag{12}$$

$$\frac{\partial L}{\partial p_i} = \lambda_i + \rho \frac{w_i (s_i(p_i))'}{(s_i(p_i))^2} = 0, \quad i = 1, 2, \ldots, n \tag{13}$$

from which it is known that:

$$\sum_{i=1}^{n} \lambda_i = 1 \tag{14}$$

$$\lambda_i = -\rho \frac{w_i (s_i(p_i))'}{(s_i(p_i))^2}, \quad i = 1, 2, \ldots, n. \tag{15}$$

Orthogonality conditions are as follows:

$$\lambda_i (p_i - p) = 0, \quad i = 1, 2, \ldots, n \tag{16}$$

Now, since $w_i > 0$, $p_i > 0$, $s_i(0) = 0$, and all functions s_i are increasing and strictly concave, thus:

$$\frac{w_i (s_i(p_i))'}{(s_i(p_i))^2} > 0, \quad i = 1, 2, \ldots, n. \tag{17}$$

which means that in (15) for any i, $\lambda_i = 0$ if and only if $\rho = 0$. However, if $\rho = 0$, then it follows from (15) that $\lambda_i = 0$ for every $i = 1, 2, \ldots, n$, which is a contradiction to (14). Consequently, $\rho \neq 0$ and therefore $\lambda_i \neq 0$ for every $i = 1, 2, \ldots, n$. If so, we can conclude from (16) that $p_i = p$ for every $i = 1, 2, \ldots, n$.

An immediate corollary follows:

Corollary 1. *The minimum amount of power p^* sufficient to execute all jobs before given deadline D can be found as the unique positive root of the equation:*

$$\sum_{i=1}^{n} \frac{w_i}{s_i(p)} = D. \tag{18}$$

After finding the optimal value of p, the minimum level of energy can be calculated from:

$$E_{\min} = p^* \cdot D. \tag{19}$$

3 Power minimization under limited energy

In this server problem, the power usage is to be minimized under an assumed limited amount E of energy and a required deadline D. We start with the condition for the existence of a feasible solution.

Lemma 1. *A feasible solution to the problem exists if there exists a solution to the system of inequalities:*

$$\sum_{i=1}^{n} \frac{w_i}{s_i(p_i)} \leq D, \tag{20}$$

$$\sum_{i=1}^{n} \frac{w_i p_i}{s_i(p_i)} \leq E. \tag{21}$$

If a feasible solution exists, the following NLP problem, using Property 2, finds a minimum power allocation:

$$\underline{\text{NLP3}}\text{: minimize} \quad P = \max_{i=1,\ldots,n} \{p_i\} \tag{22}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \frac{w_i}{s_i(p_i)} = D \tag{23}$$

$$\sum_{i=1}^{n} \frac{w_i p_i}{s_i(p_i)} \leq E. \tag{24}$$

In problem NLP3 the power usage (22) is minimized subject to the constraints that the deadline is met (23) as well as the available amount of energy is not exceeded (24). Notice that now Corollary 1 can be used to make an attempt to find an optimal solution. If p^* calculated from (18) fulfils constraint (24), it will define the optimum power allocation. However, if it does not, it means that a power allocation with different p_i, $i = 1, 2, \ldots, n$, leads to optimum and, in such a case, it is necessary to solve problem NLP3.

In any case, after finding an optimum power allocation, the energy consumption in the obtained schedule can be calculated from formula:

$$E^* = \sum_{i=1}^{n} E_i^* = \sum_{i=1}^{n} \frac{w_i p_i^*}{s_i(p_i^*)}. \tag{25}$$

However, finding the minimum level of energy sufficient to realize the power-optimal schedule requires a solution of another NLP problem:

$$\underline{\text{NLP4}}\text{: minimize} \quad E_{\min} = \sum_{i=1}^{n} \frac{w_i p_i}{s_i(p_i)} \tag{26}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \frac{w_i}{s_i(p_i)} = D \tag{27}$$

$$p_i \leq P^*, \quad i = 1, \ldots, n \tag{28}$$

where P^* is the optimal solution to problem NLP3.

4 Summary

In this work we consider a problem of minimizing the power usage while scheduling preemptable, independent jobs on a single processor to meet a schedule deadline. Each

job uses some amount of power and consumes some amount of energy. We consider two situations: when energy is not, and when it is limited. For these cases we formulate mathematical programming problems to find optimal power allocations. In the case of unlimited energy, we prove that all jobs are processed using the same power amount. We also show how to calculate the minimum amount of energy for the power-optimal schedules.

References

Hurson A., A. Memon, 2012, *Green and Sustainable Computing: Part I*, Elsevier Science, Academic Press.

Hurson A., A. Memon, 2013, *Green and Sustainable Computing: Part II*, Elsevier Science, Academic Press.

Różycki R., J. Węglarz, 2014, "Power-aware scheduling of preemptable jobs on identical parallel processors to minimize makespan", *Annals of Operations Research*, Vol. 213(1), pp. 235–252.

Węglarz J., 1976, "Time-optimal control of resource allocation in a complex of operations framework", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 6(11), pp. 783–788.

Węglarz J., 1981, "Project scheduling with continuously-divisible doubly constrained resources", *Management Science*, Vol. 27(9), pp. 1040–1053.

Scheduling resource-constrained projects with makespan-dependent revenues and costly overcapacity

André Schnabel and Carolin Kellenbrink

Department of Production Management, Leibniz Universität Hannover, Germany
andre.schnabel, carolin.kellenbrink@prod.uni-hannover.de

Keywords: RCPSP, scheduling, heuristics, local-search, genetic algorithm, overcapacity.

1 Introduction

The resource-constrained project scheduling problem (RCPSP), cf. Pritsker (1969), is a widely researched combinatorial optimization problem. Solving the RCPSP involves finding an assignment of activity starting times which minimizes the project duration without violating constraints imposed by precedence relations between activities and limited resource availabilities. There are given activities $j \in \mathcal{J}$ including dummy start activity 0 and dummy end activity $J+1$. Each activity j has an associated duration d_j, resource consumptions k_{jr} on renewable resources $r \in \mathcal{R}$, and may only start after all of its predecessors $i \in \mathcal{P}_j$ are finished. Each resource r has a constant capacity availability K_r in each time period $t \in \mathcal{T}$. In any period t, the total resource consumption of each resource r from all activities executed in this period must not exceed the capacity level K_r.

This problem setting itself is widely applicable and general. Many industrial scheduling problems like machine scheduling are special cases of the RCPSP. However, there are situations in practice, in which some assumptions of the RCPSP are invalidated or additional assumptions are required. A good overview of extensions and generalizations of the RCPSP can be found in Hartmann and Briskorn (2010).

One aspect often found in industrial applications like aircraft engine remanufacturing is the consideration of costly overcapacity in conjunction with makespan-dependent revenues. This problem setting resembles known generalizations of the RCPSP like the makespan minimization for exogenous fluctuating capacities, cf. Hartmann (2015), or like the minimization of endogenous fluctuating capacities for a given deadline, cf. Deckro (1989). However, the specific combination of simultaneous makespan and flexible capacity optimization is not yet covered in literature. Therefore, we propose a new extension of the RCPSP. We also discuss this topic in a working paper submitted to a journal, see Schnabel *et al.* (2017).

2 Problem setting

The resource-constrained project scheduling problem with makespan-specific revenues and option of overcapacity (RCPSP-ROC) extends the well-known RCPSP by allowing an increase of the freely available capacity levels on a per period basis through the utilization of costly and bounded overcapacity. This capacity level extension instrument can be interpreted either as overcapacity acquired by renting additional machines (or lease workers) or as overtime of employees. Furthermore, the RCPSP-ROC incorporates customer specific revenues depending on the time required for project completion, i.e., makespan. These revenues are assumed to be monotonically decreasing, meaning that a customer is never willing to increase his payment in case of a delay.

A new trade-off emerges when combining these two aspects in the RCPSP framework: The planner may either increase revenue through speedups obtained by additional usage

of overcapacity or decrease cost through reduction of overcapacity. However, he can never simultaneously increase revenue while decreasing costs.

2.1 Model formulation

A precise description of the RCPSP-ROC requires three additional parameters: The payment reserves of a customer $u_t \colon \mathcal{T} \mapsto \mathbb{R}$, the costs κ_r per capacity unit and per period of overcapacity, and the upper bound for overcapacity $\overline{z}_r$. The problem can then be formalized extending any mixed integer programming formulation of the RCPSP. As one possible option, we chose to modify the binary pulse variable formulation given in Pritsker (1969). By definition, x_{jt} is set to one if, and only if, activity j finishes in period t.

Model RCPSP-ROC

$$\max F = \sum_{t=EFT_{J+1}}^{LFT_{J+1}} u_t \cdot x_{J+1,t} - \sum_{r\in\mathcal{R}} \sum_{t\in\mathcal{T}} \kappa_r \cdot z_{rt} \tag{1}$$

subject to

$$\sum_{t=EFT_j}^{LFT_j} x_{jt} = 1, \qquad j \in \mathcal{J} \tag{2}$$

$$\sum_{t=EFT_i}^{LFT_i} x_{it} \cdot t \leq \sum_{t=EFT_j}^{LFT_j} x_{jt} \cdot t - d_j, \qquad j \in \mathcal{J}, i \in \mathcal{P}_j \tag{3}$$

$$\sum_{j=1}^{J} \sum_{\tau=t}^{t+d_j-1} k_{jr} \cdot x_{j\tau} \leq K_r + z_{rt}, \qquad r \in \mathcal{R}, t \in \mathcal{T} \tag{4}$$

$$x_{jt} \in \{0,1\}, \qquad j \in \mathcal{J}, t \in \{EFT_j, \ldots, LFT_j\} \tag{5}$$

$$z_{rt} \in [0, \overline{z}_r], \qquad r \in \mathcal{R}, t \in \mathcal{T} \tag{6}$$

Equation (1) captures the objective of maximizing the profit, which is computed from the realized project revenue depending on the makespan and the overcapacity costs incurred by the schedule. These costs for overcapacity are computed using the auxiliary variable z_{rt}, which is linked by equations (4) to the amount of overcapacity used.

Equations (2) enforce that each activity is executed exactly once. The required order of activity execution is incorporated through constraints (3). Restrictions (4) then limit the cumulative demands in the schedule to the freely available fixed resource availabilities supplemented by the chosen amount of overcapacity. Since the objective of this model indirectly minimizes z_{rt}, this auxiliary variable will always store the amount of overcapacity that was actually necessary in order to gain resource feasibility.

The remaining domains of binary primary variable x_{jt} and continuous variable z_{rt} are specified in equations (5) and (6) respectively, with equations (6) also enforcing an upper bound and non-negativity for overcapacity. In order to tighten the latest finishing times LFT_j without excluding the optimal solution, as a deadline we apply the makespan of the schedule generated by the serial schedule generation scheme (SGS) using the canonical activity list *without using any overcapacity at all*.

2.2 Properties of the problem setting

The RCPSP is a special case of the RCPSP-ROC, e.g., $u_t = -t \;\forall\; t$, $\overline{z}_r = 0 \;\forall\; r$. Therefore, the $\mathcal{NP}$-hardness of the RCPSP implies the impracticality of solving industrial

size instances of the more general RCPSP-ROC in acceptable time using exact solution methods.

Our problem setting combines the minimization of overcapacity (non-regular term) with the maximization of makespan-dependent revenues (regular term). The non-regular component in the objective prevents direct application of heuristics developed for the RCPSP, cf. Ballestin and Blanco (2015). This motivates the design, implementation, and evaluation of novel heuristics for the RCPSP-ROC.

3 Solution approaches

The representation or encoding of the solution is a critical core element of many heuristics. A key concern when designing a solution encoding is the following trade-off: The state space of the encoding should ideally be big enough to contain at least one optimal solution while simultaneously being as small as possible.

An example for such an efficient encoding of schedules for the RCPSP is the activity list. Unfortunately, just reusing the activity list as encoding in conjunction with the serial SGS as decoding procedure is not possible due to the non-regular objective. Therefore, new solution encodings were developed for the RCPSP-ROC. They include the amount of overcapacity permitted in certain periods or the information whether one activity is allowed to use overcapacity or not. These encodings are embedded in two different types of heuristics.

3.1 Genetic Algorithms

Genetic algorithms are very powerful for heuristically solving the RCPSP and its variants, see Kolisch and Hartmann (2006). For any encoding, a genetic algorithm can be obtained by specifying the construction of the initial population and the genetic operators (crossover, mutation, selection). An individual is simply an encoded solution and its fitness value is the objective value of that solution. The genetic algorithm for the RCPSP developed by Hartmann (1998) was used as a starting point for developing the genetic algorithms and adapted to the encodings for the RCPSP-ROC.

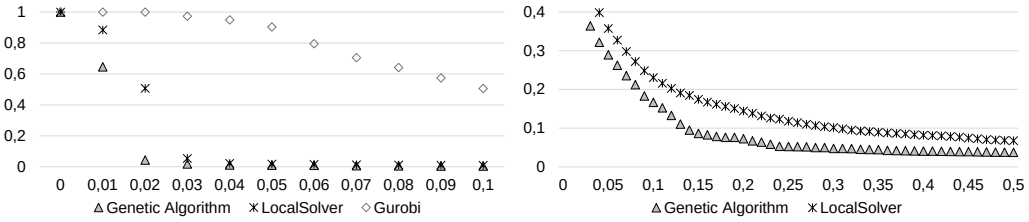
3.2 LocalSolver

Furthermore, LocalSolver, a relatively new commercial proprietary black-box heuristic solver[1], was used for solving the RCPSP-ROC. LocalSolver provides a modeling language for specifying the objectives and constraints of the model similar to GAMS or OPL. Additionally, it offers an API for specifying models. This API then provides so-called "native functions", a mechanism that allows inserting arbitrary functions implemented in a general-purpose programming language into any expression of the model. The decoding procedures (e.g. activity list $\mapsto$ starting times) already implemented for the genetic algorithms as fitness functions were plugged into small LocalSolver models. These models only specify the structure of the solution encoding.

4 Results and conclusion

The numerical experiments are based on two test sets consisting of a filtered subset of 270 and 585 project instances from j30 and j120 respectively based on the PSPLIB, cf. Schnabel *et al.* (2017) for details on the instances, e. g., the definition of the revenue

[1] http://localsolver.com



(a) ∅ optimality gaps for j30 (b) ∅ gaps to best-known solutions for j120

Fig. 1: Numerical results for extended PSPLIB test sets

function. Figure 1a shows the progression of average optimality gaps for small instances with 30 non-dummy activities in the first 0.1 seconds. The optimal reference solutions were acquired using Gurobi. Figure 1b shows the progression of average gaps to best-known solutions for larger instances with 120 non-dummy activities in the first 0.5 seconds. For each instance, the best-known solution is the highest profit schedule of all considered methods at the end of computation.

In summary, the results show that the developed genetic algorithm is very competitive and outperforming the other solution procedures evaluated both in the short and in the long run with small and large instances. Interestingly, utilizing the developed solution encodings in LocalSolver was also very efficient, although not as efficient as using the genetic algorithm. Both heuristic approaches were easily able to beat Gurobi on the MIP-formulation. These results indicate the possibility of constructing efficient methods for solving this generalized version of the RCPSP by adapting and combining ideas from scheduling literature for both cost- and time-based objective functions.

Acknowledgements

The authors thank the German Research Foundation (DFG) for financial support of this research project in the CRC 871 “Regeneration of Complex Capital Goods”.

References

Ballestin, F. and R. Blanco, 2015, “Theoretical and Practical Fundamentals”, *Handbook on Project Management and Scheduling*, Vol. 1, pp. 411-427.

Deckro, R. F. and J. E. Hebert, 1989, “Resource constrained project crashing”, *Omega*, Vol. 17, No. 1, pp. 69-79.

Hartmann, S., 1998, “A competitive genetic algorithm for resource-constrained project scheduling”, *Wiley Online Library*, Vol. 45, No. 7, pp. 733-750.

Hartmann, S. and D. Briskorn, 2010, “A survey of variants and extensions of the resource-constrained project scheduling problem”, *European Journal of Operational Research*, Vol. 207, No. 1, pp. 1-14.

Hartmann, S., 2015, “Time-Varying Resource Requirements and Capacities”, *Handbook on Project Management and Scheduling*, Vol. 1, pp. 163-176.

Kolisch, R. and S. Hartmann, 2006, “Experimental investigation of heuristics for resource-constrained project scheduling: An update”, *European Journal of Operational Research*, Vol. 174, No. 1, pp. 23-37.

Prisker A., 1969, “Multiproject scheduling with limited resources: A zero-one programming approach”, *Management Science*, Vol. 16, pp. 93-108.

Schnabel, A., C. Kellenbrink and S. Helber, 2017, “Profit-oriented scheduling of resource-constrained projects with flexible capacity constraints”, *Diskussionspapiere - Hannover Economic Papers (HEP)*, No. 593. `http://diskussionspapiere.wiwi.uni-hannover.de/index.php?number=593`.

On the complexity of scheduling start time dependent asymmetric convex processing times

Helmut A. Sedding

Institute of Theoretical Computer Science, D-89069 Ulm University, Germany
{firstname}.{lastname}@uni-ulm.de

Keywords: Time-dependent scheduling, Piecewise-linear convex processing times

1 Introduction

Time-dependent scheduling concerns with processing times that are a function of start time (Gawiejnowicz 2008). In this work, we consider a rather generic problem that allows for nonmonotonous convex processing times. With it, we aim to lay groundwork for this recent branch of time-dependent scheduling theory. The problem setting surfaced while automating production planning of continuously moving assembly lines. In it, we consider sequencing of assembly operations J for one worker at one workpiece. Notably, before an operation $j \in J$ can be performed, necessary parts need to be fetched from a corresponding container at the line side. For this purpose, the worker leaves the workpiece and walks along the conveyor. As the conveyor continually moves the workpiece, walking distance varies over time. It is minimum at time τ_j, when the moving workpiece just passes the according container. Else, it increases linearly. Thus, walk time is depicted by a V-shaped, piecewise linear function of time. After each walk, the worker performs the corresponding operation in assembly time l_j. The objective is to reduce total walk time by permuting the operations. In scheduling terms, we subsume a walk and an assembly operation by a job with a time-dependent processing time, and minimize total makespan.

Definition 1 (Problem P) *We are given slopes $a \in [0,1]$, $b \in [0,\infty)$, and a set of n jobs $J = \{1,\dots,n\}$. Each job $j \in J$ is given an assembly time $l_j \in \mathbb{Q}_{\geq 0}$ and an ideal start time $\tau_j \in \mathbb{Q}$. We decide on job sequence $S : J \to \{1,\dots,n\}$, which is a permutation of the jobs J to assign each job a distinct position. Its inverse is denoted by S^{-1}. For each job $j \in J$, we calculate start time $t_j = C_{S^{-1}(S(j)-1)}$, iteratively from the global start time $t_{\min} = C_0$ (usually zero), and completion time $C_j = t_j + p_j(t_j)$ with the start time dependent processing time $p_j(t) = l_j + \max\{-a\,(t-\tau_j),\, b\,(t-\tau_j)\}$. The objective is to find a job sequence S that minimizes makespan $\phi(S) = C_{S^{-1}(n)} = C_{\max}$.*

In three-field notation, the problem is stated as $1 \mid p_j = l_j + \max\{-a\,(t-\tau_j),\, b\,(t-\tau_j)\} \mid C_{\max}$. The processing time $p_j = p_j(t)$ of job $j \in J$ is shortest if j starts at $t = \tau_j$, increasing with slope a for decreasing $t < \tau_j$, else with slope b, thus it is asymmetric. The completion time $C_j = t + p_j(t)$ is increasing with t because $a \leq 1$ and $b \geq 0$. Therefore, idle time between jobs may only increase the objective; it is thus excluded by definition. In the literature, problem setting is introduced with symmetric factors $a = b$ in Sedding and Jaehn (2014). The variant in Jaehn and Sedding (2016) measures a job's deviation from the mid-time, when exactly half of the whole job has been processed. The case with one common ideal start time and a variable global start time $t_{\min}$ is polynomial (Farahani and Hosseini 2013). With a fixed, given $t_{\min}$ and asymmetric slopes however, we show in the following that the decision problem is NP-complete by reduction from Even Odd Partition. A highlight of our proof is its compactness compared to the approach for the similar and more specialized problem setting in Jaehn and Sedding (2016). Moreover, we introduce two important polynomial cases which additionally apply for multiple ideal start times.

2 Polynomial Cases

Lemma 1 *Given an instance of P and a job sequence S that sorts the jobs nondecreasingly by $l_j - b\tau_j$. If each job starts at or after its ideal start time $t_j \geq \tau_j$ for all $j \in J$, objective $\phi(S)$ is minimum and it is expressed by*

$$\phi(S) = \sum_{j \in J} (l_j - b\tau_j) (1+b)^{n-S(j)} . \tag{1}$$

Lemma 2 *Given an instance of P. If each job starts at or before its ideal start time $t_j \leq \tau_j$ for all $j \in J$, the objective is expressed by*

$$\phi(S) = \sum_{j \in J} (l_j + a\tau_j) (1-a)^{n-S(j)} . \tag{2}$$

If S furthermore sorts the jobs nonincreasingly by $l_j + a\tau_j$, then $\phi(S)$ is minimum.

On the other hand, given an objective value ϕ, the start time $t_{\min} = t_{S^{-1}(1)}$ of the first job in the sequence $S^{-1}(1)$ is

$$t_{\min} = (1-a)^{-n} \phi - \sum_{j \in J}^{n} (l_j + a\tau_j) (1-a)^{-S(j)} . \tag{3}$$

3 Computational Complexity

We analyze the computational complexity of P using a partition-type NP-hard problem:

Definition 2 (Even Odd Partition Problem (Garey, Tarjan and Wilfong 1988)) *We are given a set of $n = 2h$ natural numbers $X = \{x_1, \ldots, x_n\}$ where $x_{i-1} < x_i$ for all $i = 2, \ldots, n$. The question is whether there exists a partition of X into subsets X_1 and $X_2 := Y \setminus X_1$ such that $\sum_{x \in X_1} x = \sum_{x \in X_2} x$, while for each $i = 1, \ldots, h$, set X_1 contains exactly one element of set $\{x_{2i-1}, x_{2i}\}$.*

We reduce from the Even Odd Partition Problem to the decision version of P, which asks, for a given threshold $\Phi \in \mathbb{Q}$, if there exists a sequence S with makespan $\phi(S) \leq \Phi$.

Theorem 1. *The decision version of P with a common $\tau = \tau_j \ \forall j \in J$ is* NP*-complete.*

Proof. We are given an instance of the Even Odd Partition Problem as of Definition 2. Let us define a corresponding instance of P. For this, we choose an arbitrary $a \in (0,1)$, and set $b = (1-a)^{-1} - 1$. Then, $b \in (0, \infty)$ and $(1+b) = (1-a)^{-1}$. Let job set $J = \{1, \ldots, 2n+1\}$, with $l_{n+j} = 0$ for $j = 1, \ldots, n$, $l_{2n+1} = 2q$ for $q = \frac{1}{2} \sum_{i \in X} x_i$, and $l_{2k-i} = x_{2k-i} (1+b)^{k-h-1}$ for $k = 1, \ldots, h$ and $i = 0, 1$. Hence, $l_{j-1} < l_j$ for $j = 2, \ldots, n$, and $l_n < l_{2n+1}$. Moreover, we set the common ideal start time $\tau = 0$ and the global start time $t_{\min} = -q$. The decision version of this instance asks if there exists a sequence S where objective $\phi = C_{\max}$ is below threshold $\Phi = 3q$.

Solving the sequencing problem results in an optimum sequence S, which we divide into three partial sequences for our analysis. Partial sequence S_0 consists of jobs $n+1, \ldots, 2n$, and we assume this order without loss of generality. The rest is divided into S_1, consisting of jobs that start before 0, and S_2 of jobs that start at or after 0. By Lemma 2, S_1 is sorted nonincreasingly by l_j, while S_2 has nondecreasing l_j (Lemma 1). Moreover, S_0 is between S_1 and S_2 in S as the jobs in S_0 have the smallest assembly times. Let $\hat{C}$ denote the completion time of S_1 and $\hat{t} \geq 0$ the start time of S_2. Hence, sequence S_0 starts at $\hat{C}$

and completes at $\hat{t}$. As $a < 1$, if a job j with $l_j = 0$ starts at $\hat{C} < 0$, then it completes at $\hat{t} < 0$. This contradicts the sorting of the jobs in $S_1 \cup S_0$. Thus, there must be $\hat{C} > 0$. Then, $\hat{t} = \hat{C}(1+b)^n$ by Equation 1. As l_{2n+1} has the longest assembly time, job $2n+1$ is either the first job in S_1 or the last job in S_2. However, we show that job $2n+1$ is not in S_1 if $C_{\max} \leq \Phi$: for a contradiction, let $S(2n+1) = 1$. Then, job $2n+1$ starts at $-q$ and has a completion time of $-q + l_{2n+1} + aq > 0$, hence equals $\hat{C} = q(1+a)$. By Lemma 1, jobs $n+1, \ldots, 2n$ and then jobs $1, \ldots, n$ are appended in nondecreasing order of l_j, thus

$$\begin{aligned}
C_{\max} &= \hat{C}(1+b)^{2n} + \sum_{j=1,\ldots,2n} l_j (1+b)^{2n-(S(j)-1)} = \hat{C}(1+b)^{2n} + \sum_{j=1,\ldots,n} l_j (1+b)^{n-j} \\
&= \hat{C}(1+b)^{2n} + \sum_{k=1,\ldots,h} l_{2k-1}(1+b)^{n+2-2k} + l_{2k}(1+b)^{n+1-2k} \\
&> \hat{C}(1+b)^{2n} + \sum_{k=1,\ldots,h} (l_{2k-1} + l_{2k})(1+b)^{n+1-2k} \\
&= \hat{C}(1+b)^{2n} + \sum_{k=1,\ldots,h} \left(x_{2k-1}(1+b)^{k-h-1} + x_{2k}(1+b)^{k-h-1} \right)(1+b)^{n+1-2k} \\
&> \hat{C} + \sum_{k=1,\ldots,h} (x_{2k-1} + x_{2k}) = \hat{C} + 2q = q(1+a) + 2q = q(3+a) > 3q = \Phi.
\end{aligned}$$

Hence, job $2n+1$ is the last job in S_2 in any optimum S with $C_{\max} \leq \Phi$. Moreover we note as S_0 starts at or after 0, partial sequence S_1 constains at most n jobs.

Let S be optimum for the given instance and assume $C_{\max} \leq \Phi$. Define h_1 as the number of jobs in S_1, and define $h_2 = n - h_1$. Given $\hat{t} \geq 0$ and Equation 3, there is

$$t_{\min} = \hat{C}(1-a)^{-h_1} - \sum_{k=1,\ldots,h_1} l_{S_1^{-1}(k)}(1-a)^{-k} = \hat{C}(1+b)^{h_1} - \sum_{k=1,\ldots,h_1} l_{S_1^{-1}(k)}(1+b)^{k}.$$

As S_2 starts at $\hat{t}$, with Equation 1 there is

$$\begin{aligned}
C_{\max} &= \hat{t}(1+b)^{h_2+1} + \sum_{k=1,\ldots,h_2+1} l_{S_2^{-1}(k)}(1+b)^{h_2+1-k} \\
&= \hat{t}(1+b)^{h_2+1} + l_{2n+1} + \sum_{k=1,\ldots,h_2} l_{S_2^{-1}(k)}(1+b)^{h_2+1-k} \\
&= \hat{t}(1+b)^{h_2+1} + l_{2n+1} + \sum_{k=1,\ldots,h_2} l_{S_2^{-1}(h_2+1-k)}(1+b)^{k}.
\end{aligned}$$

Define $d = (1+b)^{n+h_2+1} - (1+b)^{h_1}$, and

$$f_1(k) = \begin{cases} l_{S_1^{-1}(k)}, & 1 \leq k \leq h_1, \\ 0, & \text{else}, \end{cases} \qquad f_2(k) = \begin{cases} l_{S_2^{-1}(h_2+1-k)}, & 1 \leq k \leq h_2, \\ 0, & \text{else}. \end{cases}$$

Then with $\hat{t} = \hat{C}(1+b)^n$,

$$\begin{aligned}
&\Phi - t_{\min} \geq C_{\max} - t_{\min} \\
&\iff 4q \geq \hat{t}d + l_{2n+1} + \sum_{k=1,\ldots,h_2} l_{S_2^{-1}(h_2+1-k)}(1+b)^{k} + \sum_{k=1,\ldots,h_1} l_{S_1^{-1}(k)}(1+b)^{k} \\
&\iff 2q \geq \hat{t}d + \sum_{k=1,\ldots,n} (f_1(k) + f_2(k))(1+b)^{k}. \qquad (4)
\end{aligned}$$

As $h_1 \leq n$ and $h_2 \geq 0$, there is $d > 0$. In the following, we show that the minimum of $\sum_{k=1,\ldots,n} (f_1(k) + f_2(k))(1+b)^k$ is $2q$, hence Equation 4 requires $\hat{t} = 0$.

– By Hardy, Littlewood and Pólya (1923, Theorem 368, p. 261) and as $(1+b)^k$ increases with k, sum $f_1(k) + f_2(k)$ decreases with k a optimum S.
– For any $i, j = 1, 2$ such that $i \neq j$, if $f_i(k) = 0$ for some k while $f_j(k+1) > 0$, then S is not optimum: an improved S rather has $f_i(k) > 0$ and $f_j(k+1) = 0$. By this argument and as $h_1 + h_2 = 2h$, it follows that $h_1 = h_2 = h$ for an optimum S.
– Moreover, Hardy et al.'s (1923) theorem implies $f_i(k-1) > f_j(k)$ for $k = 2, \ldots, h$ and any $i, j = 1, 2$. This is the case for an optimum S as of Lemma 1 and Lemma 2. Therefore, S has $\{S(2k-1),\, S(2k)\} = \{h+1-k,\, h+k\}$ (in any order) for $k = 1, \ldots, h$.
– It follows that an optimum S has

$$\sum_{k=1,\ldots,n} (f_1(k) + f_2(k)) (1+b)^k = \sum_{k=1,\ldots,h} (l_{2k-1} + l_{2k}) (1+b)^k$$
$$= \sum_{k=1,\ldots,h} \left(x_{2k-1} (1+b)^{-k} + x_{2k} (1+b)^{-k} \right) (1+b)^k = \sum_{k=1,\ldots,h} x_{2k-1} + x_{2k} = 2q.$$

The value of $\hat{t}$ follows from Equation 2 and $S_1^{-1}(h+1-k) \in \{2k-1,\, 2k\}$ for $k = 1, \ldots, h$:

$$\begin{aligned}
\hat{t} &= -q (1-a)^h + \sum_{j=1,\ldots,h} l_{S_1^{-1}(j)} (1-a)^{h-j} \\
&= -q (1-a)^h + \sum_{k=1,\ldots,h} l_{S_1^{-1}(h+1-k)} (1-a)^{h-(h+1-k)} \\
&= -q (1-a)^h + \sum_{k=1,\ldots,h} \left(x_{S_1^{-1}(h+1-k)} (1+b)^{k-h-1} \right) (1+b)^{1-k} \\
&= -q (1-a)^h + (1-a)^h \sum_{j=1,\ldots,h} x_{S_1^{-1}(j)}.
\end{aligned}$$

Then, $\hat{t} = 0 \iff \sum_{j=1,\ldots,h} x_{S_1^{-1}(j)} = q \iff \{x_{S_1^{-1}(j)} \mid j = 1, \ldots, h\} = X_1$ where X_1 is a solution for the Even Odd Partition Problem.

Therefore, the Even Odd Partition Problem instance solves the corresponding P instance and vice versa. As the construction is polynomial and as, given a correct partition, S and $\phi(S)$ can be obtained in polynomial time, the stated problem is NP-complete. □

References

Farahani, M. H. and Hosseini, L.: 2013, Minimizing cycle time in single machine scheduling with start time-dependent processing times, *The International Journal of Advanced Manufacturing Technology* **64**(9), 1479–1486.

Garey, M. R., Tarjan, R. E. and Wilfong, G. T.: 1988, One-processor scheduling with symmetric earliness and tardiness penalties, *Mathematics of Operations Research* **13**, 330–348.

Gawiejnowicz, S.: 2008, *Time-dependent scheduling*, Monographs in Theoretical Computer Science, Springer, Berlin and Heidelberg.

Hardy, G. H., Littlewood, J. E. and Pólya, G.: 1923, *Inequalities*, Cambridge University Press.

Jaehn, F. and Sedding, H. A.: 2016, Scheduling with time-dependent discrepancy times, *Journal of Scheduling* **19**(6), 737–757.

Sedding, H. A. and Jaehn, F.: 2014, Single machine scheduling with nonmonotonic piecewise linear time dependent processing times, *in* T. Fliedner, R. Kolisch and A. Naber (eds), *Proceedings of the 14th International Conference on Project Management and Scheduling*, TUM School of Management, pp. 222–225.

Resource-constrained project scheduling with alternative project structures

Tom Servranckx[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
tom.servranckx@ugent.be, mario.vanhoucke@ugent.be
[2] Technology and Operations Management Area, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, UK

Keywords: Project scheduling, Alternative project structure, Tabu search.

1 Introduction

Project scheduling is crucial to project success as it provides a point-of-reference for long-term resource allocation and project scope management. The resource-constrained project scheduling problem (RCPSP) is a well-known problem in the context of project scheduling (Brucker *et al.* 1999). Many research efforts have focused on the development of various extensions of the basic RCPSP (Hartmann and Briskorn 2010) as well as multiple (meta)heuristic and exact solution procedures (Kolisch and Hartmann 2006, Hartmann and Kolisch 2000). However, one assumption that is retained in most scheduling problems requires that the project structure is deterministic. This implies that the project structure, which is imposed by the activities and the precedence relations between the activities, is fixed and completely known prior to the project execution. However, this assumption has been rendered obsolete in most real-life projects due to the ever-increasing complexity and uncertainty in the project environment (Wiers V. 1997). Therefore, several researchers have already considered improving the flexibility in the execution mode of a project. This would allow certain project elements to be executed in alternative ways in order to respond to unexpected disruptions. In this regard, we should mention a well-studied extension of the RCPSP, the so-called multi-mode RCPSP (MRCPSP) (Elmaghraby S. 1977). In recent years, a more general scheduling problem has been introduced that considers alternative execution modes at the higher work package (WP) level in the project work breakdown structure. In the remainder of this abstract, we will refer to this scheduling problem as the RCPSP with alternative project structures. The most important feature of this problem formulation is the incorporation of alternative execution modes in the scheduling phase. These alternatives are necessary in order to model the uncertain project structure in future stages or are preferred in order to overcome the complex and fast-changing project environment. The objective of the research at hand is to construct a (near) optimal schedule given the alternative project structure. Therefore, the discussed problem shows how to leverage alternative project structures in order to tackle an uncertain project environment. Several research efforts on scheduling with alternative structures have been conducted in various research fields over the past decades (Kis T. 2003, Capacho *et al.* 2009, Capek *et al.* 2012, Kellenbrink and Helber 2015, Vanhoucke and Coelho 2016).

The main contributions of our research are threefold. (1) The existing research efforts on scheduling with alternative project structures have been developed largely independent. Therefore, we propose a comprehensive classification framework to uniquely identify and define different types of alternative project structures. (2) Since most of the existing datasets for the proposed problem formulation are small-scale and randomly generated, we construct a large dataset of artificial problem instance that supports the proposed framework generated using RanGen 2 (Vanhoucke *et al.* 2008). (3) We develop a metaheuristic

solution approach that is tailored to the specific characteristics of the alternative project structures in the discussed classification framework to solve the data instances in the new dataset.

2 Problem description

In this abstract, we discuss the RCPSP with alternative project structures which extends the basic RCPSP by defining alternative ways to execute a subset of interrelated activities in the project. In order to model the alternative execution modes of the WPs, we distinguish between fixed and alternative activities. *Fixed activities* should always be present in the final project schedule and, consequently, the corresponding resource and precedence constraints should always be satisfied. However, the presence of the *alternative activities* in the final project schedule is optional and depends on the selected alternative project structures. Consequently, the project scheduling problem consists of two subproblems, i.e. the decision and the scheduling subproblem. The objective is to select for each WP exactly one alternative execution mode such that the resulting precedence, resource and logical feasible schedule has a minimal project makespan.

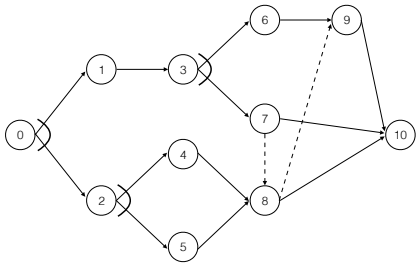


Fig. 1. Illustrative example of project network with alternative project structures

We illustrate the concept of alternative project structures based on the simple project network (see figure 1) derived from Kellenbrink and Helber (2015). This example shows a project network with 9 non-dummy activities (i.e. assume activities 0 and 10, respectively, the dummy start and end activity) and default finish-to-start precedence relations with a zero time lag. The symbol ')' in figure 1 indicates that a choice is triggered between multiple alternative execution modes of a WP. Therefore, only one of the corresponding precedence relations should be considered during project scheduling, e.g. a choice is triggered between the (alternative) activities 1 and 2. The choice for activity 2 will subsequently cause the implementation of either activity 4 or 5. Consequently, all non-dummy activities in this example can be classified as alternative activities since their presence in the final schedule is optional. Note that the choice for one alternative might enforce the implementation of an activity that also belongs to another alternative. This is represented in figure 1 by means of a dotted line, e.g. between activities 7 and 8.

3 Methodology

Based on the aforementioned terminology, we have constructed a classification matrix to unambiguously define projects with alternative project structures based on the relative number of alternative activities and the type of relations between the alternative activities.

Subsequently, we propose a tabu search (TS) procedure (Glover F. 1986) that is tailored to the characteristics of the alternative project structure based on the presented classification matrix. In this research, we first test various strategies for the initial solution generation. Each strategy will assign a weight to the alternative execution modes based on the total work content (TWC) or sum of durations (SOD), adjusted for the specifications of the alternative project structure, in order to prioritise alternatives. The improvement procedure of the TS consists of two components: a neighbourhood structure (NH) and a local search (LS). Given the nature of the overall project scheduling problem, the proposed TS alternately improves the scheduling and decision subproblem. Where the NH and LS for the scheduling subproblem are based on best practices in literature, novel heuristic improvement techniques (i.e. NH and LS) are introduced for the decision subproblem. For each of the strategies, the characteristics of the alternative project structures are used to guide the search procedure to a high-quality final solution. An overview of the procedure is given in figure 2. The main methodological contributions are threefold. First, the presented procedure will not tackle both subproblems in a sequential way, rather in an integrated way. Secondly, the strategies of the TS are adjusted to incorporate the characteristics of the problem instances. Third, different variants of the building blocks of the TS, which either focus on the scheduling or selection subproblem, are constructed.

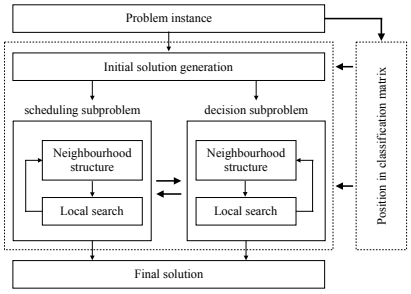


Fig. 2. Overview of TS procedure

4 Computational experiments

The aim of the computational experiments is threefold. First, we compare the performance of different strategies for the building blocks of the TS. Secondly, we validate the solution quality, expressed as the overall best project makespan with a stopping criterion of 5,000 generated schedules, obtained using the TS through a comparison with a multi-start LS routine. Third, we quantify the impact of the characteristics of alternative project structures on the solution quality. The computational experiments provide insights on the above research questions.

1. The impact of the decision subproblem outperforms the impact of the scheduling subproblem on the solution quality.
2. The computational results show that the memory structure of the TS pays off for the problem at hand as it outperforms the multi-start LS routine.
3. According to expectations, an increased relative number of alternatives significantly improve the solution quality, while the complexity of the alternative project structure,

as expressed by the type of relations between the alternatives, has a negative impact on the project makespan (see the preliminary results in table 1).

		Degree of flexibility		
		Low	Medium	High
Degree of complexity	Low	-	-6.23	-9.45
	Medium	2.52	0.48	-0.63
	High	7.06	5.97	4.71

Table 1. The impact of the degree of flexibility and complexity on the project makespan (%)

The extension of the problem formulation to incorporate other concepts discussed in the literature on project scheduling with alternatives together with a comparison of the computational results are part of future research.

References

Brucker, P., A. Drexl, R. Mohring, K. Neumann, E. Pesch, 1999, "Resource-constrained project scheduling: notation, classification, models, and methods.", *European Journal of Operational Research*, Vol. 122, pp.3-41

Capacho, L., R. Pastor, A. Dolgui and O. Guschinskaya, 2009, "An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem", *Journal of Heuristics*, Vol. 15 (2), pp. 109-132.

Capek, R., P. Sucha and Z. Hanzalek, 2012, "Production scheduling with alternative process plans", *European Journal of Operational Research*, Vol. 217 (2), pp. 300-311.

Elmaghraby S., 1977, "Activity networks: Project planning and control by network models", New York: John Wiley and Sons, Inc.

Glover F., 1986, "Future paths for integer programming and links to artificial intelligence", *Computers & Operations Research*, Vol. 13 (5), pp. 533-549.

Hartmann, S., D. Briskorn, 2010, "A survey of variants and extensions of the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 207, pp. 1-15.

Hartmann, S., R. Kolisch, 2000, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 127, pp. 394-407.

Kellenbrink C., S. Helber, 2015, "Scheduling resource-constrained projects with a flexible project structure", *European Journal of Operational Research*, Vol. 246 (2), pp. 379-391.

Kis T., 2003, "Job-shop scheduling with processing alternatives", *European Journal of Operational Research*, Vol. 174, pp. 23-37.

Kolisch, R., S. Hartmann, 2006, "Experimental investigation of heuristics for resource-constrained project scheduling: An update.", *European Journal of Operational Research*, Vol. 174, pp. 23-37.

Vanhoucke M., J. Coelho, 2016, "An approach using SAT solvers for the RCPSP with logical constraints", *European Journal of Operational Research*, Vol. 249, pp. 577-591.

Vanhoucke M., J. Coelho, D. Debels, B. Maenhout and L. Tavares, 2008, "An evaluation of the adequacy of project network generators with systematically sampled networks", *European Journal of Operational Research*, Vol. 187, pp. 511-524.

Wiers V., 1997, "A review of applicability of OR and AI scheduling techniques in practice", *Omega*; Vol. 25 (2), pp. 145-153.

A New Pre-Processing Procedure for the Multi-Mode Resource-Constrained Project Scheduling Problem

Christian Stürck

Helmut Schmidt University, Hamburg, Germany
christian.stuerck@hsu-hh.de

Keywords: Multi-Mode Resource-Constrained Project Scheduling Problem (MRCPSP), Pre-Processing Procedure, Mode Reduction, MMLIB.

1 Introduction

This paper presents a new pre-processing procedure for the Multi-Mode Resource-Constrained Project Scheduling Problem (MRCPSP). The objective is finding a feasible schedule with minimal makespan. The MRCPSP is a $\mathcal{NP}$-hard problem. Even finding a feasible mode assignment is $\mathcal{NP}$-complete if the instance has more than one non-renewable resource (Kolisch and Drexl (1997)).

To reduce the number of variables of this $\mathcal{NP}$-hard problem, different pre-processing techniques have been presented in the literature. The most cited one is the procedure of Sprecher *et al.* (1997). It deletes inefficient modes as well as redundant non-renewable resources and has shown to be very effective on the PSPLIB instances (Kolisch and Sprecher (1997)). The approaches of Zhu *et al.* (1997) and Stürck and Gerhards (2018) are based on calculating new earliest starting times which reduce the number of variables in time-indexed models. While the former approach uses heuristic methods, the latter one uses mathematical programming.

Recently, the new benchmark data set MMLIB has been presented by Van Peteghem and Vanhoucke (2014). These instances are designed in such way that the procedure of Sprecher *et al.* (1997) does not have any impact at all. As a result no procedure exists which is able to reduce the number of modes for the MMLIB instances. Therefore, we develop a technique which aims to reduce the number of modes of these instances. It can be used as a pre-processing procedure and be embedded in a solution approach.

2 Problem description

The MRCPSP consists of a set of activities $A = \{0, ..., n+1\}$. Activity 0 and $n+1$ act as dummy activities and denote the start and the end of the project. Each activity i can be executed in a mode m out of the corresponding set of modes M_i. Precedence constraints E exist among some activities. Each activity has to be assigned to exactly one mode and one starting time while minimising the makespan.

Resource restrictions have to be adhered. Therefore, a set of renewable resources $\mathcal{R}$ – available per time unit – is given. This resource type replenishes on each time unit. Furthermore, a set of non-renewable resources $\mathcal{R}^n$ exists which do not replenish. According to the chosen mode m, activity i has a duration $d_{i,m} \in \mathbb{Z}^+$ as well as a resource consumption $r_{i,m,k} \in \mathbb{Z}^+$ for each resource $k \in \mathcal{R} \cup \mathcal{R}^n$. A mode cannot be changed once it is executed.

In time-indexed models (for example the one of Talbot (1982)) each activity i has an earliest starting time ES_i and a latest starting time LS_i. Alternatively, earliest completion times EC_i and latest completion times LC_i are used. The values of ES_i and EC_i can be derived by using the critical path method (CPM, Kelley (1963)). For the values of LS_i and

LC_i a feasible time horizon T of the project is needed. Based on T the values of LS_i and LC_i can be computed with backward recursion.

3 A new pre-processing procedure

The pre-processing procedure that is presented in this work highly depends on the quality of T. Therefore, it can only be used if the makespan of a feasible solution of an instance is known. The procedure uses the relationship between LC_i and T: the value of T is set to the known makespan of the feasible solution. Then each LC_i is computed, starting with $LC_{n+1} = T$.

With all ES_i and LC_i values known, each mode m of each activity $i \in A$ can be investigated whether the following condition (1) holds:

$$LC_i \geq ES_i + d_{i,m} \; . \tag{1}$$

If an activity i' is started at its earliest starting time $EC_{i'}$ and the duration $d_{i',m'}$ of a mode m' extends the latest completion time $LC_{i'}$, the time horizon T cannot be reached any more. Since the objective is minimising the makespan of the project, using mode m' would not be reasonable. If m' is part of the mode vector it is impossible to reach a makespan which is equal or better than T. Thus, the optimal makespan can never be realised with the choice of this mode. Therefore, these modes are called *non-optimal modes*.

A similar idea of discarding modes has already been proposed in the branch-and-bound procedure of Sprecher and Drexel (1998). The idea was used for truncating the branch-and-bound tree. In contrast, the presented procedure has the advantage that it can be used as pre-processing procedure for exact and heuristic methods.

An example of a *non-optimal mode* is given in Figure 1. Consider an activity i with the $ES_i = 2$ and $LC_i = 6$. Activity i has two modes. Using the first mode, the activity has a duration of $d_{i,1} = 2$. Executed in the second mode the duration of i is $d_{i,2} = 5$. While the first mode fulfils condition (1), the second mode does not. Executed at the earliest starting time, it still extends the latest completion time for activity i. Identified as a *non-optimal mode*, the second mode of activity i can be excluded before the search for the optimal solution starts.

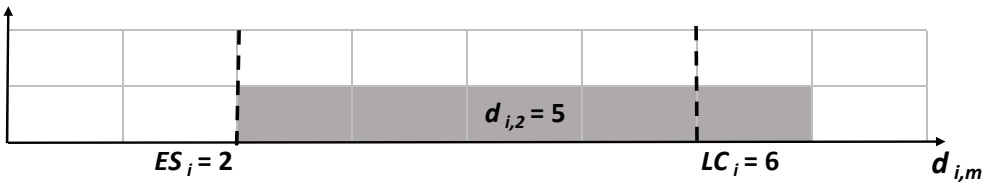


Fig. 1. Example of a *non-optimal mode*

A necessary condition for the usage of this pre-processing procedure is a known makespan of a feasible solution which can be used for T. This can be achieved in two ways:

1. Based on a known feasible solution. This could be a solution from a previous search or the best solution reported to a database.
2. Embedded into an algorithm or a heuristic each time a better solution is found.

The impact of the proposed procedure on the MMLIB is investigated in the following section.

4 Experimental investigation

To test the effectiveness of the presented procedure it is applied to the MMLIB instances. The number of reduced modes is used as a measure of effectiveness due to the fact that the number of variables decreases with the reduction of modes. The experiments were carried out on a PC with an Intel Xeon X5650 CPU at 2.66 GHz. The algorithm is implemented in C#.

First of all the procedure of Sprecher *et al.* (1997) was tested on the MMLIB instances. We can confirm the statement of Van Peteghem and Vanhoucke (2014) – neither a mode nor a resource could be deleted for any MMLIB instance.

We then tested the presented pre-processing procedure. The best known solutions (BKS) reported on the data base *www.mmlib.eu* are used as T for each instance. Based on T all LC_i values of an instance were computed. Then each mode of each activity was tested whether modes exist that extend the latest completion time of an activity if this activity is started at its earliest starting time. If an activity extends its latest completion time in a mode, this mode was identified as *non-optimal* and therefore deleted.

Table 1. *Non-optimal modes* of the MMLIB after using the BKS as T

	MMLIB50	MMLIB100	MMLIB+
Total number of instances	540	540	3,240
Number of instances with at least one *non-optimal mode*	352	347	1,327
Average reduction of *non-optimal modes*	17.69%	13.45%	13,52%
Maximal reduction	44.00%	32.33%	55.78%

Table 1 shows the number of *non-optimal modes* that can be identified for the MMLIB instances. The computational time is less than one second for each instance. Using the best known solutions of the data base a mode reduction was possible for 65.19% / 64.26% / 40.96% of the MMLIB50 / MMLIB100 / MMLIB+ instances, respectively. For one instance of the MMLIB+ 55.78% of the given modes were identified as *non-optimal*. This leads to a significant reduction of variables for this instance.

After the reduction of the modes a feasible mode assignment was done with the remaining modes of the instances. Using the MIP-based procedure presented in Gerhards *et al.* (in print) a feasible mode assignment was found for each instance. Thus, the deletion of *non-optimal modes* does not lead to infeasibility.

To emphasize the impact of the pre-processing procedure a MIP implementation of the mathematical model of Talbot (1982) was tested. IBM ILOG CPLEX 12.6.3 was used as the mathematical solver. The experiments were done for all instances with a maximal running time of 30 minutes per instance. The time horizon T of the model was computed as the sum of the maximal duration of each activity. All ES_i and LS_i (based on T) values were computed using CPM.

Table 2. Improved best known solutions

	MMLIB50	MMLIB100	MMLIB+
MIP (Talbot (1982))	2	2	9
MIP (Talbot (1982)) with pre-processing	2	2	23

Table 2 summarizes the number of improved best known solutions. In the first run the MIP was started without any further pre-processing. Although the MIP was not able to find a feasible solution for each instance, it improves the best known solutions of 13 instances. In the second run, the presented pre-processing procedure was used. This led to an additional improvement of the best known solutions for 27 instances. This indicates the effectiveness of the procedure. Due to a smaller number of variables the MIP is able to improve its solution quality. A more detailed overview of these experiments will be given during the presentation at the conference. All improvements are reported to the database *www.mmlib.eu*.

5 Conclusions

This work presents a pre-processing procedure for the MRCPSP which is based on a known makespan of a feasible solution. The computational experiments show that the procedure can be integrated into a solution approach and has a very short computation time. A mode reduction is possible for 2,026 of the 4,320 MMLIB instances.

To test the effectiveness of the procedure a MIP implementation of the mathematical model of Talbot (1982) was used. The first run did not use the pre-processing procedure. Applying the pre-processing technique in the second run before starting the MIP lead to an improvement of the best known solutions for 27 instances. Thus, the presented technique improves the solution approach. A more detailed investigation of the instances containing *non-optimal modes* as well as a more detailed explanation of the computational experiments will be presented at the conference.

References

Gerhards, P., Stürck, C. and Fink, A. in print, "An Adaptive Large Neighborhood Search as a Matheuristic for the Multi-Mode Resource-Constrained Project Scheduling Problem", *European Journal of Industrial Engineering, to appear.*

Kelley, J. E. 1963, "The critical-path method: Resources planning and scheduling", *Industrial Scheduling*, Vol. 13, no. 1, pp. 347-365.

Kolisch, R. and Drexl, A. 1997, "Local search for nonpreemptive multi-mode resource-constrained project scheduling", *IIE Transactions*, Vol. 29, no. 11, pp. 987-999.

Kolisch, R. and Sprecher, A. 1997, "PSPLIB – a project scheduling problem library: OR software – ORSEP operations research software exchange program", *European Journal of Operational Research*, Vol. 96, no. 1, pp. 205-216.

Sprecher, A. and Drexl, A. 1998, "Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm", *European Journal of Operational Research*, Vol. 107, no. 1, pp. 431-450.

Sprecher, A., Hartmann, S. and Drexl, A. 1997, "An exact algorithm for project scheduling with multiple modes", *Operations-Research-Spektrum*, Vol. 19, no. 3, pp. 195-203.

Stürck, C. and Gerhards, P. 2018, "Providing Lower Bounds for the Multi-Mode Resource-Constrained Project Scheduling Problem", In: *Operations Research Proceedings 2016* eds: Fink, A., Fügenschuh, A. and Geiger, M. J., pp. 551-557, Springer(Cham).

Talbot, F. B. 1982, "Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case", *Management Science*, Vol. 28, no. 10, pp. 1197-1210.

Van Peteghem, V. and Vanhoucke, M. 2014, "An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances", *European Journal of Operational Research*, Vol. 235, no. 1, pp. 62-72.

Zhu, G., Bard, J. F. and Yu, G. 2006, "A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem", *INFORMS Journal on Computing*, Vol. 18, no. 3, pp. 377-390.

An $\mathcal{O}^*(1.41^n)$-time algorithm for a single machine just-in-time scheduling problem with common due date and symmetric weights

Vincent T'kindt[1] and Lei Shang[1] and Federico Della Croce[2]

[1] University of Tours, Laboratory of Computer Science (EA 6300), ERL CNRS 6305, 64 avenue Jean Portalis, 37200 Tours, France
{tkindt,lei.shang}@univ-tours.fr
[2] Politecnico di Torino, DIGEP, Corso Duca degli Abruzzi 24, 10129 Torino, Italy
federico.dellacroce@polito.it

Keywords: Single machine, Just-in-Time, Exponential-time algorithm.

1 Problem statement and literature review

In this paper we revisit a well-known just-in-time scheduling problem under the light of exponential-time algorithms. We are given a set of n jobs, each job i being defined by a processing time p_i and a weight w_i, the latter reflecting the penalty induced by scheduling it early or tardy. All jobs share the same, non restrictive, common due date $d \geq \sum_i p_i$. The objective is to find a schedule s of jobs such that $\sum_i w_i(E_i(s) + T_i(s))$ is minimized, with $T_i(s) = \max(C_i(s) - d; 0)$ and $E_i(s) = \max(d - C_i(s); 0)$. Notice that the mention of schedule s may be omitted whenever there is no ambiguity. Following the standard three-fields notation, this problem can be referred to as $1|d_i = d \geq \sum_i p_i| \sum_i w_i(E_i + T_i)$.

This problem has been shown to be $\mathcal{NP}$-hard by Hall and Posner (1991) and a comprehensive survey of related problems can be found in (T'kindt and Billaut 2006) and (Jozefowska 2007). Interestingly, several remarkable properties, summarized in Property 1, have been established along the years on that problem. They notably induce that the hardness of the problem comes from deciding for each job if it is better to schedule it early or tardy.

Property 1. There exists optimal solutions to the $1|d_i = d \geq \sum_i p_i| \sum_i w_i(E_i+T_i)$ problem satisfying:

1. there are no machine idle times between two consecutive jobs,
2. the first scheduled job can start at a time greater than 0,
3. there exists a job which exactly completes at time d,
4. the class of V-shape schedules is dominant, *i.e.* all early jobs are sequenced by decreasing value of the ratio $\frac{p_i}{w_i}$ (WLPT rule) while all tardy jobs are sequenced by increasing value of the ratio $\frac{p_i}{w_i}$ (WSPT rule).

Several exact algorithms have been proposed to solve it but with the focus of proposing efficient algorithms on randomly generated instances. In this work, we focus on a more theoretical approach which consists in designing exact algorithms for which a "good behaviour" in the worst-case is sought. This relates to the area of exponential-time algorithms and the reader is referred to the book of Fomin and Kratsch (2010) for a good introduction. When dealing with worst-case complexities of exponential algorithms, we usually make use of the $\mathcal{O}^*()$ notation: let $T(\cdot)$ be a superpolynomial and $p(\cdot)$ be a polynomial, both on the instance size (usually the number of jobs n for scheduling). Then, we express running-time bounds of the form $\mathcal{O}(p(n) \cdot T(n)))$ as $\mathcal{O}^*(T(n))$. With respect to scheduling literature,

Lenté *et. al.* (2014) proposed an introduction and a first review of existing exponential algorithms. Additional works have been published by Cygan *et. al.* (2011), Lenté *et. al.* (2013), Garraffa *et. al.* (2017), Shang *et. al.* (2017a) and Shang *et. al.* (2017b). For the $1|d_i = d \geq \sum_i p_i| \sum_i w_i(E_i + T_i)$ problem it is clear that its optimal solution can be computed by means of a brute-force algorithm which enumerates all possible assignments of jobs to the sets of early and tardy jobs. Then, each set can be sorted in polynomial time by means of either WSPT or WLPT rules: the last early job then completes at time d while the first tardy jobs starts at time d. This algorithm requires $\mathcal{O}^*(2^n)$ time and polynomial space in the worst-case. Consequently, it becomes of interest to search for an exact algorithm that would be of a lower worst-case time complexity. We show in the next section that it is possible to solve the $1|d_i = d \geq \sum_i p_i| \sum_i w_i(E_i + T_i)$ problem in $\mathcal{O}^*(2^{\frac{n}{2}})$ time and space.

2 A *Sort & Search* algorithm

Among the known techniques to derive exponential-time algorithms (Fomin and Kratsch 2010), there is *Sort & Search* initially proposed by Horowitz and Sahni (1974) to solve the knapsack problem in $O^*(2^{\frac{n}{2}})$ time and space. Later on, this method has been extended to solve multiple constraints problems by Lenté *et. al.* (2013) who also applied it to a set of scheduling problems. Roughly speaking, it consists in separating an input instance into two equal-size instances, then in enumerating all partial solutions for each sub-instance and then find the optimal solution of the input instance by recombining in a suitable way all those partial solutions taking each time one from each sub-instance. This "complexity breaking" is done at the detriment of the space complexity which turns to be exponential.

Without loss of generality, assume that n is even and that jobs are indexed such that $\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \ldots \leq \frac{p_n}{w_n}$. In the remainder we implicitly make use of the results in Property 1 to elaborate our *Sort & Search* algorithm. For any given instance I of n jobs, let be $I_1 = \{1, \ldots, \frac{n}{2}\}$ and $I_2 = \{\frac{n}{2} + 1, \ldots, n\}$ a decomposition into two equal-size sub-instances. By enumeration, done in $\mathcal{O}^*(2^{\frac{n}{2}})$ time, we can build set $\mathcal{S}_1 = \{s_j^1 / j = 1, \ldots, 2^{|I_1|}\}$ (resp. $\mathcal{S}_2 = \{s_k^2 / k = 1, \ldots, 2^{|I_2|}\}$) which is the set of all possible solutions built from sub-instance I_1 (resp. I_2). We have $|\mathcal{S}_1| = |\mathcal{S}_2| = 2^{\frac{n}{2}}$. Figure 1 shows, for an instance I, a complete schedule $s = s_j^1 // s_k^2$ decomposed into two partial solutions $s_j^1 = \{\epsilon_j^1; \tau_j^1\} \in \mathcal{S}_1$ and $s_k^2 = \{\epsilon_k^2; \tau_k^2\} \in \mathcal{S}_2$, with ϵ_x^y (resp. τ_x^y) referring to a schedule of early jobs (resp. tardy jobs). Besides, t_j^b refers to the completion time of the last job in ϵ_k^2 while t_j^f refers to the starting time of the first job in τ_k^2.

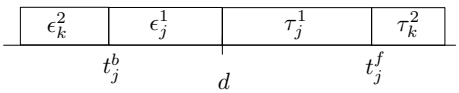


Fig. 1. Decomposition of a complete schedule into two sub-schedules s_j^1 and s_k^2

We can show the following result.

Proposition 1. *Let $s = s_j^1 // s_k^2$ be a complete schedule, and let $f_{jk} = \sum_{i \in s} w_i(E_i(s) + T_i(s))$ be the value of the objective function for schedule s. We have:*

$$f_{jk} = f_j + c_k + a_k t_j^b,$$

with $f_j = \sum_{i \in s_j^1} w_i(E_i(s_j^1) + T_i(s_j^1))$, $c_k = \sum_{i \in s_k^2} w_i(E_i(s_k^2) + T_i(s_k^2)) + d(\sum_{i \in \epsilon_k^2} w_i - \sum_{i \in \tau_k^2} w_i) + \sum_{i \in \tau_k^2} w_i \sum_{i \in I_1} p_i$, *and* $a_k = \sum_{i \in \tau_k^2} w_i - \sum_{i \in \epsilon_k^2} w_i$. *Notice that* f_j *and* t_j^b *are only dependent on* s_j^1, *while* c_k *and* a_k *are only dependent on* s_k^2.

For any s_j^1 partial schedule f_j and t_j^b can be computed in $\mathcal{O}(n)$ time, which is also the case for c_k and a_k whenever partial schedule s_k^2 is given. In the remainder we assume that these values are computed when building sets $\mathcal{S}_1$ and $\mathcal{S}_2$ which does not affect the $\mathcal{O}^*(2^{\frac{n}{2}})$ time complexity required by the *Sort & Search* algorithm to build these sets. This algorithm then proceeds by sorting set $\mathcal{S}_1$ in $\mathcal{O}^*(2^{\frac{n}{2}})$ time, so that for any s_j^1 and s_{j+1}^1 we have $t_j^b \leq t_{j+1}^b$. For a given partial schedule $s_j^1 \in \mathcal{S}_1$, starting from $j = 1$ to $j = |\mathcal{S}_1|$, the algorithm needs to find a schedule $s_k^2 \in \mathcal{S}_2$ such that f_{jk} is minimum. The optimal solution associated to instance I is then given as the best complete solution obtained. Now, let us turn to the search of the best schedule s_k^2 when s_j^1 is fixed.
We separate set $\mathcal{S}_2$ into sub-sets $\mathcal{S}_2^+ = \{s_k^2 \in \mathcal{S}_2 / a_k \geq 0\}$ and $\mathcal{S}_2^- = \{s_k^2 \in \mathcal{S}_2 / a_k < 0\}$ and the search for the best partial schedule s_k^2 complementing s_j^1 is done first in $\mathcal{S}_2^+$ and next in $\mathcal{S}_2^-$. In this abstract, we only detail how the search in $\mathcal{S}_2^+$ can be done in $\mathcal{O}^*(2^{\frac{n}{2}})$ time and we claim that the same result holds for searching in $\mathcal{S}_2^-$. Before doing the search in $\mathcal{S}_2^+$ when s_j^1 is fixed, an extra preprocessing on $\mathcal{S}_2^+$ is done. We know that the lowest possible value of t_j^b is equal to $(d - \sum_{i \in I_1} p_i)$, so let partial sequences $s_k^2 \in \mathcal{S}_2^+$ be re-indexed by increasing values of $\alpha_k = (c_k + a_k(d - \sum_{i \in I_1} p_i)) = \sum_{i \in s_k^2} w_i(E_i(s_k^2) + T_i(s_k^2)) + \sum_{i \in \epsilon_k^2} w_i \sum_{i \in I_1} p_i$. We also remove all s_k^2 such that $\alpha_k \geq \alpha_{k-1}$ and $a_k \geq a_{k-1}$. This can be done, independently from s_j^1, in $\mathcal{O}^*(2^{\frac{n}{2}})$ time. By the way, all contributions $(f_{jk} - f_j)$ of $s_k^2 \in \mathcal{S}_2^+$ depend on t_j^b as pictured in Figure 2. By a dedicated algorithm (not presented here) it is possible to compute couples $(T_\ell, s_{k_\ell}^2)$ in $\mathcal{O}^*(2^{\frac{n}{2}})$ time, with the meaning that whenever $t_j^b \in [T_\ell; T_{\ell+1}[$, partial schedule $s_{k_\ell}^2$ leads to the complete schedule $s = s_j^1 // s_{k_\ell}^2$ with minimum cost. In the worst-case scenario there are $\mathcal{O}(2^{\frac{n}{2}})$ couples, but in practice there can be less couples, depending on the c_k's and the a_k's. Searching in $\mathcal{S}_2^+$ is then equivalent to search in a list of couples $(T_\ell, s_{k_\ell}^2)$ which is assumed to be sorted by increasing values of T_ℓ. The search in this list can be done, for a given $s_j^1 \in \mathcal{S}_1$, can be done in $\mathcal{O}(\log(|\mathcal{S}_2^+|)) = \mathcal{O}(n)$ time. Then, finding the best partial schedule s_k^2 complementing s_j^1, in sets $\mathcal{S}_2^+$ and $\mathcal{S}_2^-$, can be done in $\mathcal{O}(n)$ time.

As for each $s_j^1 \in \mathcal{S}_1$ a search step in $\mathcal{O}(n)$ time has to be done, we reach a total time complexity in $\mathcal{O}^*(2^{\frac{n}{2}})$ for finding the optimal solution of the instance I whenever all sets $\mathcal{S}_1$, $\mathcal{S}_2^+$ and $\mathcal{S}_2^-$ have been built. This data processing requires a total of $\mathcal{O}^*(2^{\frac{n}{2}})$ time, leading by the way to a *Sort & Search* algorithm with $\mathcal{O}^*(2^{\frac{n}{2}})$ time and space worst-case complexities.

3 Future directions

In this abstract we have shown how to build an exponential time algorithm for the $1|d_i = d \geq \sum_i p_i| \sum_i w_i(E_i + T_i)$ problem, running in $\mathcal{O}^*(2^{\frac{n}{2}}) \approx \mathcal{O}^*(1.41^n)$ time and space in the worst case. This algorithm is based on the *Sort & Search* method which works by appropriate data processing and sorting procedures. Interestingly, the sorting procedure is elaborated on partial sequence starting times which is, finally, not surprising since timing problems play a central role in just-in-time scheduling. Besides, this is the first result known for such problems.

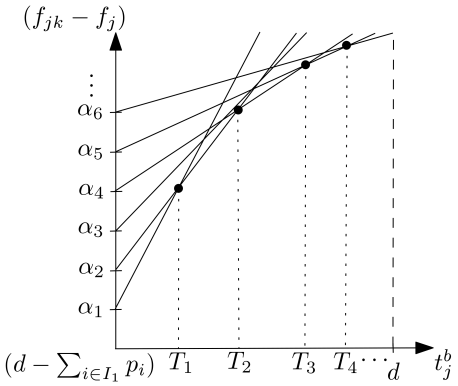


Fig. 2. Contributions of partial schedules $s_k^2 \in \mathcal{S}_2^+$

Notice that in the presence of non symmetric weights, Proposition 1 can be slightly modified which enables to adapt the proposed algorithm for the $1|d_i = d \geq \sum_i p_i| \sum_i w_i E_i + v_i T_i$ problem. Besides, the proposed algorithm suggests also the existence of a *Sort & Search* algorithm for the problem but with identical parallel machines, denoted by $P|d_i = d \geq \sum_i p_i| \sum_i w_i E_i + v_i T_i$.

References

Cygan M., and Pilipczuk M., and Pilipczuk M., and Wojtaszczyk J.O., 2011, "Scheduling partially ordered jobs faster than 2^n", In: Demetrescu C., Halldorsson M.M. (eds) Algorithms - ESA 2011. ESA 2011. *Lecture Notes in Computer Science*, vol 6942, Springer.

Garraffa M., and Shang L., and Della Croce F., and T'kindt V., 2017, "An exact exponential Branch-and-Merge algorithm for the single machine total tardiness problem", *hal.archives-ouvertes.fr/hal-01477835*.

Hall N.G., Posner M.E., 1991, "Earliness-tardiness scheduling problems, I: Weighted deviation of completion times about a common due date", *Operations Research*, Vol. 39, pp. 836-846.

Horowitz E., and Sahni S., 1974, "Computing Partitions with Applications to the Knapsack Problem", *Journal of the ACM*, Vol. 21, pp. 277-292.

Jozefowska J., 2007, "Just-in-Time Scheduling: Models and algorithms for computer and manufacturing systems", *Springer*.

Fomin F. and Kratsch D., 2010, "Exact exponential algorithms", *Springer*.

Lenté C. and Liedloff M. and Soukhal A. and T'kindt V., 2013, "On an extension of the *Sort & Search* method with application to scheduling theory", *Theoretical Computer Science*, Vol. 511, pp. 13-22.

Lenté C. and Liedloff M. and Soukhal A. and T'kindt V., 2014, "Exponential algorithms for scheduling problems", *hal.archives-ouvertes.fr/hal-00944382v1*.

Shang L., and Garraffa M., and Della Croce F., and T'kindt V., 2017a, "Merging nodes in search trees: an exact exponential algorithm for the single machine total tardiness scheduling problem", In *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, volume 89 of LIPIcs, pages 28:1-28:12, Vienna, Austria. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik

Shang L., and Lenté C. and Liedloff M. and T'kindt V., 2017b, "Exact exponential algorithms for 3-machine flowshop scheduling problems", *Journal of Scheduling*, doi.org/10.1007/s10951-017-0524-2.

T'kindt V. and Billaut J.-C., 2006, "Multicriteria Scheduling: Theory, Models and Algorithms", *Springer*.

Finding a specific permutation of jobs for a single machine scheduling problem with deadlines

TA Thanh Thuy Tien[1] and BILLAUT Jean-Charles[1]

Université de Tours, CNRS, LIFAT, ERL CNRS ROOT 6305, Tours, France.
jean-charles.billaut@univ-tours.fr, thanhthuytien.ta@etu.univ-tours.fr

Keywords: single machine, deadlines, lattice, new objective function.

1 Introduction

It is well known that a lot of scheduling problems have a huge number of optimal solutions. This is particularly true for some polynomial problems such as $1||L_{max}$, $1|r_j|C_{\max}$, $F2||C_{max}$, etc. (Smith, W.E. *et. al.* 1956). The purpose of this paper is to contribute to the characterization of all the optimal solutions of such a polynomial scheduling problem.

A general framework to characterize the set of optimal solutions has been proposed in (Billaut, J-C. *et. al.* 2011b) and (Billaut, J-C. *et. al.* 2012), based on the properties of the lattice of permutations (also called permutohedron). We consider in this paper the single machine scheduling problem with maximum lateness minimization, denoted by $1||L_{max}$ (Jackson, J-R. *et. al.* 1955). We assume that a pre-treatment in $O(n \log n)$ is performed so that the jobs are numbered in EDD order and due dates are modified into deadlines so that any optimal sequence has to be feasible with respect to these deadlines.

In the framework based on the lattice, one problem is to find a feasible sequence, as deep as possible. Indeed, any feasible sequence in the lattice is such that all its predecessors are also feasible (simple pairwise exchange argument) and it is possible to give easily the characteristics of all these predecessors. To denote the level of a feasible sequence in the lattice, a new function has been introduced and we want this level to be as small as possible. Let remember (see (Billaut, J-C. *et. al.* 2012)) that the top sequence is EDD with level $\frac{1}{2}n(n-1)$ and the bottom sequence is the inverse EDD sequence with level 0. Typically, if the inverse EDD sequence is feasible, it means that all the predecessors, i.e.e the n! sequences, are feasible.

The new objective function denoted by $\sum N_j$ has led to the introduction of some other new objective functions, based on the position of the jobs in the sequence, which have been studied in (Ta, T.T.Tien *et. al.* 2017a) and (Ta, T.T.Tien *et. al.* 2017b).

2 Definition of function $\sum N_j$ and first results

We consider a set of n jobs to schedule. To each job J_j, $1 \leq j \leq n$, is associated a processing time p_j and a deadline $\tilde{d}_j$. Without loss of generality, it is assumed that $\tilde{d}_1 \leq \tilde{d}_2 \leq ... \leq \tilde{d}_n$ and that sequence $EDD = (J_1, J_2, ..., J_n)$ is feasible.

Let σ be a sequence. The level of σ in the lattice is the number of couples (J_j, J_k) so that $j < k$ and J_j precedes J_k. Therefore, the contribution of J_j to this objective function is the number of jobs after J_j with an index greater than j. We denote this number by N_j.

Let suppose that $x_{j,k}$ is a binary variable equal to 1 if J_j is in position k. We have: $N_j = \sum_{i=j+1}^{n} \sum_{h=k+1}^{n} x_{i,h}$.

This objective function has other denominations in the litterature: the Kendall's tau distance (counts the number of pairwise disagreements between two ranking lists) and the

crossing number between the considered sequence and the inverse numbering sequence. Notice that a problem, presenting similarities with our problem, is proved NP-hard in (Biedl, T. *et. al.* 2005).

We can notice that this objective function does not depend on the jobs completion times, which is unusual in scheduling. This remark leads to some first (simple) results.

• **Problem** $1||\sum N_j$

Problem $1||\sum N_j$ (without due date or deadlines) is trivial. Scheduling the jobs in the reverse order of their numbering leads to a solution with $\sum N_j = 0$.

• **Problem** $1|p_j = p, \widetilde{d}_j|\sum N_j$

Let consider first the $1|p_j = 1, \widetilde{d}_j|\sum N_j$ problem and consider the following Backward algorithm (Alg. 1): schedule starting by the end the feasible job with minimum index. This algorithm solves problem $1|p_j = 1, \widetilde{d}_j|\sum N_j$ to optimality (the proof is admitted here).

It is easy to see that this algorithm can also solve problem $1|p_j = p, \widetilde{d}_j|\sum N_j$.

3 Properties and resolution methods for $1|\widetilde{d}_j|\sum N_j$

Property 1: An optimal solution can always be decomposed in a succession of batches defined as follows: the "head" of the batch is the last job of the batch ; the jobs in the batch are in decreasing numbering order and have an index greater than the head. Therefore, the index of the heads are increasing, starting with index 1.

Proof. admitted.

Exact resolution methods

For exact resolution, two MILP models were presented in (Billaut, J-C. *et. al.* 2012). The first model uses positional variables, the second model uses relative position variables.

In this paper, a branch-and-bound algorithm is proposed with some dominance rules.

The $B\&B$ method for $\sum N_j$ has the following characteristics. A node is defined by a partial sequence S of k jobs starting by the end of the schedule, a set of $n-k$ unscheduled jobs $\bar{S}$, a lower bound $LB(S)$, the index idx of the head of the current batch and t the starting time of the jobs in S: $t = \sum_{J_j \in \bar{S}} p_j$.

At the root node, the unscheduled jobs are $\{J_n, J_{n-1}, ..., J_1\}$. The initial upper bound UB is given by a Backward algorithm of the same type as Alg. 1. The strategy of branching consists in adding a job of $\bar{S}$ in first position of S, respecting the deadlines, and the exploration is done by $depth - first$ (the list of nodes is managed as a LIFO list).

Some dominance rules are used for this method. Let consider a current node and let us denote by J_ℓ the first job in S and by J_h the job in $\bar{S}$ to schedule before J_ℓ. The child node is created only if $\tilde{d}_h \geq t$. Furthermore, if $h < \ell$ and $h > idx$, the node is not created (see Property 1). If $h < \ell$ and $h < idx$, the idx of the child node is set to h. If $h = 1$, the sequence is completed by the jobs in $\bar{S}$ in their inverse numbering order and this node is considered immediately as a leaf of the tree (see Property 1).

The lower bound works as follows: a dummy sequence is built with the jobs in $\bar{S}$ in reverse number ordering, plus the jobs in S. The evaluation of this a priori non feasible sequence is the lower bound. However, if the set of unscheduled jobs is $(J_n, J_{n-1}, ..., J_1)$ in this order, it is possible to compute the lower bound in $O(1)$ time.

Heuristic and metaheuristic methods

Two polynomial time heuristic methods are proposed: a Backward algorithm (denoted BW, Alg. 1) and a Forward algorithm (denoted FW). BW builds a solution by the end, putting in last position the feasible job with the smallest index; FW takes the jobs in EDD

order, put each job as late as possible and insert the feasible job with the biggest index before it.

Two metaheuristic methods are proposed: a Tabu search (denoted TS) and a Simulated Annealing (SA), with several (common) neighborhoods operators. The initial solution of TS and SA is the best solution of BW and FW.

4 Computational experiments

After a study about a related problem based on jobs positions, which was proved to be strongly NP-hard ((Ta T.T.Tien, *et. al.* 2017a), (Ta, T.T.Tien, *et. al.* 2017b)), two types of instances were generated. One type of pure random instances, and one type of "difficult" instances. Even if the problems are not the same, we kept these data for our computational experiments.

Data sets For each type of instance, 30 instances have been generated for each value of n, with $n \in \{10, 20, ..., 100\}$.

• For the instances of **type I**, random data sets have been generated as follows: $p_j \in [1, 100]$, $w_j \in [1, 100]$, $d_j \in [(\alpha - \beta/2)P, (\alpha + \beta/2)P]$, with $P = \sum p_j$, $\alpha = 0.75$ and $\beta = 0.25$.

These instances receive a pre-treatment: (1) EDD rule is applied, giving L^*_{max}. Then, (2) due dates are modified to give deadlines: $\tilde{d}_j = d_j + L^*_{max}$, for any $j \in \{1, 2, ..., n\}$, limiting the deadlines to $\sum p_j$. Finally, (3) the jobs are renumbered in EDD order.

• For the instances of **type II**, random data sets have been generated as follows:

For $n' = \lfloor n/4 \rfloor$ jobs: $p_j = 1$; $w_j = 0$; $\tilde{d}_j = 4jP/n$

For the $(n - n')$ remaining jobs: $p_j \in [1, 100]$, $w_j = w_{0j} + P$, with $w_{0j} \in [1, 100]$ and $P = \sum p_j$; $\tilde{d}_j = P + \lfloor n/4 \rfloor$

These instances do not need the pre-treatment.

Results The computational experiments have been run on a HP ProBook, Intel(R) Core(TM) i5-6300 CPU @ 2.40GHz 2.50 GHz, RAM 16,0Go, System style 64 bit. The MILP models have been solved by IBM ILOG CPLEX 12.6. The CPU time to solve each instance has been limited to 180 seconds for all the resolution methods. Results for instances of type I and II are presented in Table 1. Columns MILP1, MILP2 and $B\&B$ concern the exact methods, 'cpu' indicates the average computation time and 'opt' indicates the number of instances solved to optimality in less than 180 seconds. The other columns concern the heuristic methods. Columns '$N°$B' indicate the number of times the method is the best among all the methods, and $\Delta B1$ is a relative deviation defined by: $MIN = \min(MIP1, MIP2, B\&B, BW, FW)$ and $\Delta B1(H) = \frac{H - MIN}{H}, \forall H \in \{BW, FW, TS, SA\}$

For Type I instances, one can see that MIP1 is better than MIP2 for small instances, but B&B is the best exact method, solving quite all instances up to 70 jobs. With 90 jobs the $B\&B$ remains interesting but for larger instances, the best method is the Simulated Annealing algorithm. For Type II instances, one can see that the exact methods are limited to instances with up to 20 jobs. Among the heuristic algorithms, BW is the best method and the Tabu Search and the Simulated Annealing are not able, in the limited computation time of 180 seconds, to improve the initial solution.

Table 1. Results of Type I & II instances

n	MIP1 cpu (s)	opt	MIP2 cpu (s)	opt	B&B cpu	opt	BW N°B	$\Delta B1$ (%)	FW N°B	$\Delta B1$ (%)	TS N°B	$\Delta B1$ (%)	SA N°B	$\Delta B1$ (%)
Results of Type I instances														
10	0,26	30	0,27	30	3.10^{-5}	30	22	2,00	28	0,43	30	0	30	0
20	47,2	30	105	20	4.10^{-4}	30	5	15,36	13	4,21	26	0,36	20	0,65
30	180	0	180	0	5.10^{-3}	30	1	20,66	5	6,50	13	2,39	11	1,03
40	180	0	180	0	0,014	30	0	24,09	1	8,23	13	4,04	7	1,55
50	180	0	180	0	0,077	30	0	28,92	0	7,44	5	3,68	0	1,72
60	180	0	180	0	2,391	30	0	28,90	0	7,11	10	2,42	0	1,28
70	180	0	180	0	23,79	29	0	28,49	0	7,64	3	3,59	1	1,37
80	180	0	180	0	127,3	15	0	31,33	0	7,07	7	2,90	7	0,44
90	180	0	180	0	174,5	1	0	25,85	0	1,76	14	-2,42	13	-3,30
100	180	0	180	0	180	0	0	28,79	0	0,30	12	-1,62	19	-3,10
Results of Type II instances														
10	0,001	30	0,1	30	2.10^{-3}	30	28	0,58	6	13,39	30	0	30	0
20	0.504	30	111	20	33,21	29	30	0	0	26,14	30	0	30	0
30..100	180	0	180	0	180	0	30	0	0	$\simeq$27%	30	0	30	0

5 Conclusions and Perspectives

In this paper, we have identified a new category of scheduling problems, with the definition of a new objective function. Some trivial problems are identified but the general problem with deadlines remains open. We propose some exact and exponential methods, as well as heuristic and meta-heuristic algorithms. These methods are evaluated by some computational experiments on randomly generated instances. In the future, we will continue to improve the exact methods by introducing cuts and more dominance conditions, but the most important point is to investigate the complexity of the general problem.

References

Biedl, T., Brandenburg, Franz J., Deng X., "Crossings and Permutations", In: Healy P., Nikolov N.S. (eds) Graph Drawing. GD 2005. Lecture Notes in Computer Science, vol 3843. Springer, Berlin, Heidelberg, 2005.

Billaut, J-C., Lopez, P. 2011b, "Characterization of all $\rho-$approximated sequences for some scheduling problems", *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, art.* . No. 6059026.

Billaut, J-C., Hebrard, E. and Lopez, P. 2012, "Complete Characterization of Near-Optimal Sequences for the Two Machine Flow Shop Scheduling Problem", *Ninth International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR'2012), Nantes, June 2012.*

Jackson, J-R., 1955 "Scheduling a production line to minimize maximum tardiness", *Research report 43, Management Science Research Project, University of California, Los Angeles, 1955.*

Smith, W.E. 1956 "Various optimizers for single stage production", *Naval Research Logistics Quarterly*, 3(1-2):59-66, 1956.

Ta, T.T.Tien, Billaut, J-C., Febuary 2017 "Characterization of optimal solutions for some problems scheduling", *ROADEF Conference, Metz, France.*

Ta, T.T.Tien, Ringeard, K., Billaut, J-C., October 2017 "New objective functions based on jobs positions for single machine scheduling with deadlines ", *7th IESM Conference, Saarbrucken, Germany.*

Minimizing makespan on parallel batch processing machines

Karim Tamssaouet[1,2], Stéphane Dauzère-Pérès[1], Claude Yugma[1] and Jacques Pinaton[2]

[1] Ecole des Mines de Saint-Etienne, Department of Manufacturing Sciences and Logistics
CNRS UMR 6158 LIMOS, Gardanne, France
karim.tamssaouet,dauzere-peres,yugma@emse.fr
[2] STMicroelectronics Rousset, Rousset, France
jacques.pinaton@st.com

Keywords: Scheduling, Parallel machines, Batching, Local Search, Semiconductor Manufacturing

1 Introduction and Problem Description

Batch scheduling problems arise in many industries, such as semiconductor manufacturing, aircraft manufacturing, steel casting, transportation, material handling, packaging, and storage systems. A batch is defined as a group of jobs that can be processed jointly (Brucker et al. (1998)). Batch scheduling problems are a combination of assignment and sequencing problems. The two main decisions are: Forming batches (assigning jobs to batches) and scheduling the batches on the machines.

This work is motivated by semiconductor manufacturing. Our goal is to optimize scheduling decisions in the diffusion area which is a complex and critical part of front-end processing in semiconductor manufacturing (eg. Mehta and Uzsoy (1998); Mathirajan and Sivakumar (2006); Mönch et al. (2012)). The processes in this area are performed on two types of equipment: Cleaning machines and furnaces (Yugma et al. (2012)).These machines can process several lots simultaneously. Moreover, the processing durations can be very long compared to other operations in a front-end semiconductor manufacturing facility (fab).

As a starting point, we adopt the novel approach recently proposed by Knopp et al. (2017) for complex job-shop scheduling problems with batching machines. In this approach, an adaptation of the classical conjunctive graph is introduced to model batches through arc labels. Using this new representation, called batch-oblivious graph, schedules are constructed and improved during the graph traversal. As this representation does not differ from the conjunctive graph representation for the flexible job-shop scheduling problem, it is possible to directly apply the move proposed in Dauzère-Pérès and Paulli (1997) which integrates the resequencing and reassignment of operations. However, the batch-oblivious approach in Knopp et al. (2017) considers all operations as candidates for the move while in Dauzère-Pérès and Paulli (1997) only critical operations are considered.

The contribution of this work is to improve the efficiency of the batch-oblivious approach when it is dealing with scheduling problems on parallel batch processing machines. Within the context of a neighborhood-based heuristic, which is the case of the batch-oblivious approach, the efficiency can be reached by reducing the size of the neighborhood, i.e. by reducing the set of candidate operations to move. After bringing out that the original batch-oblivious graph lacks fundamental properties that underlie efficient neighborhood structures for scheduling problems without batching, a new construction algorithm is proposed to remedy this. Using this new algorithm, we propose two efficient neighborhood functions that improve the results obtained by the original batch-oblivious approach.

We consider n jobs that arrive dynamically and have to be processed on m identical parallel machines. The maximum batch size of each machine is B jobs. The jobs belong to F incompatible families. Only jobs of the same family can be processed together in a batch due to the chemical nature of the processes. All jobs of family f have the same processing time p_f. Job j has a family denoted by $f(j)$ and a release date denoted by r_j . We are interested in minimizing the makespan C_{max}. Using the $(\alpha|\beta|\gamma)$ notation of Graham et al. (1979), the scheduling problem can be denoted by: $P|p-batch, incompatible, r_j|C_{max}$.

2 Batch Oblivious Approach

Existing disjunctive graph approaches for scheduling problems with batching rely on the introduction of dedicated nodes and arcs to explicitly represent batches. To facilitate modifications of the graph, the batch-oblivious approach reduces this complexity by encoding batching decisions into edge weights and keep unchanged the structure of the original graph. Using this new representation, an original construction algorithm that takes batching decisions and improves the schedules on the fly is proposed. As it computes the processing start dates of the operations while it traverses the graph in the topological order, the proposed construction algorithm changes dynamically the graph in order to fill up the underutilized batches by bringing backward suitable nodes. This algorithm is complemented by the integrated move of Dauzère-Pérès and Paulli (1997) to resequence and reassign operations. This combination yields a rich neighborhood that is applied within a Simulated Annealing (SA) metaheuristic. The latter is embedded in a Greedy Randomized Adaptive Search Procedure (GRASP).

3 New Construction Algorithm

While we adopt most of the batch-oblivious approach, we propose a new construction algorithm. In the original algorithm, the graph is traversed in topological order to compute the processing start dates and constitute batches. If a batch is incomplete, the algorithm searches for a node of a compatible job that can complete the batch among the set of nodes that have not been yet assigned a processing start date. If such compatible job is found, it is required to be available before the already decided start date of the incomplete batch. If it is the case, the job is moved and inserted at the end of the batch sequence.

The study of the resulting graph shows it lacks a fundamental property of efficient neighborhood functions (Van Laarhoven et al. (1992)): The removal of an operation from a machine sequence cannot increase start times. It is easier to construct an example of a batch-oblivious graph when the removal of an operation degrades the solution. The new construction algorithm proposed in this work then modifies the graph in a way that deleting an operation cannot increase start times. This algorithm uses Property 1.

Property 1. If all batch operations are sequenced in the non-increasing order of the job release dates, no operation deletion can degrade the solution quality.

So, instead of inserting an operation at the end of the batch sequence, it will be inserted in the position that leads to the satisfaction of the condition in Property 1. Note that the new construction algorithm leads to the same solution and only changes the solution representation.

4 New Neighborhood Functions

The new algorithm thus uses a solution representation where removing an operation does not increase start times. This leads to the direct use of the classical move where

only critical operations are candidates. This restriction to critical operations is justified by Property 2. This new neighborhood function can quickly lead to a good solution as it discards uninteresting moves and only focuses on promising ones.

Property 2. If a move of a non-critical operation can improve the solution, there is always a move of a critical operation that leads to a solution with the same or a better quality.

Based on the same solution representation, the size of the neighborhood of a solution that can be reached through the move of critical operations can also be reduced. This additional reduction uses Property 3. Before stating the property, two types of operation are given in Definition 1 and Definition 2. This classification is based on the position of an operation in the sequence of its batch.

Definition 1 (First Batched Operation). *It is a first operation in a batch sequence in the batch oblivious conjunctive graph representation.*

Definition 2 (Internal Batched Operation). *It is an operation in a batch which is not the First Batched Operation.*

Property 3. If a move of a critical Internal Batched Operation can improve the solution, there is always a move of a critical First Batched Operation that leads to a solution with the same or a better quality.

As the constructed solutions respect the condition in Property 1, the first operation in the batch sequence is in fact the last available among all the operations belonging to the batch. Property 3 then helps reducing the candidate set of operations to move to those that are critical and the last available in their batch.

5 Computational Results

To assess the efficiency of the two proposed neighborhood functions, the instances of Mönch et al. (2005) are used, except that the makespan is considered instead of the total weighted tardiness. Three implementations are compared:

1. (**OI**): Original implementation of Knopp et al. (2017).
2. (**NC**): The construction algorithm is modified so that Property 1 is satisfied. Moreover, only critical operations are moved instead of any operation.
3. (**NCB**): Similar to (NC), except that only critical first batched operations are considered instead of any critical operation.

Table 1 shows the results of the three implementations on the 160 instances using the same experimentation parameters. Table 1 shows that the ideas presented in this work improve the performance of the original batch-oblivious approach. For example, (NCB) obtains the best solution for 70% of the instances while the original approach finds the best solution for 43%. The dominance of (NC) and (NCB) over (OI) can also be observed when analyzing the average gap (**AverageGap**) and the maximum gap (**MaxGap**) to the best solution. When comparing (NC) and (NCB), even if the improvement is not significant, the interest of improving the local search efficiency is confirmed.

Table 1. Results for instances of Mönch et al. (2005) with makespan objective

Implementation	% Found Best	AverageGap	MaxGap
(OI)	43 %	0.83 %	6.27 %
(NC)	65 %	0.24 %	3.00 %
(NCB)	70 %	0.20 %	3.00 %

6 Conclusion

In this work, we study the scheduling problem of minimizing makespan on parallel identical batching machines with dynamic job arrivals and incompatible families. Based on the batch-oblivious approach presented in Knopp et al. (2017), two new neighborhood functions are proposed. These two functions improve the search performance by reducing the size of the neighborhood to explore. Numerical experiments on academic instances show that the local search efficiency is improved. An important perspective of this work is to extend the analysis to the case where routing precedence constraints are considered.

Bibliography

Brucker, P., A. Gladky, H. Hoogeveen, M. Y. Kovalyov, C. N. Potts, T. Tautenhahn, and S. L. Van De Velde (1998). Scheduling a batching machine. *Journal of scheduling 1*(1), 31–54.

Dauzère-Pérès, S. and J. Paulli (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research 70*, 281–306.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. R. Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics 5*, 287–326.

Knopp, S., S. Dauzère-Pérès, and C. Yugma (2017). A batch-oblivious approach for complex job-shop scheduling problems. *European Journal of Operational Research 263*(1), 50 – 61.

Mathirajan, M. and A. I. Sivakumar (2006). A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *The International Journal of Advanced Manufacturing Technology 29*(9-10), 990–1001.

Mehta, S. V. and R. Uzsoy (1998). Minimizing total tardiness on a batch processing machine with incompatible job families. *IIE transactions 30*(2), 165–178.

Mönch, L., H. Balasubramanian, J. W. Fowler, and M. E. Pfund (2005). Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. *Computers & Operations Research 32*(11), 2731–2750.

Mönch, L., J. W. Fowler, and S. J. Mason (2012). *Production planning and control for semiconductor wafer fabrication facilities: modeling, analysis, and systems*, Volume 52. Springer Science & Business Media.

Van Laarhoven, P. J., E. H. Aarts, and J. K. Lenstra (1992). Job shop scheduling by simulated annealing. *Operations research 40*(1), 113–125.

Yugma, C., S. Dauzère-Pérès, C. Artigues, A. Derreumaux, and O. Sibille (2012). A batching and scheduling algorithm for the diffusion area in semiconductor manufacturing. *International Journal of Production Research 50*(8), 2118–2132.

Order Acceptance and Scheduling Problem with Batch Delivery

İstenç Tarhan and Ceyda Oğuz

Koç University, Turkey
itarhan15@ku.edu.tr, coguz@ku.edu.tr

Keywords: Order acceptance, scheduling, batch delivery, sequence dependent setup times, metaheuristics, dynamic programming

1 Introduction

Scheduling problems are highly diverse though the principal objective of them is similar, which is satisfying the orders while utilizing the scarce resources as efficiently as possible. However, as the firms operating on a make-to-order basis, satisfaction of the entire demand may not be possible due the capacity limitations and tight delivery time requirements faced by the firm. This necessitates selecting only part of customer orders to maximize the total revenue, which gives rise to the order acceptance and scheduling (OAS) problems. We consider a make-to-order production system, where limited production capacity and order delivery requirements necessitate selective acceptance of the orders. It is often assumed that each order is delivered individually (i.e. immediately after their completion). However, orders may be sent in batches to decrease the transportation cost. Therefore, batching decisions along with order acceptance and scheduling decisions should be considered. Since batching decisions directly affect the tardiness, all decisions should be taken jointly. Herein, we study the problem called as the order acceptance and scheduling problems with batching (OASB) and present an iterated local search algorithm (ILS) to solve it.

2 Literature review

Charnsirisakskul *et. al.* (2004) define an order acceptance problem in which the customer does not place an order if the manufacturer cannot complete the order by the latest acceptable due date. Oguz *et. al.* (2010) study a different version of the problem where sequence dependent setup times and release times are included. They develop a simulated annealing based heuristic and two constructive heuristics. Cesaret *et. al.* (2012) propose a tabu search algorithm for the same problem that improves the best solution of many test instances. Chaurasia and Singh (2016) develop two hybrid metaheuristic approaches, a hybrid steady-state genetic algorithm and a hybrid evolutionary algorithm and improved the best solutions further.

The timing issue and package delivery is determined first by Cheng and Kahlbacher (1993). Although there have been studies regarding batch delivery in the following years such as Potts and Kovalyov (2000), Hall and Potts (2005) and Cakici *et. al.* (2014), batch delivery in the OAS problem is addressed for the first time by Khalili *et. al.* (2016). Authors propose an imperialist competitive algorithm for which the gap between the best solution found by CPLEX solver in 3600s and the solution found by the proposed algorithm can be up to 150%. Computational results of this study show that better solution methodologies can be developed for this problem.

3 Problem definition

In a single machine environment, we are given a set of independent orders O at the beginning of the planning period. For each order $i \in O$, we have data on its customer q_i, $q_i \in Q$, its release time r_i, processing time p_i, due date d_i, deadline $\bar{d}_i$ such that $d_i \leq \bar{d}_i$, sequence dependent setup times where each element $st_{i,j}$ is the time that has to be incurred before order j is processed, if order i immediately precedes order j , revenue e_i which denotes the maximum gain from order i, and unit tardiness penalty cost w_i. Orders are delivered in batches to the customers (it is assumed that there are infinite number of uncapacitated vehicles for delivery). Each batch can only contain the orders belonging to a single customer and it can be delivered only if all orders in that batch are completed. Therefore, the delivery time of an order i, del_i, is the completion time of the latest order in batch k_i, $k_i \in K$, that includes order i, $\bar{c}_{k_i}$; $del_i = \bar{c}_{k_i}$ and $\bar{c}_{k_i} = \max_{j:k_j=k_i} c_j$ where c_j is the completion time of order j. The manufacturer may deliver order i until its deadline $\bar{d}_i$, but for each time unit beyond its due date, she incurs a tardiness penalty cost. Accordingly, tardiness of order i, T_i is equal to $max\{0, del_i - d_i\}$.

Given a sequence σ_s of the selected orders $S \subseteq O$ and the number of batches n_b (in another sense, deliveries) of corresponding sequence, revenue generated from order i, denoted by $R_i(\sigma_s)$, is calculated as $R_i(\sigma_s) = max\{0, e_i - T_i w_i\}$. Consequently, the total revenue gained from processing orders in S in sequence σ_s is $TR(\sigma_s) = \sum_{i \in S} R_i(\sigma_s)$ and net revenue is $TR - n_b f$ where f is the fixed cost of a delivery. Hence the OASB problem is to find the set S, the sequence σ_s and its batching configuration so that the net revenue is maximized. We are not presenting the corresponding MILP here due to page limitation. In a nutshell, MILP consists of three groups of constraints regarding i) sequence of orders, ii) batch configuration of orders and iii) computation of tardiness, completion and delivery times.

An extensive set of test instances with 10, 15, 25 and 50 orders is solved by CPLEX using the MILP formulation. Results show that MILP can handle all instances with 10 orders and some instances with 15 orders. However, none of the instances with 25 and 50 orders can be solved to optimality by MILP in one hour. Therefore, we propose an iterated search algorithm that is capable of solving the majority of the instances with 10 orders to optimality and for the instances with higher number of orders, providing better solutions than CPLEX in much shorter time.

4 Proposed metaheuristic algorithm

The general steps of the proposed metaheuristic algorithm ILS are given in Algorithm 1. Solution x is encoded as a sequence including all orders. The proposed algorithm is founded on two types of neighborhoods: *Swap* and *Insertion*. Swap neighborhood of a solution x includes all solutions that can be obtained by swapping any two orders of it. Insertion neighborhood of a solution x includes all solutions that can be obtained by shifting a single order or consecutive two orders in the schedule of solution x. In the OASB problem, even though the corresponding neighborhoods may not include a solution that is better than the incumbent solution, they are likely to include a solution having the same objective value with the incumbent solution. Thus, it is promising to employ moves, which have the same objective value as the incumbent solution, simultaneously. To this end, *SwapXSwap*, *SwapXInsertion* and *InsertionXInsertion* neighborhoods include all solutions generated by simultaneously employing promising two swap moves, swap and insertion move, and two insertion moves, respectively. For each neighborhood defined, there is a function returning the best solution in the corresponding neighborhood and they are employed in variable neighborhood search manner.

If the best solution cannot be improved through these moves, $Perturbation(x^*)$ is called to perturb the best solution. $Perturbation(x^*)$ is performed by first dividing solution x^* into a set of blocks and then resequencing these blocks. The first part, that is dividing solution x^* into a set of unconnected blocks of orders, is realized by ejecting a set of orders chosen randomly. The second part, that is forming solution x again, is accomplished by obtaining the optimal sequence of these blocks via a dynamic programming algorithm. To avoid cycling, that is to ensure that the perturbed solution is different from x^*, the ejected orders are not allowed to be appended to the end of blocks that precede them in solution x^*.

The algorithm returns to Step 1 after the perturbation and terminates after a certain number of perturbations.

Algorithm 1 Pseudocode of the proposed algorithm

Input: Current solution x, Objective value of x is $f(x)$, best solution x^*, perturbation number $p = 0$, maximum number of perturbations $maxp$

$Update\_best\_solution(x')$ {$x \leftarrow x'$, $x^* \leftarrow x'$, go to line 2}

1: **while** $p < maxp$ **do**
2: $x' := Swap(x)$
3: **if** $f(x') > f(x)$ **then** $Update\_best\_solution(x')$ **end if**
4: $x' := Insertion(x)$
5: **if** $f(x') > f(x)$ **then** $Update\_best\_solution(x')$ **end if**
6: $x' := SwapXSwap(x)$
7: **if** $f(x') > f(x)$ **then** $Update\_best\_solution(x')$ **end if**
8: $x' := SwapXInsertion(x)$
9: **if** $f(x') > f(x)$ **then** $Update\_best\_solution(x')$ **end if**
10: $x' := InsertionXInsertion(x)$
11: **if** $f(x') > f(x)$ **then** $Update\_best\_solution(x')$ **end if**
12: $x := Perturbation(x^*)$
13: $p++$
14: **end while**

5 Computational results

The proposed algorithm ILS is coded in C++ and the MILP model of the OASB problem is solved by CPLEX 12.5.1 for comparison purposes. All computations are executed in an Intel Core i7 with 2.60 GHz and 8 GB of RAM running Windows 7. The proposed algorithm is tested with the benchmark instances suggested in Cesaret *et. al.* (2012). Test-problems have four different sizes (number of orders), more specifically, n = 10, 15, 25 and 50. Two additional parameters are used to create instances with varying characteristics, namely τ and R. The first is a tardiness factor, while the second is a due date range; both parameters were chosen from 3 possible values: 0.1, 0.5 and 0.9. For each combination of these parameters, ten instances are solved and average results are provided in Table 1.

CPLEX is set to terminate in at most one hour and under this limitation, it is unable to provide tight upper bounds when the the number of orders is higher than 10. Thus, the optimality gap of the proposed algorithm seems high. However, it outperforms CPLEX in terms of both solution quality and time when n is larger than 10. While CPLEX could solve all instances with 10 orders to the optimality, the proposed algorithm can solve the majority of these instances to the optimality in much shorter time.

n	R	τ	Optimality gap (%)		Solution time (s)	
			CPLEX	ILS	CPLEX	ILS
10	0,1	0,1	0,00	0,00	563,73	0,03
		0,5	0,00	0,14	158,72	0,02
		0,9	0,00	0,00	1,05	0,01
	0,5	0,1	0,00	0,22	882,63	0,03
		0,5	0,00	0,00	145,61	0,02
		0,9	0,00	0,68	2,50	0,01
	0,9	0,1	0,00	0,00	662,07	0,04
		0,5	0,00	0,08	144,50	0,02
		0,9	0,00	0,00	4,09	0,01
15	0,1	0,1	8,81	7,91	3600,00	0,06
		0,5	23,35	23,17	3600,00	0,04
		0,9	0,00	0,00	48,19	0,02
	0,5	0,1	9,24	9,19	3600,00	0,07
		0,5	24,35	23,09	3600,00	0,04
		0,9	2,34	2,34	689,15	0,02
	0,9	0,1	10,99	10,21	3600,00	0,06
		0,5	19,94	19,20	3600,00	0,04
		0,9	1,02	1,02	791,98	0,02
25	0,1	0,1	13,30	10,11	3600,00	0,60
		0,5	25,46	18,65	3600,00	0,28
		0,9	13,84	11,94	3600,00	0,12
	0,5	0,1	13,39	8,49	3600,00	0,62
		0,5	20,59	16,14	3600,00	0,31
		0,9	15,28	12,49	3600,00	0,12
	0,9	0,1	15,39	9,39	3600,00	0,56
		0,5	16,36	13,92	3600,00	0,26
		0,9	10,19	10,19	3600,00	0,13
50	0,1	0,1	18,12	7,53	3600,00	2,73
		0,5	34,34	15,04	3600,00	1,14
		0,9	*	*	*	*
	0,5	0,1	20,66	7,10	3600,00	2,96
		0,5	30,11	13,93	3600,00	0,99
		0,9	*	*	*	*
	0,9	0,1	18,18	6,19	3600,00	2,74
		0,5	11,24	6,24	3600,00	0,92
		0,9	*	*	*	*

Table 1: Computational comparison of the proposed algorithm and CPLEX
*Instances with asterisk cannot be compared since CPLEX runs out of memory.

6 Conclusion

We provide a competitive iterated local search algorithm ILS for the order acceptance and scheduling problem with batch delivery in a single machine environment. ILS includes a variable neighborhood search and a dynamic programming algorithm to perturb a solution if the algorithm is stuck at a local optima. Computational results show that the proposed algorithm can find the optimal solutions for the majority of the instances with 10 orders and can find better solutions than CPLEX for the instances with higher number of orders in much shorter time.

References

Cakici E., S. J. Mason, H. N. Geisman and J. W. Fowler, 2014, "Scheduling parallel machines with single vehicle delivery", *Journal of Heuristics*, Vol. 20, pp. 511-537.

Cesaret B., C. Oguz and F. S. Salman, 2012, "A tabu search algorithm for order acceptance and scheduling", *Computers and Operations Research*, Vol. 39, pp. 1197-1205.

Charnsirisakskul K., P. M. Griffin and P. Keskinocak, 2004, "Order selection and scheduling with leadtime flexibility", *IIE Transactions*, Vol. 36, pp. 194-205.

Chaurasia S. N., A. Singh, 2016, "Hybrid evolutionary approaches for the single machine order acceptance and scheduling problem", *Applied Soft Computing*, Vol. 177, pp. 2033-2049.

Cheng T. C. E., H. G. Kahlbacher, 1993, "Scheduling with delivery and earliness penalties", *Asia-Pacific Journal of Operational Research*, Vol. 10, pp. 145-152.

Hall N. G., C. N. Potts, 2005, "Scheduling with batching: A review", *European Journal of Operational Research*, Vol. 120, pp. 228-249.

Khalili M., M. Esmailpour and B. Naderi, 2016, "The production-distribution problem with order acceptance and package delivery: Models and algorithm", *Manufacturing Review*, Vol. 3, pp. 194-205.

Oguz C., F. S. Salman and Z. B. Yalcin, 2010, "Order acceptance and scheduling decisions in make-to-order systems", *International Journal of Production Economics*, Vol. 125, pp. 200-211.

Potts C. N., M. Y. Kovalyov, 2000, "Scheduling with batching: A review", *European Journal of Operational Research*, Vol. 120, pp. 228-249.

Energy Conscious Scheduling of Robot Moves in Dual-Gripper Robotic Cells

Nurdan Tatar[1], Hakan Gültekin[1], Sinan Gürel[2]

[1] TOBB University of Economy and Technology, Ankara, Turkey
{nurdantatar,hakan.gultekin}@gmail.com
[2] Middle East Technical University, Ankara, Turkey
gsinan@metu.edu.tr

Keywords: Robotic cell scheduling, Dual gripper, energy optimization, bicriteria optimization.

1 Introduction

In cellular manufacturing systems, one of the most used material handling devices is an industrial robot. Among the interrelated issues to be considered in using robotic cells, the scheduling of robot moves is one of the most significant ones. In such a robotic cell, the processing of the parts are performed by the machines and the transportation of the parts between the machines and the loading/unloading of the machines are performed by a material handling robot.

The robotic cells consist of an input device, a series of processing stages, an output device, and robots for material handling within the cell. Most robotic cells examined in the literature assume one of the two layouts: linear or circular (Dawande *et al.* 2005). On the other hand, two types of robots are discussed in the literature: A single gripper robot, which can hold only one part at a time, and in contrast, a dual-gripper robot, that can hold two parts simultaneously. If the robotic cell produces identical parts, we refer to it as a single part-type. The identical parts cyclic scheduling problem is then to find the shortest cyclic schedule for the robot; i.e., a sequence of robot moves that can be repeated indefinitely.

From an optimization point of view, the most widely used objective in the literature is that of maximizing the *throughput*; i.e. minimizing the cycle time. To achieve the maximum throughput (minimum cycle time) it is assumed that the robot performs the operations at its maximal speed. However, this may not be the most efficient energy consumption policy; at its maximum speed, the energy consumption is also at its maximum. On the other hand, the robot may need to wait in front of some of the machines to unload them because of reaching them earlier than necessary (before the processing is completed). Accordingly, there is a considerable potential for energy saving (Drobouchevitch *et al.* 2004).

In this study, we consider an m-machine robotic cell, where a dual-gripper robot is used, as illustrated in Figure 1. The robot moves linearly along a track (linear layout) and the system follows the flow shop assumption which means that each part goes through the same sequence of machines ($M_1 \rightarrow M_2 \rightarrow \ldots \rightarrow M_m$). However, the sequence of robot activities may be different. One of the decisions is to determine the optimal sequence of these robot activities. However, this is a complex task, because even in a two-machine dual-gripper robotic cells there is a total of 52 robot activity sequences that produce one part (1-unit cycles) (Sethi *et al.* 2001).

The other decision is to determine the robot's speeds in each of its moves to minimize the total energy consumption. Therefore, we consider a bicriteria problem. There are very few studies that consider bicriteria robotic cell scheduling problem. Gültekin *et al.* (2008, 2010) assumed that the processing times on the machines are controllable and considered

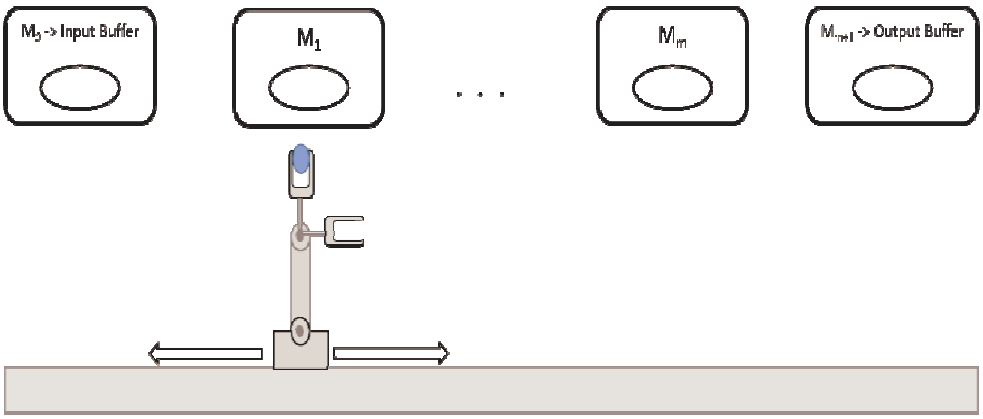


Fig. 1. A Dual Gripper Robotic Cell.

the problem of minimizing the cycle time and total manufacturing cost in a single gripper robotic cell. This study is the first one to consider the energy consumption of the robot together with the cycle time minimization objective in a dual gripper robotic cell. For this problem, we adapted the epsilon-constraint method and moved the cycle time objective to constraints by including an upper bound on it and developed a mixed integer nonlinear mathematical programming formulation (MINLP). This MINLP is solved using the BARON solver in GAMS. To improve its performance the nonlinear terms are reformulated to build conic quadratic representation. The new transformed model is a mixed integer quadratic conic programming (MIQCP) and solved with CPLEX 12.6.2.

In the next section, we define the problem and introduce the notation, in Section 3 we provide the solution methodology. Section 4 concludes the study.

2 Problem definition and notations

In this section, we develop a notational and mathematical modelling framework for the given problem. Being consistent with the existing studies in the literature, the following notation is used to describe the problem.

2.1 Parameters

ϵ: Loading/unloading time of machine
d_{ij}: Distance between machine M_i and M_j
v_{ij}: Traveling speed of the robot between machine M_i and M_j
δ_{ij}: Traveling time of the robot between machine M_i and M_j, $\delta_{ij} = d_{ij}/v_{ij}$
θ: The time for switching the robot grippers

2.2 Robot states

The following notation is used for our analysis of the robot states:

(g_1, g_2): where $g_i \in \{0, 1, \ldots, m+1\}$: represents the state of the grippers. For instance, (2,4) means the first gripper (g_1) has a part that requires processing next on machine 2 and the second gripper (g_2) has a part that requires processing next on machine 4, where $g_i = 0$ (e.g. state (0,0)) means that there is no part on the gripper i.

L_i: The robot activity that indicates a loading operation onto M_i. Just after loading a part on this machine and it has no part on the corresponding gripper. Therefore, just after this operation the corresponding gripper states are either $(0, g_2)$ or $(g_1, 0)$.

U_i: The robot activity that indicates an unloading operation from M_i. Just after performing this operation the robot has at least one part in one of its grippers that require processing next on machine M_{i+1}. Therefore the corresponding gripper states are either $((i+1), g_2)$ or $(g_1, (i+1))$.

In order to understand better, consider the following example of a dual gripper robot sequence for a two machine case: $U_0(1,0) \rightarrow U_1(1,2) \rightarrow L_1(0,2) \rightarrow U_2(3,2) \rightarrow L_2(3,0) \rightarrow L_3(0,0)$. As it can be seen, there are $2(m+1) = 6$ robot activities in this sequence in which $U_0(0,1)$ is the first. After unloading a part from the input buffer, the robot unloads another part with the other gripper from the first machine. Now, both grippers are full and the robot is in front of machine 1. Since both grippers are not empty $(g_1, g_2 \neq 0)$, the next robot activity must be a loading one. That is why we have a L_1 as the third activity. It means that the robot loads the part that it transported from the input buffer to M_1. Similar sequence of unloading and loading activities are performed on the second machine and finally the unloaded part from M_2 is transported to the output buffer.

Our mathematical programming formulation sequence the unloading and loading activities of the robot while satisfying the feasibility of this sequence. A feasible sequence must load and unload each machine once, must not try to load an already loaded machine, unload an already empty machine, unload a machine with a loaded gripper, or load a machine with an empty gripper. The model considers all such feasibility constraints, determines the starting times of all the sequenced activities together with robot move speeds.

3 Solution methodology

The minimization of the cycle time and the minimization of the energy consumption are conflicting objectives. In other words, improving one of them will sacrifice the other one and further achievement on the cycle time (energy consumption) can only be accomplished at the expense of higher energy consumption (cycle time).

To handle this bicriteria problem, we used the epsilon-constraint method, in which one of the objectives is written as a constraint with an upper bound on its value. By utilizing different upper bounds, different non-dominated solutions are generated. In this study, we considered the cycle time objective as a constraint. Therefore, the problem becomes the minimization of the total energy consumption subject to a given upper bound on the cycle time.

Figure 2 shows a set of Pareto efficient solutions for a 2-machine problem instance. It depicts the rate of change in the energy consumption and the change of optimal robot sequence when different cycle time upper bounds are used. In this figure Solution (1^*) corresponds to the problem where the robot's speeds are at their upper limits. If a machine has not completed the processing of a part when the robot arrives to unload it, the robot waits in front of the machine. In such cases, instead of waiting, the robot can make its previous moves slower. This situation corresponds to solution (2^*) in Figure 2. Which results in the same cycle time with Solution 1^* with a 28% less energy consumption. Increasing the cycle time upper bound by 5 units, changes the optimal robot activity sequence and leads to decrease in energy consumption as depicted in Solution (3^*). When the cycle time upper bound is large enough, robot's speeds of all moves becomes equivalent to their lower bounds. In this case, the second gripper is never used as depicted in Solution (4^*).

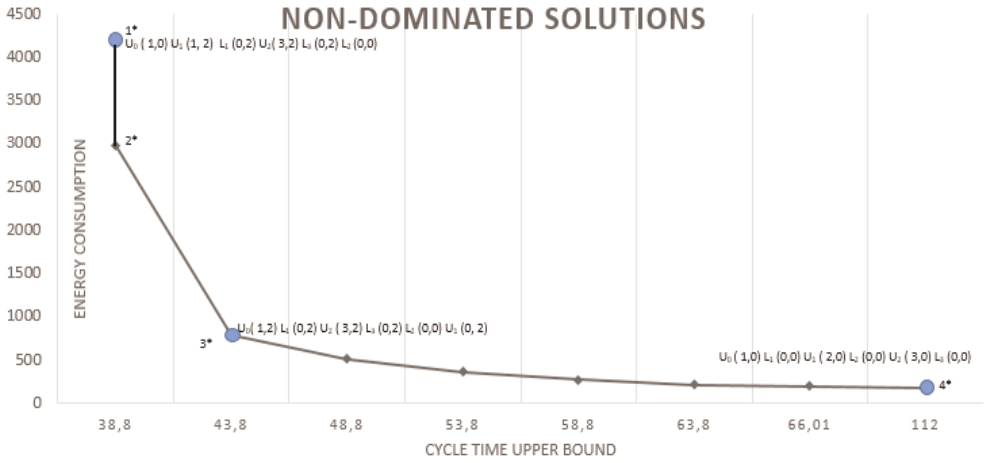


Fig. 2. Cycle Time vs. Energy Consumption.

Table 1 shows the elapsed times to attain the above solutions with MINLP and MIQCP formulations. To get all 8 non-dominated solutions took 66.7 seconds with MIQCP and 11529.4 with MINLP for this problem instance.

Table 1. Performance of MIQCP and MINLP for a 2-machine problem

	MIQCP	MINLP
CT	**Time** (s)	**Time** (s)
38.8	10.1	3646.2
43.8	9.0	3183.1
48.8	5.9	2222.4
53.8	10.8	1571.0
58.8	10.9	274.4
63.8	5.0	384.8
66.01	8.4	162.3
112	6.6	85.3
Total	66.7	11529.4

To test the performance of MIQCP and MINLP models, 100 instances of the problem are generated with different parameter values. In these test problems, the number of machines varies between 2 and 6. For two-machine instances, all non-dominated solutions are evaluated optimally with both MIQCP and MINLP. However, for 3 and 4 machine instances, MINLP could not find any solution within the given time limit of 3 hours, whereas MIQCP was able to find each non-nominated solution in 23.3 and 231.7 seconds for 3 and 4 machines, respectively. However, it was not possible to solve the instances with 5 and 6 machines with MIQCP with three-hour time limit. When robotic cells with speed control is compared with robotic cells without any speed control, our results reveal that the controllability of robot speeds yields 30% energy savings on the average.

4 Conclusion

This study, addresses a flow shop robotic cell scheduling problem consisting of m-machines, each of which performs a different operation on the parts, plus an input and output buffer, and a dual gripper robot that moves linearly along a track to transport the parts between the machines. We deal with a bicriteria scheduling problem for optimizing the cycle time and energy consumption of the robot at the same time.

We developed two mathematical models; a mixed integer nonlinear mathematical programming formulation and a mixed integer quadratic conic programming formulation. Both are evaluated with the same data sets and it is shown that MIQCP is much more efficient than the MINLP in terms of the solution time. However, for larger number of machines, the MIQCP formulation also fails to find solutions in reasonable times.

By means of the proposed approach of this study, which utilizes the controllability of the robot speeds, we can get not only an economic return but also an environmental benefit through reducing carbon emissions by decreasing the need for electric power across the manufacturing sector.

For further studies, we plan to develop a heuristic algorithm to solve large problem instances where the MIQCP formulation is not sufficient. Also, multiple part-type case can be considered instead of producing identical parts.

Acknowledgements

This research is supported by the Scientific and Technological Council of Turkey (TUBİTAK) under grant number 215M845.

References

Dawande, M., Geismar, H., Sethi, S., and Sriskandarajah, C., 2005, "Sequencing and Scheduling in Robotic Cells: Recent Developments", *Journal of Scheduling*, Vol. 8(5), pp. 387–426.

Drobouchevitch, I.G., Sethi, S., Sidney, J., and Sriskandarajah, C., 2004, "A note on scheduling multiple parts in two-machine dual gripper robotic cell: Heuristic algorithm and performance guarantee", *International Journal of Operations and Quantitative Management*, Vol. 10(4), pp. 297–314.

Gültekin, H., Akturk, M.S., Karasan, O.E., 2008, "Bicriteria robotic cell scheduling", *Journal of Scheduling*, Vol. 11(6), pp. 457–473.

Gültekin, H., Akturk, M.S., Karasan, O.E., 2010, "Bicriteria robotic operation allocation in a flexible manufacturing cell",", *Computers & Operations Research*, Vol. 40(2), pp. 639–653.

Sethi, S., Sidney, J., and Sriskandarajah, C., 2001, "Scheduling in dual gripper robotic cells for productivity gains", *IEEE Transactions on Robotics and Automation*, Vol. 17(3), pp. 324–341.

A continuous-time assignment-based MILP formulation for the resource-constrained project scheduling problem

Tom Rihm, Nadine Saner, Norbert Trautmann and Adrian Zimmermann

University of Bern, Switzerland
norbert.trautmann@pqm.unibe.ch

Keywords: Resource-constrained project scheduling problem, RCPSP, continuous-time mixed-integer linear programming model, experimental performance analysis.

1 Introduction

The resource-constrained project scheduling problem (RCPSP) can be described as follows: given is a set of completion-start precedence-related project activities that require time and scarce resources for execution; sought is a vector of start times for the activities such that all precedence relationships are respected, the total required quantity of each resource never exceeds its prescribed capacity, and the total project duration is minimized. The RCPSP poses a challenging combinatorial optimization problem; in addition to many problem-specific solution approaches, various types of mixed-integer linear programming (MILP) models have been proposed, which now receive increased attention due to the improved performance of MILP solvers and computer hardware.

Two classes of models exist (cf. Artigues *et al.* 2015): discrete-time (DT) models and continuous-time (CT) models. In DT models, the planning horizon is divided into a set of equal-length time intervals, and activities can start only at the beginning of each of these intervals; by contrast, in CT models, activities can start at any point in time over the planning horizon. In general, DT models involve time-indexed binary variables, e.g., pulse variables, cf. Pritsker *et al.* (1969) and Christofides *et al.* (1987); step variables, cf. Kaplan (1988) and Klein (2000); step variables and percentage-of-completion variables, cf. Bianco and Caramia (2013); or on/off variables, cf. Kopanos *et al.* (2014). In all these models, the number of binary variables grows with the number of time intervals considered, which is disadvantageous in the case of long activity durations. In the well-known CT model of Artigues *et al.* (2003), resource flow variables are used to model the resource constraints. According to Koné *et al.* (2011), for instances with short planning horizons, DT models exhibit a better performance than CT models; for instances with relatively long planning horizons, however, CT models provide better results than DT models.

In this paper, we present a novel CT model for the RCPSP; a preliminary version of the model, with some redundant constraints, has been published in Rihm and Trautmann (2017). To model the resource constraints, we use two types of binary variables: assignment variables specify which individual resource units are used for the execution of each activity, and sequencing variables specify the order in which pairs of activities that are assigned to the same resource unit are processed. To enhance the performance of the model, we modify the sequencing constraints for pairs and triplets of activities that cannot be processed in parallel, and we eliminate some symmetric solutions from the search space. In a comparative analysis, we have applied the new model to two standard test sets from the literature. Our computational results indicate that the model performs particularly well when resources are very scarce or when the range of activity processing times is rather high.

The remainder of this paper is structured as follows. In Section 2, we describe the novel MILP model. In Section 3, we report on the computational results. In Section 4, we present some concluding remarks and an outlook on future research.

2 Novel MILP formulation

Table 1 provides the nomenclature; activities 0 and $n+1$ are fictitious activities representing the start and the completion of the project, respectively. The basic novel continuous-time assignment-based MILP formulation, called CTAB in the following, reads as follows.

$$
\text{(CTAB)} \left\{
\begin{array}{lll}
\text{Min. } S_{n+1} & & (1)\\
\text{s.t. } \sum_{l=1}^{R_k} r_{li}^k = r_{ik} & (i \in V, k \in R) & (2)\\
r_{li}^k + r_{lj}^k \leq 1 + y_{ij} + y_{ji} & (i,j \in V, k \in R, l = 1,\ldots,R_k : i < j, (i,j) \notin TE) & (3)\\
S_i + p_i \leq S_j & ((i,j) \in E) & (4)\\
S_i + p_i \leq S_j + (LS_i + p_i)(1 - y_{ij}) & (i,j \in V : i \neq j, (i,j) \notin TE) & (5)\\
ES_i \leq S_i \leq LS_i & (i \in V) & (6)\\
y_{ij} \in \{0,1\} & (i,j \in V : i \neq j, (i,j) \notin TE) & (7)\\
r_{li}^k \in \{0,1\} & (i \in V, k \in R, l = 1,\ldots,R_k) & (8)
\end{array}
\right.
$$

The objective (1) is to minimize the makespan. Constraints (2) ensure that the required number of resource units is assigned to each activity. Constraints (3) link the resource-assignment variables to the sequencing variables: if the same resource unit is assigned to two activities i and j, then a sequence between these two activities is enforced. The sequencing variables y_{ij} are not defined for the pairs of activities (i,j) for which there is a path from i to j in the activity-on-node digraph $G = (V,E)$; the transitive closure TE of E consists of all these pairs of activities. Constraints (4) model the completion-start precedence relationships among the activities. Constraints (5) link the start time variables to the sequencing variables. Constraints (6) ensure that each activity starts between its earliest and its latest start time.

To enhance the performance of model CTAB, we include the following extensions:

1. For all pairs of activities i and j that cannot be processed in parallel, i.e., $r_{ik}+r_{jk} > R_k$ for some resource $k \in R$, we add the constraint $y_{ij}+y_{ji} = 1$. Analogously, for all triplets of activities i, j, and m that cannot be processed in parallel, i.e., $r_{ik}+r_{jk}+r_{mk} > R_k$ for some resource $k \in R$, we add the constraint $y_{ij}+y_{ji}+y_{im}+y_{mi}+y_{jm}+y_{mj} \geq 1$.
2. All units of each resource are identical. Therefore, to eliminate some symmetric solutions from the search space w.l.o.g., for each resource $k \in R$, we select an activity i with largest requirement r_{ik} for this resource and assign the first r_{ik} resource units to its execution by prescribing $r_{li}^k = 1$ for $l = 1,\ldots,r_{ik}$.

3 Computational study

In this section, we compare the performance of the MILP model proposed in Section 2 to the performance of the well-known models presented in Pritsker *et al.* (1969), Christofides *et al.* (1987), Klein (2000), Kopanos *et al.* (2014), and Artigues *et al.* (2003). We implemented the MILP models in AMPL, and we used the Gurobi Optimizer 7.5 with the default solver settings to solve the models. We performed all computations on a workstation equipped with two 6-core Intel Xeon X5650 CPUs (2.66 GHz, 24 GB RAM). We set the solver time limit to 500 seconds per instance and limited the number of used threads to 4. For the comparative analysis, we used the j30 set (480 instances) from the PSPLIB (cf. Kolisch and Sprecher 1996) and the Pack_d set (55 instances) generated by Koné *et al.* (2011). Tables 2–4 summarize the results for test set j30, for the instances of test set j30 with resource strength 0.2 (i.e. with very scarce resources), and for test set Pack_d, respectively.

For the instances of set j30, the results can be summarized as follows. All models except the model of Kopanos *et al.* (2014) provide a feasible solution to each instance (column # Feas). With respect to the number of instances for which an optimal solution is found and optimality is proven by the solver within the time limit (# Opt), the models of Pritsker *et al.* (1969), Christofides *et al.* (1987) and Kopanos *et al.* (2014) perform best; the same holds for the number of instances for which, among all models, a best solution is obtained (# Best). The average relative deviation (Gap^{bb}) between the objective function value (OFV) and the lower bound (LB) provided by the solver, i.e. $(OFV - LB)/LB$, is the lowest for the models of Pritsker *et al.* (1969) and Christofides *et al.* (1987). The lowest average relative deviation between the OFV and the critical-path based lower bound (Gap^{CPM}) and the best OFV returned by any of the models (Gap^{best}), respectively, is obtained by the model of Pritsker *et al.* (1969) and the model presented in this paper. For the instances with resource strength 0.2, the extended model presented in this paper even outperforms the other models. For the instances of set Pack_d, which have considerably longer activity durations than the instances of set j30, a feasible solution for all instances has been obtained only by the model of Klein (2000) and the model presented in this paper; however, the deviation from the lower bounds is notedly larger for the model of Klein (2000) than for the model presented in this paper.

4 Conclusions

In this paper, we have proposed a novel continuous-time MILP model for the RCPSP which is based on binary variables that represent the assignment of the project activities to individual resource units and the sequential relationships between activities that are assigned to at least one identical resource unit. In future research, further possibilities to eliminate some symmetric solutions from the search space should be exploited, and the novel model should be compared against other models known from the literature.

References

Artigues, C., Koné, O., Lopez, P., Mongeau, M., 2015, "Mixed-integer linear programming formulations", *in* C. Schwindt and J. Zimmermann (eds), *Handbook on Project Management and Scheduling Vol. 1*, Cham: Springer, pp. 17–41.

Artigues, C., Michelon, P., Reusser, S., 2003, "Insertion techniques for static and dynamic resource-constrained project scheduling", *Eur J Oper Res*, Vol. 149, No. 2, pp. 249–267.

Bianco, L., Caramia, M., 2013, "A new formulation for the project scheduling problem under limited resources", *Flex Serv Manu J*, Vol. 25, No. 1–2, pp. 6–24.

Christofides, N., Alvarez-Valdés, R., Tamarit, J. M., 1987, "Project scheduling with resource constraints: a branch and bound approach", *Eur J Oper Res*, Vol. 29, No. 3, pp. 262–273.

Kaplan, L., 1988, "Resource-constrained project scheduling with preemption of jobs", PhD thesis, University of Michigan.

Klein, R., 2000, "Scheduling of resource-constrained projects", Boston: Kluwer.

Kolisch, R., Sprecher, A., 1996, "PSPLIB-a project scheduling problem library", *Eur J Oper Res*, Vol. 96, No. 1, pp. 205–216.

Koné, O., Artigues, C., Lopez, P., Mongeau, M., 2014, "Event-based MILP models for resource-constrained project scheduling problems", *Comp Oper Res*, Vol. 38, No. 1, pp. 3–13.

Kopanos, G. M., Kyriakidis, T. S., Georgiadis, M. C., 2014, "New continuous-time and discrete-time mathematical formulations for resource-constrained project scheduling problems", *Comput Chem Eng*, Vol. 68, pp. 96–106.

Pritsker, A. A. B., Waiters, L. J., Wolfe, P. M., 1969, "Multiproject scheduling with limited resources: a zero-one programming approach", *Manage Sci*, Vol. 16, No. 1, pp. 93–108.

Rihm, T., Trautmann, N., 2017, "An assignment-based continuous-time MILP model for the resource-constrained project scheduling problem", *in* de Meyer, A., Chai, K.H., Jiao, R., Chen, N., Xie, M. (eds), *Proc 2017 IEEE Int Conf on Ind Eng Eng Mgmt*, Singapore, 35–59

Table 1. Nomenclature of the novel MILP formulation

V	Set of all activities ($V := \{0, \ldots, n+1\}$)
p_i	Processing time of activity $i \in V$
E	Set of all precedence relationships
TE	Transitive closure of E
ES_i	Earliest start time of activity $i \in V$
LS_i	Latest start time of activity $i \in V$
R	Set of all renewable resources
R_k	Capacity of resource $k \in R$
r_{ik}	Requirement of resource $k \in R$ per time period for execution of activity $i \in V$
$*$ S_i	Start time of activity i
$*$ y_{ij}	$\begin{cases} = 1, & \text{if activity } i \text{ must be completed before the start of } j; \\ = 0, & \text{otherwise.} \end{cases}$
$*$ r_{li}^k	$\begin{cases} = 1, & \text{if activity } i \text{ is processed on unit } l \text{ of resource } k; \\ = 0, & \text{otherwise.} \end{cases}$

Table 2. Overall results for test set j30 (480 instances)

Formulation	**# Feas**	**# Opt**	**# Best**	**Gap^{bb} (%)**	**Gap^{CPM} (%)**	**Gap^{best} (%)**
PriWaiWol69	480	443	451	1.0	13.8	0.2
ChrAlvTam87	480	446	453	0.9	13.9	0.3
Kle00	480	432	447	1.8	14.1	0.4
KopKyrGeo14	478	446	455	1.3	14.0	0.4
ArtMicReu03	480	354	389	10.8	16.2	1.7
CTAB basic	480	374	406	3.6	14.4	0.7
CTAB extended	480	417	444	1.8	13.8	0.2

Table 3. Results for j30 instances with resource strength 0.2 (120 instances)

Formulation	**# Feas**	**# Opt**	**# Best**	**Gap^{bb} (%)**	**Gap^{CPM} (%)**	**Gap^{best} (%)**
PriWaiWol69	120	83	91	3.8	45.7	0.7
ChrAlvTam87	120	86	93	3.5	46.3	1.0
Kle00	120	72	87	7.2	46.9	1.4
KopKyrGeo14	118	86	95	5.1	46.9	1.7
ArtMicReu03	120	39	53	40.0	54.7	6.3
CTAB basic	120	79	90	9.6	46.3	1.2
CTAB extended	120	100	112	4.6	44.8	0.2

Table 4. Overall results for test set Pack_d (55 instances)

Formulation	**# Feas**	**# Opt**	**# Best**	**Gap^{bb} (%)**	**Gap^{CPM} (%)**	**Gap^{best} (%)**
PriWaiWol69	48	6	6	99.2	218.5	59.2
ChrAlvTam87	0	0	0	-	-	-
Kle00	55	4	4	292.1	298.7	95.4
KopKyrGeo14	3	1	1	53.2	55.1	31.6
ArtMicReu03	48	5	16	103.9	103.9	4.7
CTAB basic	55	17	40	19.3	111.1	0.2
CTAB extended	55	19	53	12.4	110.7	0.1

A heuristic procedure to solve the integration of personnel staffing in the project scheduling problem with discrete time/resource trade-offs

Mick Van Den Eeckhout[1], Mario Vanhoucke[1,2,3] and Broos Maenhout[1]

[1] Faculty of Economics and Business Administration, Ghent University,Tweekerkenstraat 2, 9000 Gent (Belgium)
mick.vandeneeckhout@ugent.be,mario.vanhoucke@ugent.be,broos.maenhout@ugent.be
[2] Technology and Operations Management Area, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, UK

Keywords: Project scheduling, staffing, discrete time/resource trade-offs, decomposition.

1 Introduction

Project scheduling and personnel staffing are two complementary optimisation problems. In project scheduling, activities are scheduled given precedence relations between these activities and a constant resource availability. Hartmann and Briskorn (2010) discuss the characteristics of the resource-constrained project scheduling problem (RCPSP) and give an overview of different extensions. Personnel resources are one of the most important resources in project planning, accounting for 30-50% of the total project cost (Adrian 1987) and therefore it is essential to determine the personnel budget to carry out a project. The personnel budget results from the composition of a staffing plan and is based on the staffing requirements generated by the project schedule. However, in personnel staffing time-related constraints are imposed on the scheduling of individual workers, which complicate the scheduling of the resources (see Van den Bergh *et. al.* (2013) for a full overview).

Tackling these two problems sequentially leads to sub-optimal outcomes. On the one hand, the scheduling of activities determines the staffing requirements and should thus be in line with the personnel staffing. On the other hand, personnel supply is an important factor when activities need to be scheduled. When integrating these two-interrelated problems, benefits can thus arise in both directions. First, additional flexibility is provided for the project manager if resource scheduling is included. Second, demand management can be applied to improve the resource utilisation.

2 Problem Definition

As stated above, integrating personnel staffing in project planning increases the schedule flexibility since resource availabilities can be adapted to the project scheduling requirements. On top of that, additional schedule flexibility is considered by incorporating different modes for each activity. Each activity mode is defined by a trade-off between duration and resource demand, where a longer duration will lead to a smaller resource demand. Only one type of (renewable) resources is considered, namely personnel resources, which are divided in regular and temporary workers. The scheduling of the regular workers implies a manpower days-off scheduling problem with time-related constraints (Van den Bergh *et. al.* 2013), whereas temporary workers are hired for a single day. The imposed time-related

constraints limit the minimum and maximum weekly assignments, the minimum and maximum consecutive days-on and the minimum and maximum consecutive days off for a single worker. Given the incorporation of multiple modes for each activity and the use of personnel as only resource, the problem lies in line with the discrete time/resource trade-off problem in project management (Ranjbar *et. al.* 2009).

When integrating the two presented problems, an accurate objective function should be chosen. In project planning, makespan minimisation is the most common objective, whereas cost minimisation is the primary focus of resource scheduling. These two objectives are thus conflicting since a short makespan will lead to higher personnel costs and vice versa. To overcome this issue, a fixed deadline is proposed resulting in strategic budgeting problem which determines the size of the personnel staff. The objective function makes a trade-off between the number of regular and temporary workers, since a regular worker should be paid the entire planning period and a temporary workers has a higher daily cost.

3 Methodology

An iterative heuristic solution procedure was developed to solve the integrated project scheduling and personnel staffing problem. This procedure combines a heuristic framework, namely an iterated local search (ILS), with optimal solution procedures in the local search step. A generic framework of an iterated local search is presented in algorithm 1, where the different steps will be explained below.

Algorithm 1 Iterated Local Search (Lourenço *et. al.* 2010)

1: s_0 = Generate Initial Solution()
2: s^* = Local Search (s_0)
3: **repeat**
4: s' = Perturbation (s^*, history)
5: s'^* = Local Search (s')
6: s^* = Acceptance Criterion (s^*, s'^*, history)
7: **until** termination condition met

Due to the observation that the initial solution of the local search is of great importance for the performance of the algorithm (Lourenço *et. al.* 2010), different methods were developed to create this initial solution. Instead of generating one solution, a pool of solutions was created wherefrom the best solution was selected. This pool can be created by creating random projects or by incorporating information from the linear programming (LP) relaxation. In the first case, the assignment of a certain mode and a certain start time to an activity is based on an uniform distribution. In the second case, the probability distribution is biased by the fractional decision variable values of the LP relaxation.

The local search step is based on the two types of variables in our problem definition, namely project and personnel variables, resulting in two types of decomposition strategies, activity- and personnel-based decomposition. In activity-based decomposition, the complexity of the project scheduling problem is reduced by fixing a large set of activities and thus rescheduling only a small set of activities in a (limited) time period. In personnel-based decomposition, the complexity of the personnel scheduling problem is reduced by fixing certain staffing assignments or by fixing days-off/days-on in the underlying personnel

patterns. The iterated local search takes the solution quality of the local search and randomisation into account to perform a perturbation move.

4 Computational experiments

Different subsets of 30 instances, taking into account the different network topology measures (Vanhoucke *et. al.* 2008) and each subset having a different number of activities, were selected from the multi-mode Project Scheduling Problem LIBrary (Kolisch and Sprecher 1997) to test the quality of the solution procedure. Since only one type of renewable resources is needed, the original modes become inefficient and a random mode generation was invoked. Due to the integration of personnel scheduling, the daily resource availability is determined based on the personnel schedule instead of the defined constant resource availability. The deadline is set at the middle between the shortest and longest path, unless otherwise stated.

We compare our procedure with a branch-and-price procedure, a branch-and-bound method and a multi-start heuristic. The branch-and-price is based on Maenhout and Vanhoucke (2016) and includes an additional layer to branch on the activity modes. The branch-and-bound method only considers a limited set of personnel patterns, and is also used as initial upper bound in the previous mentioned paper. Both the branch-and-price and the branch-and-bound are truncated after 3600 seconds. To evaluate the ILS framework, a multi-start heuristics was programmed where in each iteration, the local search is applied on a random schedule. A stopping criteria of 100 iterations was imposed on the ILS and multi-start heuristic.

When the number of activities is limited to 10 and the deadline is set to the critical path, the expanded version of the branch-and-price finds the optimal solution for all considered instances. The solutions obtained by the ILS lie very close to the optimum, leading to the conclusion that the presented procedure can find near optimal solutions for small instances. The branch-and-bound does not perform well on these instances, which can be explained by the limited number of considered personnel patterns.

When the number of activities or the deadline increases, results indicate that the performance of the branch-and-price procedure deteriorates quickly. Certainly when the number of activities is high, the branch-and-price is unsuitable to find good solutions. Even the branch-and-bound procedure with a limited set of patterns has a better performance, given the time limit of one hour. The presented ILS framework outperforms the branch-and-bound, meaning that better results are obtained in smaller timeframes. Moreover, the proposed procedure outperforms the multi-start procedure, which advocates the use of the iterated local search as heuristic framework.

5 Conclusion

Integrating personnel staffing with project planning when discrete time/resource trade-offs are considered, is a challenging endeavour due to the high complexity. A heuristic procedure was developed which is based on iterated local search. The local search consists of decomposing the problem in smaller subproblems by applying different activity-based and personnel-based decomposition strategies. By relying on randomisation and solution quality to perform a perturbation move, the algorithm is able to reach good solutions in relatively small time frames. When comparing the presented algorithm with other optimal

or heuristic procedures, it is clear that the presented procedure outperforms the benchmarks on time and solution quality. Furthermore, for small instances, the proposed procedure leads to solutions close to the optimum.

References

Adrian, J., 1987, "Construction productivity improvement", *Elsevier*

Hartmann, S. and Briskorn, D., 2010, "A survey of variants and extensions of the resource-constrained project scheduling problem", *European Journal of Operations Research*

Kolisch, R. and Sprecher, A., 1997, "PSPLIB-a project scheduling problem library", *European Journal of Operations Research*

Lourenço, H. R., Martin, O.C. and Stützle, T., 2010, "Iterated Local Search: Framework and applications", *Handbook of metaheuristics*

Maenhout, B. and Vanhoucke, M., 2016, "An exact algorithm for an integrated project staffing problem with a homogeneous workforce", *Journal of Scheduling*

Ranjbar, M., De Reyck, B. and Kianfar, F., 2009, "A hybrid scatter-search for the discrete time/resource trade-off problem in project scheduling", *European Journal of Operations Research*

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E. and De Boeck, L, 2013, "Personnel scheduling: A literature review", *European Journal of Operations Research*

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B. and Tavares, L.V., 2008, "An evaluation of the adequacy of project network generators with systematically sampled networks", *European Journal of Operations Research*

Production and distribution planning for smoothing supply-chain variability

Marie-Sklaerder Vié[1], Nicolas Zufferey[1,2] and Leandro Coelho[2,3]

[1] Geneva School of Economics and Management, GSEM, University of Geneva, Switzerland
marie-sklaerder.vie@unige.ch, n.zufferey@unige.ch
[2] Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport, CIRRELT, Québec, Canada
[3] Canada Research Chair in Integrated Logistics, Université Laval, Canada
leandro.coelho@cirrelt.ca

Keywords: production planning, distribution planning, supply chain management, simulation, optimization.

1 Introduction

This study is motivated by a fast-moving consumer goods company, denoted as ABC. It has its European headquarters in Geneva and it cannot be named due to a non-disclosure agreement. We consider a flagship product widely used in the world. Currently, the supply chain is classically managed with a *decentralized pull* approach based on a *reorder-point* policy: every echelon orders to the upstream echelon whenever the available inventory level reaches the reorder point. The plant adjusts its production to these orders to minimize storage. In each echelon, *Economic Batch Quantities* (EBQs) are used according to how the products are transported (e.g., rounding up to pallets/layers/cases). Such a decentralized inventory-management approach leads to a significant volatility of the orders, and to an even stronger volatility of the production. This so-called *bullwhip effect* has been widely studied in the literature (Wang and Disney 2016). It can be caused by the following elements: (1) each ordered quantity is rounded up to an EBQ; (2) overestimation of the demand; (3) lead-time uncertainties along the supply chain. Consequently, a small demand can create a large production order because of these amplification effects. A historical review of current practices creating the bullwhip effect can be found in Geary *et. al.* (2006). It shows that most of the research on the topic either develops empirical/experimental studies that analyze historical data, or it proposes management games, or it seeks for a mathematical model to explain the effect and determine its factors. Surprisingly, only few studies tackle real problems with simulation, and none with the integrated approach proposed in this paper. However, many examples show that *integrated* optimization outperforms *sequential* optimization (Darvish and Coelho 2017, Thevenin *et. al.* 2017).

The reader is referred to Axsäter (2015) for an overview on inventory management. A basic inventory-management approach that avoids the bullwhip effect consists in daily producing the same amount, computed as the average forecasted demand over a long horizon. The production is then pushed along the supply chain down to the shops. However, even if this *push* approach perfectly smoothes the production, it is far from optimality wiht respect to shortage and inventory costs. Hence, a balance needs to be found between these *pull* and *push* methods. The main contribution of this work is the design of an integrated *planning-by-range* (PBR) approach for a real-world problem (P) described in Section 2 (relying on an efficient simulation-optimization algorithm), along with the adaptations of the classical *pull* and *push* approaches for (P). These methods are numerically compared in Section 3. As highlighted in the conclusion (Section 4), PBR can be easily adapted to other

supply chain networks, for which reducing the following features is important: shortage, bullwhip effect and inventory level.

2 Presentation of the problem (P) and of the proposed PBR approach

The considered 3-echelon supply chain is made of one plant, one distribution center (DC), and dozens of shops. EBQ constraints have to be satisfied: for each day t, the shipments $s^t_{P,DC}$ from the plant to the DC (resp. $s^t_{DC,x}$ from the DC to each shop x) have to be in number of layers (resp. cases). The number of cases per layer (resp. item per cases) is n_l (resp. n_c). Storage is allowed at the DC (resp, in each shop x), and the current inventory for each day t is denoted i^t_{DC} (resp. i^t_x). At the plant level, storage is not allowed, and for each day t, the production p^t must be in the ideal production range $[P_{\min}, P_{\max}]$ (given in number of layers per day). This range aims to mitigate the bullwhip effect. Producing out of these limits is penalized. For ABC, the daily range is set to $\pm 20\%$ of the average daily demand computed over a planning horizon of 100 days (assuming the daily demand follows a normal distribution). The considered lead-times (in days) are the following: $L(P) = 1$ between production and availability (at the plant level) for shipment to DC, $L(P, DC) = 2$ from the plant to the DC, $L(DC) = 1$ for cross-docking through the DC, $L(DC, x) = 1$ from the DC to any shop x. Finally, the expected demand (resp. the corresponding lost sales) for each day t and each shop x is denoted d^t_x (resp. l^t_x).

The only structural constraints for the supply chain are the material flow conservation in each echelon (first in the plant, second in the DC, and third in each shop):

$$\left\{ \begin{array}{rl} s^t_{P,DC} & = p^{t+L(P)} \\ i^t_{DC} & = i^{t-1}_{DC} + n_l \cdot s^{t+L(P,DC)}_{P,DC} - \sum_x s^t_{DC,x} \\ i^t_x & = i^{t-1}_x + n_c \cdot s^{t+L(DC)+L(DC,x)}_{DC,x} - d^t_x + l^t_x \end{array} \right. \quad (C)$$

Instead of the usual reorder-point approach, we propose to associate a daily MIN-MAX range $S(x)$ with each shop x. The MIN is what x needs to satisfy its daily demand, whereas the MAX is the largest *desired* daily inventory for the considered product. MAX is defined as the available part of two quantities: the shelf capacity (which is used first) plus the back-room capacity assigned to the product (it is not a hard constraint, as each shop has other products and therefore can find a place in the back-room if there is too much inventory of the considered product). The sum of these two quantities is denoted $M(x)$ for each shop x. Consequently, the production can be planned based on the aggregation of the $S(x)$ ranges, and then pushed down to the DC (where the product can be temporarily stored but a storage penalty is due) and finally to the shops. The production plan is further calibrated in order to perfectly satisfy the EBQ constraints. This approach should allow (1) the DC having a much more stable response (even while keeping a low inventory), and (2) the plant smoothing its peaks of production (mitigating the bullwhip effect). On the one hand, PBR is innovative in the sense that from the shop perspective, any stock within its associated range $S(x)$ is not penalized (in contrast with the usual unit storage costs). On the other hand, from the plant perspective, the production variability within the range $[P_{\min}, P_{\max}]$ is not penalized.

Three different objective functions are considered, ranked in a lexicographic order from f_1 to f_3 (i.e., a higher-level objective f_i is infinitely more important than a lower-level objective f_{i+1}). The lexicographic approach $f_1 > f_2 > f_3$ was validated by ABC. The three objective functions are, for each day t:

- $f_1(t) = \sum_x l^t_x$: shortage at the shop level (i.e., less than MIN);
- $f_2(t) = \max\{P_{\min} - p^t, 0\} + \max\{p^t - P_{\max}, 0\}$: production out of the ideal range;

– $f_3(t) = i^t + \sum_x \max\{i_x^t - M(x), 0\}$: excess of inventory in the shops and at the DC (i.e., more than MAX in the shops, and more than zero stock in the DC).

As the demand is non-deterministic, simulation involving a rolling horizon H is used (over a planning horizon of 100 days). The size of H is fixed to $L(P)+L(P,DC)+L(DC)+\max_x L(DC,x)+1$ (which results in 6 days for ABC). This way, two production decisions (i.e., involving the first two days of H) can reach the shop level within H. Note that each decision that cannot impact the stocks in the shops within H are set to zero. When simulating, for each shop x, only the demand of today $d_x^{t_0}$ and those of the previous days are known, and the forecast for the next day d_x^t is simply the average daily demand plus its standard deviation (as neither a trend nor a seasonality characterize the considered flagship product). On each day of the planning horizon, a 3-step optimization is performed, each step being solved by CPLEX. The resulting PBR approach is summarized in Algorithm 1.

Algorithm 1 Planning-by-range approach

Set $t_0 \leftarrow 1$

While $t_0 + |H| \leq 100$**, do:**

– for each shop x, set $d_x^{t_0}$ as the actual known demand (instead of forecast);
  1. minimize $F_1 = \sum_{t\in[t_0,t_0+|H|]} f_1(t)$ while satisfying the constraint set (C); let $F_1^\star$ be the obtained minimum;
  2. minimize $F_2 = \sum_{t\in[t_0,t_0+|H|]} f_2(t)$ while satisfying (C), and such as $F_1 = F_1^\star$; let $F_2^\star$ be the obtained minimum;
  3. minimize $F_3 = \sum_{t\in[t_0,t_0+|H|]} f_3(t)$ while satisfying (C), and such as $(F_1, F_2) = (F_1^\star, F_2^\star)$;

– freeze shipments and production of day t_0;

– set $t_0 \leftarrow t_0 + 1$.

3 Results

PBR is compared with the two standard supply-chain-management methods *pull* and *push* described in Section 1. On the one hand, *pull* minimizes the retailer's costs (i.e., shortage and inventory in the shops). On the other hand, *push* minimizes the manufacturer's costs (i.e., irregular production, and inventory in the DC).

Method *pull* uses reorder points for both the DC and the shops. Each time the available inventory level is below its reorder point, an order is placed to the upstream level. A production batch is launched each time an order comes from the DC, as the plant does not hold inventory. The reorder point is set equal to $(D + \sigma) \cdot L$, where D is the average daily demand from the downstream echelon, σ is the daily standard deviation of D, and L is the total lead-time from the upstream echelon (i.e., $L(P) + L(P, DC)$ for the DC, and $L(DC) + L(DC, x)$ for any shop x).

Method *push* first computes its ideal production rate p (in layers), which is the average daily demand over the whole planning horizon of 100 days. It is usually not an integer. To satisfy the EBQ constraint while having an almost constant production rate (i.e., around p), each day t (from $t = 1$ to $t = 100$), $\lceil p \cdot t - Q \rceil$ layers are produced, where Q is the number of layers produced until day t. The daily produced quantity is shipped to the DC as soon as possible, and whenever the DC has any inventory, it ships it to the shops without exceeding their *desired* inventories (as previously defined).

The three methods are compared for 20 instances I_1 to I_{20}, generated randomly based on the real data provided by ABC. Each instance is made of N shops. Instances I_1 to I_{10} have $N = 20$ shops, with a large daily average demand per shop (in [6, 12] cases), and a desired inventory per shop of 6 cases (2 for the shelf and 4 for the back-room). Instances I_{11} to I_{20} have $N = 50$ shops, with a small daily average demand per shop (in [1, 4] cases), and a desired inventory per shop of 3 cases (1 for the shelf and 2 for the back-room). For instances I_1 to I_5 and I_{11} to I_{15}, we have $\sigma \in [50, 100]\%$ of the average daily demand. In contrast, $\sigma \in [100, 150]\%$ for the other instances. For each group of five instances, the number of items per case/layer differs. More precisely, the numbers of items per case are $(6, 8, 12, 16, 20)$, whereas the corresponding numbers of cases per layer is $(14, 10, 14, 10, 12)$, those numbers being real data from five different pack materials of ABC. Table 1 presents the results for the considered methods. The following averaged indicators (over the planning horizon of 100 days) are given to capture the performance on f_1, f_2 and f_3, respectively. F_1 is the shortage percentage with respect to the average daily demand over all the shops. F_2 is the out-of-range production percentage. F_3 is the percentage of exceeding stock with respect to the total storage capacity that is not penalized. The latter is the sum of all the *desired* inventories in the shops (knowing that the desired inventory in the DC is always zero).

All the algorithms were coded with C++ under Linux, and run on 3.4 GHz Intel Quad-core i7 processor with 8 GB of DDR3 RAM. For each time window H, each method is able to find its solution within seconds, including CPLEX that always provides an optimal solution. CPLEX is based on the 3-step algorithm presented in Section 2, and it has a time limit of one minute per objective. For all three methods, the shortage penalty is logically smaller for the instances with a smaller σ (i.e., instances I_1 to I_5, and I_{11} to I_{15}). As expected, *pull* shows a very small shortage (on average, 0.76% of the demand), but a very irregular production pattern (on average, 61.03% of the production is out of the ideal range). The relatively big inventory (on average, 23.07% of the free-of-cost capacity) is due to storage at the DC. Unsurprisingly, *push* has no out-of-range production, but it has the biggest shortage (on average, 12.45% of the demand) as it does not adapt its production to the demand pattern. Interestingly, *push* shows much better results with a larger number of shops: the shortage indicator roughly goes down from 25% (with $N = 20$) to 5% (with $N = 50$). Indeed, the more shops there are, the more possibilities a fixed production has to be pushed down to the shops, and hence the better the dispatching of the production between the shops can be. PBR offers the best results: the average shortage is only 0.32% of the demand, the average out-of-range production is limited to 0.02%, and the average costly inventory is only 0.26% of the free-of-charge capacity. Remarkably, none of the PBR performance indicator exceeds 1%, and even 0.11% for the out-of-range production. The bullwhip effect is thus removed, while almost always avoiding shortage and inventory penalties.

Table 1. Comparison of *pull*, *push* and PBR approaches for realistic instances.

	Pull			Push			PBR		
Instance	F_1	F_2	F_3	F_1	F_2	F_3	F_1	F_2	F_3
I_1	0.55	75.00	18.12	18.82	0.00	11.50	0.09	0.01	0.50
I_2	0.63	59.00	30.90	19.19	0.00	17.12	0.17	0.00	0.00
I_3	0.44	46.67	68.08	22.81	0.00	46.13	0.07	0.00	0.00
I_4	0.55	54.50	18.79	16.28	0.00	10.37	0.07	0.01	0.17
I_5	0.73	24.00	14.72	1.46	0.00	0.09	0.11	0.00	0.00
I_6	1.28	70.50	26.81	27.92	0.00	15.95	0.51	0.01	0.50
I_7	1.03	70.00	43.71	27.97	0.00	28.69	0.20	0.02	0.82
I_8	1.53	77.00	89.22	28.71	0.00	53.61	0.74	0.04	0.82
I_9	1.24	42.33	36.05	30.69	0.00	23.92	0.64	0.00	0.17
I_{10}	0.88	8.00	25.76	4.66	0.00	3.11	0.02	0.00	0.00
I_{11}	0.59	49.50	5.27	4.17	0.00	0.00	0.43	0.01	0.30
I_{12}	0.35	110.00	7.50	4.80	0.00	0.00	0.23	0.04	0.42
I_{13}	0.43	105.00	14.45	4.67	0.00	0.00	0.28	0.11	0.63
I_{14}	0.42	113.00	5.51	5.14	0.00	0.00	0.31	0.03	0.25
I_{15}	0.34	15.00	4.83	6.85	0.00	0.00	0.00	0.00	0.00
I_{16}	0.79	57.50	6.56	4.20	0.00	0.00	0.56	0.00	0.12
I_{17}	0.80	126.00	10.03	4.75	0.00	0.00	0.67	0.02	0.21
I_{18}	1.18	47.50	20.94	5.94	0.00	0.00	0.81	0.02	0.21
I_{19}	0.99	48.00	8.04	4.75	0.00	0.00	0.48	0.00	0.00
I_{20}	0.46	22.00	6.13	5.22	0.00	0.00	0.03	0.00	0.00
Average	**0.76**	**61.03**	**23.07**	**12.45**	**0.00**	**10.52**	**0.32**	**0.02**	**0.26**

4 Conclusion

In this work, a planning-by-range (PBR) approach is proposed for the production and distribution of a flagship product of a real company. PBR is specifically well adapted for controlling the bullwhip effect. In this context, a lexicographic optimization problem is designed for minimizing: (1) shortage, (2) out-of-range production, (3) undesired inventory. PBR was tested for 20 realistic instances and very favorably compared with the well-known *pull* and *push* approaches. Future works include the investigation of more complex situations (e.g., multiple DCs, variable lead-times, promotional weeks).

References

Axsäter, S., 2015, "Inventory Control", *Springer*.

Darvish M. and L.C. Coelho, 2017, "Sequential versus Integrated Optimization: Lot Sizing, Inventory Control and Distribution", *European Journal of Operational Research*, forthcoming.

Geary S., S.M. Disney and D.R. Towill, 2006, "On bullwhip in supply chains - Historical review, present practice and expected future impact", *International Journal of Production Economics*, Vol. 101, pp. 2-18.

Thevenin S., N. Zufferey and R. Glardon, 2017, "Model and Metaheuristics for a Scheduling Problem Integrating Procurement, Sale and Distribution", *Annals of Operations Research*, Vol. 259(1), pp. 437-460.

Wang X., and S.M. Disney, 2016, "The bullwhip effect: Progress, trends and directions", *European Journal of Operational Research*, Vol. 250, pp. 691-701.

Modeling Non-preemptive Parallel Scheduling Problem with Precedence Constraints

Tianyu Wang[1] and Odile Bellenguez-Morineau[1]

LS2N(Le Laboratoire des Sciences du Numerique de Nantes), France
tianyu.wang@ls2n.fr; odile.bellenguez@imt-atlantique.fr

Keywords: parallel scheduling, modeling, precedence constraints.

1 Introduction

The scheduling problem we study deals with n jobs to be processed on m machines while satisfying the precedence constraints. Considering the makespan, this problem is $\mathcal{NP}$-hard even with no precedence constraints and two machines(Lenstra et al. 1977). To the best of our knowledge, there is no exact method for this problem even when m is fixed.

In this paper, we adapt some models to parallel scheduling problems and propose a new one. Then, we compare their performance by testing them on benchmarks with precedence constraints from PSPLIB.

2 Mathematical Models

In this section, we present different models. Each model uses the variables C_j and S_j as starting time and completion time of j. The objective is $C_{\max}$, the following constraints hold for all models, and they are omitted hereafter:

$$\begin{aligned} C_j &= S_j + p_j, \quad \forall j \in \mathcal{J} \\ C_{max} &\geq C_j, \quad \forall j \in \mathcal{J} \\ S_j &\geq C_i, \quad \forall i, j \in \mathcal{J} \text{ and } i \prec j \end{aligned}$$

where p_j is the processing time of job j and $i \prec j$ means i precedes j. $\mathcal{J}$ and $\mathcal{M}$ are set of all jobs and machines. In the following sections, $i, j \in \mathcal{J}$ and $k \in \mathcal{M}$. M represents a large number, which can be defined as $\sum p_j$.

2.1 Relative-Order-Indexed Model1 (ROIM1)

This model uses binaries $y_{i,j}^k$ and $z_{i,j}^k$ as decision variables. $y_{i,j}^k = 1$ if i is executed immediately before j on k; $z_{i,j}^k = 1$ if i is executed before j on k. Different formulations of this model can be seen in Blazewicz et al. (1991) and Unlu and Mason (2010) for problems without precedence constraints. We introduce S_j and C_j for precedence constraints by adding (1b), and we propose the following formulation:

$$y_{i,j}^k \leq z_{i,j}^k, \quad \forall i, j \in \mathcal{J}, k \in \mathcal{M} \tag{1a}$$

$$M(1 - z_{i,j}^k) + S_j \geq C_i, \quad \forall i, j \in \mathcal{J}, k \in \mathcal{M} \tag{1b}$$

$$M(1 - z_{i,j}^k) \geq \sum_{k' \neq k} \sum_{q} (z_{i,j}^{k'} + z_{q,j}^{k'} + z_{q,i}^{k'} + z_{j,q}^{k'} + z_{i,q}^{k'}), \quad \forall i, j \in \mathcal{J}, k \in \mathcal{M} \tag{1c}$$

$$\sum_k (y_{i,j}^k + y_{j,i}^k) \leq 1, \quad \forall i, j \in \mathcal{J} \tag{1d}$$

$$\sum_k \sum_i y_{i,j}^k = 1, \quad \forall j \in \mathcal{J} \tag{1e}$$

$$\sum_k \sum_j y_{i,j}^k = 1, \quad \forall i \in \mathcal{J} \tag{1f}$$

(1a) ensures that $z_{i,j}^k = 1$ if $y_{i,j}^k = 1$. (1c), (1d), (1e) and (1f) force each job has exactly one predecessor and one successor. Notice that in (1e) and (1f), each job has to be executed before(after) some other job. In practice, some dummy jobs are created to represent jobs after(before) the last(first) executed jobs on each machine.

In fact, (1e) and (1f) convey the same meaning: if every job follows another(except the first one), then every job has a follower(except the last one). If we remove one of (1e) and (1f), the model still works well, but (16% in our experiment) slower. We call this a *redundant constraint*.

2.2 Relative-Order-Indexed Model2 (ROIM2)

This model uses binaries $z_{i,j}^k$ and x_i^k as decision variable. $x_j^k = 1$ if j is on k. To associate these two variables, a non-linear constraint, $x_i^k x_j^k = z_{i,j}^k + z_{j,i}^k, \quad \forall i, j, k$, is given in Low et al. (2006) and Gao et al. (2006) A linear version in Özgüven et al. (2010) introduces new integer variables.

We proposed a new formulation which is superior to the others both theoretically and in our practice as well:

$$\sum_k x_j^k = 1, \quad \forall j \in \mathcal{J} \tag{2a}$$

$$C_i \leq S_j + M(1 - z_{i,j}^k), \quad \forall i, j \in \mathcal{J}, k \in \mathcal{M} \tag{2b}$$

$$M x_i^k \geq \sum_j (z_{i,j}^k + z_{j,i}^k), \quad \forall i \in \mathcal{J}, k \in \mathcal{M} \tag{2c}$$

$$\sum_j (z_{i,j}^k + z_{j,i}^k) + M(1 - x_i^k) \geq 1, \quad \forall i \in \mathcal{J}, k \in \mathcal{M} \tag{2d}$$

$$M(z_{i,j}^k + z_{j,i}^k) \geq x_i^k + x_j^k - 1, \quad \forall i, j \in \mathcal{J}, k \in \mathcal{M} \tag{2e}$$

(2a) forces each job to be executed once. (2b) ensures that $C_i \leq S_j$ if $z_{i,j} = 1$. (2c) and (2d) consider two cases: $x_i^k = 0$ then $\forall j \in \mathcal{J}$, $z_{i,j}^k = 0$; $x_i^k = 1$ then $\exists j$, $z_{i,j}^k \vee z_{j,i}^k = 1$. (2e) works when both i, j are on k, and forces one to precede the other.

2.3 Absolute-Order-Indexed Model(AOIM)

This model uses $\beta_{k,j}^l$, which equals 1 if j is the l^{th} job on k, as a principal variable. It was originally designed for parallel scheduling problem without precedence constraints by Blazewicz et al. (1991). To add precedence constraints, Demir and İşleyen (2013) introduces T_k^l to the model, which is the starting time of the l^{th} job of k.

Here, we propose a similar formulation, which uses fewer variables:

$$T_k^{l+1} - T_k^l \geq p_j \beta_{k,j}^l, \quad \forall l \leq n, j \in \mathcal{J}, k \in \mathcal{M} \tag{3a}$$

$$T_k^l + M(1 - \beta_{k,j}^l) \geq S_j, \quad \forall l \leq n, j \in \mathcal{J}, k \in \mathcal{M} \tag{3b}$$

$$T_k^l \leq M(1 - \beta_{k,j}^l) + S_j, \quad \forall l \leq n, j \in \mathcal{J}, k \in \mathcal{M} \tag{3c}$$

$$\sum_j \beta_{k,j}^l \leq 1, \quad \forall l \leq n, k \in \mathcal{M} \tag{3d}$$

$$\sum_k \sum_l \beta_{k,j}^l = 1, \quad \forall j \in \mathcal{J} \tag{3e}$$

(3e) forces each job to be executed once. (3d) ensures that only one job can be executed as the l^{th} job on k. (3c), (3b) and (3a) works when $\beta_{k,j}^{l} = 1$, they guarantee $T_k^l = S_j$ and $T_k^{l+1} - T_k^l = p_j$.

2.4 Compact Model(CM)

We propose a new order-indexed model here. It uses $\delta_{i,j}$, which equals 1 if i is executed before j on the same machine, and x_j^k as decision variables.

$$C_i - S_j \leq M(1 - \delta_{i,j}), \quad \forall i, j \in \mathcal{J} \tag{4a}$$

$$M(2 - x_i^k - x_j^k) + \delta_{i,j} + \delta_{j,i} \geq 1, \quad \forall i, j \in \mathcal{J}, k \in \mathcal{M} \tag{4b}$$

$$M(2 - x_i^{k_1} - x_j^{k_2}) \geq \delta_{i,j} + \delta_{j,i}, \quad \forall i, j \in \mathcal{J}, k_1, k_2 \in \mathcal{M} \text{ and } k_1 \neq k_2 \tag{4c}$$

$$\sum_k x_j^k = 1, \quad \forall j \in \mathcal{J} \tag{4d}$$

(4a) connects $\delta_{i,j}$, C_i and S_j(if $\delta_{i,j} = 1$ then $C_i \leq S_j$). When both i, j are on k, (4b) forces one precedes the other. (4d) make each job be executed once. (4c) sets $\delta_{i,j}$ and $\delta_{j,i}$ as 0 when i, j are on different machines.

Notice that when minimizing $C_{\max}$, (4c) is unnecessary because when i, j are on different machines, $\delta_{i,j} = 1$ or $\delta_{j,i} = 1$ can not reduce $C_{\max}$. However, it helps to give $\delta_{i,j}$ a comprehensible meaning($\delta_{i,j} = 1$ if and only if i precedes j on the same machine) and has a positive impact on the model (which is improved by 19% according to our tests).

3 Test Result and Analysis

We tested the models with benchmarks we built from precedence constraints in PSPLIB. The platform we used is: IBM ILOG CPLEX Optimization Studio V12.6.0 on Intel Core i7-4600U @2.10GHz. We compared their average time consumed to solve instances with different scales of jobs on $m = 4$ machines in the following table:

Table 1. Models' performance, where BV/IV/C means number of binary variables/integer variables/constraints; TC means average time consumed to solve the instances; '-' means the model did not solve any instance on this scale within 6000sec

Model	BV	IV	C	TC	TC	TC
	$n = 15$				$n = 30$	$n = 60$
CM	270	31	1125	0.92	1.22	13.32
ROIM2	900	31	1875	1.77	6.41	97.23
AOIM	900	91	2775	3.19	14.66	–
ROIM1	1920	31	2828	15.41	79.23	–
TIM	4920	31	5338	48.39	–	–

As can be seen, CM stays ahead of the others and requires less space. The different speed of models results mostly from the different decision variables used.

Inspired by the Time-Indexed Model (TIM) by Thomalla (2001), we also formulated a version of TIM for parallel scheduling problem. It uses binary variable $x_{t,j}^k$, which is 1 if j starts on k at t. It was compared with other models, however, not developed in the previous section due to its poor performance. It requires an estimation of an upper bound of C_{max}. It is set as $\sum p_j$ for the worst case(single machine) in practice, which leads to

large amounts of variables. In fact, the number of variables could be extremely large if p_j is not an integer. However, TIM is still the slowest even for instances of unit-processing-time jobs. Both ROIM2 and ROIM2 use $O(n^2m)$ binary variables: ROIM1 uses two 3-dimension variables, while ROIM2 and AOIM use only one. AOIM is the only one who requires extra integer variable. The fastest model CM uses only 2-dimension binary variables and requires fewest variables, which may be the principal advantage of CM.

We tested also instances with different number of machines. When judging the models, comparison results are similar as Table 1. Additionally, we find that when m is set as 4, the models took longest time for solving.

4 Conclusion and Perspective

In this paper, we adapted models to the parallel scheduling problem with precedence constraints and proposed a new one which outperforms the others by our test.

In addition of redundant constraints, we tested and find that sometimes the redundant variables, such as C_j which could totally be replaced by $S_j + p_j$, ameliorates the models. How an redundant constraint or variable affects the model is worthy of being further discussed.

Besides, the solving time does not depend merely on instance's scale, but also on number of machines, the processing time of jobs, and the shape of precedence constraints. For example, a subproblem of scheduling equal-processing-jobs with in-tree precedence graphs can be finished in polynomial time Hu (1961). The model is helpful to study experimentally how they impact the solving time. Our next work direction follows this approach.

Bibliography

Blazewicz, J., Dror, M. and Weglarz, J.: 1991, Mathematical programming formulations for machine scheduling: A survey, *European Journal of Operational Research* **51**(3), 283–300.

Demir, Y. and İşleyen, S. K.: 2013, Evaluation of mathematical models for flexible job-shop scheduling problems, *Applied Mathematical Modelling* **37**(3), 977–988.

Gao, J., Gen, M. and Sun, L.: 2006, Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm, *Journal of Intelligent Manufacturing* **17**(4), 493–507.

Hu, T. C.: 1961, Parallel sequencing and assembly line problems, *Operations research* **9**(6), 841–848.

Lenstra, J. K., Kan, A. R. and Brucker, P.: 1977, Complexity of machine scheduling problems, *Annals of discrete mathematics* **1**, 343–362.

Low, C., Yip, Y. and Wu, T.-H.: 2006, Modelling and heuristics of fms scheduling with multiple objectives, *Computers & operations research* **33**(3), 674–694.

Özgüven, C., Özbakır, L. and Yavuz, Y.: 2010, Mathematical models for job-shop scheduling problems with routing and process plan flexibility, *Applied Mathematical Modelling* **34**(6), 1539–1548.

Thomalla, C. S.: 2001, Job shop scheduling with alternative process plans, *International Journal of Production Economics* **74**(1), 125–134.

Unlu, Y. and Mason, S. J.: 2010, Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems, *Computers & Industrial Engineering* **58**(4), 785–800.

A Branch-and-Bound Procedure for the Resource-Constrained Project Scheduling Problem with Partially Renewable Resources and Time Windows

Kai Watermeyer and Jürgen Zimmermann

Clausthal University of Technology, Germany
kai.watermeyer, juergen.zimmermann@tu-clausthal.de

Keywords: Project scheduling, Partially renewable resources, Branch-and-bound.

1 Introduction

In this paper we present a branch-and-bound procedure for the resource-constrained project scheduling problem with partially renewable resources and time windows (RCPSP/max,π). For the first time the concept of partially renewable resources is embedded in the context of projects with general temporal constraints.

Partially renewable resources were introduced by Böttcher *et al.* (1996) and have just been considered for projects restricted to precedence constraints (RCPSP/π). For each partially renewable resource a resource capacity for a subset of time periods of the planning horizon is given. In this way timetabling and complex labor regulation problems can be modeled as project scheduling problems (Álvarez-Valdés *et al.* 2006). For the RCPSP/π a branch-and-bound procedure has been developed in Böttcher *et al.* (1999) and also approximation procedures in Schirmer (1999) and Álvarez-Valdés *et al.* (2006, 2008) have been investigated.

In Section 2 the RCPSP/max,π is described formally. Section 3 presents the enumeration scheme the developed branch-and-bound procedure is based on and in Section 4 the branch-and-bound procedure is outlined. Finally, in Section 5 the results of a computational study are presented where we compared the performance of our branch-and-bound procedure with the outcome of the mixed-integer linear programming solver *IBM CPLEX*.

2 Problem description

The resource-constrained project scheduling problem with time windows and partially renewable resources (RCPSP/max,π) can be modeled as an activity-on-node network where the nodes correspond to all activities of the project $V = \{0, 1, \ldots, n+1\}$ with n real activities and the fictitious activities 0 and $n+1$ representing the start and end of the project, respectively. Each activity $i \in V$ is assigned a non-interuptible processing time $p_i \in \mathbb{Z}_{\geq 0}$ and a resource demand $r_{ik}^d \in \mathbb{Z}_{\geq 0}$ for each partially renewable resource $k \in \mathcal{R}$ considered in the project. The arcs of the network given by the set $E \subseteq V \times V$ represent the temporal constraints between the activities where the arc weight $\delta_{ij} \in \mathbb{Z}$ for arc $\langle i, j \rangle \in E$ implicates a minimal time lag between the start times of activity i and activity j which has to be fulfilled. For each resource $k \in \mathcal{R}$ a resource capacity R_k and a subset of time periods of the whole planning horizon $\Pi_k \subseteq \{1, 2, \ldots, \bar{d}\}$ is given with $\bar{d}$ as a given maximal project duration. It is assumed that an activity i just consumes a resource k with r_{ik}^d units in each time period of Π_k activity i is in execution where the start times of all activities are restricted to integer values. The number of time periods an activity i with start time point S_i is in execution during the defined time periods of resource k is given by the so

called resource usage $r^u_{ik}(S_i) := |\{S_i+1, S_i+2, \ldots, S_i+p_i\} \cap \Pi_k|$ so that the corresponding resource consumption can be determined by $r^c_{ik}(S_i) := r^u_{ik}(S_i) \cdot r^d_{ik}$.

The objective of the problem is to assign each activity $i \in V$ a start time S_i so that all time and resource constraints are fulfilled and the project duration is minimized. In the following a sequence of start times of all activities $S = (S_0, S_1, \ldots, S_{n+1})$ with $S_0 := 0$ is called a schedule where it is said to be time-feasible, resource-feasible or feasible if it fulfills all temporal constraints, all resource constraints or all constraints, respectively. The problem RCPSP/max,π can be stated as follows:

$$
\begin{array}{lll}
\text{Minimize} & f(S) = S_{n+1} & \\
\text{subject to} & S_j - S_i \geq \delta_{ij} & (\langle i,j \rangle \in E) \\
& S_0 = 0 & \\
& \sum\nolimits_{i \in V} r^c_{ik}(S_i) \leq R_k & (k \in \mathcal{R}) \\
& S_i \in \mathbb{Z}_{\geq 0} & (i \in V)
\end{array}
$$

3 Enumeration scheme

The enumeration scheme of the developed branch-and-bound procedure is based on a stepwise restriction of the allowed resource usages of the activities of the project. The procedure starts with the determination of the earliest possible start times ES_i of all activities $i \in V$ for the resource-relaxation of RCPSP/max,π. If this schedule is resource-feasible the optimal solution is already found. Otherwise there is at least one resource k whose resource capacity R_k is exceeded so that the resource usage of at least one activity consuming resource k have to be decreased to get a feasible schedule. The enumeration scheme makes use of the start time dependency of the resource usage $r^u_{ik}(\cdot)$ of all activities $i \in V$ for resource k. It is easy to see that for a feasible schedule S the resource usage of at least one activity $i \in V$ has to be lower than the resource usage of the resource-infeasible schedule ES, i.e., $r^u_{ik}(S_i) \leq r^u_{ik}(ES_i) - 1$. So we preserve all feasible schedules by branching the resource-relaxation in subproblems where each subproblem restricts the resource usage of an activity i with $r^u_{ik}(ES_i) > 0$ to $r^u_{ik}(ES_i) - 1$. The resource usage restriction of activity i for resource k is achieved by permitting only start time points t with $r^u_{ik}(t) \leq r^u_{ik}(ES_i) - 1$. In order to save these permitted start time points for all activities in the enumeration process a so called start time restriction W_i for each activity is introduced. This is set to $W_i := \{ES_i, ES_i+1, \ldots, LS_i\}$ for each activity at the beginning of the process with LS_i as the latest possible start time point of activity i for the resource-relaxation of RCPSP/max,π. For the subproblem in which the resource usage of activity i is restricted the start time restriction is set to $W_i := W_i \cap \{t \in \{0, 1, \ldots, \bar{d}\} \mid r^u_{ik}(t) \leq r^u_{ik}(ES_i) - 1\}$ so that the resource usage of activity i of resource k is lower or equal to $r^u_{ik}(ES_i) - 1$ if activity i starts at time point $t \in W_i$. For each achieved subproblem the earliest possible start time points of all activities have to be determined so that all temporal constraints of the RCPSP/max,π are fulfilled and also $S_i \in W_i$ for each $i \in V$ is satisfied. This can be done by a modified label correcting algorithm which determines the earliest possible start time points denoted by $ES_i(W)$ of all activities $i \in V$ with a worst-case time complexity of $\mathcal{O}(|V||E|\,(1 + \mathcal{B}))$ with $\mathcal{B}$ as the number of interruptions of consecutive time points in W_i over all activities $i \in V$. If all determined and all following subproblems are tackled like described for the resource-relaxation of the RCPSP/max,π it can be shown that the procedure determines after a finite number of iterations an optimal schedule or shows the infeasibility if there is no optimal schedule.

4 Branch-and-bound procedure

The enumeration scheme describes the decomposition of the currently considered part of the solution space in one or more components for a chosen conflict resource, i.e., a resource whose capacity is exceeded. The strategy to decide which of the conflict resources is used next to decompose the solution space is called branching strategy. The way to determine which node in the enumeration tree is considered next is called search strategy. For both strategies different approaches have been investigated on benchmark test sets.

Before the branch-and-bound procedure is started a preprocessing phase is conducted. In this step start time points of activities are eliminated for which it can be shown that they cannot be part of any of the feasible schedules. For this a start time point of an activity is eliminated if the resource consumption of the activity started at this time point and the sum of the minimal resource consumptions of all other activities over all start time points satisfying the temporal constraints to the considered activity exceeds the capacity of at least one resource.

Furthermore, for each node in the search tree two lower bounds for the project duration are determined to be able to prune this node and the following parts of the enumeration tree if one of these lower bounds is greater or equal to the project duration of the best found solution so far. The first lower bound is given by the minimal possible project duration taking the start time restrictions of all activities into consideration. The second lower bound is equal to the minimal project duration for which at least one resource-feasible schedule in the currently considered part of the search tree exists so that all temporal constraints to the start and the end of the project are satisfied.

To reduce the search tree even further a dominance rule is used in addition. For this an unexplored node is called dominated by another node if the restrictions of the resource usages over all activities and resources are lower or equal to the resource usage restrictions of the other node. In this case the unexplored node is pruned from the search tree.

5 Performance analysis

In order to evaluate the performance of our branch-and-bound (*BnB*) procedure we have compared the obtained results with the outcome of the mixed-integer linear programming (MILP) solver *IBM CPLEX* in the latest version 12.7.1. The computational study was conducted on a PC with Intel Core i7-3820 CPU with 3.6 GHz and 32 GB RAM under Windows 7. The *BnB* procedure was coded in C++ and compiled with the 64-bit Visual Studio 2015 C++-Compiler. The instance sets we have used are adaptions of the well-known benchmark test set UBO (Schwindt 1998) where we replaced the included renewable resources by 30 partially renewable resources using the generation procedure described in Schirmer (1999). Note that there is no instance with a project network containing a cycle of positive length. In this manner we have generated 729 instances with 10, 20, 50, 100, and 200 activities, respectively. For the computational study we set the runtime limit to 60 seconds and used an adaption of the MILP given in Böttcher *et al.* (1999) for the *IBM CPLEX* solver. The mathematical program is given as follows:

$$
\begin{aligned}
\text{Minimize} \quad & \sum\nolimits_{t\in\mathcal{T}_{n+1}} t\cdot x_{n+1,t} && \\
\text{subject to} \quad & \sum\nolimits_{t\in\mathcal{T}_i} x_{it} = 1 && (i\in V)\\
& \sum\nolimits_{t\in\mathcal{T}_j} t\cdot x_{jt} \geq \sum\nolimits_{t\in\mathcal{T}_i} t\cdot x_{it} + \delta_{ij} && (\langle i,j\rangle\in E)\\
& \sum\nolimits_{i\in V} r^d_{ik} \sum\nolimits_{v\in\Pi_k} \sum\nolimits_{\tau\in Q_{i,(v-1)}\cap\mathcal{T}_i} x_{i\tau} \leq R_k && (k\in\mathcal{R})\\
& x_{it}\in\{0,1\} && (i\in V, t\in\mathcal{T}_i)
\end{aligned}
$$

The MILP is a time-indexed formulation with binary variables x_{it} for each activity $i \in V$ and each start time point t of the activity in the set $\mathcal{T}_i := \{ES_i, ES_i + 1, \ldots, LS_i\}$. The binary variable x_{it} takes the value 1 exactly if activity i starts at time point t, i.e., $t = S_i$. The set Q_{it} contains all time points activity i could be started so that activity i would be in execution at time point t, i.e., $Q_{it} := \{t - p_i + 1, \ldots, t\}$.

Table 1. Results of the computational study

	$UBO10^{\pi}$		$UBO20^{\pi}$		$UBO50^{\pi}$		$UBO100^{\pi}$		$UBO200^{\pi}$	
	BnB	*CPLEX*	*BnB*	*CPLEX*	*BnB*	*CPLEX*	*BnB*	*CPLEX*	*BnB*	*CPLEX*
#opt	511	565	288	391	116	113	58	34	53	5
#feas	55	1	259	160	352	65	333	6	312	1
#infeas	129	132	30	57	0	19	0	3	0	0
#noSol	3	0	34	3	59	330	93	441	101	460
#trivial	31	31	118	118	202	202	245	245	263	263
$\varnothing^{\text{CPU}}_{\text{opt}}$	1.60	0.56	2.51	6.78	1.72	4.89	1.76	15.41	6.21	40.51
$\varnothing^{\text{CPU}}_{\text{infeas}}$	0.44	0.03	2.31	0.45	–	2.44	–	14.90	–	–

The results of the computational study are given in Tab. 1 where for each test set, for instance $UBO10^{\pi}$ with 10 activities, the results of the *BnB* procedure and the *IBM CPLEX* solver (*CPLEX*) are listed. The term #opt stands for the number of optimal solved instances for which the schedule ES is not optimal, term #feas describes the number of instances the solution procedure was able to find a solution which could not be proofed to be optimal and #infeas gives the number of instances the procedure could proof the infeasibility for. In the following two rows, the number of instances the solution procedure was not able to find any feasible solution (#noSol) and the number of so called trivial instances for which the schedule ES is already optimal (#trivial) are given. Finally, the last rows show the average used CPU time in seconds over all optimal solved ($\varnothing^{\text{CPU}}_{\text{opt}}$) and over all instances for which the infeasibility could be proofed ($\varnothing^{\text{CPU}}_{\text{infeas}}$).

In Tab. 1 it can be seen that the *IBM CPLEX* solver dominates the developed BnB procedure for the instance sets $UBO10^{\pi}$ and $UBO20^{\pi}$. In contrast, the BnB procedure is able to obtain optimal and feasible solutions for more instances of the test sets $UBO50^{\pi}$, $UBO100^{\pi}$ and $UBO200^{\pi}$.

References

Álvarez-Valdés R., E. Crespo, J.M. Tamarit and F. Villa, 2006, "A scatter search algorithm for project scheduling under partially renewable resources", *Journal of Heuristics*, Vol. 12, pp. 95-113.

Álvarez-Valdés R., E. Crespo, J.M. Tamarit and F. Villa, 2008, "GRASP and path relinking for project scheduling under partially renewable resources", *European Journal of Operational Research*, Vol. 189, pp. 1153-1170.

Böttcher J., A. Drexl, R. Kolisch, F. Salewski, 1996, "Project scheduling under partially renewable resource constraints", Technical Report, *Manuskripte aus den Instituten für Betriebswirtschaftslehre 398*, University of Kiel.

Böttcher J., A. Drexl, R. Kolisch and F. Salewski, 1999, "Project scheduling under partially renewable resource constraints", *Management Science*, Vol. 45, pp. 544-559.

Schirmer A., 1999, "Project scheduling with scarce resources: models, methods and applications", Dr. Kovač, Hamburg.

Schwindt C., "Generation of resource-constrained project scheduling problems subject to temporal constraints", *Technical Report WIOR-543*, University of Karlsruhe.

Fixed interval multiagent scheduling problem with rejected costs

B. Zahout, A. Soukhal and P. Martineau

Université de Tours, France
LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002
boukhalfa.zahout,ameur.soukhal,patrick.martineau@univ-tours.fr

Keywords: Multiagent scheduling, single machine, Pareto optimization, MILP, linear combination of criteria, ε-constraint, heuristics.

1 Introduction

"*Multiagent scheduling problems*" consider that several agents are competing for the use of the same resources. Each agent is responsible for a set of jobs, and aims at minimizing one objective function that depends only on the completion times of its associated subset of jobs. When the subsets of jobs are disjoint, the problems are called *Competing scheduling problems* (Agnetis, Mirchandani, Pacciarelli and Pacifici 2004). Such problems corresponds to some real world situations as introduced in (Agnetis, Billaut, Gawiejnowicz, Pacciarelli and Soukhal 2014).

In this study, we consider two agents A and B. Agent A (resp. B) is associated with the set of n_A (resp. n_B) jobs, denoted by $\mathcal{N}^{\mathcal{A}} = \{J_1, J_2, ..., J_{n_A}\}$ (resp. $\mathcal{N}^{\mathcal{B}} = \{J_{n_A+1}, J_{n_A+2}, ..., J_n\}$), where $n = n_A + n_B$.

The n independent jobs should be scheduled without preemption on a single machine. Additional renewable resources are however necessary to process each job. Several types of such resources are needed, denoted $R_j, j = 1 \dots k$. Hence, at execution time of job i, r_{ij} units of available resource are required. For each job i, the start time s_i and its finished time f_i $(i = 1, \dots, n)$ are fixed where its processing time $p_i = f_i - s_i$. w_i is the weight of job i. Dealing with each type of resources, the machine can process more than one job at a time provided the resource consumption does not exceed a given value R_j $(j = 1 \dots k)$. This machine is continuously available during time interval $[0, \infty)$. All data are assumed positive integers. The processing times of jobs is formatted in slotted windows. The total time period $[0, T]$ is partitioned into equal length slots (l_0) with $T = max_{i,i=1,\dots,n}(f_i)$. We suppose that: $s_i < f_i$ and $r_{i,j} \leq R_j$ for all $i = 1, \dots, n$ and $j = 1 \dots k$. The objective of each agent is to minimize its total rejected costs. Let x_i be the binary decision variable where $x_i = 1$ if job i is rejected; 0 otherwise. We denote the rejected cost of agents A and B by $Z^A = \sum_{i=1}^{n_A} w_i x_i$ and $Z^B = \sum_{i=n_A+1}^{n} w_i x_i$, respectively. In this study, both linear combination of criteria approach and ε-constraint approach are used to determine one Pareto optimal solution.

According to the three-field notation of multiagent scheduling problems introduced in (Agnetis, Billaut, Gawiejnowicz, Pacciarelli and Soukhal 2014), problems we address are denoted by: $1|CO|F_\ell(Z^A, Z^B)$ with $F_\ell = \lambda Z^A + (1 - \lambda)Z^B$; And $1|CO|\varepsilon(Z^B/Z^A)$. These problems are all NP-hard even if only one agent is considered (monocriterian case) (Zahout, Soukhal and Martineau 2017).

The addressed problem can be met in a data center where the objective is to optimize the objective function of each user (agent). Virtual Machines VMs (jobs) submitted by the users should be executed on the same cluster (only one cluster is considered in this study). For example, this cluster owns three limited types of renewable resources CPU, MEMORY and STORAGE with capacities equal to Q_1 CPU, a certain quantity of memory Q_2 and

a certain storage capacity Q_3. In this case, to execute VM_i, a number of virtual CPUs r_{i1}, virtual memory r_{i2} and hard drives r_{i3} are needed. The monocriterian case has been addressed in (Angelelli, Bianchessi and Filippi 2014) where the authors consider only one additional resource (memory) and develop methods to determine one feasible solution.

In the context of grid computing, (Cordeiro, Dutot, Mounié and Trystram 2011) considers organizations that share clusters to distribute peak workloads among all the participants. Each cluster is associated with one agent and the global objective function is to minimize the makespan. The authors propose a 2-approximation algorithm for finding collaborative solutions.

2 Exact methods

2.1 Linear combination of criteria

Consider the classical scheduling problem $1||Z$ where $Z = \sum_{1 \leq i \leq n} w'_i x_i$. The two following scheduling problems are equivalent: $1|CO|F_\ell(Z^A, Z^B)$ and $1||Z$. In fact, we set $w'_i = \lambda w_i$ for all $J_i \in \mathcal{N}^{\mathcal{A}}$ and $w'_i = (1-\lambda)w_i$ for all $J_i \in \mathcal{N}^{\mathcal{B}}$. Hence, we propose the following time indexed integer linear programming formulation (ILP) where: x_i is a binary variable equal to 1 if job J_i is rejected, 0 otherwise; And y_{it} is a binary variable equal to 1 if job J_i is executed at time t, and 0 otherwise.

$$\text{Minimize:} \quad \sum_{i \in \mathcal{N}} w'_i x_i$$

$$\text{subject to:} \quad \sum_{t=s_i}^{f_i-1} y_{it} = (f_i - s_i) * (1 - x_i) \qquad \forall i \in \mathcal{N} \qquad (1)$$

$$\sum_{i \in \mathcal{N}} y_{it} * r_{ij} \leq R_j \qquad \forall j \in \mathcal{R}\,; \forall t \in [0,T] \qquad (2)$$

$$x_i \in \{0,1\}, \quad y_{it} \in \{0,1\}\,, \forall i \in \mathcal{N}\,, \forall t \in [0,T].$$

The constraints (1) ensure that if job J_i is not rejected then it is scheduled during its time interval. The constraints (2) ensure that no more than R_j quantities of the required resources are consumed at time t.

2.2 ε-constraint approach

To determine a non-dominated solution, we propose to use previous ILP where the objective function is now: Minimize $Z^B = \sum_{i \in \mathcal{N}^{\mathcal{B}}} w_i x_i$. Then to the two inequalities (1) and (2), we add following new constraint: $Z^A \leq Q_A$. It means that the total rejected cost defined by non-scheduled jobs of agent A does not exceed a given value Q_A.

This ILP is also used to compute the optimal Pareto front.

3 Greedy heuristics

The scheduling problems under this study have many applications, but they were motivated by research into on-line system and integrated-services networks, where the number of jobs to be processed can be extremely large, so low computational running time is essential. Hence, the resolution methods must have low complexity, not just polynomial complexity. In tis section, we presente low-complexity ($O(nlogn)$) greedy algorithms. Roughly, this algorithm works as follow. If ε-constraint approach is used, jobs of each agent are sorted according to a given priority rule. At first, we try to schedule jobs of agent A with respect of its objective (i.e. $Z^A \leq Q_A$). Jobs are taken according to their priority order. Job is

rejected if it can not be scheduled. Then, within the obtained solution, we try to schedule jobs of agent B minimizing its objective function Z^B.

When linear combination of criteria approach is used, whole jobs are sorted according to a given priority rule. Then, we solve the scheduling problem $1||Z$ where $Z = \sum_{1 \leq i \leq n} w'_i x_i$. It means that jobs with higher priority are scheduled first, if possible.

3.1 Priority rules

1. **Weighted Shortest Processing Time First ($WSPT$)**: Jobs are sorted in non-decreasing order of $(f_i - s_i)/w_i$, in case of ties, job with the smallest finished time come first, otherwise lexicographical order is considered. This $WSPT$ rule allows resources to be released as soon as possible.
2. **Weighted Capacity-Makespan (WCM)**: Jobs are sorted in non-decreasing order of their occupied space divided by w_i given by the following formula: $(\sum_{j \in R} r_{ij} * (f_i - s_i))/w_i$, in case of ties, the job with the smallest finished time come first, otherwise lexicographical order is considered. The idea of using WCM rule is to minimize the space occupied by jobs defined by processing time per quantities of consumed resources.

4 Computational experiments

We implemented our algorithms in $C++$ language and executed experiments on a workstation with a 2.8 Ghz Intel Core i7 processor and 8 GB of memory. We used IBM ILOG CPLEX Optimization Studio version 12.6.3 to solve the ILP models.

We assessed the performance of the algorithms on 50 instances, with a number of jobs $n \in \{20, 40, \ldots, 100\}$ where 30% of n are jobs of agent A (10 instances are generated per n). We generated the job-starting times s_i using a discrete uniform distribution between 1mn and 1400mn. Similarly, we generated the job-finishing time of each job J_i has using a discrete uniform distribution between $(s_i + 1)$mn and $(1440 - s_i)$mn. We considered three types of resources. Without lost of generality, we normalize the units of a renewable resource to 1000. Hence, $R_j = 1000, j = 1, 2, 3$. For each job J_i, r_{ij} randomly generated in $[1, 1000]$, $i = 1, \ldots, n$ and $j = 1, 2, 3$.

The experimental results are summarized in Table 1.

Table 1. Computational results for problem $1|CO|\varepsilon(Z^A, Z^B)$ with $n_A = 30\%n$

n	ILP		$WSPT$					WCM				
	CPU_s	$\|S^*\|$	CPU_s	$\|S\|$	GD	$\%S$	$\%wS$	CPU_s	$\|S\|$	GD	$\%S$	$\%wS$
20	0,8	4,1	0	3,5	1,16	47	40	0	3,6	1,05	47	43
40	3,0	6,5	0	4,0	3,20	34	25	0	4,0	3,20	32	41
60	8,0	10,2	0	5,3	8,86	17	40	0	5,3	8,77	15	49
80	15,6	12,8	0	5,3	13,00	10	12	0	4,9	10,00	14	13
100	35,1	21,9	0	6,6	21,00	2	37	0	6,9	19,00	3	40

The first column in Table 1 shows the size of the instance (number of jobs). We computed the average computation time in seconds required to obtain the Pareto front for each method: ILP model, Weighted Shortest Processing Time First ($WSPT$) and Weighted Capacity-Makespan (WCM). This computation time is denoted by CPU_s. The size of exact (resp. approximate) Pareto front is denoted by $|S^*|$ (resp. $|S|$).

For each instance, we compare the exact front S^* generated by ILP model with the Pareto front S generated by heuristics (WCM and $WSPT$). Different performance measures of heuristics are used and described as follow. Given $S^* = \{a_1, \ldots, a_{|S^*|}\}$ and $S = \{b_1, \ldots, b_{|S|}\}$, we categorize these measures in two classes:

- *Cardinality measure*: we calculate the size of the optimal Pareto front $|S^*|$, and the approximated Pareto front $|S|$. We then combine these metrics to obtain the percentage of strict non-dominated solutions generated by WCM and $WSPT$.

$$\%S = \frac{|S \cap S^*|}{|S|} * 100$$

  and $\%wS$, the percentage of weak non-dominated solutions generated by WCM and $WSPT$.
- *Average minimum Euclidian distance GD*: let d_i be the minimum Euclidian distance between the element $b_i \in S$ and some element of S^*. GD is given by:

$$GD = \frac{1}{|S|}(\sum_{i=1}^{|S|} d_i)$$

The first result concerns the performance of the proposed mathematical model ILP. CPLEX delivers optimal solutions for the 50 instances. The required average computation time per instance is less than 35 seconds (1s for instances of 20 jobs). In addition, we conducted additional tests to analyze the performance of ILP on large size instances (up to 500 jobs). For example, the maximum computation time needed by ILP to calculate the exact Pareto front with instances of 500 jobs is 45 minutes.

According to the data displayed on the Table 1 and according to cardinality measure, the number of Pareto solutions increases with increasing number of jobs. Over all 50 instances, 22% of Pareto solutions generated by heuristic WCM or $WSPT$ are optimal solutions, while 37% of solutions are weakly Pareto solutions generated by WCM when $WSPT$ finds 31%. Dealing with GD measure, the average minimum Euclidean distance given by WCM (resp. $WSPT$) is 14,5 (resp. 17,0).

We observe that when the number of jobs increases, the average percentage of the exact/weakly Pareto solutions generated by heuristics WCM and $WSPT$ decreases. For example, on the 20-job (resp.100-job) instances, 90% (resp. 43%) of the Pareto solutions generated by heuristic WCM are exact or weakly solutions. Therefore, GD increases to an average of 1.05 (resp. 19). We observe almost the same performances with $WSPT$. In fact, dealing with $\%wS$ measure WCM obtains some advantage.

However, these methods are very useful to solve studied problem given its complexity.

References

Agnetis, A., P. Mirchandani, D. Pacciarelli, A. Pacifici, 2004, "Scheduling problems with two competing agents", *Operations Research*, Vol. 52, pp. 229-242.

Agnetis A., J.-C. Billaut, S. Gawiejnowicz, D. Pacciarelli, A. Soukhal, 2014, "Multiagent Scheduling, Models and Algorithms", *Springer-Verlag*, Berlin Heildelberg New York.

Angelelli E., N. Bianchessi, C. Filippi, 2014, "Optimal interval scheduling with a resource constraint", *Computers & Operations Research*, Vol. 51, pp. 268-281.

Cordeiro D., P.-F. Dutot, G. Mounié, D. Trystram, 2011, "Tight Analysis of Relaxed Multi-Organization Scheduling Algorithms", *In Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS), IEEE Computer Society*, Anchorage, AL, USA, pp. 1177-1186.

Zahout E., N. Soukhal, C. Martineau, 2017, "Fixed jobs scheduling on a single machine with renewable resources", *MISTA'2017*, Kuala Lumpur, Malaysia, pp. 1-9.

Integrating case-based analysis and fuzzy programming for decision support in project risk response

Yao Zhang, Fei Zuo, and Xin Guan

Department of Operations and Logistics Management, School of Business Administration, Northeastern University, Shenyang, China
yzhang@mail.neu.edu.cn, zfsy30@163.com, guanxin1016@126.com

Keywords: project risk management, Case-based, risk response action (RRA), fuzzy mathematical programming, decision support system.

1 Introduction

Project performance is constantly affected by risks, and thus the operation of effective project risk management (PRM) is significant for the success of the whole project. In general, PRM consists of three phases: risk identification, risk assessment and risk response. Though the three processes are of equal importance to the success of PRM, risk response is always considered to have direct influence in reducing the risk exposure, which should be conducted right after risks being identified and analysed.

Risk response action (RRA) selection, which plays an important role in project risk management (PRM), has attracted much scholarly attention. Some methods such as the matrix-based method (Datta and Mukherjee 2001, Elkjaer and Felding 1999, Flanagan and Norman 1993, Miller and Lessard 2001, Piney 2002), the trade-off method (Chapman and Ward 2003, Klein 1993, Haimes 2015, Kujawski 2002, Pipattanapiwong and Watanabe 2000), the decision tree method (Dey 2002, 2012, Marmier *et al.* 2013, 2014, Kujawski and Angelis 2010), and the optimization method (Ben-David and Raz 2001, Ben-David *et al.* 2002, Fang *et al.* 2013, Kayis *et al.* 2007, Sherali *et al.* 2008, Zhang and Fan 2014, Zhang 2016) are proposed in last several decades. These methods have made significant contributions to RRA selection from different perspectives. However, there are some limitations in the existing methods. The matrix-based method and the trade-off method only consider two criteria and the characteristics of the RRAs are not considered. In addition, in the two methods, it is not convinced whether the determined RRAs are optimal. Although the RRA selected from the decision tree method is optimal to each risk, it would be a difficult and time-consuming task to construct a decision tree in the situation of complex projects or multiple risks. The optimization method can avoid the above limitations and obtain an optimal set of RRAs. However, in all the above methods, the candidate RRAs are developed based on the evaluation of PMs and experts. It is very likely that some RRAs with better effects may be left out, which may further reduce the overall risk response effect. Therefore, it is crucial to determine proper RRA alternatives in RRA selection. The case-based analysis provides some insights to determine proper RRA alternatives. The core idea of the case-based analysis is to solve current problems by reusing and referring the knowledge, information, and the experiences from historical similar situations. It allows PMs to retrieve similar historical risks and corresponding RRAs from historical cases. Thus, in this study, the authors attempt to propose a decision support method that combines both the case-based analysis and the optimization model in helping PMs choose appropriate RRAs, and managerial suggestion and implication can be drawn.

2 Methodology

The framework for the integrated method in this study is shown in Figure 1.

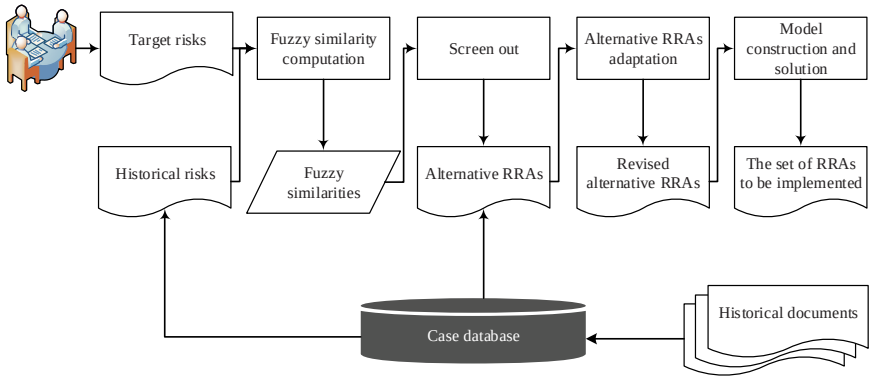


Fig. 1. Framework for the integrated method.

2.1 RRA alternatives formulation

This section presents a case-based method to retrieve historical risks and their corresponding RRAs. To do this, the target risks and representation of historical cases are described firstly. In PRM, project risk can generally be characterized by the product of its probability and impact. In our research, linguistic terms are used to describe risk probability and impact in the process of risk evaluation, and the fuzzy set representation for each linguistic term is used.

In the case database, each historical case includes three types of information: risk information, project information, and RRA information. Take Case 1 (a historical case) for example, R_1^1 denotes the 1-st historical risk in the 1-st historical project and A_1^1 denotes its corresponding RRA. The probability and impact of R_1^1 are evaluated as "unlikely" and "low" by experts, respectively. Similar to the approach of describing the risk probability and impact, the implementation effect of each RRA is also represented in linguistic terms and the fuzzy set representation for linguistic terms are used.

The fuzzy similarity between two fuzzy numbers can be obtained by Equation (1), in which $\tilde{d}(A, B)$ is the fuzzy distance between fuzzy number A and fuzzy number B and can be calculated according to the study of Guha and Chakraborty (2010), $0 \leq \tilde{d}(A, B) \leq 1$.

$$\tilde{s}_{A,B} = 1 - \tilde{d}(A, B) \tag{1}$$

Besides the risk probability and impact, the categories that target risks and historical risks belong to and the time interval between the evaluations of target risks and historical risks are also factors that may affect similarities between target risks and historical risks. With the consideration of the above mentioned factors, the similarity between two risks can be calculated by Equation (2).

$$\tilde{s}_{ijk} = \epsilon_{ijk}^{h} \frac{w_1 \, \tilde{s}_{ijk}^{P} + w_2 \, \tilde{s}_{ijk}^{L}}{(1+d)^{\Delta y_{ijk}}} \tag{2}$$

In Equation (2), $\tilde{s}_{ijk}^{P}$ and $\tilde{s}_{ijk}^{L}$ denote the similarities between the risks R_j^k and R_i in terms of the probability and impact, respectively. w_1 and w_2 denote the weights of the risk probability and impact, respectively, reflecting the PM's preferences for the two attributes, in which $w_1 \in [0,1]$, $w_2 \in [0,1]$ and $w_1 + w_2 = 1$. d denotes the value of the time factor, and a higher value of d means that more attention should be paid to historical risks that have occurred in recent years. Δy_{ijk} denotes the time interval between the evaluations of R_j^k and R_i.

The historical risks whose similarities are higher than a predefined threshold (σ) can be screened out. After screening out similar historical risks, their corresponding historical RRAs can be obtained. However, because the historical RRAs were formulated with respect to the historical risks rather than the target risks, it is necessary for PMs to make adaptation before adopting them to mitigate the target risks. The adaptation can be made from three aspects: RRA itself, the implementation cost of the RRA and the estimated effect of implementing the RRA. Let $\hat{D}_j^k$ and $\hat{C}_j^k$ denote the revised implementation effect and cost of A_j^k respectively. After the adaptation process, the historical RRAs can be thought of as the alternatives for the next step of optimal selection.

2.2 RRA selection optimization model

Before model construction, several assumptions need to be made. Assumption 1: The target risks are independent mutually. Assumption 2: In the case database, a historical risk has only one corresponding RRA. Assumption 3: The budget is considered to be the only resource constraint. Since fuzzy numbers are used to describe the similarities between target risks and historical risks and the implementation effects of RRAs, a fuzzy optimization model is developed below.

$$\max \ z = \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{m} \xi_{ijk}\tilde{s}_{ijk}\hat{D}_j^k x_{ijk} \tag{3}$$

$$\text{s.t.} \ \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ijk}\hat{C}_j^k \le B \tag{4}$$

$$\sum_{k=1}^{K}\sum_{j=1}^{m} x_{ijk} \le 1, \quad i = 1, \ldots, n \tag{5}$$

$$\xi_{ijk} = \begin{cases} 0, & \tilde{s}_{ijk} < \sigma \\ 1, & \tilde{s}_{ijk} \ge \sigma \end{cases} \tag{6}$$

$$x_{ijk} \in \{0,1\}, \quad i = 1, \ldots, n, \ \ j = 1, \ldots, m, \ \ k = 1, \ldots, K. \tag{7}$$

In the model, objective function (3) maximizes the estimated effect of implementing the revised historical RRAs. Equation (4) is the budget constraint. Constraint (5) ensures that no more than one RRA will be used to mitigate each target risk. Equation (6) means the screening process. Constraint (7) is a binary mode indicator. The fuzzy optimization model can be transformed into a single-objective crisp linear programming model according to the study of Zimmermann (1978). The transformed crisp model can be solved by LP solvers and the optimal RRAs can be obtained.

3 Case study

In order to demonstrate the feasibility of applying the proposed model in actual projects, a metro construction project S in city D is presented. According to the proposed methodology, alternative RRAs are retrieved and an optimal set of RRAs can be obtained. From results, it can be seen that the higher effects of project risk response can be achieved when more budget is allocated. In addition, in order to achieve better effects of project risk response, the value of the threshold needs to be set lower. However, the more RRA alternatives result from lower threshold will incur additional costs of manpower and other financial resources in the adaptation process. Since the total budget for project risk response is fixed, the budget allocated for implementing RRAs would reduce due to the increased adaptation cost and then the total effect of project risk response may decrease. Therefore, to achieve the maximized risk response effect within the limited budget, the reasonable thresholds should be set and the tradeoffs between the budget for RRA implementation and for historical RRA adaptation should be considered.

4 Conclusions

In this paper, a method integrates the case-based method and the optimization method is proposed for decision support in project risk response. Compared with the existing RRA selection methods, the advantages and contributions of the proposed method are three facets. First, the proper RRA alternatives with better effects can be developed by using the case-based method. Second, the optimal set of RRAs can be obtained easily by the optimization model. Third, the fuzzy set theory is applied to evaluate the risk probability, risk impact and the similarity between risks in the RRA selection process. The advantage of using the fuzzy set theory is that the PMs and experts can make the evaluations with linguistic terms, which is more suitable for human perception in actual situations. The proposed method can be applied to many projects with some information of historical projects for reference.

Some managerial suggestion and implication can also be drawn. First, in order to provide better decision support for PMs, organizations should always capture a long-term perspective with an awareness of keeping documents of all handled historical projects. The integration of knowledge management and the PRM process to some extent would provide better help for PMs. Second, the relatively lower thresholds may incur considerably higher adaptation costs with respect to the retrieved RRAs. Thus, the budget for RRA implementation may be insufficient and the effect of project risk response may decrease. However, if the thresholds are higher, few or even no RRA alternatives will be screened out. Therefore, to achieve the maximized response effect within limited budget, PMs should set reasonable thresholds with discretion and consider the tradeoffs between the budget for RRA implementation and for historical RRA adaptation.

References

Ben-David I., G. Rabinowitz and T. Raz, 2002, "Economic optimization of project risk management efforts", The Israel Institute of Business Research.

Ben-David I., T. Raz, 2001, "An integrated approach for risk response development in project planning", *Journal of the Operational Research Society*, Vol. 52, pp. 14–25.

Chapman C., S. Ward, 2003, *Project risk management: processes, techniques, and insights*, Wiley.

Datta S., S.K. Mukherjee, 2001, "Developing a risk management matrix for effective project planning-an empirical study", *Project Management Journal*, Vol. 32, pp. 45–57.

Dey P. K., 2002, "Project risk management: a combined analytic hierarchy process and decision tree approach", *Cost Engineering*, Vol. 44, pp. 13–27.

Dey P. K., 2012, "Project risk management using multiple criteria decision-making technique and decision tree analysis: a case study of Indian oil refinery", *Production Planning & Control*, Vol. 23, pp. 903–921.

Elkjaer M., F. Felding, 1999, "Applied project risk management-introducing the project risk management loop of control", *Project Management*, Vol. 5, pp. 16–25.

Fang C., F. Marle, M. Xie, and E. Zio, 2013, "An integrated framework for risk response planning under resource constraints in large engineering projects", *IEEE Transactions on Engineering Management*, Vol. 60, pp. 627–639.

Flanagan R., G. Norman, 1993, *Risk management and construction*, Blackwell, Oxford.

Guha D., D. Chakraborty, 2010, "A new approach to fuzzy distance measure and similarity measure between two generalized fuzzy numbers", *Applied Soft Computing*, Vol. 10, pp. 90–99.

Haimes Y. Y., 2015, *Risk modelling, assessment, and management*, John Wiley & Sons.

Kayis B., G. Arndt, M. Zhou and S. Amornsawadwatana, 2007, "A risk mitigation methodology for new product and process design in concurrent engineering projects", *CIRP Annals-Manufacturing Technology*, Vol. 56, pp. 167–170.

Klein J. H., 1993, "Modelling risk trade-off", *Journal of the Operational Research Society*, Vol. 44, pp. 445–460.

Kujawski E., 2002, "Selection of technical risk responses for efficient contingencies", *Systems Engineering*, Vol. 5, pp. 194–212.

Kujawski E., D. Angelis, 2010, "Monitoring risk response actions for effective project risk management", *Systems Engineering*, Vol. 13, pp. 353–368.

Marmier F., I. F. Deniaud and D. Gourc, 2014, "Strategic decision-making in NPD projects according to risk: application to satellites design projects", *Computers in Industry*, Vol. 65, pp. 1107–1114.

Miller R., D. Lessard, 2001, "Understanding and managing risks in large engineering projects", *International Journal of Project Management*, Vol. 19, pp. 437–443.

Piney C., 2002, "Risk response planning: select the right strategy", In *Fifth European Project Management Conference*, pp. 1–7.

Pipattanapiwong J., T. Watanabe, 2000, "Multi-party risk management process (MRMP) for a construction project financed by an international lender", In *Proceeding of Construction Engineering and Management Symposium*, pp. 85–92.

Sherali H. D., J. Desai and T. S. Glickman, 2008, "Optimal allocation of risk-reduction resources in event trees", *Management Science*, Vol. 54, pp. 1313–1321.

Zhang Y., 2016, "Selecting risk response strategies considering project risk interdependence", *International Journal of Project Management*, Vol. 34, pp. 819–830.

Zhang Y., Z. P. Fan, 2014, "An optimization method for selecting project risk response strategies", *International Journal of Project Management*, Vol. 32, pp. 412–422.

Zimmermann H. J., 1978, "Fuzzy programming and linear programming with several objective functions", *Fuzzy Sets and Systems*, Vol. 1, pp. 45–55.

Multi-Level Tabu Search for Job Scheduling in a Variable-Resource Environment

Nicolas Zufferey

GSEM - University of Geneva, Switzerland, n.zufferey@unige.ch

Keywords: tabu search, variable resources, job scheduling.

1 Introduction

Tabu Search (TS) is a popular local search that works as follows. In each iteration, a *neighbor* solution s' is generated from the *current* solution s by modifying s according to a predefined rule. Such a modification is called a *move*. In order escape from a local optimum, when a move is performed, its reverse is forbidden (it is said to be *tabu*) during *tab* (parameter) iterations. TS generally performs the best non-tabu move in each iteration, and it is usually stopped when a time limit is reached. The reader is referred to (Gendreau and Potvin 2010) for more information on TS and other well-known metaheuristics.

In this paper, *Multi-Level Tabu Search* (MLTS) is formulated (see Section 2) in the context where various resources can be used, for instance in production scheduling (see Section 3 relying on (Hertz *et. al.* 2009)) or in transportation (see Section 4 relying on (Zufferey *et. al.* 2016)). A *level* is defined here as a set of available resources. The main idea of MLTS consists in performing successive applications of TS for different levels. As TS is usually time-consuming, a challenge would be to identify the most promising levels to investigate, as pointed out in the conclusion (see Section 5).

The success of two existing solution methods are discussed in Sections 3 and 4, which can however be considered as belonging to the MLTS methodology. The reader is referred to the two previous references for having more information on the literature review, the technical details of the problems, the experiments and the numerical results (which are state-of-the art). For these reasons, simplifications and shortcuts are proposed in Sections 3 and 4 in order to better focus on (1) the main characteristics of the problems, and (2) the relevance of the MLTS methodology for such types of problems. A unified terminology is employed in order to fit with the proposed MLTS framework.

2 Multi-Level Tabu Search (MLTS)

Let f be the objective function to minimize (e.g., a cost function), k be the number of available resource types (e.g., machines, vehicles), and J be the set of jobs to perform with the selected resources. The goal consists in performing all the jobs while minimizing f (if a job is not performed, a rejection cost has to be minimized). A *solution space* or *level* S is denoted as $S = (r_1, r_2, \ldots, r_k)$, where r_i is the number of available resources of type i. MLTS is formulated in Algorithm 1, which returns the best encountered solution $s^\star$. It first requires an initial level S and an initial solution $s \in S$. Such a solution is then improved with TS (i.e., TS tries to use the available resources as well as possible according to f). At the end of the main loop, the level is updated (i.e., the available resources are modified), and the process is restarted with a new solution space as long as a stopping condition is not met. The two main steps are *resource utilization* (RU) and *resource modification* (RM). On the one hand, any solution method can be employed for RU, but TS is proposed here as it usually finds a good compromise between various criteria (e.g., quality, speed, robustness). A classic neighborhood structure for TS is the *reinsertion move* (i.e., perform a job j earlier/later with the same resource type, or perform j with a different resource type). On the other hand, various strategies can be investigated for RM, ranging from

a basic resource augmentation/reduction (e.g., increase/decrease a single r_i by one unit) to more refined variations (e.g., perform significant but structured modifications as in *variable neighborhood search* or *large neighborhood search* (Gendreau and Potvin 2010)). The stopping condition of MLTS can be simply a time limit. For some cases, it can be a guarantee that all the promising levels have been explored, which is likely to be possible if k is small and f varies in a convex fashion according to the modifications of S.

Algorithm 1 Multi-Level Tabu Search (MLTS)

Initialization

1. generate an initial level S;
2. set $f(s^\star) = +\infty$;

While a stopping condition is not met, **do**:

1. generate a solution s in level S;
2. *resource utilization* (RU): improve s with TS (within S);
3. update the best encountered solution: if $f(s) < f(s^\star)$, set $s^\star = s$;
4. *resource modification* (RM): modify the solution space S (e.g., augment or reduce it);

Return $s^\star$

3 MLTS in production scheduling

A set J of jobs have to be performed in a plant within a planning horizon H (typically several days). The sum of the below-presented costs has to be minimized. $M(r)$ is the set of machines of type r that are available from the very beginning of H. $f_s(r)$ is the storage cost (per time slot) of a machine of type r in the plant. At any time slot $t \in H$, a new machine of any type r can be purchased at a cost of $f_p(r)$. But for each type r, there is a limit on the number of machines that can be bought. Observe that the later is t, the lower will be the associated storage cost of the new machine.

The following information is associated with each job j: a strict time window $[e_j, d_j]$, a processing time $p_j = d_j - e_j$, the required type of machine m_j, the number n_j of required machines, and the rejection cost $f_r(j)$ that is encountered if job j is not performed with the available resource level S. The rejection cost depends on p_j and on m_j. A typical job j can thus be: produce two batches (i.e., $n_j = 2$) from 1 pm to 8 pm (i.e., $p_j = 7$ hours, $e_j = 1$ pm, $d_j = 8$ pm) on machine type $m_j = 2$. Substitution is possible. In other words, if a job j requires machine type r, it can also be processed with machine type r' if r' covers r (i.e., r' is able to perform any job assigned to r). This is denoted as $r' \to r$. If a machine of type r is allocated to a job j, two types of costs are encountered: the fixed allocation cost $f_a^{fix}(r)$, and the variable allocation cost $f_a^{var}(r)$, which is time-dependent. Unsurprisingly, substitution is not advantageous. Formally, if $r' \to r$, then $f_a^{fix}(r') > f_a^{fix}(r)$ and $f_a^{var}(r') > f_a^{var}(r)$.

Maintenance constraints have also to be satisfied. In this context, $M_u(r)$ is the maximum time of use of a machine of type r without maintenance, $M_d(r)$ is the associated maintenance duration, $M_w(r)$ is the associated number of requested workers to do it, and M_c is the capacity of the maintenance workshop (i.e., the number of available workers, which corresponds to the number of maintenances that can be performed concurrently). Anytime a maintenance is performed on a machine of type r, a maintenance cost $f_m(r)$ is encountered.

For each job j, the following decisions are possible, from the best to the worst: (1) perform j with the requested machine type r (regular allocation costs $f_a^{fix}(r)$ and $f_a^{var}(r)$);

(2) perform j with a machine of type $r' \rightarrow r$ (larger allocation costs); (3) buy a new machine of type r (at time t) and perform j with this new machine (regular allocation costs, but additional purchasing and storage costs); (4) reject job j (rejection cost $f_r(j)$). From a global perspective, it is however better to reject a few jobs and use substitution if one can avoid purchasing a new machine.

The solution space S is defined here as the set of machines available in the stock (including the possibly purchased machines with their associated purchasing times). RM simply consists in purchasing a machine of type r (especially if the rejection costs were strongly impacted by machine type r), or by removing a previously bought machine of type r (especially if RU is finally able to find a solution that does not employ too much one machine of type r). RU relies on TS with the following main neighborhood structure. A move (j, u) consists in inserting a rejected job j (which needs machine type r) in the schedule of machine u (of type r or of type $r' \rightarrow r$). The jobs of any type in conflict (because of time-window or maintenance constraints) with job j are either rescheduled in an allowed machine (which might modify the allocation and maintenance costs) or rejected. The maintenances are greedily rescheduled when testing any move, and non-feasible solutions (because of the maintenance constraints) are discarded. If move (j, u) is performed, it is then tabu to remove job j from machine u for some iterations.

4 MLTS in vehicle routing

Consider an urban network (typically with distances below 30 km between each pair of locations) with n (typically 20) client locations and a central depot. Each client represents a medical facility (e.g., the office of a doctor, a hospital) and the depot represents the laboratory LAB that is in charge of analyzing blood samples. The planning horizon H is a day (typically from 8 am to 6 pm) and the fleet of available vehicles is initially located at LAB. Dynamic travel times are considered (i.e., the *actual* travel time between two locations has to be simulated by perturbing the *expected* travel time). Two types of job exists (and can involve the same client location several times during H): (A) move a blood sample j from a client to LAB (where j has to be analyzed); (B) move a medical equipment j from LAB to a client (where j has to be used the next day). For type (A), in order to keep the chemical properties of each blood sample j, a time window $[e_j, d_j]$ is associated, where e_j is the time at which j becomes available, and d_j is the latest time at which j should be delivered to LAB. If d_j is exceeded, the blood sample is lost as its chemical properties are not preserved. Such a situation is totally forbidden by LAB. For type (B), the time window is simply the full planning horizon H (meaning that the delivery of medical equipment can be performed anytime during the day). Capacity constraints have to be satisfied. Each j of type (A) has a volume of $q_j \in [1, 10]$ liters, whereas $q_j \in [10, 60]$ liters for each j of type (B). Two types of vehicles are available, namely scooters (capacity of 60 liters, speed of 18.7 km/h) and cars (capacity of 900 liters, speed of 17 km/h). The daily average demand is characterized by the following features: 300 *static* (i.e., known before H) blood requests, 200 *dynamic* (i.e., revealed during H with the use of simulation) blood requests, and 25 *static* equipment request. Three objective functions have to be minimized in a lexicographic order (i.e., a higher-level objective is infinitely more important than a lower-level objective): (f_1) rejection costs (corresponding to the use of an external taxi service to pick up a blood sample j that cannot be collected on time by the LAB fleet, or if the capacity constraint does not allow to perform j); (f_2) car costs (corresponding to the number of used cars); (f_3) travel costs of all the vehicles of the fleet (which is proportional to fuel consumption).

The solution space S is defined here as the fleet of vehicles (i.e., set of cars plus set of scooters), but LAB gives upper bounds on the number of vehicles of each type that can be

used (i.e., it is not always possible for f_1 to reach 0). RM simply consists in adding a vehicle (especially if f_1 has to be significantly decreased), or in removing a vehicle (especially if f_2 can be reduced without augmenting f_1). An initial solution has to be generated before H with all the static requests. For this purpose, two procedures are used, namely GR (a greedy heuristic) and DLS (a descent local search based on the well-known CROSS exchanges (Taillard *et. al.* 1997)). In each step of GR, the next job to insert at best is the one that minimizes the augmentation of f_2. If there are several equivalent options, the goal is then to minimize the augmentation of f_3. Again, if there are still more than one option, the objective is finally to balance the load of the various involved vehicles. Indeed, if a vehicle is too much loaded, its availability to serve a new job becomes poor, which has to be avoided especially if the vehicle is in the vicinity of a newly revealed job. If a job cannot be inserted in the schedule without violating the deadline constraint or the capacity constraint, a taxi is called and f_1 is augmented accordingly. A move in RU simply consists in inserting a job (that dynamically appears during the day) in the current solution s as explained above (i.e., use GR for the insertion and DLS for the improvement).

5 Conclusions and future work

In this work, MLTS (for *Multi-Level Tabu Search*) is formulated. It is specifically dedicated for situations where variable resources are available (any fixed set of resources being called a *level*), and it is well adapted either if various objective functions have to be considered jointly (with or without lexicographic optimization). Under the light of MLTS and with a unified view, the success of two existing solutions methods is presented (one for production, one for transportation). Note that other applications can be seen as adaptations of the MLTS framework (e.g., (Amrani *et. al.* 2011, Schindl and Zufferey 2015)). Among the future works, one could imagine filtering techniques and local search procedures for moving from one promising level to another. This would be particularly relevant for situations where an efficient level is difficult to find and a small computation-time limit is imposed.

References

Amrani H., Martel A., Zufferey N., Makeeva P., 2011, "A Variable Neighborhood Search Heuristic for the Design of Multicommodity Production-Distribution Networks with Alternative Facility Configurations", *Operations Research Spectrum*, Vol. 33 (4), pp. 989-1007.

Gendreau M., J.-Y. Potvin, 2010, "Handbook of Metaheuristics", *Springer*.

Hertz A., Schindl D., Zufferey N., 2009, "A Solution Method for a Car Fleet Management Problem with Maintenance Constraints", *Journal of Heuristics*, Vol. 15 (5), pp. 425-450.

Schindl D., Zufferey N., 2015, "A Learning Tabu Search for a Truck Allocation Problem with Linear and Nonlinear Cost Components", *Naval Research Logistics*, Vol. 61 (1), pp. 42-45.

Taillard ED., Badeau P., Gendreau M., Guertin F-, Potvin J-Y., 1997, "A tabu search heuristic for the vehicle routing problem with soft time windows", *Transportation Science*, Vol. 31, pp. 170–186.

Zufferey N., Cho B.Y., Glardon R., 2016, "Dynamic Multi-Trip Vehicle Routing with Unusual Time-Windows for the Pick-Up of Blood Samples and Delivery of Medical Material", *Proceedings of the 5th International Conference on Operations Research and Enterprise Systems*, ICORES 2016, Rome, Italy, February 23-25

Scheduling a forge with due dates and die deterioration

Ösz O.[1], Ferenczi B.[2], and Hegyháti M.[1]

[1] Department of Information Technology, Széchenyi István University, Hungary
osz.oliver@sze.hu, hegyhati@sze.hu
[2] Department of Logistics and Forwarding, Széchenyi István University, Hungary

Keywords: scheduling, forge, due dates, deterioration, MILP.

1 Motivation

We have been studying the production processes of an axle-manufacturer. Their main products are heavy-duty truck axles for various trucks, buses and other vehicles. The equipment required to manufacture these products is very expensive. Therefore, in order to increase sales on the short term, optimization of the production process is required. For more substantial, long-term growth, investment in more equipment is necessary. The presented methods can also be used to estimate the impact of the investment on efficiency.

The production starts with steel rods of different sizes, made by external suppliers. The process consists of 4 main stages: forging, heat treatment, preparation, and machining.

In the forging stage, the steel rod is heated up, placed into a forging die and a hammering machine forms it into the desired shape. After some cooling and grinding, it goes through a heat treatment furnace.

In preparation phase, a hydraulic press gives the product its final geometry. After several surface inspection and treatment steps, then painting, the product is ready for machining and packaging.

Forging and heat treatment are the most critical stages of the process, as later stages have higher flow-rates and they are also less expensive to upgrade. Therefore, we aim to optimize the manufacturing process by scheduling the forging and heat treatment jobs for a more efficient operation.

2 Scheduling problem

Forging is the first step of the process. The intermediates need to cool down after that. Before heat treatment, the intermediates can be stored indefinitely but with incurred storage costs. A grinding step is also necessary sometime during this storage period. It is assumed that the resources required for grinding are always available. This way, we can simplify the scheduling problem by introducing a minimum storage time between forging and heat treatment, that is enough for cooling and grinding.

A similar simplification is made for the stages after heat treatment. These steps are assumed to have a constant flow-rate with no equipment constraints.

The forge produces different intermediate products which require different resources. Product orders are given as input with product type, quantity and due date information. These product demands must be fulfilled without delays.

Some input materials of the process may not be available at the start but their release dates are given a priori.

The goal is to find a resource-feasible schedule which fulfils product demands on time, and minimizes production and storage costs.

The following subsections describe the scheduling aspects of the production steps.

2.1 Die forging

The forge operates on one steel rod at a time. Different products require different sized rods, and different forging dies. Each die has a production capacity, given in number of uses. After this many uses, the die cannot be used until it is restored. Restoration is a long and expensive process which fully restores the production capacity of the die.

Switching dies is an expensive and time-consuming operation which also decreases the remaining capacity of the die. Moreover, when forging starts on a die (due to necessary calibrations), the first batches will result in waste.

2.2 Heat treatment

The heat treatment furnace is operated in cycles. It is active for a certain time, then shut down for maintenance. The length of these cycles is a few weeks.

The intermediates enter the furnace in batches of a given maximum size and leave it after a given time. The processing time and costs are independent from the batch size.

3 Literature

Gascon and Leachman (1988) proposed a dynamic programming approach for minimizing changeover and inventory holding costs with deterministic demands. The authors studied a process with unit sized batches and equal processing times. These assumptions do not hold for the forge scheduling problem, so despite the similar objective function, their algorithm is not applicable to our problem.

Many different methods have been presented in the literature for scheduling manufacturing processes. Méndez and Cerdá (2006) made a comprehensive review of Mixed-Integer Linear Programming (MILP) approaches. They compared the advantages and disadvantages of various time and event representation models.

There are various publications addressing the deterioration of resources (Tang and Liu 2009, Zhao and Tang 2010) in the sense that processing times increase as time progresses. Zhao and Tang (2010) presented an approach which takes into account restoration jobs to revert the deterioration. However, in our problem, the resources, namely the dies, deteriorate through changeovers and usages, not as a function of time, and their production capacity decreases, not their flow rate.

4 Modeling

We decided to use a MILP model with a discrete uniform time grid. This time representation has some modelling advantages to continuous time models, when dealing with constraints similar to our problem. One of the modeling difficulties is that we do not know in advance, how many times a die will be changed before its restoration, and in what ratio its capacity will be subdivided. This would make it harder to determine the number of required time points in a continuous time formulation. Inventory costs, and resource demands, arrivals are also harder to model in continuous time models, leading to more complex constraints and worse LP-relaxation.

The forge is modeled with 2 binary variables for each pair of time point and die type, the first is active during the setup phase of the die, and the second when forging is active, i.e. intermediates are being created. (1) shows the constraint for ensuring that at most 1 die can be in use at a time.

$$\sum_{d \in Dies} (forge^s_{t,d} + forge^a_{t,d}) \leq 1, \;\; t = 1, \ldots, T \tag{1}$$

The required setup time is enforced by (2): forging can only be active if either it was active in the previous time point, or it was set up in the previous number of time points equal to the setup time given in time periods.

$$forge^a_{t,d} \cdot setup \leq forge^a_{t-1,d} \cdot setup + \sum_{t'=\max(1,t-setup)}^{t-1} forge^s_{t',d}, \;\; t = 2, \ldots, T, \forall d \in Dies \tag{2}$$

Heat treatment is modeled by a continuous variable, which represents the quantity under treatment from each axle type during a time period. The total quantity in a time period cannot exceed the capacity of the furnace, this constraint is shown in (3). The furnace capacity is not constant over time, hence it is subscripted by the time point. In the case study, it was a periodic availability but the model allows more complex discrete functions.

$$\sum_{a \in Axles} heat_{t,a} \leq furnace_t, \;\; t = 1, \ldots, T \tag{3}$$

The minimum waiting time necessary for cooling and grinding between forging and heat treatment is ensured by constraint (4). It only allows heat treatment of intermediates that were forged at least as long ago as the required waiting time given in number of time periods.

$$heat_{t,a} \leq res_{t-cooling,a}, \;\; t = cooling + 1, \ldots, T, \forall a \in Axles \tag{4}$$

Input and output materials, equipment, and capacity levels of forging dies are modelled as resources in a Resource Task Network (RTN). The RTN formulation was proposed by Pantelides (1994), and it is a general representation of material flows and production equipment. Treating die capacities as continuous resources allows handling the special deterioration constraints associated with die forging. Changing a die decreases the capacity, not only production usage. Die restoration produces the resource representing die capacity. RTN formulation can also deal with multiple resource types (steel rods, die capacity, hammering machine) used by the same process.

Resource levels are calculated in every time point (res_t), and used to compute storage costs and enforce meeting product demands. The starting levels ($t = 1$) are given as input, the later levels are computed from the previous time point and the resource usage / production of the last period. This is shown in equation (5). Note that *demand* is adjusted by the fixed time requirement of the finishing stages following heat treatment.

$$\begin{array}{l} res_{t+1,r} = res_{t,r} + input_{t+1,r} \\ -demand_{t+1,r} - heat_{t,r} + heat_{t,product(r)} \\ -\sum_{d \in Dies} forge^a_{t,d} \cdot usage_{d,r} - \sum_{d \in Dies} forge^s_{t,d} \cdot usage^s_{d,r} \end{array} \quad t = 1, \ldots, T, \forall r \in R \tag{5}$$

Die capacities must be handled separately, as restoration increases it to the maximum capacity, regardless of its current level. Therefore, the resource balance for die capacities is split into a lower and upper bound inequality, shown in (6) and (7). The res variable is bounded from the top by the maximum die capacity. Deterioration is included in $usage^s$, which is subtracted during the setup phase.

$$res_{t+1,d} \geq res_{t,d} - \sum_{d \in Dies} forge^a_{t,d} \cdot usage_{d,d} - \sum_{d \in Dies} forge^s_{t,d} \cdot usage^s_{d,d} \qquad t = 1, \ldots, T, \ \ \forall d \in Dies \tag{6}$$

$$res_{t+1,d} \leq res_{t,d} - \sum_{d \in Dies} forge^a_{t,d} \cdot usage_{d,d} - \sum_{d \in Dies} forge^s_{t,d} \cdot usage^s_{d,d} + restored_{t+1,d} \cdot maxCap_d \qquad t = 1, \ldots, T, \ \ \forall d \in Dies \tag{7}$$

The binary variable, $restored$ is set to 1 in the last time period of the restoration. Restoration time is enforced in a similar manner as the setup time shown in (2). Solution time is improved if another lower bound (8) for res is added, which ensures that the die is fully restored.

$$res_{t,d} \geq restored_{t,d} \cdot maxCap_d, \ \ t = 1, \ldots, T, \ \ \forall d \in Dies \tag{8}$$

The objective function to be minimized is the sum of different costs calculated from the variables introduced above. Costs include setup costs, storage costs and die restoration costs.

5 Computational results

The model was tested on 2 problem instances. The problems were solved on a computer with 4 CPU cores (Intel i7-6700HQ, 2.60GHz) and 8 GB RAM, using Gurobi 7.5.2. The length of time periods was set to 8 hours in each instance, to correspond with the work shifts.

In the first instance, 22 products were considered over a time horizon of 4 months. The solver could not obtain a proven optimal solution in the time limit of 1000 s but provided feasible schedules with under 5% MIP gap.

The second instance focused on the 16 products that had the highest demand, and the planning horizon was lowered to 1 month. An optimal solution was found in 646 s.

6 Conclusion

We modeled the forging process of an axle-manufacturing factory. The special constraints of deteriorating forging dies were modeled. These constraints originate from a real industrial environment. To the extent of our knowledge, this scheduling problem has not been studied in optimization research, prior to this work.

The presented model can be used to find the optimal schedule for a real-life, industrial problem. For long-term planning, finding the optimal solution may require unacceptably long time but good near-optimal solutions can be obtained in under an hour.

7 Acknowledgements

Supported by the ÚNKP-17-3 New National Excellence Program of the Ministry of Human Capacities.

References

Gascon A., R. C. Leachman, 1988, "A Dynamic Programming Solution to the Dynamic, Multi-Item, Single-Machine Scheduling Problem", *Operations Research*, Vol. 36(1), pp. 50–56.

Méndez C., Cerdá J., 2006, "State-of-the-art review of optimization methods for short-term scheduling of batch processes", *Computers & Chemical Engineering*, Vol. 30(6-7), pp. 913–946.

Pantelides C. C., 1994, "Unified frameworks for optimal process planning and scheduling", *Proceedings on the second conference on foundations of computer aided operations*, Cache Publications, New York, pp. 253–274.

Tang L., P. Liu, 2009, "Flowshop scheduling problems with transportation or deterioration between the batching and single machines", *Computers & Industrial Engineering*, Vol. 56(4), pp. 1289–1295.

Zhao C., H. Tang, 2010, "Single machine scheduling with general job-dependent aging effect and maintenance activities to minimize makespan", *Applied Mathematical Modelling*, Vol. 34(3), pp. 837–841.