ADAM MICKIEWICZ
UNIVERSITY
IN POZNAŃ

THE SECOND INTERNATIONAL WORKSHOP ON DYNAMIC SCHEDULING PROBLEMS

ADAM MICKIEWICZ UNIVERSITY IN POZNAŃ
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
JUNE 26TH – 28TH, 2018, POZNAŃ, POLAND

EXTENDED ABSTRACTS

IWDSP
POZNAŃ 2018

ADAM MICKIEWICZ
UNIVERSITY
IN POZNAŃ

THE SECOND INTERNATIONAL WORKSHOP ON DYNAMIC SCHEDULING PROBLEMS

ADAM MICKIEWICZ UNIVERSITY IN POZNAŃ
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
JUNE 26TH – 28TH, 2018, POZNAŃ, POLAND

EXTENDED ABSTRACTS

IWDSP
POZNAŃ 2018

This book contains extended abstracts of a plenary lecture, a tutorial and papers presented at the Second International Workshop on Dynamic Scheduling Problems, June 26th–28th, 2018, Poznań, Poland.

Layout, maps and cover design by Bartłomiej Przybylski
Edited by Stanisław Gawiejnowicz

ISBN: 978-83-937220-8-2 (printed version)
ISBN: 978-83-937220-9-9 (eBook)
ISBN: 978-83-951298-0-3 (printed version + eBook)

Publisher:

Polish Mathematical Society
Śniadeckich 8, 00-956 Warsaw
Poland

Welcome to IWDSP 2018

Dear participant,

on behalf of the Programme and Local Committees, I am pleased to welcome you to IWDSP 2018, the Second International Workshop on Dynamic Scheduling Problems, and to the Faculty of Mathematics and Computer Science, Adam Mickiewicz University in Poznań, which is the host of this event.

The IWDSP 2018 workshop is the second event in the series started in 2016, focused on dynamic scheduling problems defined by parameters whose values are varying in time. Problems of this kind appear in many applications. The most common examples are scheduling problems with time-, position- and resource-dependent job processing times. The aim of this workshop is to present the recent research in this important domain of scheduling theory.

The Programme Committee, supported by the members of the Advisory Committee and external reviewers, selected for presentation at IWDSP 2018 papers submitted by the authors from Belarus, Belgium, France, Germany, Israel, Poland, the Russian Federation and the United Kingdom. These papers, together with a plenary lecture on energy-efficient scheduling and a tutorial on mechanism design in scheduling, allowed the Programme Committee to prepare an attractive scientific programme of the event.

I wish you a pleasant stay in Poznań and a fruitful workshop, expressing the hope that you will find IWDSP 2018 stimulating for your further research.

Stanisław Gawiejnowicz
The Chair of the Programme Committee
The Chair of the Local Committee

Committees

Committees

Programme Committee

Stanisław GAWIEJNOWICZ (chair), Adam Mickiewicz University in Poznań, Poland

Gur MOSHEIOV, Hebrew University of Jerusalem, Jerusalem, Israel

Vitaly A. STRUSEVICH, Greenwich University, London, United Kingdom

Advisory Committee

Alessandro AGNETIS, University of Siena, Siena, Italy

Hans KELLERER, University of Graz, Graz, Austria

Alexander KONONOV, Sobolev Institute of Mathematics, Novosibirsk, Russian Federation

Mikhail Y. KOVALYOV, National Academy of Sciences of Belarus, Minsk, Belarus

Bertrand M-T. LIN, National Chiao Tung University, Hsinchu, Taiwan

Michael L. PINEDO, Stern School of Business, New York University, New York, USA

Dvir SHABTAY, Ben-Gurion University of the Negev, Beer Sheva, Israel

Zhiyi TAN, Zhejiang University, Hangzhou, P. R. China

Prudence W-H. WONG, University of Liverpool, Liverpool, United Kingdom

Local Committee

Stanisław GAWIEJNOWICZ (chair), Adam Mickiewicz University in Poznań

Marta KOLIŃSKA, Adam Mickiewicz University in Poznań

Bartłomiej PRZYBYLSKI, Adam Mickiewicz University in Poznań

Contents

Venue

Venue

Location

IWDSP 2018 takes place at the Faculty of Mathematics and Computer Science, Adam Mickiewicz University in Poznań, Umultowska 87, 61-614 Poznań.

The faculty is located in the Morasko campus, the new part of Adam Mickiewicz University in Poznań (for details see the attached map).

Communication

The quickest way to reach the venue of IWDSP 2018 is to take the tram no. 12, 14 or 16 in the direction of 'Os. Sobieskiego', get off at the last stop ('Os. Sobieskiego', for timetable see `www.mpk.poznan.pl`) and walk to the Morasko campus (for suggested route see the attached map).

Registration

Registration desk for IWDSP 2018 will be located in the main hall of the Faculty of Mathematics and Computer Science.

Registration will be possible on Tuesday, June 26th, 8:00–8:30, and on Wednesday, June 27th, 8:30–9:00.

Presentation room

The plenary lecture, the tutorial and all presentations will take place in the Faculty Council Room (A1-33, Level 1, see the attached map).

The room is equipped with a projector for handling presentations in typical formats, such as PDF or PPT.

Coffee breaks room

Coffee breaks will take place in the Professors' Club (A0-13, Level 0) which is located one floor below the Faculty Council Room (see the attached map).

Internet access

Wireless network is available in the whole building of the Faculty of Mathematics and Computer Science.

Details concerning usernames and passwords will be given during registration.

Get-together party, lunches and conference dinner

A get-together party will be held on Tuesday, June 26th, 19:00–21:00, in the Professors' Club (A0-13, Level 0, see the attached map).

Lunches on June 26th, 27th and 28th will be served in the Professors' Club.

The conference dinner will be organized on June 27th, 19:00–21:30, outside the Morasko campus. Details will be given during registration.

Social programme

On Thursday, June 28th, 8:00–12:00, there will be organized a walking guided tour showing the main tourist attractions of Poznań.

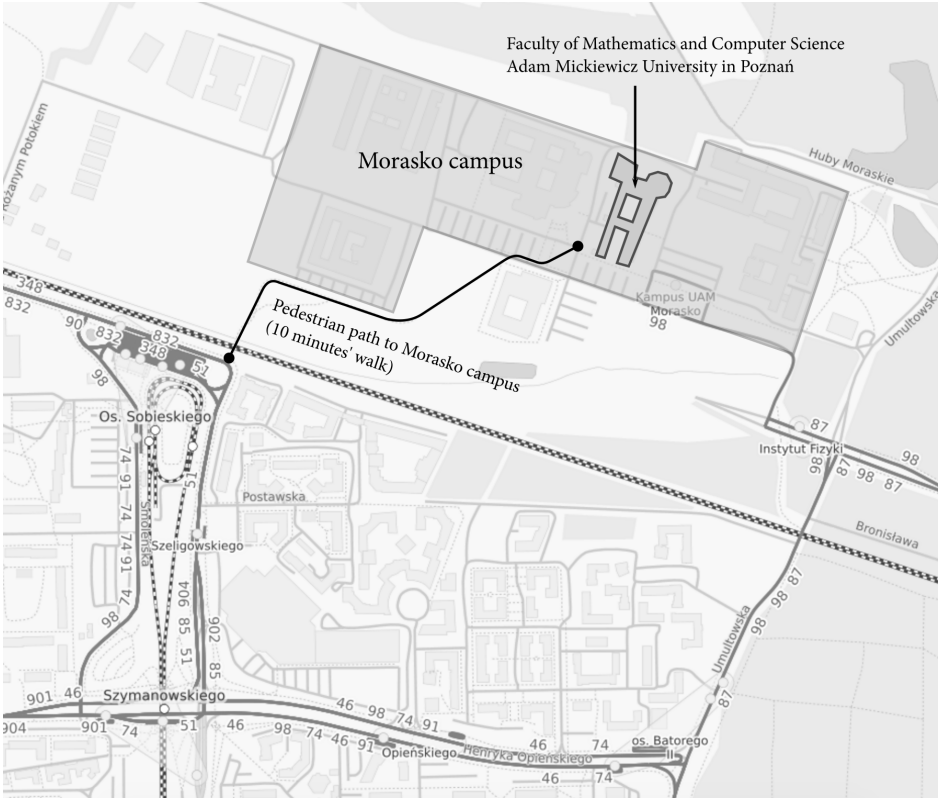


Figure 1: The main part of the Morasko campus (source: Open Street Map)

Level 1 (ground floor)

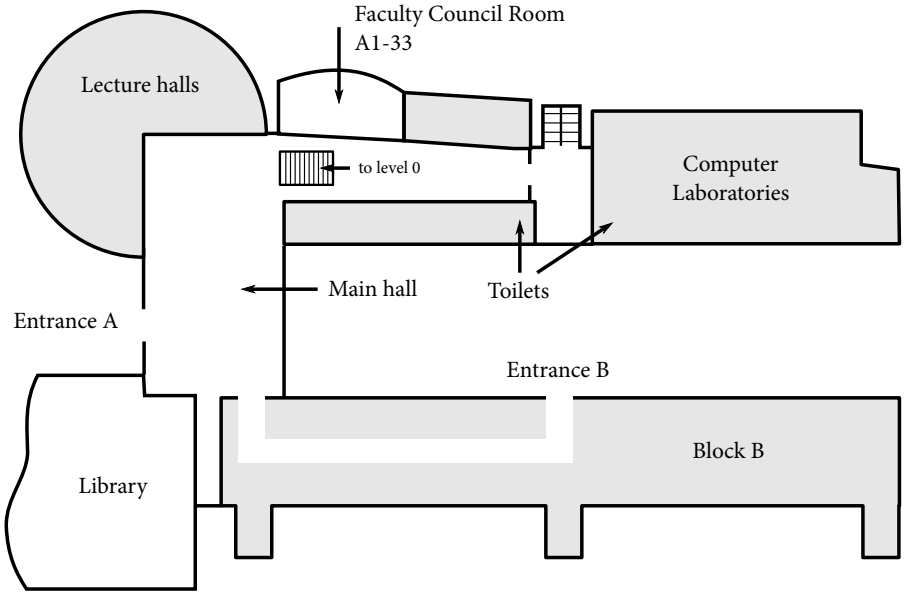


Figure 2: Level 1 (ground floor) of IWDSP 2018 venue

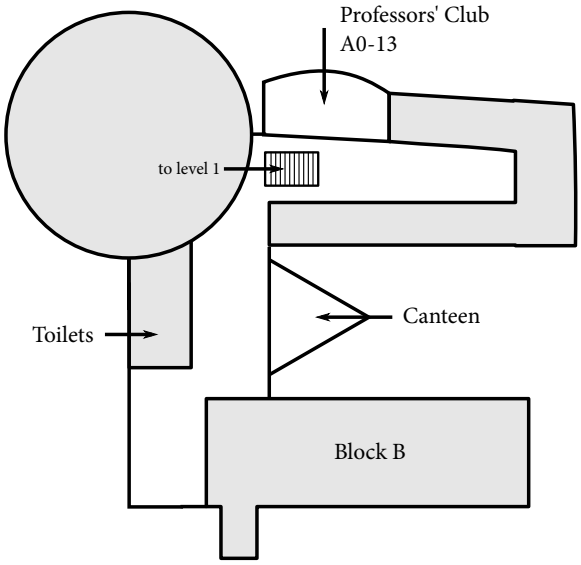


Figure 3: Level 0 (basement) of IWDSP 2018 venue

Programme

Tuesday, June 26th, 2018

08:00 – 08:30	Registration
08:30 – 09:00	Opening
09:00 – 10:20	**Session no. 1** *Speakers:* Nir Halman, Stanisław Gawiejnowicz *Chair:* Gur Mosheiov
10:20 – 10:40	Coffee break
10:40 – 12:00	**Session no. 2** *Speakers:* Farzaneh Karami, Helmut A. Sedding *Chair:* Evripidis Bampis
12:00 – 14:00	Lunch break
14:00 – 15:20	**Session no. 3** *Speakers:* Joanna Berlińska, Alexander Kononov *Chair:* Alan Soper
15:20 – 15:40	Coffee break
19:00 – 21:00	Get-together party

Wednesday, June 27th, 2018

08:30 – 09:00	Registration
09:00 – 10:20	**Session no. 4** *Speakers:* Bartłomiej Przybylski, Gur Mosheiov *Chair:* Alexander Kononov
10:20 – 10:40	Coffee break
10:40 – 12:00	**Plenary lecture** *Speaker:* Evripidis Bampis *Chair:* Stanisław Gawiejnowicz
12:00 – 14:00	Lunch break
14:00 – 16:30	**Tutorial** *Speaker:* Ruben Hoeksma *Chair:* Stanisław Gawiejnowicz
19:00 – 21:30	Conference dinner

Thursday, June 28th, 2018

08:00 – 12:00	Guided tour
12:00 – 14:00	Lunch break
14:00 – 15:20	**Session no. 5** *Speakers:* Baruch Mor, Alan Soper *Chair:* Nir Halman
15:20 – 15:40	Coffee break
15:40 – 17:40	**Session no. 6** *Speakers:* Marcin Żurowski, Maksim Barketau, Dvir Shabtay *Chair:* Stanisław Gawiejnowicz
17:40	Closing

Plenary lecture

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

Algorithms for energy-efficient scheduling

Evripidis Bampis *
Sorbonne Université, Paris, France

Keywords: scheduling, single machine, parallel machines, energy consumption models

1 Introduction

Saving energy is one of the major issues in modern computer science. Different hardware- and system-based mechanisms have been developed for reducing the energy consumption, including speed scaling or the use of multiple power states. From an algorithmic viewpoint, various models have been proposed in which are analyzed algorithms exploiting these mechanisms. We will focus on the most studied models:

- the *speed scaling* model,
- the *power-down* model, and
- the *power-down with speed scaling* model.

The first two models are independent, the third one is their combination.

2 Main energy consumption models

In the *speed scaling* model (Yao et al. [14]), the speed of a processor may be dynamically changed over time. The power of a processor running at speed s is $f(s)$, where f is a non-decreasing function of the speed, and the energy is the integral of power over time. In many references it is assumed that the power is $f(s) = s^\alpha$, where $\alpha > 1$ is a constant, or it is an arbitrary convex function of s.

In the *power-down* model (Baptiste [9]), the processors run at a *fixed* speed, but they can be turned to a *sleep state*. Hence, every processor has two states, the On state and the Off state. During the wake-up of a processor from the Off state to the On state, a *start-up* energy consumption is assumed, denoted by L. Hence,

* Plenary speaker, email: evripidis.bampis@lip6.fr

suspending the processor is only beneficial when the idle periods are long enough to compensate the consumed start-up energy.

The *power-down with speed scaling* model (Irani et al. [11]) combines the previous two models by considering speed scalable processors with a sleep state. The power function in this model is defined as $g(s) = f(s) + c$, where $f(s)$ is as in the speed-scaling model and $c > 0$ is a constant specifying the power consumed when the processor is in the On state.

Different machine environments are considered in the literature, ranging from the single-machine environment (Yao et al. [14]) to parallel heterogeneous environments (Albers et al. [3]). In the parallel machine case, we distinguish between homogeneous and heterogeneous environments. In the *homogeneous* case, the characteristics of every job are independent of the machine on which the jobs will be executed and the speed-to-power function is the same for all the machines. In the *heterogeneous* case, the following subcases have been studied:

- the *fully heterogeneous* environment, where the jobs' characteristics are machine-dependent and every machine has its own power function;
- the *power-heterogeneous* environment, where the characteristics of every job are independent of the machine on which the job will be executed, while every machine has its own speed-to-power function, and
- the *unrelated-heterogeneous* environment, where the processing volumes of the jobs are machine-dependent while all the other characteristics are independent of the machine on which the job is executed.

We will be interested in both the *preemptive* and the *non-preemptive* variants of the considered problems (Bampis et al. [6]). In the preemptive case, the execution of a job may be interrupted and resumed later. When more than one machine are considered, the preemptive case gives rise to two subcases:

- the *migratory* case, where the execution of the jobs may allow the *migration* of the jobs, i.e. the possibility to execute a job on more than one machine, without allowing its parallel execution, and
- the *non–migratory case*, where the execution of a preempted job must be continued on the same machine.

3 Lecture contents

In the lecture, we will focus on algorithms with provably good performance for various objective functions and machine environments. We will first consider *deadline-based* problems, where we aim to find a feasible schedule minimizing the energy consumption, i.e. a schedule in which each job is executed in the interval between its release date and its deadline.

We will next consider budget approaches, where we are given a budget of energy and we aim to optimize a scheduling criterion such as the *throughput*, i.e. the number of jobs that complete before their deadlines (Angel et al. [4]), the *makespan*, i.e. the time at which the last job completes its execution (Bampis et al. [7]), the *sum of* (*weighted*) *completion times* (Megow and Verschae [13]), etc.

We will also consider more general problems, where the objective will be the minimization of a linear combination of the energy and of some scheduling criterion, e.g. the *lateness* (Bampis et al. [8]).

For more details, the reader is invited to consult recent surveys on energy-efficient scheduling (Albers [1, 2], Bampis [5], Gerards et al. [10], Irani and Pruhs [12]).

References

[1] S. Albers, Energy efficient algorithms, *Communications of the ACM*, **53** (2010), 86–96.

[2] S. Albers, Algorithms for dynamic speed scaling, *Proceedings of the 28th International Symposium of Theoretical Aspects of Computer Science*, Dortmund, March 10–12, 2011, 1–11.

[3] S. Albers, E. Bampis, D. Letsios, G. Lucarelli, R. Stotz, Scheduling on power-heterogeneous processors, *Information and Computation*, **257** (2017), 22–33.

[4] E. Angel, E. Bampis, V. Chau, Throughput maximization in the speed-scaling setting, *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science*, Lyon, March 5–8, 2014, 53–62.

[5] E. Bampis, Algorithmic issues in energy-efficient computation, *Proceedings of the 9th International Conference on Discrete Optimization and Operations Research*, Vladivostok, September 19–23, 2016, 3–14.

[6] E. Bampis, A. V. Kononov, D. Letsios, G. Lucarelli, I. Nemparis, From preemptive to non-preemptive speed-scaling scheduling, *Discrete Applied Mathematics*, **181** (2015), 11–20.

[7] E. Bampis, D. Letsios, G. Lucarelli, A note on multiprocessor speed scaling with precedence constraints, *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, Prague, June 23–25, 2014, 138–142.

[8] E. Bampis, D. Letsios, I. Milis, G. Zois, Speed scaling for maximum lateness, *Theory of Computing Systems*, **58** (2016), 304–321.

[9] Ph. Baptiste, Scheduling unit tasks to minimize the number of idle periods: a polynomial time algorithm for offline dynamic power management, *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, Miami, January 22–24, 2006, 364–367.

[10] M. E. T. Gerards, J. L. Hurink, Ph. K. F. Hölzenspies, A survey of offline algorithms for energy minimization under deadline constraints, *Journal of Scheduling*, **19** (2016), 3–19.

[11] S. Irani, R. K. Gupta, S. K. Shukla, Competitive analysis of dynamic power management strategies for systems with multiple power savings states, *Proceedings 2002 Design, Automation and Test in Europe: Conference and Exhibition*, Paris, March 4–8, 2002, 117–123.

[12] S. Irani, K. Pruhs, Algorithmic problems in power management, *ACM SIGACT News*, **36** (2005), 63–76.

[13] N. Megow, J. Verschae, Scheduling on a machine with varying speed: Minimizing cost and energy via dual schedules, *Lecture Notes in Computer Science*, **7965** (2013), 745–756.

[14] F. F. Yao, A. J. Demers, S. Shenker, A scheduling model for reduced CPU energy, *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, Milwaukee, October 23–25, 1995, 374–382.

Tutorial

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

A tutorial on mechanism design in scheduling

Ruben Hoeksma *
University of Bremen, Bremen, Germany

Keywords: mechanism design, payment rule, Bayes-Nash equilibrium, agents, scheduling, single machine, total weighted completion time

1 Introduction

The analysis and design of auctions and mechanisms originated in early 1960s in the paper by Vickrey [11]. In particular, Vickrey studied single item auctions, which later became known as *Vickrey auctions*. Among the mayor early contributions are the works by Clarke [1] and Groves [3]. What we call today *optimal mechanism design* sees its foundation twenty years after Vickrey's work in the seminal paper by Myerson [7]. He studied revenue maximizing single item auctions for continuous, single-dimensional type spaces. The revenue optimal auction turns out to be one with an easy description: a second price auction with a reserve price. This celebrated result instigated a whole field of research.

The goal of this tutorial is to give an introduction to *mechanism design*, using scheduling as the leading example. We will see how the famous results by Myerson [7] for single item auctions translate to single machine scheduling with single-dimensional private information, where only the weight of a job is private, and how this extends to the setting with two-dimensional private information, where both the weight and the processing time are private.

We will consider the single machine scheduling problem, where a set N of n jobs $1, \ldots, n$ needs to be scheduled non-preemptively on a single machine. A job $j \in N$ has a processing time p_j and a weight w_j. We are interested in the setting where the owner of this single machine (the *mechanism designer*) sells the usage of the machine to different *agents* who each own a single job. (Hence, we identify jobs with agents.) The mechanism designer, applying a *scheduling rule*, decides the order in which the jobs are processed and, using a *payment rule*, subsequently reimburses the agents for their waiting time.

We assume that the price that each agent pays for the service is sufficiently larger than the cost that these agents incur because they have to wait for processing.

* Tutorial speaker, email: hoeksma@uni-bremen.de

However, the extra cost of waiting needs to be compensated by the mechanism designer. Therefore, we will consider the setting where payments are made by the machine owner to the agents. In view of several technical implications that we do not detail here, we assume that the agents' utility u_j of job $j \in N$, is determined solely not by the completion time but by the start time of the job, s_j. The payment made by the mechanism designer for $j \in N$ will be denoted by π_j. Then

$$u_j(s_j, \pi_j) = \pi_j - w_j s_j \,,$$

where the payment π_j should at least exceed the cost $w_j s_j$. Of course, the mechanism designer wants to minimize the payments made to the agents, while the agents wish to maximize their own utility. Thus, the mechanism designer has an incentive to minimize the agents' loss of utility. When all parameters are known to the mechanism designer, it is an easy task to find the schedule that minimizes $\sum_{j \in N} w_j s_j$. It is well known that scheduling the jobs in non-increasing order of $\frac{w_j}{p_j}$ achieves this goal (Smith [10]).

Here, however, we consider the situation where some of the parameters are private information. For each agent, this may consist of a single-dimensional private information such as the weight of a job or a two-dimensional private information such as both the weight and the processing time of a job. We refer to the set of private parameters of an agent j as the *type* of the agent, denoted t_j. We assume that the only way for the mechanism designer to gain information on the type of the agents is by asking them to reveal this information. The only choices available to the agents are which type they report. Note that they can report their type truthfully or misrepresent their type, in order to obtain a more beneficial outcome.

The mechanism designer now has to consider the strategic behavior of the agents, when deciding on the schedule and the payments.

2 Preliminaries

Let t_j be the type of a job j. We assume that, while the exact type of a job is only known to its agent, the mechanism designer and the other agents have a common knowledge of the distribution of job j's possible weights. Let $T_j = \{t_j^1, \ldots, t_j^{m_j}\}$ be the set of possible types of job j and let $\varphi_j(t_j^i)$ denote the probability that job j is of type i. Symbol $t = (t_1, \ldots, t_n)$ will represent the vector of types of all the jobs.

A *mechanism* is a pair (f, π) and consists of a scheduling rule f and a payment rule π. From the scheduling rule f, we can compute the start time $s_j^f(t)$ of a job $j \in N$, depending on the vector t of types reported by the agents. A *strategy* σ of

an agent is a function that indicates which type $\sigma(t_j)$ the agent reports if the true type of this agent is t_j.

We are interested in mechanisms that create *Bayes-Nash equilibria* in which the revenue of the machine owner is maximized. In a Bayes-Nash equilibrium all agents have a single combination of strategies such that none of them can unilaterally change their strategy to improve their expected utility, given the strategies of the other agents. Since in a Bayes-Nash equilibrium the agents care only about their expected utility, we can consider the utility function of agent j to be

$$u_j(t_j, t'_j) = E\pi_j(t'_j) - w_j(t_j)Es_j^f(t'_j)\,.$$

Here, $w_j(t_j)$ denotes the weight of job j if it is of type t_j, and $Es_j^f(t'_j)$ and $E\pi_j(t'_j)$ denote the *expected* start time of job j and *expected* payment made to agent j, respectively, if the agent reports type t'_j. The expectation is taken over the distribution of the types of the other jobs. Note that the utility of job j depends only on the type t_j and the reported type t'_j, not on the complete type vector t.

Myerson's revelation principle (cf. [7]) tells us that we can focus on mechanisms that are *truth telling*, i.e., mechanisms which create a Bayes-Nash equilibrium when the strategy of all agents is to report their type truthfully. This can be expressed with the following linear constraint for all j, t_j and t'_j:

$$E\pi_j(t_j) - w_j(t_j)Es_j^f(t_j) \geq E\pi_j(t'_j) - w_j(t_j)Es_j^f(t'_j)\,. \tag{1}$$

Inequality (1) says that no agent has an incentive to misrepresent their type, when all other agents report their type truthfully and is known as *Bayes-Nash incentive compatibility*. That our mechanism should reimburse the agents for their waiting cost is better known as *individual rationality*. This means that agents who tell the truth must have non-negative utility for all j and t_j:

$$E\pi_j(t_j) - w_j(t_j)Es_j^f(t_j) \geq 0\,. \tag{2}$$

3 Single-dimensional setting

In the case of the single-dimensional setting with private information, constraints (1) and (2), for a given scheduling rule f, can be interpreted as a graph. This idea goes back to a paper by Rochet [9] and works for discrete type spaces. For scheduling rule f, we construct the *type graph* of a job with a node for each type in T_j, with directed arcs (t_j^i, t_j^k) between all pairs of nodes t_j^i and t_j^k. The length of arc (t_j^i, t_j^k) is equal to the difference in cost for agent j if reports his true type t_j^i instead of

misrepresenting type t_j^k. Note that this does not take into account the payment rule and it does not depend on the distribution φ_j. Fig. 1 gives an example of a simple type graph for a job with just two types.

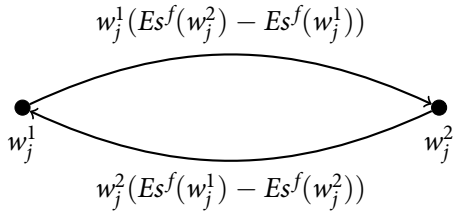


Figure 1: Type graph for a job with two types, w_j^1 and w_j^2

By rewriting (1), we see that it describes the difference in expected payments that should be made to agent j if he reports t_j or t'_j. Moreover, this difference is exactly the length of the arc between t_j and t'_j in the type graph. Thus we can interpret the type graph in such a way that it implies (1). Adding to the type graphs a sink, i.e. a dummy node d with $Es_j^f(d) = 0$, implies (2) within the type graph as well. Computing the length of the shortest paths from each node to the sink in the type graph gives us the minimal payments for a given scheduling rule f.

Among other things, the above implies that given a scheduling rule f there exists a feasible mechanism (f, π) if and only if there are no negative cycles in the type graph. This is true if and only if $Es_j^f(w_j) \geq Es_j^f(w'_j) \Leftrightarrow w'_j \leq w_j$. Moreover, it turns out that, for any such feasible f, the shortest paths are always the same. Namely, for $w_j^1 \geq w_j^2 \geq \ldots \geq w_j^{m_j}$, the shortest path from w_j^i to the sink goes through $w_j^{i+1}, \ldots, w_j^{m_j}$ in order of the indices. From this, after some additional computations, it is not hard to conclude the following result.

Theorem 1. (Duives et al. [2]) *Let* $\overline{w}_j^1 = w_j^1$ *and* $\overline{w}_j^i = w_j^i + (w_j^i - w_j^{i-1}) \frac{\sum_{k=1}^{i-1} \varphi_j(w_j^k)}{\varphi_j(w_j^i)}$ *for* $i = 2, \ldots, m_j$. *If regularity of weights holds, i.e. for any* j *inequality* $\overline{w}_j \geq \overline{w}'_j$ *holds iff inequality* $w_j \geq w'_j$ *holds, then the scheduling rule* f *in which jobs are scheduled in non-increasing order of* $\frac{\overline{w}_j}{p_j}$ *can be complemented by the payment rule* π *such that* (f, π) *satisfies* (1) *and* (2) *and minimizes the payments made to the agents.*

4 Two-dimensional setting

The two-dimensional setting with private information has both the weight and the processing time as information that is private to the agents. As opposed to the

single-dimensional problem described in the previous section, which was fairly easy to solve and even possessing results in a closed form, the two-dimensional problem is very different. The main difficulty lies in the fact that it is not clear how to define the utility of an agent, when the agent receives less processing time than the true processing time, e.g. when the agent reports a smaller processing time. This leads to two different models: constrained and unconstrained ones. In the *constrained model* only reporting processing times larger than the true processing time is allowed, while in the *unconstrained model* any processing time can be reported. It turns out that the unconstrained model leads to a very similar analysis as the single-dimensional model. Hence, we focus on the constrained model.

Though also for the constrained two-dimensional model a type graph can be constructed for each job, the shortest paths now differ depending on the scheduling rule. Hence, an approach similar to the one for the single-dimensional model is doomed to fail. Moreover, we know that in general an optimal scheduling rule for the two-dimensional case does not satisfy a condition called *independence of irrelevant alternatives*. This means that the order in which two jobs get scheduled may need to depend on the types of other jobs, instead of just the types of those two jobs. Thus, a scheduling rule based on a comparison of parameters can be excluded up front. Nevertheless, the following result has been proved recently.

Theorem 2. (Hoeksma and Uetz [5]) *The two-dimensional scheduling mechanism design problem can be solved in polynomial time.*

Instead of a closed form solution, we can construct for any instance of the two-dimensional scheduling mechanism design problem a linear program of polynomial size in the input that solves the problem exactly. Arriving at this polynomial size LP features several intricate techniques. Especially, it turns out that while the order of two jobs may need to depend on the types of other jobs, the expected start time of the jobs in the LP can be expressed without this dependency.

5 Further reading

For a more complete write-up on auctions and mechanism design, we direct the reader to the book by Krishna [6]. For a more computer science focused discussion, one may read the book by Nisan et al. [8] or the manuscript by Hartline [4].

References

[1] E. H. Clarke, Multipart pricing of public goods, *Public Choice*, **11** (1971), 17–33.

[2] J. Duives, B. Heydenreich, D. Mishra, R. Müller, M. Uetz, On optimal mechanism design for a sequencing problem, *Journal of Scheduling*, **18** (2015), 45–59.

[3] T. Groves, Incentives in teams, *Econometrica: Journal of the Econometric Society*, **41** (1973), 617–631.

[4] J. D. Hartline, Mechanism design and approximation, unpublished manuscript, `http://jasonhartline.com/MDnA/`, 2017.

[5] R. Hoeksma, M. Uetz, Optimal mechanism design for a sequencing problem with two-dimensional types, *Operations Research*, **64** (2016), 1438–1450.

[6] V. Krishna, *Auction Theory*, 2^{nd} ed., Academic Press, Cambridge, 2009.

[7] R. B. Myerson, Optimal auction design, *Mathematics of Operations Research*, **6** (1981), 58–73.

[8] N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani, *Algorithmic Game Theory*, Cambridge University Press, New York, 2007.

[9] J.-C. Rochet, A necessary and sufficient condition for rationalizability in a quasi-linear context, *Journal of Mathematical Economics*, **16** (1987), 191–200.

[10] W. E. Smith, Various optimizers for single-stage production, *Naval Research Logistics Quarterly*, **3** (1956), 59–66.

[11] W. Vickrey, Counterspeculation, auctions, and competitive sealed tenders, *The Journal of Finance*, **16** (1961), 8–37.

Extended abstracts

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

General inequalities in the set-dependent scheduling

Maksim Barketau *
National Academy of Sciences of Belarus, Minsk, Belarus

Keywords: set-dependent scheduling, convex programming, maximum completion time

1 Introduction

We consider the following scheduling problem. There are n jobs and m parallel machines, where m is a constant. The minimal processing time $p_i, 1 \leq i \leq n$, is given for each job. The preemption of jobs is allowed but only a finite number of job preemptions is possible. The solution of the problem is a feasible schedule for the jobs assigned to the parallel machines in an ordinary sense. A feasible schedule is a sequence of intervals with the lengths $\Delta_l, 1 \leq l \leq L$, where L is the number of the intervals. The assignment of the jobs to the machines is fixed for each interval and no more than one particular job is allowed for the processing on each machine during each interval. We denote the set of jobs that are assigned to the lth interval as J_l. The non-negative value $v(J)$ is given for each subset of jobs J of a cardinality not greater than m. The value of a feasible schedule is calculated as follows. The length of the lth interval is augmented by $v(J_l)\Delta_l$. The aim is to find a feasible schedule with the minimum augmented length, which is equivalent to the minimization of the makespan of the augmented schedule.

2 Conic inequalities

We consider a special system of conic inequalities that can be used for defining the above mentioned value function $v(J)$. This system is in the form of

$$X_i - X_j \geq_K A_{i,j}, i, j \in \{1, 2, \ldots, n\}, i \neq j, \tag{1}$$

where $X_i, 1 \leq i \leq n$, is a variable vector in a particular vector space and $A_{i,j}$ is a given constant vector in this vector space, K is a convex cone in that vector space and $\geq_K$ denotes a partial ordering in this space, induced by cone K. Examples of

* Speaker, email: barketau@mail.ru

the vector space are the real vector space R^m and the vector space of the symmetric matrices S^m. The convex cone can be the non-negative cone R^m_+, the second-order cone L^m or the semidefinite cone S^m_+.

Already in the case when each X_i, $1 \leq i \leq n$, is a real variable, system (1) may have no solution (see Cormen et al. [3]). A possible way to define $v(J)$ for a particular set J is to narrow system (1) to the system $X_i - X_j \geq_K A_{i,j}, i, j \in J, i \neq j$. If the latter system is feasible, then we set $v(J) = 0$, otherwise $v(J) = v$, where v is a constant. One can note that by narrowing a feasible system we always obtain a feasible system. Thus, all the subsets of the feasible subset J are feasible. Note that the job sets $\{j\}, j \in \{1, 2, \ldots, n\}$, always have zero augmented length.

Another way to set the possible value to a job set J is the following. Let $C \in K^*$, where K^* is a dual cone to the cone K, be a constant vector. Consider the following conic program:

$$\begin{aligned} &\min \max\{\max\{\langle C, X_i\rangle - \langle C, Y_i\rangle | i \in \{1, 2, \ldots, n\}\}, 0\} \\ &X_i - Y_j \geq_K A_{i,j}, i, j \in \{1, 2, \ldots, n\}, i \neq j, \end{aligned} \tag{2}$$

where $\langle X, Y\rangle$ is the inner product of vectors. In program (2) instead of vector variable X_i, $1 \leq i \leq n$, we introduce two vector variables X_i and Y_i. Now, we may set the value of a job set J equal to the optimal value of program (2), narrowed to the set J in the same way as we did with the system (1).

3 Our results

Program (2) has a lower bound and is strictly feasible, thus it has an optimal solution (Ben-Tal and Nemirovski [1]). One can use interior point methods for the solution of the program in case of the non-negative cone, the second-order cone or the semidefinite cone (Nemirovski [2]). The problem has quite a large number of constraints, namely $n(n-1)$. We propose two potentially faster approximate approaches for the problem, both having the same first stage.

The first stage. Arrange the jobs in some order and renumber them according to this order. Solve the following program and take the optimal solution with the minimal Euclidean distance from the vector X_1 to the vector Y_1 :

$$\begin{aligned} \min \max\{\max\{ &\langle C, Y_1 + A_{i,1}\rangle - \langle C, X_1 - A_{1,i}\rangle | i \in \{2, \ldots, n\}\}, \\ &\langle C, X_1\rangle - \langle C, Y_1\rangle, 0\}. \end{aligned}$$

Essentially, in the case of all considered vector spaces this problem reduces to the solution of the linear program with n linear inequality constraints plus taking a

point on a hyperplane and taking the nearest point on the parallel hyperplane. As a result of the first stage we find two vectors, X_1 and Y_1.

The second stage: the first heuristic. In the second stage of the first heuristic, we will sequentially look for the next pair of vectors with possible modification of the previous pairs of vectors.

Assume we have found the first k pairs of vectors $X_1, Y_1, X_2, Y_2, \ldots, X_k, Y_k$. Let us solve the following program to find the pair of vectors X_{k+1}, Y_{k+1}:

$$\begin{aligned}
&\min\langle C, X_{k+1}\rangle - \langle C, Y_{k+1}\rangle \\
&X_{k+1} - Y_j \geq_K A_{k+1,j}, 1 \leq j \leq k, \\
&X_i - Y_{k+1} \geq_K A_{i,k+1}, 1 \leq i \leq k.
\end{aligned}$$

Note that vectors X_{k+1} and Y_{k+1} are the only variables of this program, which is solvable and has no more than $2k$ constraints. If

$$\max\{\langle C, X_{k+1}\rangle - \langle C, Y_{k+1}\rangle, 0\} \leq \max\{\max\{\langle C, X_i\rangle - \langle C, Y_i\rangle | i \in \{1, \ldots, k\}\}, 0\},$$

then we simply fix vectors X_{k+1}, Y_{k+1} and repeat the second stage for the next index. Otherwise, let

$$\begin{aligned}
a &= \max\{\langle C, X_{k+1}\rangle - \langle C, Y_{k+1}\rangle, 0\}, \\
b &= \max\{\max\{\langle C, X_i\rangle - \langle C, Y_i\rangle | i \in \{1, \ldots, k\}\}, 0\}.
\end{aligned}$$

Let $X_i' := X_i + Ct$, $1 \leq i \leq k$, $Y_{k+1}' := Y_{k+1} + Ct$, $Y_i' = Y_i$, $1 \leq i \leq k$, $X_{k+1}' = X_{k+1}$, where $t = \frac{a-b}{2\langle C,C\rangle}$. It is easy to see that by adjusting variables this way, the constraints $X_i - Y_j \geq_K A_{i,j}, 1 \leq i, j \leq k+1$, remain feasible, while $\min\max\{\max\{\langle C, X_i\rangle - \langle C, Y_i\rangle | i \in \{1, 2, \ldots, k+1\}\}, 0\}$ decrease. Now, we set $X_i = X_i', Y_i = Y_i', 1 \leq i \leq k+1$, and repeat the second stage for the next index.

The second stage: the second heuristic. The second approach also sequentially sets the values of pairs of variables, but it tries to get into account the potential distance to each pair, not only to those that have been considered. The other important feature of the second approach is that it approximates the set R defined with the convex system $X \geq_K A_i, 1 \leq i \leq l$, with the set $X^{'} + K$, where $X^{'}$ is the vector with minimal value $\langle C, X\rangle$ from the set R. In fact, we have $X^{'} + K \subseteq R$ (Barketau and Pesch [4]).

At the beginning, set $X_1^1 = X_1$, $Y_1^1 = Y_1$, $X_i^1 = Y_1 + A_{i,1}$, $Y_i^1 = X_1 - A_{1,i}$, $2 \leq i \leq n$, where X_1 and Y_1 are the vectors obtained during the first stage. Assume we set k pairs of vectors and have the vectors $X_i^k, Y_i^k, 1 \leq i \leq n$. The first k pairs of them are the fixed vectors we have obtained from the first k iterations. In the

$(k+1)$st iteration, we solve the following program with the variables X_i, Y_i, where $k+1 \leq i \leq n$:

$$\begin{aligned}
&\min \max\{\max\{\langle C, X_i\rangle - \langle C, Y_i\rangle | i \in \{k+1, \ldots, n\}\}, 0\} \\
&X_i \geq_K X_i^k, k+1 \leq i \leq n, \\
&X_i \geq_K Y_{k+1} + A_{i,k+1}, k+2 \leq i \leq n, \\
&Y_i^k \geq_K Y_i, k+1 \leq i \leq n, \\
&X_{k+1} - A_{k+1,i} \geq_K Y_i, k+2 \leq i \leq n,
\end{aligned}$$

After linearizing the criterion function, we have a conic program with no more than $4(n-k)$ convex constraints plus no more than n linear constraints. After solving the system, we set $X_i^{k+1} = X_i^k$, $1 \leq i \leq k$, $Y_i^{k+1} = Y_i^k$, $1 \leq i \leq k$, $X_i^{k+1} = X_i$, $k+1 \leq i \leq n$, $Y_i^{k+1} = Y_i$, $k+1 \leq i \leq n$ and repeat the iteration.

4 Future research

In future research it would be of interest to compare the above heuristics.

References

[1] A. Ben-Tal, A. Nemirovski, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, SIAM, Philadelphia, 2001.

[2] A. Nemirovski, *Interior Point Polynomial Time Methods in Convex Programming*, lecture notes, `https://www2.isye.gatech.edu/~nemirovs/Lect_IPM.pdf`, spring semester 1996.

[3] T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press/McGraw-Hill, Cambridge-London, 2001.

[4] M. Barketau, E. Pesch, Looking for the interset distance, talk presented at *The 20th International Conference on Operations Research 2017*, Berlin, September 6-8, 2017.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

Scheduling non-preemptive data gathering affected by background communications

Joanna Berlińska *
Adam Mickiewicz University in Poznań, Poznań, Poland

Keywords: scheduling, data gathering networks, variable communication speed

1 Introduction

Scheduling in a data gathering network is usually analyzed under the assumption that the network performance is constant. For example, Choi and Robertazzi [4], Moges and Robertazzi [7] proposed algorithms for partitioning the total amount of gathered data between the network nodes, in order to minimize the makespan. Scheduling algorithms for gathering fixed amounts of data from the network nodes were proposed, e.g., by Berlińska [1, 2], Luo et al. [6]. However, real communication parameters of a network may change in time. Preemptive scheduling in data gathering networks with variable communication speed was studied by Berlińska [3]. This work considers non-preemptive scheduling in a data gathering network with performance affected by background communications.

2 Problem formulation

We study a star data gathering network that consists of m worker nodes $P_1, \ldots, P_m$ and a single base station P_0. Each worker P_i holds dataset D_i of size α_i, which has to be transferred to the base station in a single message. At most one node can communicate with the base station at a time. The communication rate, i.e. the inverse of speed, of the link between P_i and P_0 in an otherwise unloaded network is C_i. However, background communications required by other applications may degrade the link performance. We will be calling a link *loaded* if it is used by background communications, and *free* in the opposite case. We assume that the network implements QoS Percentage-Based Policing (Szigeti et al. [8]), hence the communication rate perceived by the analyzed data gathering application for a loaded link between P_i and P_0 is δC_i, for some fixed $\delta > 1$. Thus, the

* Speaker, email: Joanna.Berlinska@amu.edu.pl

maximum time that may be necessary to gather data from all worker nodes is $\overline{T} = \delta \sum_{i=1}^{m} C_i \alpha_i$. For each node P_i, we are given a set of n_i disjoint time intervals $[t'_{ij}, t''_{ij}]$, where $j = 1, \ldots, n_i$ and $t'_{ij} < \overline{T}$, in which the corresponding communication link is loaded. The total number of such intervals is $n_1 + \cdots + n_m = n$.

The scheduling problem is to choose the sequence of datasets sent consecutively to the base station, such that all data is transferred in the shortest possible time.

3 Complexity and algorithms

Let us first observe that transferring data of size x with communication rate C is equivalent to sending data of size Cx with communication rate 1. Thus, from now on we will assume without loss of generality that $C_i = 1$ for $i = 1, \ldots, m$.

We prove that the analyzed problem is strongly $\mathcal{NP}$-hard, using a pseudo-polynomial transformation from the 3-PARTITION problem (Garey and Johnson [5]). Then, we propose the following exponential-time dynamic programming algorithm. Let $\tau(D_i, t)$ be the time necessary to transfer dataset D_i, starting at moment t. For each subset $\mathcal{D} \subset \{D_1, \ldots, D_m\}$, we compute the shortest time $T(\mathcal{D})$ in which the datasets from $\mathcal{D}$ can be transferred to the base station, using the following formulas:

$$T(\mathcal{D}) = \begin{cases} 0 & \text{if } \mathcal{D} = \emptyset, \\ \min_{D_i \in \mathcal{D}} \{T(\mathcal{D} \setminus \{D_i\}) + \tau(D_i, T(\mathcal{D} \setminus \{D_i\})\} & \text{if } \mathcal{D} \neq \emptyset. \end{cases}$$

The minimum schedule makespan is $T(\{D_1, D_2, \ldots, D_m\})$, and the optimum dataset sequence can be easily tracked. This algorithm runs in $O((m+n)2^m)$ time.

Furthermore, we propose the following three greedy heuristic algorithms, each of which has $O(m(m+n))$ complexity.

1. Algorithm *gTime* always chooses to send the dataset that will be transferred in the shortest time.
2. Algorithm *gRate* selects the dataset that will be sent with the best average communication rate.
3. Algorithm *gSlowtime* chooses the dataset for which the time when data is transferred over a loaded link will be the shortest.

In all the three heuristics ties are broken by selecting a larger dataset. We also implement algorithm *Rnd*, which constructs a random dataset sequence.

4 Experimental results

The quality of solutions delivered by our greedy heuristics and algorithm *Rnd* was tested in computational experiments. We generated two groups of tests. In the *periodic* instances, for a link between P_i and P_0 we selected randomly the common length of its free intervals $f_i \in [1, F]$, and the common length of its loaded intervals $l_i \in [1, L]$. The link was loaded periodically, in intervals $[t'_{ij}, t''_{ij}] = [jf_i + (j-1)l_i, j(f_i + l_i)]$, for $j = 1, 2, \ldots n_i$. In the *random* tests, the lengths of all free intervals $f_{ij} \in [1, F]$, and the lengths of all loaded intervals $l_{ij} \in [1, L]$ for a link between P_i and P_0 were selected independently. The analyzed values of maximum lengths of free and loaded intervals were $F = 10, 30$, and $L = 5, 10, \ldots, 50$. We used $\delta = 2$, $m = 20$, and dataset sizes α_i were chosen randomly from the interval $[1, 20]$. For each tested setting, 100 instances were generated. Solution quality was measured by the ratio of the makespan delivered by a given heuristic to the optimum computed by the exact algorithm.

Table 1: Average solution quality for random instances

L	$F = 10$ *gTime*	$F = 10$ *gRate*	$F = 10$ *gSlowtime*	$F = 30$ *gTime*	$F = 30$ *gRate*	$F = 30$ *gSlowtime*
10	1.124	1.056	1.073	1.064	1.019	1.013
20	1.153	1.074	1.098	1.100	1.043	1.034
30	1.168	1.079	1.109	1.137	1.056	1.052

A subset of the obtained results can be found in Table 1. The complete results can be summarized as follows.

1. As expected, the quality of solutions delivered by all heuristics deteriorates with increasing L, and tests with big F are easier than those with small F.
2. It is easier to find good schedules for periodic instances than for the random ones, although the problem remains strongly $\mathcal{NP}$-hard in the periodic case.
3. The greedy heuristics obtain much better results than algorithm *Rnd*.
4. Algorithm *gTime* is significantly outperformed by *gRate* and *gSlowtime*.
5. For $F = 10$, algorithm *gRate* obtains better results than *gSlowtime*.
6. For $F = 30$, the results delivered by algorithms *gRate* and *gSlowtime* are very similar for periodic instances, and algorithm *gSlowtime* slightly outperforms algorithm *gRate* on random tests.

5 Future research

In the future, we want to analyze in more detail the subproblem with periodic background communications, and design for it a dedicated polynomial-time heuristic.

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

References

[1] J. Berlińska, Communication scheduling in data gathering networks with limited memory, *Applied Mathematics and Computation*, **235** (2014), 530–537.

[2] J. Berlińska, Scheduling for data gathering networks with data compression, *European Journal of Operational Research*, **246** (2015), 744–749.

[3] J. Berlińska, Scheduling data gathering with variable communication speed, *Proceedings of the 1st International Workshop on Dynamic Scheduling Problems*, Poznań, June 30–July 1, 2016, 29–32.

[4] K. Choi, T. G. Robertazzi, Divisible load scheduling in wireless sensor networks with information utility, *Proceedings of the IEEE International Performance Computing and Communications Conference 2008*, Austin, December 7–9, 2008, 9–17.

[5] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

[6] W. Luo, Y. Xu, B. Gu, W. Tong, R. Goebel, G. Lin, Algorithms for communication scheduling in data gathering network with data compression, *Algorithmica*, 2017, doi: 10.1007/s00453-017-0373-6, in press.

[7] M. Moges, T. G. Robertazzi, Wireless sensor networks: scheduling for measurement and data reporting, *IEEE Transactions on Aerospace and Electronic Systems*, **42** (2006), 327–340.

[8] T. Szigeti, C. Hattingh, R. Barton, K. Briley, *End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks*, 2nd ed., Cisco Press, Indianapolis, 2013.

Two matheuristics for problem $1|p_j = 1 + b_j t| \sum C_j$

Stanisław Gawiejnowicz *
Adam Mickiewicz University in Poznań, Poznań, Poland

Wiesław Kurc
Adam Mickiewicz University in Poznań, Poznań, Poland

Keywords: time-dependent scheduling, single machine, total completion time, Grey code, matheuristics

1 Introduction

We consider the following non-classical scheduling problem. A set of independent jobs $J_0, J_1, \ldots, J_n$ has to be scheduled non-preemptively on a single machine starting from time 0. The processing time of job J_j equals $p_j(t) = 1 + b_j t$, where $t \geq 0$ is the job starting time, deterioration rate $b_j > 0$ and $0 \leq j \leq n$. Let $\beta = (\beta_0, \beta_1, \ldots, \beta_n)$ denote the initial sequence of coefficients $\beta_j = 1 + b_j$ for $0 \leq j \leq n$, where $\beta_i \neq \beta_j$ whenever $i \neq j$ and $0 \leq i, j \leq n$, and let $\mathcal{P}(\beta)$ denote the set of all permutations of β. (Since each sequence $\sigma \in \mathcal{P}(\beta)$ corresponds to a schedule for the problem, any such a sequence can be identified with the schedule corresponding to the sequence.) Job completion times in a given schedule $\sigma \in \mathcal{P}(\beta)$ are as follows:

$$\begin{aligned} C_{[0]}(\sigma) &= 1, \\ C_{[j]}(\sigma) &= C_{[j-1]}(\sigma) + p_{[j]}(C_{[j-1]}(\sigma)) = 1 + \beta_{[j]} C_{[j-1]}(\sigma), \end{aligned}$$

where $1 \leq j \leq n$. The goal is to find $\sigma^\star \in \mathcal{P}(\beta)$ such that

$$\sum_{j=0}^{n} C_{[j]}(\sigma^\star) = \min_{\sigma \in \mathcal{P}(\beta)} \left\{ \sum_{j=0}^{n} C_{[j]}(\sigma) \right\},$$

where $C_{[j]}$ are defined as above. The problem, denoted as $1|p_j = 1 + b_j t| \sum C_j$, is a *time-dependent scheduling problem* (Agnetis et al. [1], Gawiejnowicz [2]), in which variable job processing times depend on when the jobs start. For this problem, we present a new necessary condition of schedule optimality (Gawiejnowicz and Kurc [3]) and two *matheuristics* (Maniezzo et al. [7]) which combine mathematical programming algorithms with local search techniques.

* Speaker, email: stgawiej@amu.edu.pl

2 Related research

Problem $1|p_j = 1 + b_j t| \sum C_j$ has been formulated by Mosheiov [8], its time complexity is unknown. There are known, however, some properties of the problem.

Theorem 1. (Mosheiov [8]) *Let $\sigma^\star \in \mathcal{P}(\beta)$ be an optimal schedule for problem $1|p_j = 1 + b_j t| \sum C_j$. Then (i) $\sigma^\star$ is V-shaped, (ii) the first job in $\sigma^\star$ is the job with the maximal deterioration rate, (iii) $\sigma^\star$ is symmetric starting from the second position.*

Theorem 1 gives us a necessary condition of optimality, it also decreases the number of possibly optimal schedules from $O(n!)$ to $O(2^n)$. Recently, a stronger necessary condition has been shown.

Theorem 2. (Gawiejnowicz and Kurc [3]) *Let $\sigma^\star$ be an optimal schedule for problem $1|p_j = 1 + b_j t| \sum C_j$. Then (i) $\sigma^\star$ is V-shaped, the minimal element in $\sigma^\star$ is $\sigma^\star_m$, where $1 < m < n$, and (ii) for any r and q such that $1 \leq r < m < q \leq n$ we have*

$$\Delta_1(r,q) = \sum_{j=1}^{q-2} \prod_{k=j}^{q-2} \sigma_k^\star - \sum_{i=q+1}^{n} \prod_{k=q+1}^{i} \sigma_k^\star \geq 0$$

and

$$\nabla_1(r,q) = \sum_{j=1}^{r-1} \prod_{k=j}^{r-1} \sigma_k^\star - \sum_{i=r+2}^{n} \prod_{k=r+2}^{i} \sigma_k^\star \leq 0.$$

Let $V_I(\beta)$ and $V_{II}(\beta)$ denote the sets of all $\sigma \in \mathcal{P}(\beta)$ satisfying Theorem 1 and Theorem 2, respectively. Let $u = \min_{1\leq i\leq n}\{\beta_i\}$ and $v = \max_{1\leq i\leq n}\{\beta_i\}$. Let $V_D(\beta)$ denote the set of all $\sigma = (\sigma_1, \ldots, \sigma_m, \ldots, \sigma_n) \in \mathcal{P}(\beta)$ such that σ is V-shaped and $m \in D$, where D is a set of indices (cf. Gawiejnowicz and Kurc [3]).

Theorem 3. (Gawiejnowicz and Kurc [3]) *Let $c(n) = \sqrt{\frac{2}{\pi n}}\, 2^n \left(1 + O\left(\frac{1}{n}\right)\right)$ and let u and v, $1 < u < v$, be defined as above, respectively. Then*

$$|V_{II}(\beta)| \leq |V_D(\beta)| \leq \left(1 + \frac{\log v - \log u}{\log v + \log u} n\right) \times c(n)$$

and, if v is sufficiently close to u, $|V_D(\beta)| \geq c(n)$.

Some algorithmic results for problem $1|p_j = 1 + b_j t| \sum C_j$ are also known: a variant of interior point method (Gawiejnowicz and Kurc [4]), a greedy algorithm (Gawiejnowicz et al. [5]), a heuristic (Kubale and Ocetkiewicz [6]), an FPTAS (Ocetkiewicz [9]). We refer the reader to Gawiejnowicz [2], Gawiejnowicz and Kurc [3] for reviews of these results.

3 Matheuristic H_1

Our first matheuristic, H_1, works as follows. Let $\sigma \in V_I(\beta) \subset \mathcal{P}(\sigma)$ be a V-shaped schedule with the minimal element σ_m, and let indices r and q be *m-conjugated*, i.e. r and q are such that $1 \leq r \leq q \leq n$ and after exchange of elements σ_r and σ_q schedule σ still is V-shaped. We can apply to σ *cyclic shift* $\overline{\sigma}^{q-r}$, obtaining a new schedule $e_L = \sigma(\sigma_r \leftarrow \sigma_q)$ such that $||C(e_L)||_1 < ||C(\sigma)||_1$ (see Gawiejnowicz and Kurc [3] for details). Alternatively, we can apply to σ *cyclic shift* $\underline{\sigma}^{q-r}$, obtaining a new schedule $e_R = \sigma(\sigma_r \rightarrow \sigma_q)$ such that $||C(e_R)||_1 < ||C(\sigma)||_1$. Since schedules e_L and e_R still are V-shaped with the minimal element σ_m, we can apply to them cyclic shifts again, etc.

Given an initial V-shaped sequence σ, matheuristic H_1 does assignment $u \leftarrow \sigma$ and repeats operations

$$
\begin{aligned}
R &\leftarrow \{\underline{e}^{q-r} : ||C(\underline{e}^{q-r})||_1 < ||C(u)||_1, 1 \leq r \leq m-1\},\\
e_R &\leftarrow \arg\min\{||C(e)||_1 : e \in R\},\\
L &\leftarrow \{\overline{e}^{q-r} : ||C(\overline{e}^{q-r})||_1 < ||C(u)||_1, m+1 \leq q \leq n\},\\
e_L &\leftarrow \arg\min\{||C(e)||_1 : e \in L\},\\
e &\leftarrow \arg\min\{||e_R||_1, ||e_L||_1\},\\
u &\leftarrow e
\end{aligned}
$$

until the moment when results start to become stabilized. After termination, H_1 returns a suboptimal V-shaped sequence $e = (e_1, \ldots, e_m, \ldots, e_n)$.

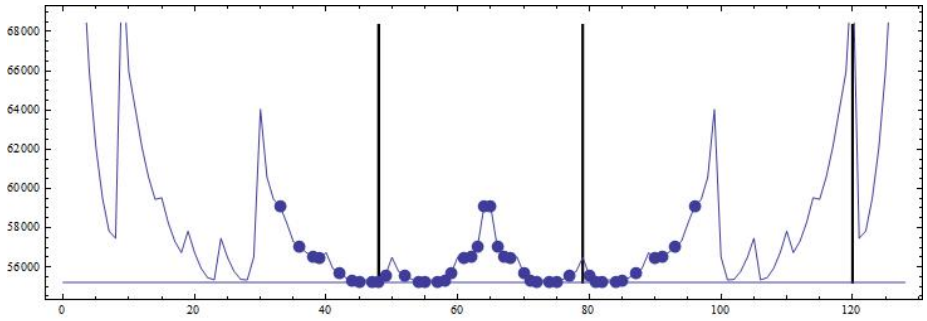


Figure 1: Example results of the execution of heuristic H_1

Fig. 1 depicts 128 V-shaped sequences from set $V_I(\beta^\circ) \subset \mathcal{P}(\beta^\circ)$ for sequence $\beta = (8, 7, 6, 5, 4, 2, 3)$. On the vertical axis there are given total completion times. Bold dots denote sequences satisfying Theorem 2, vertical lines indicate solutions constructed by H_1 in subsequent iterations.

4 Matheuristic H_2

Our second matheuristic, H_2, exploits the fact that looking for better schedules for our problem we deal with minimization over the set G of vertices $g_1, \ldots, g_{2^n}$ of hypercube $Q = [0,1]^n$, where $g_1 = (0, \ldots, 0, 0)$, $g_2 = (0, \ldots, 0, 1)$, ..., $g_{2^n} = (1, 0, \ldots, 0)$ denote (*binary reflected*) *Gray codes* (Savage [10]).

Given any vertex $g \in G$, adjacent vertices of g differ precisely at single position in n-tuples, similarly as in the case of consecutive Gray codes. Moreover, the vertices of G belong to a Hamiltonian path corresponding to a sequence of consecutive Gray codes. Finally, given any $\sigma \in \mathcal{P}(\beta)$, we can always assume that $\sigma = \sigma^{\downarrow}$, where $\sigma^{\downarrow}$ denotes schedule σ with decreasing order of elements.

We also can construct mapping $\gamma : G \to V$ between all vertices of G in Q and the set $V_I(\sigma^{\downarrow})$ of all V-shaped sequences built from $\sigma^{\downarrow}$. The γ mapping is defined as follows: given $g_i \in G$, $\gamma(g_i) = (\sigma^L|\sigma^R)$ is such a concatenation of σ^L and σ^R that $\sigma^L = \sigma^{\downarrow} \mid g_i^0$ is the subset of all elements of $\sigma^{\downarrow}$ corresponding to elements equal to 0 in g_i, while $\sigma^R = \sigma^{\downarrow} \mid g_i^1$ is composed of these elements in $\sigma^{\downarrow}$ which correspond to elements equal to 1 in g_i arranged non-decreasingly.

Example 4. Let $\sigma^{\downarrow} = (7, 5, 4, 2)$ and $g_3 = (0, 0, 1, 1)$. Then $\sigma^L = (7, 5)$ and $\sigma^R = (2, 4)$. Consequently, $\gamma(g_3) = ((7, 5)|(2, 4)) = (7, 5, 2, 4)$. If $g_1 = (0, 0, 0, 0)$, then $\gamma(g_1) = ((7, 5, 4, 2)|())$.

Heuristic H_2 is based on the following result (we omit a technical proof).

Lemma 5. *(i) Mapping $\gamma : G \to V_I(\sigma^{\downarrow})$ is an isomorphism between the set of vertices of hypercube Q and the set $V_I(\sigma^{\downarrow})$ of all V-shaped schedules constructed from a given sequence $\sigma^{\downarrow}$. (ii) To any two adjacent vertices in Q there correspond two V-shaped schedules in the form of $(\sigma^L|\sigma^R)$, obtained by a cyclic shift in which element from σ^L was moved to σ^R or vice-versa.*

Given $g = \gamma^{-1}(\sigma) \in Q$, matheuristic H_2 considers a broader neighborhood $U(g)$ of g and determines e such that $e = \arg\min\{||C(e)||_1 : e \in \gamma^{-1}(U(\sigma))\}$. Next, it considers a neighborhood of $U(\tau)$, where $\tau = \gamma^{-1}(e) \in Q$ and determines u such that $u = \arg\min\{||C(e)||_1 : e \in \gamma^{-1}(U(\tau))\}$, etc.

The main advantage of algorithm H_2 is that we can easier omit local minima. On the other hand, the broader set $U(g) \subset Q$ increases the time needed for finding $e = \arg\min\{||C(e)||_1 : e \in \gamma^{-1}(U(\sigma))\}$. Notice also that H_2 coincides with H_1, when $U(g) := \gamma^{-1}(R \cup L)$ for $g = \gamma^{-1}(\sigma) \in Q$. In both the algorithms, however, we can apply the same stop criterion.

5 Numerical experiment results

We have conducted a numerical experiment with matheuristics H_1 and H_2, testing sequences $(\beta_1, \beta_2, \ldots, \beta_n)$ with $n \in \{8, 10, 12\}$. For each n, we generated 100 instances for each of k iterations of H_1 and H_2, where $k = 4$ or $k = 8$. Integer components of each instance were generated randomly from interval $[1, 99]$ with uniform distribution.

Table 1: Results of numerical experiment with matheuristics H_1 and H_2

n	$err(H_1, 4)$	$err(H_1, 8)$	$err(H_2, 4)$	$err(H_2, 8)$
8	8.13229×10^{-6}	5.25047×10^{-6}	0.0	0.0
10	3.45228×10^{-6}	2.88174×10^{-6}	0.0	0.0
12	1.75432×10^{-6}	1.29454×10^{-6}	8.65080×10^{-11}	0.0

Results of the experiment are presented in Table 1, where $err(H_i, k)$ denotes an average relative error for k iterations.

6 Future research

In our experiment algorithm H_2 turned out to be significantly better than H_1, since in most of cases it generated an optimal schedule already for $k > 4$. However, since only instances with $n \leq 12$ jobs have been tested, more extensive experiments are needed in order to confirm this claim.

References

[1] A. Agnetis, J-C. Billaut, S. Gawiejnowicz, D. Pacciarelli, A. Soukhal, *Multiagent Scheduling*: *Models and Algorithms*, Springer, Berlin-Heidelberg, 2014.

[2] S. Gawiejnowicz, *Time-Dependent Scheduling*, Springer, Berlin-New York, 2008.

[3] S. Gawiejnowicz, W. Kurc, A new necessary condition of optimality for a time-dependent scheduling problem, *Proceedings of the 13th Workshop on Models and Algorithms for Planning and Scheduling Problems*, Seeon-Sebruck, June 12–16, 2017, 177–179.

[4] S. Gawiejnowicz, W. Kurc, Solving a time-dependent scheduling problem by interior point method, *Proceedings of the 1st International Workshop on Dynamic Scheduling Problems*, June 30-July 1, Poznań, 2016, 33–36.

[5] S. Gawiejnowicz, W. Kurc, L. Pankowska, Analysis of a time-dependent scheduling problem by signatures of deterioration rate sequences. *Discrete Applied Mathematics*, **154** (2006), 2150–2166.

[6] M. Kubale, K. M. Ocetkiewicz, A new optimal algorithm for a time-dependent scheduling problem, *Control & Cybernetics*, **38** (2009), 713–721.

[7] V. Maniezzo, T. Stützle, S. Voß (eds.), *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, Springer, Berlin-Heidelberg, 2010.

[8] G. Mosheiov, V-shaped policies in scheduling deteriorating jobs, *Operations Research*, **39** (1991), 979–991.

[9] K. Ocetkiewicz, A FPTAS for minimizing total completion time in a single machine time-dependent scheduling problem, *European Journal of Operational Research*, **203** (2010), 316–320.

[10] C. Savage, A survey of combinatorial Gray codes, *SIAM Review*, **39** (1997), 605–629.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

FPTASes for minimizing makespan of deteriorating jobs with non-linear processing times

Nir Halman *
Hebrew University of Jerusalem, Jerusalem, Israel

Keywords: time-dependent scheduling, non-linear processing times, FPTAS

1 Introduction

There are n independent jobs that need to be processed by a single machine. Each job J_j has basic processing time $p_j, j = 1, \ldots, n$. The jobs have a given common critical date d, after which they start to deteriorate and a common maximum deteriorating date D ($D > d$) after which they deteriorate no further. The actual processing time $a_j(t)$ of J_j depends on its start time t in the following way:

$$a_j(t) = \begin{cases} p_j, & \text{if } t \leq d, \\ p_j + w_j(\min\{t, D\} - d), & \text{otherwise,} \end{cases}$$

where $w_j \geq 0$ is the deteriorating rate of J_j. If $D < \infty$, the deterioration is called *bounded*, which is the case handled by Kovalyov and Kubiak [5]; otherwise – as considered by Cai et al. [1], Kovalyov and Kubiak [4] it is called *unbounded*. We assume that d, D, and all p_j and w_j are integral for $j = 1, \ldots, n$. If we set $L = \max_j\{p_j, w_j, D\}$ (or $L = \max_j\{p_j, w_j, d\}$ in the unbounded case) to be the largest numerical parameter in the input, then the problem (binary) size is $O(n \log L)$. The problem is to schedule the jobs to minimize the makespan, i.e., the completion time of the last job in the schedule. We shall denote this problem by Det. We shall assume that $\sum_{j=1}^{n} p_j > d$. Otherwise all jobs can start by d and the problem becomes trivial. Furthermore, there is no machine idle time in any optimal schedule. Under these conditions, there is a unique job for any given schedule that starts by d and completes after d. We call such a job *pivotal*. Any job that ends by d is called *early*. Any job that starts after d and ends by (after) D is called *tardy* (respectively, *suspended*).

Cai et al. [1] developed an $O(\frac{n^6}{\epsilon^2} \log^2 L)$ FPTAS for the unbounded case. Kovalyov and Kubiak [4] derived an $O(\frac{n^5}{\epsilon^3} \log^4 L)$ FPTAS for the unbounded case, which is faster than the FPTAS of [1] for $n >> \frac{\log^2 L}{\epsilon}$. Note that the authors erroneously

* Speaker, email: halman@huji.ac.il

claim their FPTAS for the bounded case (see [5, Sect. 4]). Kovalyov and Kubiak [5] addressed the bounded case and developed an $O(\frac{n^6}{\epsilon^4}\log^5 L)$ FPTAS.

In this paper, we give a simple $O\left(\frac{n^4}{\epsilon^2}\log^3 L \log\frac{n\log L}{\epsilon}\right)$ FPTAS for the unbounded case. Compared to the FPTAS by Cai et al. [1] it is faster by a factor of $\frac{n^2}{\log L}$, up to log terms (i.e., our algorithm is faster as long as $L < 2^{n^2}$). Compared to the FPTAS of Kovalyov and Kubiak [4], our FPTAS is faster by a factor of $\frac{n\log L}{\epsilon}$, up to log terms. For the bounded case, we give an $O\left(\frac{n^3\log^2 L\log(nL)}{\epsilon^2}\log\frac{n\log(nL)}{\epsilon}\right)$ FPTAS with a speedup of a factor of $\frac{n^3\log^2 L}{\epsilon^2}$, up to log terms.

2 K-approximation sets and functions

In this section, we provide an overview of the technique of *K-approximation sets and functions*, the tool we use to construct an FPTAS for DET.

Related research. Halman et al. [3] have introduced the technique of K-approximation sets and functions, and used it to develop an FPTAS for a certain stochastic inventory control problem. Halman et al. [2] have applied this tool to develop a framework for constructing FPTASes for three rather general classes of stochastic DPs. This technique has been used to yield FPTASes to various optimization problems (cf. [2]).

Notation. For any pair of integers $A \le B$, let $\{A,\ldots,B\}$ denote the set of integers between A and B. For a function $\varphi : \{A,\ldots,B\}\to\mathbb{R}$ that is not identically zero we denote $\varphi^{\min} := \min_{A\le x\le B}\{|\varphi(x)| : \varphi(x)\neq 0\}$, and $\varphi^{\max} := \max_{A\le x\le B}\{|\varphi(x)|\}$. Let t_φ be the time needed to calculate $\varphi(x)$, for any x.

K-approximation sets and functions. Let $K \ge 1$ and $\varphi,\tilde{\varphi} : \{A,\ldots,B\}\to\mathbb{R}^+$ be arbitrary functions. We say that $\tilde{\varphi}$ is a *K-approximation function of* φ if $\varphi(x) \le \tilde{\varphi}(x) \le K\varphi(x)$ for all $x = A,\ldots,B$. The following property (cf. [2, Prop. 5.1]) provides a set of general computational rules of K-approximation functions.

Property 1. ([2, Proposition 5.1]) *Let $K_i \ge 1$, let $\varphi_i,\tilde{\varphi}_i : \{A,\ldots,B\}\to\mathbb{R}^+$ and let $\tilde{\varphi}_i$ be a K_i-approximation of φ_i for $i = 1,2$. Let $\psi_1 : \{A',\ldots,B'\}\to\{A,\ldots,B\}$ be an arbitrary function and $\alpha,\beta\in\mathbb{R}^+$ be arbitrary positive reals. Then the following properties hold: (i) Linearity of approximation: $\alpha+\beta\tilde{\varphi}_1$ is a K_1-approximation function of $\alpha+\beta\varphi_1$; (ii) Summation of approximation: $\tilde{\varphi}_1+\tilde{\varphi}_2$ is a $\max\{K_1,K_2\}$-approximation function of $\varphi_1+\varphi_2$; (iii) Composition of approximation: $\tilde{\varphi}_1(\psi_1)$ is a K_1-approximation of $\varphi(\psi_1)$; (iv) Minimization of approximation: $\min\{\tilde{\varphi}_1,\tilde{\varphi}_2\}$ is a $\max\{K_1,K_2\}$-approximation of $\min\{\varphi_1,\varphi_2\}$; (v) Approximation of approximation: if $\varphi_2=\tilde{\varphi}_1$, then $\tilde{\varphi}_2$ is a K_1K_2-approximation function of φ_1.*

As DET has a non-increasing monotone structure over intervals of integers, we concentrate on definitions for K-approximation sets and functions specialized to non-increasing functions over intervals of integers (cf. [2]).

Definition 2. ([2, Definition 4.4]) *Let $\varphi : \{A, \ldots, B\} \to \mathbb{R}$ be a non-increasing function. For any subset $W \subseteq \{A, \ldots, B\}$ satisfying $A, B \in W$, the* approximation of φ induced by W *is the function $\hat{\varphi}(x) = \varphi(\max_{y \in W}\{y \leq x\}),\ \forall x \in \{A, \ldots, B\}$.*

Definition 3. ([2, Definition 4.2 and Proposition 4.5]) *Let $K \geq 1$ and let $\varphi : \{A, \ldots, B\} \to \mathbb{R}^+$ be a non-increasing function. Let $W \subseteq \{A, \ldots, B\}$ be such that $A, B \in W$ and let $\hat{\varphi}$ be the approximation of φ induced by W. We say that W is a* K-approximation set of φ *if for every $x \in \{A, \ldots, B\}$, we have $\hat{\varphi}(x) \leq K\varphi(x)$.*

The above two definitions tell us that the approximation of φ induced by a K-approximation set of φ is a K-approximation function of φ.

Proposition 4. ([2, Proposition 4.6]) *Let $\varphi : \{A, \ldots, B\} \to \mathbb{R}^+$ be a non-increasing function. Then for every $K > 1$, it is possible to compute a K-approximation set of φ of size $O(\log_K \frac{\varphi^{\max}}{\varphi^{\min}})$ in $O(t_\varphi(1 + \log_K \frac{\varphi^{\max}}{\varphi^{\min}}) \log(B - A))$ time.*

A procedure for the construction of a K-approximation function for function $\varphi : \{A, \ldots, B\} \to \mathbb{R}^+$ is stated as Algorithm 1, giving a pseudo-code of function COMPRESS which returns a non-increasing K-approximation of φ.

Algorithm 1 FUNCTION COMPRESS($\varphi, \{A, \ldots, B\}, K$)

1: find a K-approximation set W of φ over domain $\{A, \ldots, B\}$
2: **return** the approximation of φ induced by W as an array $\{(x, \tilde{\varphi}) \mid x \in W\}$ sorted in increasing order of x

By applying the calculus of approximation and the discussion above, we get the following result (see also [2, Proposition 4.5]).

Proposition 5. *Let $K_1, K_2 \geq 1$ be real numbers and let $\varphi : \{A, \ldots, B\} \to \mathbb{R}^+$ be a non-increasing function. Let $\bar{\varphi}$ be a non-increasing K_2-approximation function of φ. Then* COMPRESS*$(\bar{\varphi}, \{A, \ldots, B\}, K_1)$ returns in $O(t_\varphi(1 + \log_{K_1} \frac{\varphi^{\max}}{\varphi^{\min}}) \log(B - A))$ time a non-increasing step function $\tilde{\varphi}$ with $O(\log_{K_1} \frac{\varphi^{\max}}{\varphi^{\min}})$ steps that K_1K_2-approximates φ, and of which the query time is $t_{\tilde{\varphi}} = O(\log\log_{K_1} \frac{\varphi^{\max}}{\varphi^{\min}})$.*

3 A DP formulation

Let DET(k, s), with $d \leq s \leq d + p_k$, denote a DET problem in which the pivotal job is J_k and the start time of the earliest tardy job is s. Kovalyov and Kubiak [4] noticed that there exists $1 \leq k^* \leq n$ and s^*, where $s_V^{k^*} := \max\{d + 1, p_{k^*}\} \leq$

$s^* \le s_U^{k^*} := d + p_{k^*}$, such that any optimal solution to $\textsc{Det}(k^*, s^*)$ is an optimal solution to Det. Thus, Det reduces to solving $N^* := \sum_{k=1}^{n}(s_U^k - s_V^K + 1)$ problems $\textsc{Det}(k,s),\ s = s_V^K, \ldots, s_U^K,\ k = 1, \ldots, n$. Note that $N^* \sim nd = O(nL)$ is pseudo-polynomial in the input size, so solving (or even approximating) all N^* $\textsc{Det}(k,s)$ problems and taking the best solution results in a pseudo-polynomial algorithm. Therefore, Kovalyov and Kubiak [4] turn to approximating only a number of $\textsc{Det}(k,s)$ problems that is logarithmic in N^*.

Theorem 6. ([4, Lemma 2]) *The best solution in the family of* $(1+\frac{\epsilon}{3})$*-approximate solutions for* $\textsc{Det}(k, d + (\max\{d+1, p_k\} - d)(1+\frac{\epsilon}{3})^{i_k}),\ k = 1, \ldots, n,$ $i_k = 1, \ldots, 1 + \frac{3\log p_k}{\epsilon}$*, is a* $(1+\epsilon)$*-approximate solution for* Det.

Thus, by Theorem 6, all we need to do in order to obtain an FPTAS for Det is to design an FPTAS for $\textsc{Det}(k,s)$. Kovalyov and Kubiak [4, Theorem 1] design an FPTAS for $\textsc{Det}(k,s)$ with running time $O(n^4 \frac{\log^3 L}{\epsilon^2})$, resulting in an FPTAS for Det with running time $O(n^5 \frac{\log^4 L}{\epsilon^3})$.

To develop our FPTAS, we first show how to compute the makespan of a schedule, with pivotal job J_k and the earliest tardy job starting at s, using a set of simple recursive equations similar to the ones given in [4, page 291]. The equations will be developed for the indexing of jobs as follows. Let us index all jobs, except for the pivotal job J_0, according to Smith's rule so that $\frac{p_1}{w_1} \le \ldots \le \frac{p_{n-1}}{w_{n-1}}$ (Smith [7]). Kubiak and van de Velde [6] observe that there exists an optimal schedule, where early jobs are sequenced arbitrarily followed by the pivotal job, which in turn is followed by tardy and suspended jobs, if any. The former are sequenced in increasing order of their indices, the latter's order is arbitrary. We formulate our recursive equations for $\textsc{Det}(k,s)$ as follows. Let $M = \sum_{j=0}^{n-1} p_j + (D-d)\sum_{j=1}^{n-1} w_j = O(nL^2)$ be an upper bound on the makespan of any schedule with no idle time in the bounded case, and let $M = O(nL^n)$ be such an upper bound in the unbounded case. Denote collectively the following definitions as (1):

$$
\begin{aligned}
&f_0(y) = s,\ g_0(y) = s - d, && y = 0, \ldots, s - p_0,\\
&f_j(y) = \begin{cases} f_{j-1}(y) + p_j + w_j g_{j-1}(y), & \text{if } 0 \le y < p_j,\\ \min\{f_{j-1}(y - p_j), f_{j-1}(y) + p_j + w_j g_{j-1}(y)\}, & \text{if } y = p_j, \ldots, s - p_0, \end{cases}\\
&g_j(y) = \min\{f_j(y), D\} - d, && y = 0, \ldots, s - p_0.
\end{aligned}
$$

The problem is a knapsack-type problem, where the items considered to be placed in the knapsack are the jobs, the knapsack size is the time $s - p_0$ to schedule early jobs, item i size is p_i, the cost of placing an item in the knapsack is zero (it is an early job), and the cost of not placing an item in the knapsack is the time to process it as either a tardy or a suspended job. Then, $f_j(y)$ is the makespan of an optimal schedule of jobs $0, \ldots, j$, where at most y time is allocated to early jobs.

If job j is scheduled early, then it does not change the makespan, but it decreases the time allocated for the remaining early jobs, i.e., $f_j(y) = f_{j-1}(y - p_j)$. Then, $g_{j-1}(y)$ is the delay of job j when its starting time is the makespan of an optimal schedule of jobs $0, \ldots, j-1$ with at most y time allocated to early jobs. In this way, if job j is scheduled as either tardy or suspended, then it finishes at $f_j(y) = f_{j-1}(y) + p_j + w_j g_{j-1}(y)$. Note that the makespan of $\textsc{Det}(k, s)$ equals $f_{n-1}(s - p_0)$.

4 An FPTAS for the bounded case

Now, based upon (1), we are ready to state and analyze the FPTAS for $\textsc{Det}(k, s)$.

Algorithm 2 Function $\textsc{SolveDet}(k, s, \epsilon)$

1: Let $K \leftarrow \sqrt[n-1]{1+\epsilon}$, $\tilde{f}_0(\cdot) \equiv s$
2: **for** $j = 1$ to $n-1$ **do**
3: $\quad \bar{f}_j(y) \leftarrow \begin{cases} \tilde{f}_{j-1}(y) + p_j + w_j(\min\{\tilde{f}_{j-1}(y), D\} - d), & \text{if } 0 \le y < p_j, \\ \min\{\tilde{f}_{j-1}(y - p_j), \tilde{f}_{j-1}(y) + p_j + w_j(\min\{\tilde{f}_{j-1}(y), D\} - d)\}, & \text{if } y \ge p_j, \end{cases}$
4: $\quad \tilde{f}_j(\cdot) \leftarrow \textsc{Compress}(\bar{f}_j(\cdot), \{0, \ldots, s - p_0\}, K)$ /* $\bar{f}_j(\cdot)$ as defined in line 3 */
5: **return** $\tilde{f}_{n-1}(s - p_0)$

Theorem 7. *Function* $\textsc{SolveDet}(k, s, \epsilon)$ *computes a* $(1+\epsilon)$*-approximation for* $\textsc{Det}(k, s)$ *in* $O\left(\frac{n^2}{\epsilon} \log L \log(nL) \log \frac{n \log(nL)}{\epsilon}\right)$ *time.*

Proof. We note first that all calls to Compress are well defined, because $\bar{f}_j(\cdot)$ are non-increasing functions.

We start by analyzing the error bound. We show by induction that $\tilde{f}_j(\cdot)$ is a K^j-approximation of $f_j(\cdot)$, $j = 0, \ldots, n-1$. The base case for $j = 0$ holds true due to the initialization of (1) and Step 1 of Algorithm 2. We assume by induction correctness for $m-1$, i.e., $\tilde{f}_{m-1}(\cdot)$ is a K^{m-1}-approximation of $f_{m-1}(\cdot)$, and prove that $\tilde{f}_m(\cdot)$ is a K^m-approximation of $f_m(\cdot)$. When the value of the index of the **for** loop is m, we get by the induction hypothesis and the calculus of approximation that $\bar{f}_m(y)$ is a K^{m-1}-approximation of $f_m(\cdot)$. Calling Compress in Step 4, by Proposition 5 with parameters set to $K_1 = K$ and $K_2 = K^{m-1}$, we get that $\tilde{f}_j(\cdot)$ is a K^m-approximation of $f_j(\cdot)$ as needed.

Now, we analyze the running time. First, note that no actual computation is involved in Step 3, because we did not fix a value of y yet, but including this step in the algorithm helps us for the analysis. Due to Proposition 5, the running time of Step 4 is $O\big(t_{\tilde{f}_m} \log_K M \log(s - p_0)\big)$. By the definition in Step 3, we get that $t_{\tilde{f}_m} = O(t_{\bar{f}_m}) = O(\log \log_K M)$, where the second equality is due to Proposition 5. Moving to base 2 logarithm, substituting M with nL^2 and $s - p_o$ with L, using the

equation $\log K = \log \sqrt[n-1]{1+\epsilon} = O(\frac{\epsilon}{n})$ and taking into account that there are n iterations, the claimed running time follows. □

Remark. Regarding line 1 in Algorithm 2, we assume (by setting the rounding mode of the floating point unit to "round down", so we get an overestimate) the root operation takes constant time. Alternatively, we can set $K = 1 + \epsilon/2(n-1)$ and proceed as in the proof of [2, Theorem 8.2].

5 An FPTAS for the unbounded case

In this case, the upper bound on the makespan of any schedule with no idle time turns to $M = O(nL^n)$, instead of $M = O(nL^2)$, so the $\log(nL)$ terms in the running time of SolveDet for the bounded case turn to $O(n \log L)$. Therefore, we get for SolveDet the running time of $O\left(\frac{n^3}{\epsilon} \log^2 L \log \frac{n \log L}{\epsilon}\right)$, and the resulting FPTAS for Det runs in $O\left(\frac{n^4}{\epsilon^2} \log^3 L \log \frac{n \log L}{\epsilon}\right)$ time.

Acknowledgement. The author is grateful for partial support from the Israel Science Foundation (Grant 399/17).

References

[1] J-Y. Cai, P. Cai, Y. Zhu, On a scheduling problem of time deteriorating jobs, *Journal of Complexity*, **14** (1998), 190–209.

[2] N. Halman, D. Klabjan, C.-L. Li, J. Orlin, D. Simchi-Levi, Fully polynomial time approximation schemes for stochastic dynamic programs, *SIAM Journal on Discrete Mathematics*, **28** (2014), 1725–1796.

[3] N. Halman, D. Klabjan, M. Mostagir, J. Orlin, D. Simchi-Levi, A fully polynomial time approximation scheme for single-item stochastic inventory control with discrete demand, *Mathematics of Operations Research*, **34** (2009), 674–685.

[4] M. Y. Kovalyov, W. Kubiak, A fully polynomial approximation scheme for minimizing makespan of deteriorating jobs, *Journal of Heuristics*, **3** (1998), 287–297.

[5] M. Y. Kovalyov, W. Kubiak, A generic FPTAS for partition type optimisation problems, *International Journal of Planning and Scheduling*, **1** (2012), 209–233.

[6] W. Kubiak, S. L. van de Velde, Scheduling deteriorating jobs to minimize makespan, *Naval Research Logistics*, **45** (1998), 511–523.

[7] W. E. Smith, Various optimizers for single-stage production, *Naval Research Logistics Quarterly*, **3** (1956), 59–66.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

A dynamic scheduling approach to internal hospital logistics

Farzaneh Karami *
KU Leuven, Leuven, Belgium

Wim Vancroonenburg **
KU Leuven, Leuven, Belgium and
Research Foundation Flanders – FWO Vlaanderen

Greet Vanden Berghe
KU Leuven, Leuven, Belgium

Keywords: dynamic pickup and delivery, internal hospital logistics, heuristics

1 Introduction

This study is motivated by the problem of optimizing internal logistic flows in hospitals (Vancroonenburg et al. [5]). *Internal hospital logistics* (*IHL*) include assigning, routing and scheduling decisions, in which pickup and delivery problems with time windows (PDPTWs) appear on a daily basis. IHL require patients, medicine or medical equipment to be transported between different locations in the hospital. These transportation requests must be handled while respecting their associated time constraints, and may be interpreted as scheduling tasks to a fleet of porters while minimizing tasks' lateness, porters' overtime and travel time.

One important variant of the PDPTW is the *dynamic* PDPTW in which requests are unknown in advance and instead arrive as time progresses. In an IHL context, requests are inherently dynamic given that not all requests' locations and time windows are known at the beginning of the planning horizon.

In the paper, we propose a *rolling-horizon-based* dynamic optimization approach which considers a buffering strategy for the dynamic PDPTW and seeks to iteratively group together sets of arrival requests and allocate them to available porters. Re-optimization is performed with respect to the input which consists of only those requests known at that particular moment in time and utilizes a two-step heuristic-based scheduling algorithm.

* Speaker, email: farzaneh.karami@cs.kuleuven.be
** A postdoctoral research fellow at Research Foundation Flanders – FWO Vlaanderen

2 Problem formulation

To model IHL in a PDPTW context, its elements must first be introduced. Based on the work by Vancroonenburg et al. [5], *locations* in the hospital and its layout are represented by an undirected graph $G(V, A)$. V is the set of vertices corresponding to all locations. A is the set of edges connecting locations and junctions inside hospital. d_{ij} is the time it takes a porter to travel from location $v_i \in V$ to location $v_j \in V$.

A *request* represents a transport order and is denoted by $r = (i, j) \in R$, where R denotes the set of all requests and $i, j \in T$, with T representing the set of all tasks. A request consists of a pickup task i at location $v_i \in V$ and a delivery task j at location $v_j \in V$ and is announced at time a_r, before which it is unknown to the system. In order to schedule and plan routes of tasks servicing, it is necessary to consider each task' time windows, which is a time slot in the planning horizon. A semi-soft time window $[st_i, dt_i)$ is specified for each task i, signifying that the task may not be scheduled before the start of its corresponding time window; however, it can be scheduled after the end of this time window. A task's service time is denoted by p_i. Finally, each request has a special type corresponding to its transportation facility, for example 'bed', 'walking' or 'wheelchair'.

All *porters* are associated with an availability time window $[st_k, dt_k)$ during which they execute tasks. Porters begin and conclude their workday at a certain depot $v_k \in V$. The last-known location of each porter is the beginning of each route, what increases the complexity of the model depending on the number of porters. Each porter has a certain set of skills and each request may require its associated porter to have that skill in order to perform in.

A *forbidden combination* is a set of requests $\{r, \ldots, r'\}$ which cannot be performed simultaneously by a single porter. The combination of 'wheelchair' and 'bed' is an example of a forbidden request combination.

If a request r has *precedence* over request r', and both of them are assigned to the same porter, then the delivery of r must be executed prior to the pickup of r'. A very common example of this is that the transportation of a wheelchair to a patient's room must occur before the transportation of the patient themselves.

Some tasks may be combined together, for example when a nurse is walking with a patient, they may also carry certain medicine. Such task combinations may be permitted in some cases while prohibited in other cases.

The weighted objective function which must be minimized is the sum of requests' tardiness, porters' overtime and travel time. For each task i, the difference between

its completion time Ct_i and its time window end is called tardiness and equals $l_i = Max\{0, Ct_i - dt_i\}$. A porter's overtime is the completion time of their last request performed minus the end of their availability time window (or 0, if negative). For each porter, travel time corresponds to the total time it takes to travel between pickup/delivery locations in addition to their travel time to/from the depot.

In the context of IHL, routing corresponds to selecting appropriate paths for porters to travel along, while scheduling corresponds to assigning tasks to the porters. The total travel time largely depends on the hospital's size, whereas tardiness largely depends on the time windows of tasks. With the same objective weight, if travel times are larger than tardiness, then requests with urgent time windows may still be scheduled far in the future, thereby improving routing. By assigning proper weights in the objective function, it is possible to emphasize either the routing (minimizing the travel time) or the scheduling (minimizing the lateness).

3 Related research

Warren et al. [6] develop guidelines for inter- and intra-hospital transportations. Beaudry et al. [1] demonstrate an application of the dynamic dial-a-ride problem in terms of organizing intra-hospital patient transport in a German hospital. Hanne et al. [3] focus on transportations between different buildings within a hospital site with more than 100 individual buildings. Their research ultimately resulted in the development of the Opti-TRANS system. Fiegl and Pontow [2] presented an algorithm which dynamically schedules pickup and delivery tasks within hospitals, focusing on minimizing the average weighted flow time.

4 Our results

Real-word dynamic applications may benefit by applying more advanced scheduling algorithms. Due to the dynamic nature of these applications the scheduling algorithm must be implemented in a dynamic optimization framework.

In this study, we present a MILP model for scheduling IHL. Next, we propose a dynamic optimization method which employs two-step heuristic scheduling algorithms. A computational study on different problem instances using the proposed approach will be presented at the workshop. Results on real-life and generated hospital instances which reveal how urgency levels and dynamism degrees of request arrivals impact cost will be presented. Solutions of the proposed approach will be evaluated both locally (at each re-optimization step) and globally (over

the entire planning horizon) against the MILP solutions. In addition, the relative local and global gaps of the heuristic scheduling algorithm to the best found MILP solution for which all information is known in advance are also evaluated.

MILP model. The IHL is formulated as a mixed integer linear programming (MILP) model for the PDPTW introduced by Savelsberg and Sol [4]. First indices, sets, parameters, decision variables and the model formulation are defined.

Indices: i, j: Pickup/delivery task at a specific location. k: A specific porter. t: A request type.

Sets: R: The set of all requests. RT: The set of request types. T: The set of tasks. T^p, T^d: The set of pickup/delivery tasks and $T = T^p \cup T^d$. P: The set of porters. P_i: The set of porters with required skills for task $i \in T$. T_k: The set of tasks which may be assigned to porter k. $Prec$: The set of precedences between tasks. F: The set of forbidden request pairs (t, t'), where $t, t' \in RT$.

Parameters and constants: ω_i: The tardiness weight of pickup/delivery task i. M: A suitably large constant. C: Each porter capacity. N^0_{tk}: Number of items of type t carried by porter k.

Variables: $x_{ijk} \in \{0, 1\}$: Porter k advances to service j after performing service i. $x_{0ik} \in \{0, 1\}$: Porter k advances to service i from his depot location. $x_{i0k} \in \{0, 1\}$: Porter k advances to depot location after servicing task i (end of route). $y_{ijt} \in \mathbb{N}$: Number of items of type t travelling on edge (i, j). $z_{ijt} \in \{0, 1\}$: An item of type t travelling on edge (i, j). $s_{ik} \geq 0$: The start time of servicing the task i, where s_{ik} has no meaning if i is not assigned to k. $l_i \geq 0$: Tardiness of task i. $ot_k \geq 0$: Overtime of porter k.

Our MILP model of IHL can be formulated as follows:

$$\textbf{Minimize} \sum_{i \in T} \omega_i \cdot l_i + \sum_{k \in P} ot_k + \sum_{k \in P} \sum_{i,j \in V \cup \{0\}} d_{ij} \cdot x_{ijk} \tag{1}$$

Subject to

$$\sum_{i \in T_k} x_{0ik} = 1 \qquad \forall k \in P \tag{2}$$

$$\sum_{i \in T_k} x_{i0k} = 1 \qquad \forall k \in P \tag{3}$$

$$\sum_{j \in T_k} x_{ijk} = \sum_{j \in T_k} x_{jik} \qquad \forall k \in P, i \in T_k \tag{4}$$

$$\sum_{j\in T_k}\sum_{k\in P_i} x_{ijk} = 1 \qquad \forall i \in T \tag{5}$$

$$\sum_{j'\in T_k} x_{ij'k} \leq \sum_{j'\in T_k} x_{jj'k} \qquad j \neq j',\ (i,j) \in R, k \in P_i \cap P_j \tag{6}$$

$$st_i \cdot \sum_{j\in T_k} x_{ijk} \leq s_{ik} \qquad \forall i \in T, k \in P_i \tag{7}$$

$$s_{ik} + p_i - dt_i \leq l_i \qquad \forall i \in T, k \in P_i \tag{8}$$

$$s_{ik} + d_{i0} + p_i - dt_k \leq ot_k \qquad \forall k \in P, i \in T_k \tag{9}$$

$$s_{ik} + p_i + d_{ij} \leq s_{jk} + M \cdot (1 - x_{ijk}) \qquad \forall k \in P, i, j \in T_k \tag{10}$$

$$s_{ik} + p_i + d_{ij} \leq s_{jk} \qquad \forall (i,j) \in Prec, k \in P_i \cap P_j \tag{11}$$

$$\sum_{j\in T} y_{ijt} - \sum_{j\in T} y_{jit} = 1 \qquad \forall i \in T^p, t = type(i) \tag{12}$$

$$\sum_{j\in T} y_{ijt} - \sum_{j\in T} y_{jit} = -1 \qquad \forall i \in T^d, t = type(i) \tag{13}$$

$$N^0_{tk} \cdot x_{0ik} \leq y_{0it} \qquad \forall i \in T, k \in P_i, t \in RT^0_k \tag{14}$$

$$y_{ijt} \leq C \cdot z_{ijt} \qquad \forall i, j \in T, t \in RT \tag{15}$$

$$\sum_{t\in RT} y_{ijt} \leq C \cdot \sum_{k\in P_i\cap P_j} x_{ijk} \qquad \forall i, j \in T \tag{16}$$

$$z_{ijt} + z_{ijt'} \leq 1 \qquad \forall (t,t') \in F, i, j \in T \tag{17}$$

$$x_{ijk} \in \{0,1\} \qquad \forall k \in P; i, j \in T_k \tag{18}$$

$$u_i, l_i, ot_k \geq 0 \qquad \forall i \in T \tag{19}$$

Constraints (2), (3) and (4) are flow constraints. Constraints (5) guarantee the assignment of each task. Inequalities (6) guarantee that if a pickup task is assigned to porter k, then its corresponding delivery task is also assigned to the same porter. Constraints (7) enforce the commitment to the beginning of the time window. Constraints (8) and (9) derive tasks tardiness and porters overtime, respectively. Inequalities (10) relate the sequencing and scheduling of tasks, inequalities (11) enforce the precedence constraint. Constraints (12) and (13) are flow constraints regarding 'request type flows' which ensure that each pickup of a request type increases its flow by one and each delivery reduces it by one. Constraints (14) define the initial flow. Constraints (15) are constraints linking y and z variables. Porter capacities are defined by (16), and inequalities (17) concern the forbidden combinations of tasks which cannot be simultaneously assigned to the same porter. Constraints (18) and (19) specify the scope of variables x_{ijk}, u_i, l_i and ot_k.

Dynamic optimization approach. This study introduces a *rolling-horizon-based* optimization approach based on a buffering strategy. This method seeks to iteratively group together sets of dynamic arrival requests and allocate them to available porters, thereby reducing operational expenses while maintaining high quality

levels of service. The positive effect of an inherent delay is that it helps compensate for request arrival information uncertainty. One negative effect, however, is that it may introduce tardiness regarding urgent requests. Consequently, the amount of delay must be set considering the urgency level of request arrivals. A visual representation of the buffering-strategy-based approach is illustrated in Fig. 1.

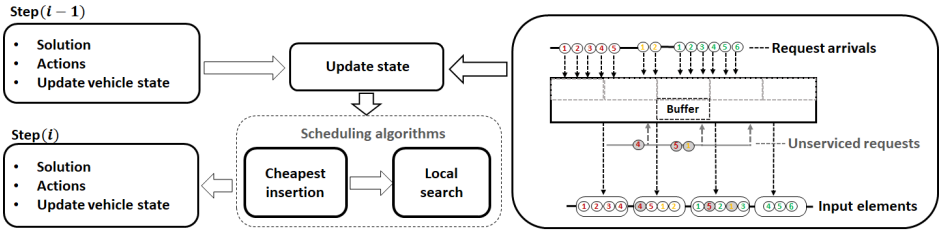


Figure 1: A buffering-strategy-based dynamic optimization approach

Acknowledgement. Editorial consultation provided by Luke Connolly (KU Leuven).

References

[1] A. Beaudry, G. Laporte, T. Melo, S. Nickel, Dynamic transportation of patients in hospitals, *OR Spectrum*, **32** (2010), 77–107.

[2] C. Fiegl, C. Pontow, Online scheduling of pick-up and delivery tasks in hospitals, *Journal of Biomedical Informatics*, **42** (2009), 624–32.

[3] T. Hanne, T. Melo, S. Nickel, Bringing robustness to patient flow management through optimized patient transports in hospitals, *Interfaces*, **39** (2009), 241–255.

[4] M. W. Savelsbergh, M. Sol, The general pickup and delivery problem, *Transportation Science*, **29** (1995), 17–29.

[5] W. Vancroonenburg, E. Esprit, P. Smet, G. Vanden Berghe, Optimizing internal logistic flows in hospitals by dynamic pick-up and delivery models, *Proceedings of the 11th International Conference on the Practice and Theory of Automated Timetabling*, Udine, August 23–26, 2016, 371–383.

[6] J. Warren, R. E. Fromm Jr, R. A. Orr, L. C. Rotello, H. M. Horst, Guidelines for the inter- and intra-hospital transport of critically ill patients, *Critical Care Medicine*, **32** (2004), 256–262.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

Approximation algorithms for energy efficient scheduling of parallel jobs without migration

Alexander Kononov *
Sobolev Institute of Mathematics, Novosibirsk, Russian Federation

Yulia Kovalenko
Sobolev Institute of Mathematics, Novosibirsk, Russian Federation

Keywords: speed scaling, parallel jobs, approximation algorithm

1 Introduction

We consider the energy-efficient scheduling of parallel jobs on m speed-scalable processors. A job set $\mathcal{J} = \{1, \ldots, n\}$ is given, where each job j is characterized by four parameters: release date r_j, deadline d_j, processing volume (work) W_j, and number of required processors $size_j$. Note that parameter $size_j$ for job $j \in \mathcal{J}$ indicates that the job can be processed on any subset of parallel processors of the given size. Such jobs are called *rigid* jobs (Drozdowski [9]). In this paper, migration of a job j among different subsets of $size_j$ processors is disallowed but job preemption might or might not be allowed.

A schedule is feasible if each processor executes at most one job at the time, and each job is processed in the required work between its release time and deadline. We assume that the homogeneous model in speed-scaling is considered. When a processor runs at a speed s, then the the energy consumed per unit time is s^α. In most applications α is a constant not exceeding 3 (Gerards et al. [10]). Each of m processors may operate at variable speed, but if processors execute the same job simultaneously, then all these processors run at the same speed. It is supposed that a continuous spectrum of processor speeds is available. The objective is to find a feasible schedule that minimizes the total energy consumed on all the processors.

2 Related research

For the preemptive single-processor case, Yao et al. [16] proposed an optimal algorithm for finding a feasible schedule with minimum energy consumption. The

* Speaker, email: alvenko@math.nsc.ru

case, where there are m available parallel processors and single-processor jobs has been solved optimally in polynomial time when both the preemption and the migration of jobs are allowed (Albers et al. [1, 4], Bingham and Greenstreet [6], Shioura et al. [15]). There are several variations of the multiprocessor speed scaling scheduling of single-processor jobs considered in the literature. They differ depending on whether preemption and migration of jobs are allowed or not. The preemptive problem with migration is polynomially solvable (see, e.g., Albers et al. [1], Angel et al. [4], Bingham and Greenstreet [6], Shioura et al. [15]). As far as we know, the algorithm by Shioura et al. [15] has the best time complexity among the above-mentioned algorithms.

The preemptive multiprocessor scheduling without migration has been widely studied by Albers et al. [3], Chen et al. [7], Greiner et al. [11] and others. Chen et al. [7] provided a greedy 1.13-approximation algorithm for the problem with a common release date and a common deadline for all jobs. Albers et al. [3] proposed $\left(\alpha^\alpha 2^{4\alpha}\right)$-approximation algorithms for jobs with unit works and arbitrary deadlines, or arbitrary works and agreeable deadlines, and an $2\left(2-\frac{1}{m}\right)^\alpha$-approximation algorithm for jobs with common release dates, or common deadlines. The authors also proved that the problem with unit works is polynomially solvable for agreeable deadlines, but becomes $\mathcal{NP}$-hard in the strong sense for arbitrary deadlines. Greiner et al. [11] further gave a generic reduction transforming any ρ-approximate schedule for parallel processors with migration to a $\rho B_{\lceil\alpha\rceil}$-approximate solution for parallel processors without migration, where $B_{\lceil\alpha\rceil}$ is the $\lceil\alpha\rceil$-th *Bell number.*

Let us notice that the non-preemptive problem is equivalent to the preemptive problem without migration in the case of agreeable deadlines. Hence, the results by Albers et al. [3], Chen et al. [7], Greiner et al. [11] for agreeable deadlines also hold for the non-preemptive problem. For general non-preemptive instances Cohen-Addad et al. [8] present an $(\frac{5}{2})^{\alpha-1}(1+\varepsilon)^\alpha \tilde{B}^\alpha(1+\rho)$-approximation algorithm, where ρ is a ratio between maximal and minimal job volume.

Bampis et al. [5] initiated the research of heterogeneous problems, where each processor has each own power function and job's parameters are processor dependent. For the non-migratory problem, they proposed an approximation algorithm of ratio $(1+\varepsilon)^\alpha \tilde{B}_\alpha$. For the case where the migration is permitted, Bampis et al. [5] present an algorithm, that returns a solution within an additive error ε in time polynomial in the problem size and in $\frac{1}{\varepsilon}$. Later, Albers et al. [2] gave a faster flow-based algorithm for the migratory scheduling of jobs, which solves the problem within any desired accuracy.

Kononov and Kovalenko [12, 13] proposed an approximation algorithm for the

speed scaling scheduling of rigid jobs with preemption and migration. The algorithm returns a solution within an additive error $\varepsilon > 0$ and runs in time polynomial in m, $\frac{1}{\varepsilon}$ and the input size. Note that this algorithm is pseudopolynomial and it is based on solving a configuration linear program using the ellipsoid method. Recently, Kononov and Kovalenko [14] developed a strongly polynomial time algorithm that achieves a $\left(2 - \frac{1}{m}\right)^{\alpha-1}$-approximation ratio.

3 Our Results

In practice, instances with agreeable release dates and deadlines form a natural class where jobs arriving at later times may be completed later (see, e.g., Gerards et al. [10]). Formally, release dates and deadlines are called agreeable if, for any two jobs i and j, relation $r_i < r_j$ implies $d_i \leq d_j$. We propose approximation algorithms for the following problems with agreeable release dates and deadlines:

- a factor $\left(4 - \frac{4}{m+1}\right)^{\alpha-1}$ approximation algorithm for the preemptive problem without migration;
- a factor $3^{\alpha-1}$ approximation algorithm for the non-preemptive problem with equal-work jobs;
- a factor $\left(3 - \frac{4}{m+1}\right)^{\alpha-1}$ approximation algorithm for the non-preemptive problem, where each job j requires at most $\frac{m}{2}$ processors.

We also give a factor $2^{\alpha}(\log m)^{\alpha-1}B_{\lceil\alpha\rceil}$ approximation algorithm for the preemptive problem without migration, but with arbitrary release dates and deadlines.

Acknowledgement. The research of the first author was partially supported by RFBR grant 17-07-00513.

References

[1] S. Albers, A. Antoniadis, G. Greiner, On multi-processor speed scaling with migration, *Journal of Computer and System Sciences*, **81** (2015), 1194–1209.

[2] S. Albers, E. Bampis, D Letsios, G. Lucarelli, R. Stotz, Scheduling on power-heterogeneous processors, *Information and Computation*, **257** (2017), 22–33.

[3] S. Albers, F. Müller, S. Schmelzer, Speed scaling on parallel processors, *Algorithmica*, **68** (2014), 404–425.

[4] E. Angel, E. Bampis, F. Kacem, D. Letsios, Speed scaling on parallel processors with migration, *Lecture Notes in Computer Science*, 7**484** (2012), 128–140.

[5] E. Bampis, A. Kononov, D. Letsios, G. Lucarelli, M. Sviridenko, Energy efficient scheduling and routing via randomized rounding, *Journal of Scheduling*, **21** (2018), 35–51.

[6] B. D. Bingham, M. R. Greenstreet, Energy optimal scheduling on multiprocessors with migration, *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing with Applications*, Sydney, December 8–10, 2008, 153–161.

[7] J. J. Chen, H. R. Hsu, K. H. Chuang, C. L. Yang, A.C. Pang, T. W. Kuo, Multiprocessor energy-efficient scheduling with task migration considerations, *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, Catania, June 30 – July 2, 2004, 101–108.

[8] V. Cohen-Addad, Z. Li, C. Mathieu, I. Milis, Energy-efficient algorithms for non-preemptive speed-scaling, *Lecture Notes in Computer Science*, **8952** (2015), 107–118.

[9] M. Drozdowski, *Scheduling for Parallel Processing*, Springer, London, 2009.

[10] M. E. T. Gerards, J. L. Hurink, P. K. F. Hölzenspies, A survey of offline algorithms for energy minimization under deadline constraints, *Journal of Scheduling*, **19** (2016), 3–19.

[11] G. Greiner, T. Nonner, A. Souza, The bell is ringing in speed-scaled multiprocessor scheduling, *Theory of Computing Systems*, **54** (2014), 24–44.

[12] A. Kononov, Yu. Kovalenko, An 'almost-exact' solution to speed scaling scheduling of parallel jobs with preemption, *Proceedings of the 1st International Workshop on Dynamic Scheduling Problem*, Poznań, June 30 – July 1, 2016, 41–44.

[13] A. Kononov, Yu. Kovalenko, On speed scaling scheduling of parallel jobs with preemption, *Lecture Notes in Computer Science*, **9869** (2016), 309–321.

[14] A. Kononov, Yu. Kovalenko, An approximation algorithm for preemptive speed scaling scheduling of parallel jobs with migration, *Lecture Notes in Computer Science*, **10556** (2017), 351–357.

[15] A. Shioura, N. Shakhlevich, V. Strusevich, Energy saving computational models with speed scaling via submodular optimization, *Proceedings of the 3rd International Conference on Green Computing, Technology and Innovation*, Serdang, December 8–10, 2015, 7–18.

[16] F. F. Yao, A. J. Demers, S. Shenker, A scheduling model for reduced CPU energy, *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, Milwaukee, October 23–25, 1995, 374–382.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

Minimizing total absolute deviation of job completion times on unrelated machines with general position-dependent processing times and job rejection

Baruch Mor *
Ariel University, Ariel, Israel

Gur Mosheiov
The Hebrew University of Jeruslaem, Jerusalem, Israel

Keywords: scheduling, unrelated machines, total absolute deviation of completion times, position-dependent processing times, job rejection

1 Introduction

The scheduling measure considered in this talk is that of *Total Absolute Deviation of Completion times* (*TADC*). This measure arises in many service systems, where the main objective is to provide customers with identical or similar quality of service. It reflects their total time-in-system or total waiting time. TADC is considered here in the most general form studied so far: the machine setting is that of parallel unrelated, job processing time are assumed to be position-dependent with no restrictions on the functional form and we allow the option of job rejection.

2 Related research

TADC was introduced by Kanet [8]. He showed that minimizing of TADC on a single machine can be solved in $O(n \log n)$, where n is the number of jobs. A number of extensions of the result have been published.

Mosheiov [11] solved TADC on a single machine and on parallel identical machines with position-dependent job processing times. Mor and Mosheiov [10] solved TADC on uniform and unrelated machines. The authors also studied a more general bi-criteria objective function, containing a linear combination of TADC and total completion time. Chen et al. [4] studied the same bi-criteria objective with past-sequence-dependent setup times and a learning effect. Recently,

* Speaker, email: baruchm@g.ariel.ac.il

Ben-Yehoshua et al. [3] focused on minimizing TADC on a two-machine no-wait proportionate flow shop.

We study a scheduling problem with the TADC objective in a more general setting. First, we allow the option of *job rejection*, which became a popular topic among researchers in recent years. This is reflected in the recently published survey by Shabtay et al. [14], where it is claimed that in many systems the acceptation of all jobs may cause a delay in the completion of orders, which may lead to high tardiness cost. In such systems, the firm may wish to reject the processing of some jobs by either outsourcing them or rejecting them all together. Thus, the method of job rejection provides the operations manager with an analytical tool that enables him to improve the overall performance of the firm, known as *Quality of Service*. Utilizing job rejection techniques, the operations manager can improve the operational expense, enhance the distribution of the resources, and reduce the time-to-market. Technically, in scheduling models considering rejection, the rejected jobs are penalized, and this cost becomes a factor in the objective function. Some of the very recent papers dealing with various models of scheduling with job rejection are Mor and Mosheiov [9], Agnetis and Mosheiov [2], Gerstl and Mosheiov [7], and Mosheiov and Strusevich [12].

Second, following another popular topic in scheduling research, we also consider *position-dependent* job processing times. Scheduling problems with *variable* job processing times (*starting-time-dependent* or position-dependent) have attracted many authors in recent years; see, e.g., Gawiejnowicz [5], Agnetis et al. [1], Gerstl et al. [6] and Pei et al. [13]. In most cases, such extensions increase significantly the complexity of the problems, even if the processing times are restricted to monotone functions of the job starting times or the job positions (reflecting *aging* or *learning*), or to specific functions (such as linear or exponential).

3 Our results

In this paper, we consider general position-dependent processing times, and no restrictions on their functional form are assumed. Thus, we assume that the processing time of a job is a function of

- the job itself,
- the machine to which it is assigned, and
- the position in the sequence processed on this machine.

Moreover, the machine setting assumed is that of parallel unrelated.

Compared to results published so far, we consider a more general version of a scheduling problem with a TADC objective. Specifically, we extend TADC in three aspects simultaneously:

- we minimize TADC on parallel–unrelated machines,
- position–dependent job processing times are general ones, and
- we allow the option of job rejection.

We show that minimizing of TADC in this very general form remains polynomially solvable in the number of jobs, n, provided that the number of machines, m, is given. Our algorithm for solving this problem consists of solving a sequence of *linear assignment problems* (*LAP*). Each LAP is defined and solved for a given *allocation vector*, including the numbers of jobs assigned to each machine and a given number of rejected jobs. This procedure is repeated for all allocation vectors. Since the number of allocation vectors, equal to the number of partitions of n items into $m+1$ sets, is known to be polynomial in n (Stirzaker [15]), the entire solution procedure is polynomial in n. More specifically, the number of assignment problems to be solved does not exceed $O(n^m)$, which is polynomial in the number of jobs for a given number of machines.

4 Future research

Solving TADC with position- or time-dependent job processing time and job rejection is a challenging topic for future research.

References

[1] A. Agnetis, J-C. Billaut, S. Gawiejnowicz, D. Pacciarelli, A. Soukhal, *Multiagent Scheduling: Models and Algorithms*, Springer, Berlin-Heidelberg, 2014.

[2] A. Agnetis, G. Mosheiov, Scheduling with job-rejection and position-dependent processing times on proportionate flowshops, *Optimization Letters*, **11** (2017), 885–892.

[3] Y. Ben-Yehoshua, E. Hariri, G. Mosheiov, A note on minimising total absolute deviation of job completion times on a two-machine no-wait proportionate flowshop, *International Journal of Production Research*, **53** (2015), 5717–5724.

[4] S. H. Chen, V. Mani, Y. H. Chen, Bi-criterion single machine scheduling problem with a past-sequence-dependent setup times and learning effect, *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management*, Singapore, December 6–9, 2015, 185–189.

[5] S. Gawiejnowicz, *Time-Dependent Scheduling*, Springer, Berlin-Heidelberg, 2008.

[6] E. Gerstl, B. Mor, G. Mosheiov, Minmax scheduling with acceptable lead-times: Extensions to position-dependent processing times, due-window and job rejection, *Computers & Operations Research*, **83** (2017), 150–156.

[7] E. Gerstl, G. Mosheiov, Single machine scheduling problems with generalised due-dates and job-rejection, *International Journal of Production Research*, **55** (2017), 3164–3172.

[8] J. J. Kanet, Minimizing variation of flow time in single machine systems, *Management Science*, **27** (1981), 1453–1459.

[9] B. Mor, G. Mosheiov, Minimizing maximum cost on a single machine with two competing agents and job rejection, *Journal of the Operational Research Society*, **67** (2016), 1524–1531.

[10] B. Mor, G. Mosheiov, Total absolute deviation of job completion times on uniform and unrelated machines, *Computers & Operations Research*, **38** (2011), 660–665.

[11] G. Mosheiov, Minimizing total absolute deviation of job completion times: extensions to position-dependent processing times and parallel identical machines, *Journal of the Operational Research Society*, **59** (2011), 1422–1424.

[12] G. Mosheiov, V. A. Strusevich, Determining optimal sizes of bounded batches with rejection via quadratic min-cost flow, *Naval Research Logistics*, **64** (2017), 217–224.

[13] J. Pei, B. Cheng, X. Liu, P. M. Pardalos, M. Kong, Single-machine and parallel-machine serial-batching scheduling problems with position-based learning effect and linear setup time, *Annals of Operations Research*, 2017, doi: 10.1007/s10479-017-2481-8, in press.

[14] D. Shabtay, N. Gaspar, M. Kaspi, A survey on online scheduling with rejection, *Journal of Scheduling*, **16** (2013), 3–28.

[15] D. Stirzaker, *Elementary Probability*, Cambridge University Press, Cambridge, 1994.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

Minmax scheduling and due-window assignment with position-dependent processing times and job rejection

Gur Mosheiov *
Hebrew University of Jerusalem, Jerusalem, Israel

Assaf Sarig
The Center for Academics Studies, Or Yehuda, Israel

Vitaly A. Strusevich
University of Greenwich, London, United Kingdom

Keywords: scheduling, sequencing, due-window assignment, position-dependent processing times, job rejection

1 Introduction

A *due-window* is a generalization of the classical job due-date, where a time interval is assumed, in which the job should be completed with no penalty. If the job is completed prior to or after its due-window, it is penalized according to its earliness/tardiness values. In most studied models, the objective function contains four cost components: for earliness, tardiness, due-window starting time, and due-window size. It is clear that if the supplier/scheduler provides a late and/or a wide due-window for the delivery of goods, competitors become more attractive to customers. Hence, in general, the scheduler tries to minimize its cost by assigning a relatively early and narrow window. The recent survey by Janiak et al. [4] covers a wide range of scheduling models with due-windows, considering various combinations of machine settings and objective functions.

The vast majority of the published papers on due-window assignment problems is focused on classical *minsum* measures, where the objective is to minimize the total cost incurred by all the jobs. Very few researchers studied *minmax* objectives in this context, where the goal is to minimize the largest cost among all scheduled jobs. Minmax models are mainly relevant to service systems, where one tries to provide service of equal level to all customers. Such models are also known as those maximizing fairness. In this paper, we focus on the model introduced

* Speaker, email: msomer@huji.ac.il

in Mosheiov and Sarig [5], where a common due-window for all the jobs is assumed, both its starting time and size are decision variables, and the minmax objective consists of the costs of earliness, tardiness, due-window starting time and size. Mosheiov and Sarig [5] introduced a polynomial time solution for the single machine problem, and proposed and tested heuristics for the settings of parallel identical and uniform machines.

In this work, we extend the model solved in [5] in two aspects: (i) we consider *position-dependent* job processing times, and (ii) we allow *job rejection*.

2 Related research

Position- and time-dependent processing times reflect many real-life applications. As claimed in Yin et al. [7] and in a number of other recently published papers: "*…recent empirical studies in several industries have demonstrated that unit cost decline as companies produce more of a product and gain knowledge or experience*". Such applications, known also as a *'learning effect'*, are reflected by a monotonic non-increasing job processing times as a function of the job-position. The opposite phenomenon, where job processing times increase as a function of the job position or starting time, is known in the literature as *'deterioration'*. Cheng et al. [2] claim that *"real-life situations (…) can be found in financial management, steel production, resource allocation and national defense, where any delay in tackling a task may result in an increasing effort (time, cost, etc.)"*. Gawiejnowicz [3] was the first who studied scheduling with position-dependent processing times, defined by the speed of the processor, which is a function of the number of executed jobs. Biskup [1] focused on a specific (non-increasing) exponential function. Since then, a large number of extensions have been solved and analyzed, mostly with specific position-independent job processing time functions.

3 Our results

We assume here that the job processing times are position-dependent in the most general way, i.e., they are not considered to be either non-increasing (a learning effect) or non-decreasing (a deterioration or aging effect), and they are not restricted to any specific function.

We show that this first extension, to a minmax due-window assignment problem with general position-dependent processing times, remains polynomially solved.

Specifically, we prove that regarding the due-window location and size, there are four candidates for the optimal solution:

- Case (i): The due-window is reduced to a due-date at time zero.
- Case (ii): The due-window is reduced to a due-date at time some intermediate point.
- Case (iii): A large due-window consisting of the interval $[C_{\min}, C_{\max}]$, where $C_{\min}$ is the completion of the first scheduled job, and $C_{\max}$ is the completion time of the last scheduled job.
- Case (iv): A large due-window consisting of the interval $[0, C_{\max}]$.

The solution of each of these cases is reduced to that of solving a single or a series of linear assignment problems. Thus, Cases (i) and (iv) are solved in $O(n^3)$, whereas Cases (ii) and (iii) are solved in $O(n^4)$. Therefore, the total running time required for solving the problem is $O(n^4)$.

We further extend the model to allow the option of job rejection. In their recent survey on scheduling with job rejection, Shabtay et al. [6] claim that *"in many practical cases, mostly in highly loaded make-to-order production systems, accepting all jobs may cause a delay in the completion of orders which in turn may lead to high inventory and tardiness costs. Thus, in such systems, the firm may wish to reject the processing of some jobs by either outsourcing them or rejecting them altogether."* In the model studied here, a job which is not processed is penalized by a job-dependent rejection cost.

In all scheduling models with job rejection, there is a trivial trade-off between the rejection cost and the scheduling measure considered. Clearly, when a job is rejected, the scheduling performance measure of the remaining jobs is improved. However, it can be easily shown in our setting, where general position-dependent job processing times are assumed, that this is not necessarily the case. (Note that after rejecting jobs, the number of positions for the remaining jobs is reduced, and consequently the actual processing times of some jobs may increase.) Hence, we focus here on general job deterioration, i.e., the job processing times are non-decreasing functions of their position. We prove that under the assumption of deterioration, rejecting a job can only improve the minmax cost of the remaining jobs. Consequently, assuming both general deterioration and job rejection, we propose a polynomial time solution for this more general setting, which, as above, consists of solving a series of linear assignment problems. Specifically, the running time of our proposed solution algorithm is $O(n^4 \log n)$.

References

[1] D. Biskup, Single machine scheduling with learning considerations, *European Journal of Operational Research*, **115** (1999), 173–178.

[2] T.-C. E. Cheng, Q. Ding, B. M.-T. Lin, A concise survey of scheduling with time-dependent processing times, *European Journal of Operational Research*, **152** (2004), 1–13.

[3] S. Gawiejnowicz, A note on scheduling on a single processor with speed dependent on a number of executed jobs, *Information Processing Letters*, **57** (1996), 297–300.

[4] A. Janiak, W. A. Janiak, T. Krysiak, T. Kwiatkowski, A survey on scheduling problems with due windows, *European Journal of Operational Research*, **242** (2015), 347–357.

[5] G. Mosheiov, A. Sarig, Minmax scheduling problems with a common due-window, *Computers & Operations Research*, **36** (2009), 1886–1892.

[6] D. Shabtay, N. Gaspar, M. Kaspi, A survey on offline scheduling with rejection, *Journal of Scheduling*, **16** (2013), 3–28.

[7] Y. Yin, D. Xu, K. Sun, H. Li, Some scheduling problems with general position-dependent and time-dependent learning effects, *Information Sciences*, **179** (2009), 2416–2425.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

Precedence constrained parallel-machine scheduling of position-dependent unit jobs

Bartłomiej Przybylski *
Adam Mickiewicz University in Poznań, Poznań, Poland

Keywords: scheduling, parallel machines, position-dependent processing times, precedence constraints, maximum completion time, total completion time

1 Introduction

In many real-world scheduling problems job processing times are not fixed but variable. Two groups of theoretical models approximating such variability are widely considered in literature: *time-dependent* models, where the processing time of a job depends on its starting time (see Gawiejnowicz [5] for a review), and *position-dependent* models, where the processing time of a job depends on a set of already executed jobs (see Biskup [2], Strusevich and Rustogi [11] for a survey).

Recently, scheduling problems with the *learning effect*—where the processing time of a job decreases as its position in a schedule increases—are gaining more and more attention. A dual group of problems with the *ageing effect* is also analyzed in literature. Having this in mind, we focus on a number of scheduling problems where the job processing time may, but does not have to give in to such a monotonic impact. In particular, we present some polynomial algorithms for scheduling problems in which job processing times are described by any positive and discrete function.

It is worth noticing that most of the papers concerning position-dependent job processing times focus either on single machine problems of scheduling precedence-constrained jobs (see, e.g., Wang [13]), or on parallel-machine problems, but with no precedence constraints between jobs (see, e.g., Mosheiov [8]). During the presentation, we show some new results related to parallel-machine scheduling of precedence-constrained identical jobs, continuing the line of research initiated by Przybylski [10].

* Speaker, email: bap@amu.edu.pl

2 Problem formulation

The group of problems to be discussed during the presentation can be formulated as follows. We are given a set of parallel machines and n non-preemptable jobs with or without precedence constraints. The processing time of the i-th job depends on its position in a schedule. This dependency is described by a positive function φ, common to all the jobs. In other words, the processing time of the i-th job holding the r-th position on a machine is equal to $p_{i,r} = \varphi(r)$. The aim is to find an optimal schedule in the sense of at least one of the following objective functions: maximum completion time or total completion time. Using the three-field notation (see Gawiejnowicz [5] for a brief description), we can denote this group of problems by $P|prec, p_{i,r} = \varphi(r)|f \in \{C_{\max}, \sum C_i\}$.

Some special cases of the above problems, when $\varphi(r) = 1$ for every argument r, are the $P|prec, p_i = 1|f \in \{C_{\max}, \sum C_i\}$ problems of scheduling unit jobs. It is known that both these problems are at least $\mathcal{NP}$-hard (see Ullman [12], Hoogeveen and Schuurman [6]). However, restricting either the graph of precedence constraints or the number of machines can lead to polynomially-solvable problems: the $P|tree, p_i = 1|C_{\max}$ and $P2|prec, p_i = 1|f \in \{C_{\max}, \sum C_i\}$ problems can be solved in $O(n)$ and $O(n^2)$ time, respectively (see Hu [7] and Coffman and Graham [3]). On the other hand, the $Pm|in\text{-}tree, p_i = 1|\sum C_i$ problem can be solved in $O(n^m)$ time (see Baptiste et al. [1]).

Therefore, as the unit jobs scheduling problems set the lower bounds on the complexity of corresponding problems with position-dependent jobs, the natural question is which special cases of the $P|prec, p_{i,r} = \varphi(r)|f \in \{C_{\max}, \sum C_i\}$ problems are polynomially-solvable. It is also worth studying which problems that are polynomially-solvable when the jobs have fixed unit processing times, become $\mathcal{NP}$-hard when the processing times are position-dependent. We are to give a partial answer to both these questions.

3 Our results

Let $\varphi\colon \mathbb{N}_+ \to \mathbb{Q}_+$ be any discrete function satisfying the condition $\varphi(r) > 0$ for every $r \in \mathbb{N}_+$, where $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. In order to simplify the notation, we write $\varphi_\downarrow$ if it is assumed a priori that the φ function is weakly decreasing, and $\varphi_\uparrow$ if it is assumed that the φ function is weakly increasing. Finally, if it is only assumed that the φ function is weakly monotonous, but it may be either weakly increasing or weakly decreasing, we denote it by $\varphi_\updownarrow$.

One can obtain some special cases of the $P|prec, p_{i,r} = \varphi(r)|f \in \{C_{\max}, \sum C_i\}$ problems by restricting either the graph of precedence constraints, the number of machines, or the characteristics of the φ function. Other special cases can be obtained by mixing such restrictions. The following results are going to be discussed during the presentation.

Theorem 1 (Corollary from Przybylski [9]). *The $P|in\text{-}tree, p_{i,r} = \varphi_{\downarrow}(r)|C_{\max}$ and $Pm|in\text{-}tree, p_{i,r} = \varphi_{\downarrow}(r)|\sum C_i$ problems can be solved in polynomial time, provided that the values of the φ function can be computed in polynomial time.*

Theorem 2. *The $P|chains, p_{i,r} = \varphi_{\updownarrow}(r)|f \in \{C_{\max}, \sum C_i\}$ problems can be solved in polynomial time, provided that the values of the φ function can be computed in polynomial time.*

By an *in-forest* we understand an in-tree or a parallel composition of in-trees. Please notice that, as any chain is a special case of an in-tree, any set of chains is a special case of an in-forest. Thus, the following theorem holds.

Theorem 3. *The $P2|in\text{-}forest, p_{i,r} = \varphi(r)|f \in \{C_{\max}, \sum C_i\}$ problems can be solved in polynomial time, provided that the values of the φ function can be computed in polynomial time.*

Let us notice that we do not make any assumptions about the characteristics of the φ function in Theorem 3.

Polynomial algorithms solving all the above problems are to be presented together with some intuitive examples. For example, it is to be shown that the problem stated in Theorem 3 can be solved in polynomial time based on algorithms presented by Hu [7] and Conway et al. [4]. Moreover, we will discuss a set of open questions, including the following.

Open questions. Are the $P2|prec, p_{i,r} = \varphi(r)|f \in \{C_{\max}, \sum C_i\}$ problems polynomially solvable? If it is not easy to determine, then what about corresponding problems in which the graph of precedence constraints is reduced to a series-parallel graph or an out-tree, or the φ function is limited to the case of it being weakly monotonous?

4 Future research

The above results can be a base for further research on the problems of parallel-machine scheduling of position-dependent unit jobs. In the case of the open questions, some special cases of the stated problem can be further analyzed. Finally, the results can be generalized to the case of scheduling problems, in which the

basic processing times of jobs are different. It is also worth analyzing, what is the complexity of problems in which the φ function can differ between jobs or between machines.

References

[1] P. Baptiste, P. Brucker, S. Knust, V. Timkovsky, *4OR Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, **2** (2004), 111–127.

[2] D. Biskup, A state-of-the-art review on scheduling with learning effects, *European Journal of Operational Research*, **188** (2008), 315–329.

[3] E. G. Coffman Jr., R. L. Graham, Optimal scheduling for two-processor systems, *Acta Informatica*, **1** (1971/1972), 200–213.

[4] R. W. Conway, W. L. Maxwell, L. W. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, 1967.

[5] S. Gawiejnowicz, *Time-Dependent Scheduling*, Springer, Berlin-Heidelberg, 2008.

[6] H. Hoogeveen, P. Schuurman, G. J. Woeginger, Non-approximability results for scheduling problems with minsum criteria, *INFORMS Journal on Computing*, **13** (2001), 157–168.

[7] T. C. Hu, Parallel sequencing and assembly line problems, *Operations Research*, **3** (1961), 841–848.

[8] G. Mosheiov, Minimizing total absolute deviation of job completion times: extensions to position-dependent processing times and parallel identical machines, *Journal of the Operational Research Society*, **59** (2008), 1422–1424.

[9] B. Przybylski, Precedence constrained parallel-machine scheduling of position-dependent jobs, *Optimization Letters*, **11** (2017), 1273–1281.

[10] B. Przybylski, Precedence constrained position-dependent scheduling on parallel machines via schedule transformations, *Proceedings of the 1st International Workshop on Dynamic Scheduling Problems*, Poznań, June 30 – July 1, 2016, 65–67.

[11] V. A. Strusevich, K. Rustogi, *Scheduling with Time-Changing Effects and Rate-Modifying Activities*, Springer, Berlin, 2017.

[12] J. D. Ullman, NP-complete scheduling problems, *Journal of Computer and System Science*, **10** (1975), 384–393.

[13] J.-B. Wang, Single-machine scheduling with learning effect and deteriorating jobs, *Computers & Industrial Engineering*, **57** (2009), 1452–1456.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

Scheduling non-monotonous convex piecewise-linear time-dependent processing times of a uniform shape

Helmut A. Sedding *
Ulm University, Ulm, Germany

Keywords: time-dependent scheduling, single machine, piecewise-linear convex processing times

1 Introduction

Time-dependent scheduling deals with job processing times that are functions of the job starting times (Gawiejnowicz [4]). In this work, we analyze the computational complexity of a certain generic problem of this kind, with non-monotonous convex job processing times. This problem is denoted in three-field notation as $1 \mid p_j = l_j + \max\{-a\,(t-\tau),\ b\,(t-\tau)\} \mid C_{\max}$, where $a \in [0,1]$ and $b \in [0,\infty)$ denote rational slopes of each job, while τ and $l_j \in \mathbb{Q}_{\geq 0}$ denote the ideal start time and the basic processing time for each job $j \in J$ in job set $J = \{1,\ldots,n\}$, respectively. The objective is to find a sequence $S : J \to \{1,\ldots,n\}$ that minimizes makespan $C_{\max}$. The processing time $p_j = p_j(t)$ of job $j \in J$ is smallest if j starts at $t = \tau$. If $t < \tau_j$ decreases, $p_j(t)$ increases with slope a, else with slope b. Hence, $p_j(t)$ is asymmetric. Thus, the processing time functions are of the same shape, differing only by the l_j value. The completion time $C_j = t + p_j(t)$ of job j is increasing with t as $a \leq 1$ and $b \geq 0$. Therefore, idle time may only increase the objective and thus it is excluded by definition. The first job starts at global start time $t_{\min}$, being zero if not stated otherwise. In short, we denote this problem as $\hat{P}$.

In the following, we show that problem $\hat{P}$ is $\mathcal{NP}$-complete by a reduction from the EVEN-ODD PARTITION problem (Garey et al. [3]). We also present two polynomial cases of $\hat{P}$. With these results, we aim to lay groundwork for time-dependent scheduling with non-monotonous job processing times.

The $\hat{P}$ problem setting is motivated by computer assisted production planning of continuously moving assembly lines. There, we consider sequencing 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 central part storage at the line side. For this purpose, the worker leaves the workpiece and walks along the

* Speaker, email: Helmut.Sedding@uni-ulm.de

conveyor. As the conveyor continuously moves the workpiece, walking distance varies over time. It is minimal at time τ, when the moving workpiece just passes the storage position; otherwise, it increases linearly. Thus, walk time is depicted by a V-shaped, piecewise-linear function of time. It is asymmetric because worker velocity and conveyor velocity are added to a greater amount of time rather than subtracted if the storage position is down-versus upstream. 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 with a constant part, and the goal is to minimize the makespan.

2 Related research

Non-monotonous processing times are considered by Farahani and Hosseini [2], who assume a variable global start time $t_{\min}$ for $\hat{P}$ with symmetric factors $a = b$, and provide a polynomial time algorithm for minimizing $C_{\max} - t_{\min}$. Also related is the setting in Jaehn and Sedding [5], which measures a job's deviation from the mid-time, when exactly half of the whole job has been processed. Problem $\hat{P}$ with $a = 0$ or $b = 0$, i.e. with monotonous processing times, is solvable in polynomial time (Melnikov and Shafransky [7]), but it is $\mathcal{NP}$-hard with job-specific slopes (Kubiak and Van de Velde [6], Cheng et al. [1]). Hence, we are suprised to discover that already the non-monotonous case with uniform slopes is $\mathcal{NP}$-hard.

3 Our results

Polynomial cases. We introduce two polynomial cases of $\hat{P}$, which next we use to show $\mathcal{NP}$-hardness of $\hat{P}$.

Lemma 1. *Given an instance of $\hat{P}$ and a sequence S where $t_j \geq \tau$ for all $j \in J$, then*

$$C_{\max} = t_{\min} (1+b)^n + \sum_{j \in J} \left(l_j - b\tau\right) (1+b)^{n-S(j)} . \tag{1}$$

If jobs in S are ordered non-decreasingly with respect to l_j, then $C_{\max}$ is minimal.

Lemma 2. *Given an instance of $\hat{P}$ and a sequence S where $t_j \leq \tau$ for all $j \in J$, then*

$$C_{\max} = t_{\min} (1-a)^n + \sum_{j \in J} \left(l_j + a\tau\right) (1-a)^{n-S(j)} . \tag{2}$$

If jobs in S are ordered non-increasingly with respect to l_j, then $C_{\max}$ is minimal.

Corollary 3. *Eq. (2) in Lemma 2 is equivalent to*

$$t_{\min} = C_{\max}\,(1-a)^{-n} - \sum_{j\in J}\left(l_j + a\tau\right)(1-a)^{-S(j)}\,. \tag{3}$$

$\mathcal{NP}$-complete case. The decision version of problem $\hat{P}$, denoted $\hat{P}_\Phi$, asks whether there exists a schedule S with $C_{\max} \leq \Phi$ for a threshold Φ. We prove $\mathcal{NP}$-completeness of problem $\hat{P}_\Phi$ by a reduction from problem EVEN-ODD PARTITION (Garey et al. [3]): given set $X = \{x_1, \ldots, x_n\}$ of $n = 2h$ natural numbers, where $x_{i-1} < x_i$ for all $i = 2, \ldots, n$, exists there a partition of X into subsets X_1 and $X_2 := Y \backslash X_1$ 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 EVEN-ODD PARTITION to $\hat{P}_\Phi$, and an advantage of our proof is its brevity compared to the one in Jaehn and Sedding [5] for a similar problem.

Theorem 4. *Problem $\hat{P}_\Phi$ is $\mathcal{NP}$-complete.*

Proof. Given an arbitrary instance of problem EVEN-ODD PARTITION, we define a corresponding instance of problem $\hat{P}_\Phi$, and show that a solution for $\hat{P}_\Phi$ exists if and only if there exists a solution for EVEN-ODD PARTITION.

In the corresponding instance, 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 ideal start time $\tau = 0$, global start time $t_{\min} = -q$, and 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 of S is divided into S_1, consisting of jobs that start before 0, and S_2, consisting of jobs that start at or after 0.

By (2), sequence S_1 is sorted non-increasingly with respect to l_j, while by (1) sequence S_2 is in non-decreasing order with respect to l_j. Moreover, S_0 is in S between S_1 and S_2, since the jobs in S_0 have the smallest assembly times. Let $\tilde{C}$ denote the completion time of S_1 and $\tilde{t} \geq 0$ be the start time of S_2. Hence, sequence S_0 starts at $\tilde{C}$ and completes at $\tilde{t}$. As $a < 1$, if a job j with $l_j = 0$ starts at $\tilde{C} < 0$, then it completes at $\tilde{t} < 0$. This contradicts the order of jobs in $S_1 \cup S_0$. Thus, there must be $\tilde{C} \geq 0$. Then, $\tilde{t} = \tilde{C}\,(1+b)^n$ by (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 us assume that $S(2n+1) = 1$. Then, job $2n+1$ starts at $-q$ and has a completion time of $-q + l_{2n+1} + aq > 0$, equaling $\tilde{C} = q(1+a)$. By (1), jobs $n+1, \ldots, 2n$ and then jobs $1, \ldots, n$ are appended in non-decreasing order with respect to l_j. Thus

$$\begin{aligned}
C_{\max} &= \tilde{C}(1+b)^{2n} + \sum_{j=1,\ldots,2n} l_j (1+b)^{2n-(S(j)-1)} \\
&= \tilde{C}(1+b)^{2n} + \sum_{j=1,\ldots,n} l_j (1+b)^{n-j} \\
&= \tilde{C}(1+b)^{2n} + \sum_{k=1,\ldots,h} l_{2k-1}(1+b)^{n+2-2k} + l_{2k}(1+b)^{n+1-2k} \\
&> \tilde{C}(1+b)^{2n} + \sum_{k=1,\ldots,h} (l_{2k-1} + l_{2k})(1+b)^{n+1-2k} \\
&= \tilde{C}(1+b)^{2n} + \sum_{k=1,\ldots,h} (x_{2k-1} + x_{2k})(1+b)^{k-h-1}(1+b)^{n+1-2k} \\
&= \tilde{C}(1+b)^{2n} + \sum_{k=1,\ldots,h} (x_{2k-1} + x_{2k})(1+b)^{h-k} \\
&> \tilde{C} + \sum_{k=1,\ldots,h} (x_{2k-1} + x_{2k}) = \tilde{C} + 2q = q(3+a) > 3q = \Phi.
\end{aligned}$$

Hence, job $2n+1$ is the last job in S_2 in any optimal S with $C_{\max} \leq \Phi$. Moreover, note that since S_0 starts at or after 0, partial sequence S_1 contains at most n jobs.

Let S be optimal 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 $\tilde{C} \geq 0$ and (3), we have

$$\begin{aligned}
t_{\min} &= \tilde{C}(1-a)^{-h_1} - \sum_{k=1,\ldots,h_1} l_{S_1^{-1}(k)} (1-a)^{-k} \\
&= \tilde{C}(1+b)^{h_1} - \sum_{k=1,\ldots,h_1} l_{S_1^{-1}(k)} (1+b)^{k}.
\end{aligned}$$

Because S_2 starts at $\tilde{t} = \tilde{C}(1+b)^n$, by (1) we have

$$\begin{aligned}
C_{\max} &= \tilde{t}(1+b)^{n+h_2+1} + \sum_{k=1,\ldots,h_2+1} l_{S_2^{-1}(k)} (1+b)^{h_2+1-k} \\
&= \tilde{t}(1+b)^{n+h_2+1} + l_{2n+1} + \sum_{k=1,\ldots,h_2} l_{S_2^{-1}(k)} (1+b)^{h_2+1-k} \\
&= \tilde{t}(1+b)^{n+h_2+1} + l_{2n+1} + \sum_{k=1,\ldots,h_2} l_{S_2^{-1}(h_2+1-k)} (1+b)^{h_2+1-(h_2+1-k)}
\end{aligned}$$

$$= \tilde{t}\,(1+b)^{n+h_2+1} + l_{2n+1} + \sum_{k=1,\dots,h_2} l_{S_2^{-1}(h_2+1-k)}\,(1+b)^k.$$

Define $\bar{f} = \sum_{k=1,\dots,n} (f_1(k) + f_2(k))\,(1+b)^k$, where

$$f_1(k) = \begin{cases} l_{S_1^{-1}(k)}, & 1 \le k \le h_1, \\ 0, & \text{else}, \end{cases} \quad f_2(k) = \begin{cases} l_{S_2^{-1}(h_2+1-k)}, & 1 \le k \le h_2, \\ 0, & \text{else}. \end{cases}$$

Then, with $d = (1+b)^{n+h_2+1} - (1+b)^{h_1}$, we have

$$\begin{aligned} &\Phi - t_{\min} \ge C_{\max} - t_{\min} \\ &\iff 4q \ge \tilde{C}d + l_{2n+1} + \sum_{k=1,\dots,h_1} l_{S_1^{-1}(k)}\,(1+b)^k + \sum_{k=1,\dots,h_2} l_{S_2^{-1}(h_2+1-k)}\,(1+b)^k \\ &\iff 2q \ge \tilde{C}d + \sum_{k=1,\dots,n} (f_1(k) + f_2(k))\,(1+b)^k = \tilde{C}d + \bar{f}. \end{aligned} \tag{4}$$

Because $h_1 \le n$ and $h_2 \ge 0$, there is $d > 0$. In the following, we show that the minimum of $\bar{f}$ equals $2q$, hence (4) requires $\tilde{C} = 0$. For any $i, j = 1, 2$ such that $i \ne j$, if $f_i(k) = 0$ for some k while $f_j(k+1) > 0$, then $\bar{f}$ is not optimal: it decreases for $f_i(k) > 0$ and $f_j(k+1) = 0$ because $(1+b)^k < (1+b)^{k+1}$. By this argument and because $h_1 + h_2 = 2h$, it follows that $h_1 = h_2 = h$. Moreover, a minimal $\bar{f}$ satisfies $f_i(k-1) \ge f_j(k)$ for $k = 2, \dots, h$ and any $i, j = 1, 2$, because $(1+b)^{k-1} < (1+b)^k$. This is the case for an optimal S in view of (1) and (2). Therefore, S satisfies $\{S(2k-1),\ S(2k)\} = \{h+1-k,\ h+k\}$ for $k = 1, \dots, h$, in any order, and

$$\begin{aligned} \bar{f} &= \sum_{k=1,\dots,n} (f_1(k) + f_2(k))\,(1+b)^k \\ &= \sum_{k=1,\dots,h} (l_{2k-1} + l_{2k})\,(1+b)^k \\ &= \sum_{k=1,\dots,h} \left(x_{2k-1}\,(1+b)^{-k} + x_{2k}\,(1+b)^{-k} \right) (1+b)^k \\ &= \sum_{k=1,\dots,h} x_{2k-1} + x_{2k} = 2q. \end{aligned}$$

As the minimum of $\bar{f}$ equals $2q$, (4) requires $\tilde{C} = 0$. In view of (2) and because $S_1^{-1}(h+1-k) \in \{2k-1,\ 2k\}$ for $k = 1, \dots, h$ for a minimal $\bar{f}$, there is

$$\tilde{C} = -q\,(1-a)^h + \sum_{j=1,\dots,h} l_{S_1^{-1}(j)}\,(1-a)^{h-j}$$

$$
\begin{aligned}
&= -q\,(1-a)^h + \sum_{k=1,\ldots,h} l_{S_1^{-1}(h+1-k)}\,(1-a)^{h-(h+1-k)} \\
&= -q\,(1-a)^h + \sum_{k=1,\ldots,h} \left(x_{S_1^{-1}(h+1-k)}\,(1+b)^{k-h-1}\right)(1+b)^{1-k} \\
&= -q\,(1-a)^h + (1-a)^h \sum_{j=1,\ldots,h} x_{S_1^{-1}(j)}.
\end{aligned}
$$

Then, $\tilde{C} = 0 \iff \sum_{j=1,\ldots,h} x_{S_1^{-1}(j)} = q \iff \{x_{S_1^{-1}(j)} \mid j = 1,\ldots,h\} = X_1$, where X_1 is a solution for the EVEN-ODD PARTITION problem.

Hence, if for the given instance of EVEN-ODD PARTITION there exists a solution, then there exists a solution for the corresponding $\hat{P}_\Phi$ instance and vice versa. Because the above reduction is polynomial and because, given a correct partition, S and $C_{\max}$ can be obtained in polynomial time, $\hat{P}_\Phi$ is $\mathcal{NP}$-complete. □

References

[1] T.-C. E. Cheng, Q. Ding, M. Y. Kovalyov, A. Bachman, A. Janiak, Scheduling jobs with piecewise linear decreasing processing times, *Naval Research Logistics*, **50** (2003), 531–554.

[2] M. H. Farahani, L. Hosseini, Minimizing cycle time in single machine scheduling with start time-dependent processing times, *International Journal of Advanced Manufacturing Technology*, **64** (2013), 1479–1486.

[3] M. R. Garey, R. E. Tarjan, G. T. Wilfong, One-processor scheduling with symmetric earliness and tardiness penalties, *Mathematics of Operations Research*, **13** (1988), 330–348.

[4] S. Gawiejnowicz, *Time-Dependent Scheduling*, Springer, Berlin-Heidelberg, 2008.

[5] F. Jaehn, H. A. Sedding, Scheduling with time-dependent discrepancy times, *Journal of Scheduling*, **19** (2016), 737–757.

[6] W. Kubiak, S. L. van de Velde, Scheduling deteriorating jobs to minimize makespan, *Naval Research Logistics*, **45** (1998), 511–523.

[7] O. I. Melnikov, Y. M. Shafransky, Parametric problem in scheduling theory, *Cybernetics and System Analysis*, **15** (1980), 352–357.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

The multi-scenario scheduling problem to maximize the weighted number of just-in-time jobs

Dvir Shabtay *
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Miri Gilenson
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Keywords: just-in-time scheduling, single machine, flow shop, multi-scenario scheduling

1 Introduction

In this paper we study a set of single-machine and flow-shop scheduling problems with scenario-dependent jobs' parameters. Our aim is to find a schedule that is feasible under all scenarios and maximizes the weighted number of jobs completed *exactly* at their due-date, i.e., completed in a *just-in-time* (*JIT*) mode. Our set of problems can be formally defined as follows.

We are given a set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ of n independent, non-preemptive jobs that are available for processing at time zero and have to be processed either on a single machine or in a two-machine flow-shop scheduling system. In a two-machine flow-shop scheduling system, all jobs have to be processed on the first machine, M_1, and only then on the second machine, M_2. We assume that there is a set of q different scenarios $\mathcal{S} = \{s_1, s_2, \ldots, s_q\}$, each of which defines a different possible set of job's parameters. Let $p_{kj}^{(s_i)}$ be the *processing time* of job J_j on machine M_k under scenario $s_i \in \mathcal{S}$. Moreover, let $d_j^{(s_i)}$ be the *due date* and $w_j^{(s_i)}$ be the *weight* of job J_j under scenario $s_i \in \mathcal{S}$. Throughout the paper we assume that all due-dates are scenario independent, that is, $d_j^{(s_i)} = d_j$ for $j = 1, \ldots, n$.

A partition of set $\mathcal{J}$ into two disjointed subsets $\mathcal{E}$ and $\mathcal{T}$ is considered to be a *feasible partition* (or a *feasible schedule*), if it is possible to schedule the jobs belonging to set $\mathcal{E}$ on the machines, such that they are all completed in a JIT mode, under *any* of the q different scenarios. In such a case we say that $\mathcal{E}$ is a feasible set of JIT jobs. We use the weighted number of jobs in set $\mathcal{E}$, i.e. the set of JIT jobs, as our

* Speaker, email: dvirs@bgu.ac.il

scheduling criterion. Accordingly, the quality of any feasible solution is measured by q different objective values, $F^{(s_i)} = \sum_{J_j \in \mathcal{E}} w_j^{(s_i)}$ for $i = 1, \ldots, q$.

Multi-scenario scheduling problems were mainly studied by using the robust optimization approach (see, e.g., Yang and Yu [11], Aloulou and Della Croce [1] and Mastrolilli et al. [8]). The main disadvantage, however, of this approach is that decisions are made based on the worst-case realizations of the uncertain parameters. We suggest using an alternative approach, where the objective value under each of the possible scenarios is taken into account. Accordingly, we subdivide our problems into the following set of problem variations which are commonly analyzed in the multi–criteria literature:

- *Problem Variation 1* (*PV1*): Given a set of non-negative θ_i parameters, find a feasible schedule that maximizes a linear combination of objectives $F^{(s_1)}$, $F^{(s_2)}, \ldots, F^{(s_q)}$, i.e., that maximizes $\sum_{i=1}^{q} \theta_i F^{(s_i)}$.
- *Problem Variation 2* (*PV2*): Find a feasible schedule that maximize $F^{(s_1)}$ subject to $F^{(s_i)} \geq K_i$ for $i = 2, \ldots, q$, where K_i is a given lower bound on the value of $F^{(s_i)}$.
- *Problem Variation 3* (*PV3*): Given parameters $K_1, \ldots, K_q$, determine if there exists a feasible schedule with $F^{(s_i)} \geq K_i$ for $i = 1, \ldots, q$.
- *Problem Variation 4* (*PV4*): Identify a single *Pareto-Optimal* (*PO*) schedule for each *PO* point, where a *feasible* schedule σ is considered to be a *PO schedule* with respect to $F^{(s_1)}, \ldots, F^{(sq)}$ if there does not exist another *feasible* schedule σ' such that $F^{(s_i)}(\sigma') \geq F^{(s_i)}(\sigma)$ for $i = 1, \ldots, q$, with at least one of these inequalities being strict. The corresponding *PO point* of schedule σ is given by $\phi(\sigma) = (K_1, \ldots, K_q)$, where $K_i = F^{(s_i)}(\sigma)$ for $i = 1, \ldots, q$.

Note that if the weights are scenario-independent, the multi-criteria approach becomes irrelevant and our objective is simply to find a feasible schedule that maximizes $\Sigma_{J_j \in \mathcal{E}} w_j$. If in addition to being scenario-independent, all weights are identical the objective is simply to find a feasible schedule that maximizes $|\mathcal{E}|$.

2 Related research

In this section we review the relevant results for the single machine and the flow-shop scheduling systems with the objective of maximizing the weighted number of JIT jobs, starting with the single-scenario counterpart. In the review, we use the

standard three-field notation $\alpha\,|\beta|\,\gamma$, introduced by Graham et al. [5] to describe scheduling problems. The α field describes the machine environment. In this field 1 stands for a single-machine environment, while *Fm* stands for a flow-shop machine environment with m machines. The β field defines the job-processing characteristics and constraints, and the γ fields includes the objective.

Lann and Mosheiov [7] study the $1||\sum_{J_j\in\mathcal{E}} w_j$ problem and presented an $O(n^2)$ time optimization algorithm for its solution. They also provided a faster $O(n\log n)$ time optimization algorithm to solve the non-weighted version of the problem, i.e., to solve the $1||\,|\mathcal{E}|$ problem. Choi and Yoon [2] proved that the $F2||\sum_{J_j\in\mathcal{E}} w_j$ problem is $\mathcal{NP}$-hard by a reduction from the PARTITION problem. However, they left an open question whether the $F2||\sum_{J_j\in\mathcal{E}} w_j$ problem is strongly or ordinary $\mathcal{NP}$-hard. Shabtay and Bensoussan [10] resolved this question by constructing a pseudo-polynomial time algorithm to solve the $F2||\sum_{J_j\in\mathcal{E}} w_j$ problem (showing that the problem is ordinary $\mathcal{NP}$-hard). Shabtay and Bensoussan also showed how the pseudo-polynomial time algorithm can be converted into a fully polynomial time approximation scheme (FPTAS), which provides a $(1-\varepsilon)$-approximation in $O(\frac{n^4}{\varepsilon}\log\frac{n}{\varepsilon})$ time. This result was later improved by Elalouf et al. [4], who accelerated the running time of the FPTAS by a factor of $n\log\frac{n}{\varepsilon}$. Shabtay [9] provided an $O(n^3)$ time algorithm for solving the $F2||\,|\mathcal{E}|$ problem, improving an earlier result by Choi and Yoon [2] by a linear factor. Choi and Yoon [2] proved that the $F3||\sum_{J_j\in\mathcal{E}} w_j$ problem is strongly $\mathcal{NP}$-hard. Shabtay [9] studied several special cases of the $Fm||\sum_{J_j\in\mathcal{E}} w_j$ problem. He proved that the problem is $\mathcal{NP}$-hard even when processing times are machine-independent. He also showed that this case becomes tractable (specifically, solvable in $O(n^4)$ time) when all weights are identical. If the processing times are job-independent (but machine-dependent), then the $Fm||\sum_{J_j\in\mathcal{E}} w_j$ problem is solvable in $O(n^3)$ time. This time complexity is pushed down to $O(n\log n)$ if in addition to job-independent processing times, it holds that all weights are identical. Finally, Shabtay [9] showed that the $Fm||\sum_{J_j\in\mathcal{E}} w_j$ problem is solvable in $O(mn^2)$ time if there is a no-wait restriction, i.e., if jobs are not allowed to wait between machines.

As far as we know, only Chung and Choi [3] studied a multi-scenario scheduling problem with the scheduling criterion of maximizing the weighted number of JIT jobs. They focused on the single machine case with the objective of finding a feasible schedule that minimizes the *maximal regret* $F=\max_{i\in\{1,\ldots,q\}}\{F^{(s_i)}-F^{(s_i)}_{opt}\}$, where $F^{(s_i)}_{opt}$ is the optimal solution value under scenario s_i. Chung and Choi study both cases where the number of scenarios, q, is either fixed or arbitrary. For arbitrary q, Chung and Choi proved that (*i*) the $1|w^{(s_i)}_j|\max_{i\in\{1,\ldots,q\}}\{F^{(s_i)}-F^{(s_i)}_{opt}\}$ problem is strongly $\mathcal{NP}$-hard even if $p_j=1$ for $j=1,\ldots,n$; and that (*ii*) the

$1|p_j^{(s_i)}|\max_{i\in\{1,\dots,q\}}\{F^{(s_i)}-F_{opt}^{(s_i)}\}$ problem is solvable in $O(n^2)$ time. For fixed q, Chung and Choi proved that (*i*) the $1|w_j^{(s_i)}, q=2|\max_{i\in\{1,\dots,q\}}\{F^{(s_i)}-F_{opt}^{(s_i)}\}$ problem is ordinary $\mathcal{NP}$-hard; that (*ii*) the $1|d_j^{(s_i)}|\max_{i\in\{1,\dots,q\}}\{F^{(s_i)}-F_{opt}^{(s_i)}\}$ problem is strongly $\mathcal{NP}$-hard even if $w_j^{(s_i)}=1$ and $p_j^{(s_i)}=1$; that (*iii*) the $1|d_j^{(s_i)}, p_j=1, q=2|\max_{i\in\{1,\dots,q\}}\{F^{(s_i)}-F_{opt}^{(s_i)}\}$ problem is solvable in polynomial time; and that (*iv*) the $1|p_j^{(s_i)}, w_j^{(s_i)}|\max_{i\in\{1,\dots,q\}}\{F^{(s_i)}-F_{opt}^{(s_i)}\}$ problem admits an FPTAS.

3 Our results

Our complexity results are summarized in Tables 1 and 2 below. The first column in each table presents the set of uncertain parameters, while the first raw presents the problem variation. In all cases the term 'Hard' implies that the corresponding problem is ordinary $\mathcal{NP}$-hard (or $\mathcal{NP}$-complete) when q is fixed, and strongly $\mathcal{NP}$-hard (or $\mathcal{NP}$-complete) when q is arbitrary. Note that when weights are scenario-independent, there is a single complexity result for the corresponding problem in which the objective value is scenario-independent.

Table 1: Computational complexity results for single machine setting

	PV1	*PV2*	*PV3*	*PV4*
$p_j^{(s_i)}$	$O(n\max\{n,q\})$	$O(n\max\{n,q\})$		
$w_j^{(s_i)}$	$O(n\max\{n,q\})$	Hard		
$p_j^{(s_i)}, w_j^{(s_i)}$	$O(n\max\{n,q\})$	Hard		

Table 2: Computational complexity results for the two-machine flow-shop setting

	PV1	*PV2*	*PV3*	*PV4*
$p_{1j}^{(s_i)}$		Hard*		
$p_{2j}^{(s_i)}$		Hard**		
$w_j^{(s_i)}$		Hard		
$p_{1j}^{(s_i)}, p_{2j}^{(s_i)}, w_j^{(s_i)}$		Hard		

* Even if $w_j=1$ for $j=1,\dots,n$.
** Solvable in $O(n^3)$ time if $j=1,\dots,n$.

References

[1] M. A. Aloulou, F. Della Croce, Complexity of single machine scheduling problems under scenario-based uncertainty, *Operations Research Letters*, **36** (2008), 338–342.

[2] B. C. Choi, S. H. Yoon, Maximizing the weighted number of just-in-time jobs in flow-shop scheduling, *Journal of Scheduling*, **10** (2007), 237–243.

[3] D. Y. Chung, B. C. Choi, Just-in-time scheduling under scenario-based uncertainty, *Asia-Pacific Journal of Operational Research,* **30** (2013), 1250055.

[4] A. Elalouf, E. Levner, H. Tang, An improved FPTAS for maximizing the weighted number of just-in-time jobs in a two-machine flow shop problem, *Journal of Scheduling*, **16** (2013), 429–435.

[5] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, **5** (1979), 287–326.

[6] H. Kung, F. Luccio, F. Preparata, On finding the maxima of a set of vectors, *Journal of the Association for Computing Machinery*, **22** (1975), 469–476.

[7] A. Lann, G. Mosheiov, Single machine scheduling to minimize the number of early and tardy jobs, *Computers & Operations Research*, **23** (1996), 769–781.

[8] M. Mastrolilli, N. Mutsanas, O. Svensson, Single machine scheduling with scenarios, *Theoretical Computer Science,* **477** (2013), 57–66.

[9] D. Shabtay, The just-in-time scheduling problem in a flow-shop scheduling system, *European Journal of Operational Research*, **216** (2012), 521-532.

[10] D. Shabtay, Y. Bensoussan, Maximizing the weighted number of just-in-time jobs in several two-machine scheduling systems, *Journal of Scheduling*, **15** (2012), 39–47.

[11] J. Yang, G. Yu, On the robust single machine scheduling problem, *Journal of Combinatorial Optimization*, **6** (2002), 17–33.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

Single machine scheduling to minimize total completion time under positional and cumulative deterioration effects

Alan Soper *
Greenwich University, London, United Kingdom

Vitaly A. Strusevich
Greenwich University, London, United Kingdom

Keywords: scheduling, single machine, cumulative effect, total completion time, V-shape

1 Introduction

Since the early 1990s, there has been a considerable interest in enhanced scheduling models in which the processing times of jobs are affected by their locations in the schedule. Mathematically, this is formalized in terms of various time-changing effects. In this paper, we clarify the status of one of the single machine problems with a time-changing effect.

In the range of problems with changing times, we are given set $N = \{1, 2, \ldots, n\}$ of jobs to be processed on a single machine. Each job $j \in N$ is associated with its *normal* processing time p_j. It is convenient to think of normal processing times as the time required under normal processing conditions of the machine, which might change during the processing, thereby affecting the actual processing times.

In the literature on scheduling with changing processing times, traditionally there is a distinction between the so-called deterioration effects and learning effects. Informally, under a *deterioration* effect, the later a job is placed in a schedule, the longer it takes to process it. This phenomenon is often found in manufacturing: if a machine loses its initial processing quality, it increases the actual processing times of some later scheduled jobs. Under a *learning* effect, the opposite is observed: the later a job is scheduled, the shorter its actual processing time is.

Consideration of time-changing effects should not be limited to monotone effects only, such as deterioration and learning. For example, if a human operator processes jobs on certain equipment, then during the process the equipment might be subject to wear and tear, i.e., it might deteriorate with time, however, the operator

* Speaker, email: A.J.Soper@gre.ac.uk

simultaneously gains additional skills by learning from experience. This gives rise to a combined effect having a non-trivial influence on the actual processing times.

Time-changing effects are represented by explicit formulae for the actual processing time of a job affected. There are three main types of so-called pure effects studied in the literature, which in accordance with the recent monograph by Strusevich and Rustogi [7] can be informally classified as follows:

- *positional* effects: the actual processing time of a job is a function of its normal processing time and the position it takes in a schedule; see a focused survey by Rustogi and Strusevich [6] and a discussion in the book by Agnetis et al. [1];
- *start-time dependent* effects: the actual processing time of a job is a function of its normal processing time and its start time in a schedule; see the book by Gawiejnowicz [2] which gives a detailed exposition of scheduling models with this effect;
- *cumulative* effects: the actual processing time of a job depends on its normal processing time and a function of the normal processing times of previously scheduled jobs; see Kuo and Yang [3, 4], where a similar effect is introduced.

In this paper, we focus on job-independent positional and cumulative effects.

If job j is sequenced in position $\pi(r)$ of permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, its completion time is denoted either by $C_j(\pi)$ or by $C_{\pi(r)}$, whichever is more convenient. Let $\Phi(\pi)$ denote an objective function to be minimized. Popular objective functions include the maximum completion time $C_{\max}(\pi)$, also known as the makespan and the sum of the completion times $F(\pi) = \sum_{j \in N} C_j(\pi)$, also known as the total completion time. Given a permutation $\pi = (\pi(1), \ldots, \pi(n))$ of jobs, let the actual processing time of a job $j = \pi(r)$ scheduled in position r, $1 \leq r \leq n$, be denoted by $p_j(r)$.

A *job-dependent positional* effect is given by

$$p_j(r) = p_j g(r), \ j \in N, 1 \leq r \leq n, \tag{1}$$

where $g(r)$ is called a (job-independent) *positional factor*. The values $g(r)$, where $1 \leq r \leq n$, form an array of positional factors that is common for all jobs. If array $g(r)$, $1 \leq r \leq n$, is monotone non-decreasing (or non-increasing), then we have a situation of positional deterioration (or of positional learning, respectively). For the most general job-independent positional effect, we make no assumption regarding the monotonicity of array $g(r)$, $1 \leq r \leq n$. It is often assumed without loss of generality that $g(1) = 1$, which guarantees that for a job that is sequenced first, i.e., in position $r = 1$, the actual processing time is equal to its normal time.

We denote the single machine problems of minimizing an objective function Φ subject to the effect (1) by $1\left|p_j(r)=p_j g(r)\right|\Phi$.

One of the most general variants of a pure *cumulative* effect defines $p_j(r)$ as

$$p_j(r)=p_j f(P_r), \tag{2}$$

where $P_r=\sum_{h=1}^{r-1}p_{\pi(h)}$ is the sum of the normal processing times of the earlier sequenced jobs. In (2), f is a continuous differentiable function, common to all jobs. In the case of learning $f:[0,+\infty)\rightarrow(0,1]$ is a non-increasing function, while in the case of deterioration $f:[0,+\infty)\rightarrow[1,+\infty)$ is a non-decreasing function. We denote the single machine problems of minimizing an objective function Φ subject to the effect (2) by $1\left|p_j(r)=p_j f(P_r)\right|\Phi$.

Quite often a permutation of jobs that defines an optimal schedule is found by applying a *priority rule,* i.e., by sorting the jobs in accordance with certain priorities. The most popular rules are the SPT and the LPT rule. Recall that if the jobs are numbered in accordance with the *Shortest Processing Time* (*SPT*) rule then $p_1\leq p_2\leq\cdots\leq p_n$, while if they are numbered in accordance with the *Longest Processing Time* (*LPT*) rule then $p_1\geq p_2\geq\cdots\geq p_n$.

A permutation of jobs $\pi=(\pi(1),\pi(2),\ldots,\pi(n))$ is called *V-shaped* with respect to p_j if it consists of a non-increasing subsequence followed by a non-decreasing subsequence, where one of the subsequences may be empty. Often an optimal permutation belongs to the class of V-shaped sequences, and it is quite common to see the term 'V-shaped' in the title of papers; see, e.g., Mosheiov [5].

2 Our results

Positional effects. Problem $1\left|p_j(r)=p_j g(r)\right|\sum C_j$ with a pure positional effect (1) is known to be solvable in $O(n\log n)$ time by a matching algorithm; see Rustogi and Strusevich [6], Strusevich and Rustogi [7]. In the case of a positional learning effect defined by $1=g(1)\geq g(2)\geq\cdots\geq g(n)$ an SPT permutation is optimal, while in the case of the positional deterioration effect $1=g(1)\leq g(2)\leq\cdots\leq g(n)$ the problem in general does not admit a solution by a priority rule (cf. Strusevich and Rustogi [7]). In this paper, for problem $1\left|p_j(r)=p_j g(r)\right|\sum C_j$ under an arbitrary deterioration effect we derive sufficient conditions on g, which guarantee that an optimal permutation is V-shaped. In particular, the conditions hold for (*i*) polynomial function $g(r)=r^A$, $A>1$; (*ii*) exponential function $g(r)=\gamma^r$, $1<\gamma<2$; (*iii*) arbitrary concave non-decreasing function g. On the other hand, for a non-decreasing convex function g an optimal permutation need not be V-shaped.

Cumulative effects. The conditions under which problems $1\left|p_j(r)=p_j f(P_r)\right|\Phi$ and their generalizations can be solved by a priority rule are presented by Strusevich and Rustogi [7]. In particular, for problem $1\left|p_j(r)=p_j f(P_r)\right|\sum C_j$ under a learning effect the SPT permutation is optimal if f is convex non-increasing. On the other hand, under a cumulative deterioration effect defined by a concave non-decreasing function an optimal solution cannot be found by a priority rule. In this paper, we additionally show that for the latter problem an optimal permutation need not even be V-shaped.

Let us define $P=\sum_{j=1}^{n} p_j$. One of popular polynomial normalized concave effects is in the form of

$$f\left(\frac{P_r}{P}\right)=\left(1+\frac{P_r}{P}\right)^A,\ 0<A<1. \tag{3}$$

A cumulative effect given by a function $f(X/P):[0,P]\to[0,+\infty]$ is called normalized. A normalized function f provides a more realistic rate of growth of actual processing times. In this paper, we prove that problem $1\left|p_j(r)=p_j f(P_r/P)\right|\sum C_j$ under a cumulative deterioration effect defined by a normalized concave twice differentiable function $f(y)$, where $y=P_r/P$, can be solved by the SPT rule provided

$$4f(y)+f''(y)\geq 0.$$

The established sufficient condition holds for normalized concave effect (3).

References

[1] A. Agnetis, J-C. Billaut, S. Gawiejnowicz, D. Pacciarelli, A. Soukhal, *Multiagent Scheduling: Models and Algorithms*, Springer, Berlin-Heidelberg, 2014.

[2] S. Gawiejnowicz, *Time-Dependent Scheduling*, Springer, Berlin-Heidelberg, 2008.

[3] W.-H. Kuo, D.-L. Yang D-L, Minimizing the makespan in a single machine scheduling problem with a time-based learning effect, *Information Processing Letters*, **97** (2006), 64–67.

[4] W.-H. Kuo, D.-L. Yang, Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect, *European Journal of Operational Research*, **174** (2006), 1184–1190.

[5] G. Mosheiov, *V*-shaped policies for scheduling deteriorating jobs, *Operations Research*, **39** (1991), 979–991.

[6] K. Rustogi, V. A. Strusevich, Simple matching vs linear assignment in scheduling models with positional effects: a critical review, *European Journal of Operational Research*, **22** (2012), 393–407.

[7] V. A. Strusevich, K. Rustogi, *Scheduling with Times-Changing Effects and Rate-Modifying Activities*, Springer, Berlin-Heidelberg, 2017.

The Second International Workshop on Dynamic Scheduling Problems
Adam Mickiewicz University in Poznań, June 26th – 28th, 2018

Some remarks on preemptive scheduling of jobs with a learning effect

Marcin Żurowski *
Adam Mickiewicz University in Poznań, Poznań, Poland

Keywords: scheduling, parallel machines, job preemption, learning effect, maximum completion time

1 Introduction

Let J be a set of $n \geq 2$ independent jobs scheduled without idle times on two parallel identical machines M_1 and M_2 available from time 0. The actual processing times of jobs are position-dependent. The jobs may be preempted in the way described in Section 3. Both parts of a preempted job are treated like two new independent and non-preemptable jobs, which cannot be executed on both machines at the same time or on the same machine in different time intervals. The objective is to minimize the schedule length $C_{\max}$. Following [9], I call the problem *the Preemptive Scheduling with a Learning Effect* (*PSLE*) *problem*.

In this paper, I consider problem PSLE with the learning effect in which the actual processing time of job J_j is equal to $p_{j,r} = p_j r^a$, where p_j is the basic processing time of job J_j, r is the position of job J_j in a schedule and $a < 0$ is the learning index of the job. To the best of my knowledge, with the exception of [9], no preemptive scheduling problems with a learning effect were considered earlier.

2 Related research

Job preemption is important in scheduling theory, since it usually decreases the schedule length (Coffman [3]). In the literature there are described a few models of job preemption. The most known is the one related to McNaughton's algorithm (McNaughton [7]), where any job can be preempted at any time and resumed later with no cost. There are also known other models of job preemption, such as restricted job preemption (Ecker and Hirschberg [4]) or job preemption at integer

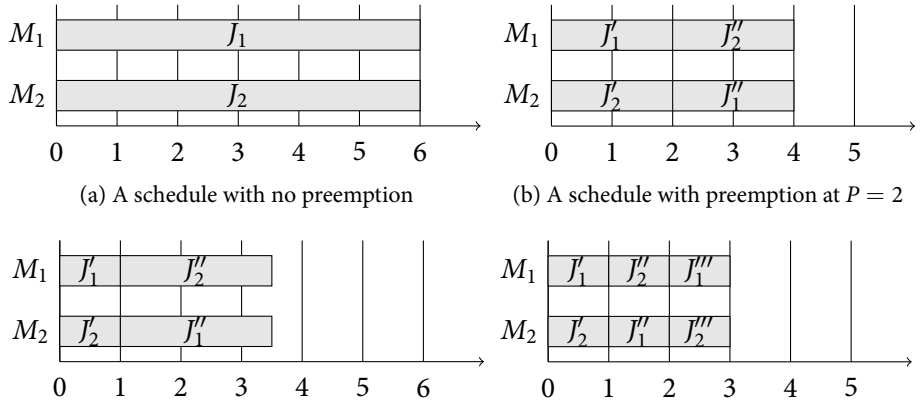
* Speaker, email: marcin.zurowski@amu.edu.pl

time moments (Baptiste et al. [2]). All these models are applied only to scheduling problems with fixed job processing times.

In practice we often deal with problems having variable job processing times. The variable job processing times may depend on the number of already executed jobs (Gawiejnowicz [5]), the starting times of the jobs (Gawiejnowicz [6]) or the positions of the jobs in a schedule (Agnetis et al. [1], Strusevich and Rustogi [8]). Job processing times which depend on the positions of the jobs are called *position-dependent* processing times. The position-dependent processing times are related to so-called *learning effect* (Agnetis et al. [1], Strusevich and Rustogi [8]).

3 Preemption of position-dependent jobs

Let us assume that in problem PSLE we are given $n = 2$ jobs with basic processing times $p_1 = p_2 = 6$ and the learning index is equal to $a = -1$. Schedule without preemption is illustrated in Fig. 1a. If we preempt both jobs at time $P = 2$ and resume these jobs on opposite machines, then we get a shorter schedule (see Fig. 1b). The length of schedule is dependent on the moment when preemption occurred (see Fig. 1c). Furthermore, if we preempt the jobs twice, the length of the schedule will be decreased yet more (see Fig. 1d).



(a) A schedule with no preemption

(b) A schedule with preemption at $P = 2$

(c) A schedule with preemption at $P = 1$

(d) A schedule with preemption at $P_1 = 1$ and $P_2 = 2$

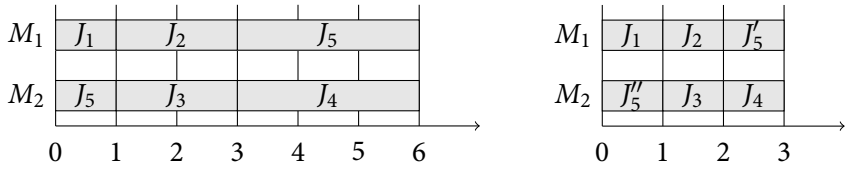
Figure 1: Example preemptive schedules with position-dependent jobs

Summarizing the considerations, we can state that in order to obtain an optimal preemptive schedule for problem PSLE, we should preempt some jobs infinitely many times, which is impossible in practice.

In view of the above, I assume further that one can preempt only the first job scheduled on machine M_2 and only once. By J_k'' I denote the first part of preempted job J_k scheduled in the first position on machine M_2. Similarly, by J_k' I denote the second part of preempted job J_k scheduled in the last position on machine M_1. The basic processing times of jobs J_k' and J_k'' satisfy the equation $p_k = p_k' + p_k''$.

The next two examples concern the instance of problem PSLE with $n = 5$ jobs and basic processing times $p_1 = 1, p_2 = 2, p_3 = 2, p_4 = 3$ and $p_5 = 4$.

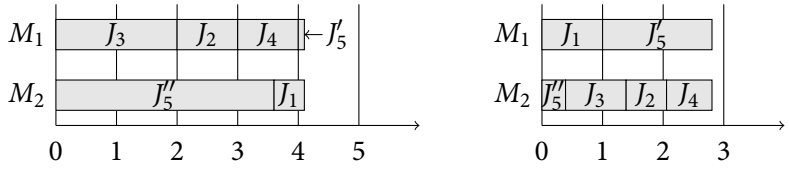
In Fig. 2 two preemptive schedules for the instance when jobs have fixed processing times ($a = 0$) are presented. In such a case, an optimal schedule can be obtained using McNaughton's algorithm [7]. One of 120 optimal schedules for this instance, σ, is presented in Fig. 2a, where $C_{\max}(\sigma) = 6$.



(a) Schedule σ (b) Schedule σ_1

Figure 2: Classic preemption vs. position-dependent preemption

However, if actual processing times of jobs are position-dependent, e.g. when $a = -1$, then finding an optimal preemptive schedule is a harder problem. An example schedule σ_1 for this case is presented in Fig. 2b, where $C_{\max}(\sigma_1) = 3$; however, the schedule σ_1 is not optimal.



(a) Schedule σ_2 (b) Optimal schedule $\sigma^\star$

Figure 3: Two different schedules with a single preempted job

The main cause of non-optimality of the schedule presented in Fig. 2b is the fact that the equality of total loads of both machines in a schedule does not guarantee that the schedule is optimal, as there may exist a shorter schedule with equal total loads of both machines. This is shown in Fig. 3a, which illustrates a schedule σ_2 with $C_{\max}(\sigma_2) = 4\frac{1}{10}$. Let us notice that machines M_1 and M_2 complete the processing of jobs at the same time. This schedule is longer than the schedule σ_1.

However, we can find a shorter schedule, $\sigma^\star$, if we change the sequence of jobs on machines M_1 and M_2 (see Fig. 3b). Then, both the machines complete the processing of jobs at the same time again, but schedule $\sigma^\star$ is shorter than σ_1, i.e. $C_{\max}(\sigma^\star) = 2\frac{29}{36} < C_{\max}(\sigma_1)$. Let us also notice that among all feasible schedules $\sigma^\star$ is a unique optimal schedule for this instance.

The above examples show that the change of a sequence of jobs assigned to a machine in a schedule for problem PSLE affects the length of the schedule. Hence, problem PSLE is not a direct generalization of problem $P2|pmtn|C_{\max}$.

4 Future research

Future research may concern the time complexity of problem PSLE or the construction of an exact algorithm for this problem.

Acknowledgement. I thank Stanisław Gawiejnowicz for help in improving the English of the paper.

References

[1] A. Agnetis, J-C. Billaut, S. Gawiejnowicz, D. Pacciarelli, A. Soukhal, *Multiagent Scheduling: Models and Algorithms*, Springer, Berlin-Heidelberg, 2014.

[2] Ph. Baptiste, J. Carlier, A. Kononov, M. Queyranne, S. Sevastyanov, M. Sviridenko, Integer preemptive scheduling on parallel machines, *Operations Research Letters*, **40** (2012), 440–444.

[3] E. G. Coffman jr (ed.), *Computer and Job-Shop Scheduling Theory*, John Wiley & Sons, New York, 1976.

[4] K. Ecker, R. Hirschberg, Task scheduling with restricted preemptions, *Lecture Notes in Computer Science*, **694** (1993), 464–475.

[5] S. Gawiejnowicz, A note on scheduling on a single processor with speed dependent on a number of executed jobs, *Information Processing Letters*, **57** (1996), 297–300.

[6] S. Gawiejnowicz, *Time-Dependent Scheduling*, Springer, Berlin-Heidelberg, 2008.

[7] R. McNaughton, Scheduling with deadlines and loss functions, *Management Science*, **6** (1959), 1–12.

[8] V. A. Strusevich, K. Rustogi, *Scheduling with Time-Changing Effects and Rate-Modifying Activities*, Springer, Berlin-Heidelberg, 2017.

[9] M. Żurowski, Preemptive scheduling of jobs with a learning effect on two parallel machines, *Proceedings of the 20th International Conference on Operations Research 2017*, Berlin, September 6-8, 2017, 587–592.

Indexes

Index of authors

Index of keywords