Adam Mickiewicz University in Poznań

THE FIRST INTERNATIONAL WORKSHOP ON DYNAMIC SCHEDULING PROBLEMS

ADAM MICKIEWICZ UNIVERSITY IN POZNAŃ
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
JUNE 30TH – JULY 1ST, 2016, POZNAŃ, POLAND

EXTENDED ABSTRACTS

IWDSP
POZNAŃ 2016

Adam Mickiewicz
University
in Poznań

The First International Workshop on Dynamic Scheduling Problems

Adam Mickiewicz University in Poznań
Faculty of Mathematics and Computer Science
June 30th – July 1st, 2016, Poznań, Poland

Extended Abstracts

IWDSP
Poznań 2016

This book contains extended abstracts of a plenary lecture and papers presented at the First International Workshop on Dynamic Scheduling Problems, June 30th–July 1st, 2016, Poznań, Poland.

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

ISBN: 978-83-937220-7-5 (eBook)

Publisher:

Polish Mathematical Society
Śniadeckich 8, 00-956 Warsaw
Poland

Welcome to IWDSP 2016

Dear participants,

on behalf of the Programme and Local Committees, I am pleased to welcome you to IWDSP 2016, The First 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 the event.

This workshop is 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 IWDSP 2016 is to present recent research in this important domain of scheduling theory.

The Programme Committee, supported by the members of Advisory Committee and external reviewers, selected for presentation at IWDSP 2016 talks submitted by the authors from Australia, Belarus, Canada, China, France, Israel, Poland, Russian Federation, Taiwan and United Kingdom. The quality of these talks allowed to prepare an attractive scientific programme of IWDSP 2016.

I wish you all the pleasant stay in Poznań and an enjoyable workshop, expressing the hope that you will find IWDSP 2016 stimulating for your further research.

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

Committees

Programme Committee

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

Gur MOSHEIOV, Hebrew University, Jerusalem, Israel

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

Advisory Committee

Florian JAEHN, University of Augsburg, Augsburg, Germany

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

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

Natalia SHAKHLEVICH, University of Leeds, Leeds, United Kingdom

Ji-Bo WANG, Shenyang Aerospace University, Shenyang, People's Republic of China

Local Committee

Stanisław GAWIEJNOWICZ (chair), Adam Mickiewicz University in Poznań

Jan KACZMAREK, Adam Mickiewicz University in Poznań

Zbigniew PALKA, Adam Mickiewicz University in Poznań

Bartłomiej PRZYBYLSKI, Adam Mickiewicz University in Poznań

Cezary SUWALSKI, Adam Mickiewicz University in Poznań

Marcin ŻUROWSKI, Adam Mickiewicz University in Poznań

Contents

Venue

Location

IWDSP 2016 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 a new part of Adam Mickiewicz University in Poznań, called Morasko campus (for details see attached map).

Communication

The quickest way to reach the venue of IWDSP 2016 is to take tram no. 12, 14 or 16 of Poznań Rapid Tram, get off at final stop ('Os. Sobieskiego', for timetable see `www.mpk.poznan.pl`) and make a short walk to the Morasko campus (for suggested route see attached map).

Registration

Registration desk for IWDSP 2016 will be located in the main hall of the Faculty of the Mathematics and Computer Science.

Registration will be possible on Wednesday, June 29th at evening, and on Thursday, June 30th at morning.

Presentation room

Plenary lecture and all presentations during IWDSP 2016 will take place in the Faculty Council Room (A1-33, see the attached map).

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

Coffee breaks room

Coffee breaks will take place in Professors' Club (A0-13) that is located one floor below the Faculty Council Room (for details see attached map).

Internet access

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

Details concerning usernames and passwords will be given during registration.

Welcome party, lunches and conference dinner

On Wednesday, June 29th in the afternoon, in Professors' Club of the Faculty of Mathematics and Computer Science there will be organized Welcome Party.

Lunches at June 30th and July 1st will be served in Professors' Club.

Conference dinner will be organized outside the Morasko campus. Details concerning the dinner will be given during registration.

Social programme

On Friday, July 1st in the morning, there will be organized a walking guided tour including the main tourist attractions of Poznań.

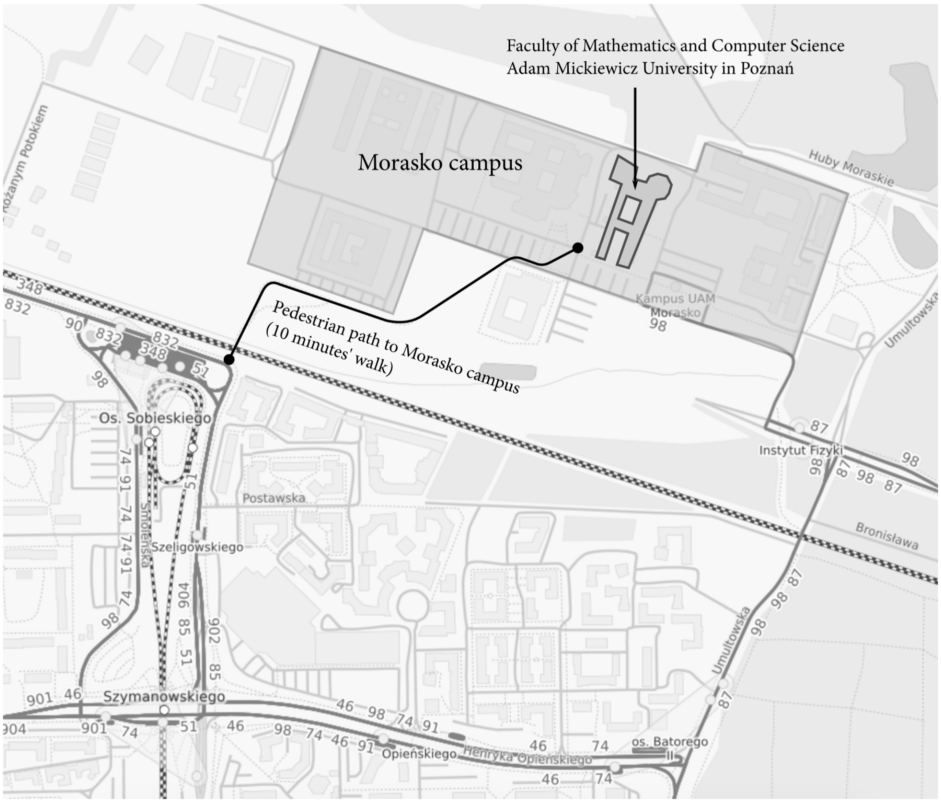

Figure 1: The main part of Morasko campus (source: Open Street Map)

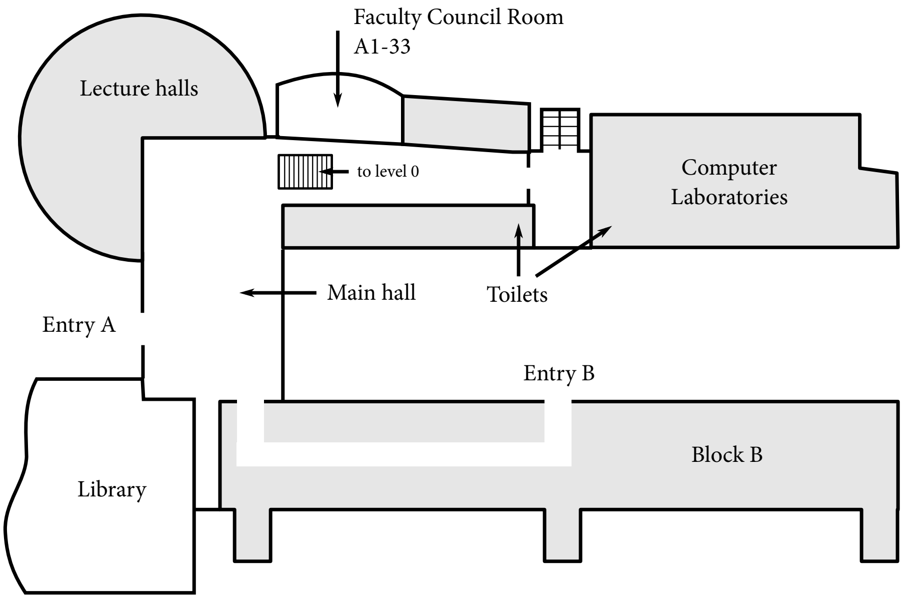

Figure 2: Level 1 (ground floor) of IWDSP 2016 venue

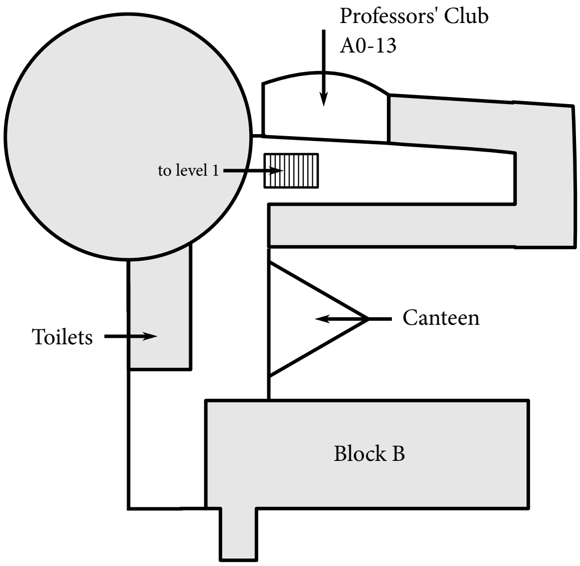

Figure 3: Level 0 (basement) of IWDSP 2016 venue

Programme

Thursday, June 30th, 2016

08:00 – 08:30	Registration
08:30 – 09:00	Opening
09:00 – 10:20	**Session no. 1**
	Speakers: Yakov M. Shafransky, Gur Mosheiov
	Chair: Vitaly A. Strusevich
10:20 – 10:40	Coffee break
10:40 – 12:00	**Plenary lecture**
	Speaker: Marc Uetz
	Chair: Stanisław Gawiejnowicz
12:00 – 13:30	Lunch break
13:30 – 14:50	**Session no. 2**
	Speakers: Bartłomiej Przybylski, Vitaly A. Strusevich
	Chair: Mikhail Y. Kovalyov
14:50 – 15:10	Coffee break
15:10 – 16:30	**Session no. 3**
	Speakers: Stanisław Gawiejnowicz, Alexander Kononov
	Chair: Gur Mosheiov
16:30 – 16:50	Coffee break
19:30 – 22:00	Conference dinner

Friday, July 1st, 2016

08:00 – 12:00	Guided tour
12:00 – 13:30	Lunch break
13:30 – 14:50	**Session no. 4**
	Speakers: Dvir Shabtay, Wiesław Kurc
	Chair: Alexander Kononov
14:50 – 15:10	Coffee break
15:10 – 16:30	**Session no. 5**
	Speakers: Mikhail Y. Kovalyov, Joanna Berlińska
	Chair: Dvir Shabtay
16:30 – 16:50	Coffee break
16:50 – 18:10	**Session no. 6**
	Speakers: Bertrand M-T. Lin, Krzysztof M. Ocetkiewicz
	Chair: Stanisław Gawiejnowicz
18:10	Closing

Plenary lecture

Approximation algorithms for scheduling under uncertainty

Marc Uetz[*]
University of Twente, Enschede, The Netherlands

Keywords: stochastic scheduling, approximation algorithms, linear programming

1 Introduction

We address machine scheduling problems with *stochastic* processing times, when the objective is to minimize the expected value of the total weighted completion time. We give an overview of *LP-based scheduling policies* with provable performance guarantees. The obtained performance bounds depend linearly on the squared coefficient of the variation of underlying processing time distributions.

We are focused on the problem to minimize the total weighted completion time on identical parallel machines, denoted $\mathrm{P}\,|\;|\sum w_jC_j$ in the three-field notation of Graham et al. [4], or on unrelated parallel machines, denoted $\mathrm{R}\,|\;|\sum w_jC_j$. Both problems are among the most-studied problems in the theory of scheduling. The former is strongly $\mathcal{NP}$-hard [8] and admits a PTAS [1], while the latter is Max$\mathcal{SNP}$-hard [6], and a $\frac{3}{2}$-approximation algorithm is known for it, e.g. [2].

We here address the variant where the processing times of jobs are random variables. The solution is then no longer a schedule, but a more complicated object called *non-anticipatory scheduling policy* [9]. Roughly speaking, for any state that the system might be in, a policy prescribes which job is to be scheduled next. For a given scheduling policy, the objective function $\sum_j w_jC_j$ is then a random variable, too, and our goal is to minimize its expected value, which by linearity of expectation equals $\sum_j w_j\mathbb{E}[C_j]$.

The problem under consideration is defined as follows. We are given a set of jobs J of cardinality n with job weights $w_j \in \mathbb{Z}_{>0}$, and a set of identical or unrelated parallel machines M of cardinality m. Moreover, we are given a random variable P_{ij} that describes the possible outcomes for job j's processing time on machine i, for every job $j \in J$ and every machine $i \in M$. For the special case with identical parallel machines, P_j denotes the random variable for job j's processing time. Each job j needs to be executed on any one of the machines $i \in M$, and each machine can process at most one job at a time. The random variables P_{ij} and P_j are stochastically independent across jobs.

[*] Plenary speaker, email: `m.uetz@utwente.nl`

2 Performance guarantees

We can compute, in polynomial time, scheduling policies with provable expected performance by means of *LP relaxations*. The basic idea is first to solve a certain LP relaxation, then derive from its solution a scheduling policy, and finally compare its expected performance with that of the relaxation. As the LP relaxation lower bounds the expected performance of *any* scheduling policy, this yields the desired performance guarantee.

The following table summarizes our (multiplicative) performance bounds for the two problems with identical and unrelated machines, respectively. Here, $\varepsilon > 0$ can be chosen arbitrarily small, and parameter Δ upper bounds the squared coefficient of variation $\mathbb{CV}^2[P_{ij}] = \frac{\mathbb{V}\mathrm{ar}[P_{ij}]}{\mathbb{E}^2[P_{ij}]}$ for all P_{ij}. The third column shows the results for uniform and exponential distributions for $\mathbb{CV}[P_{ij}] \le 1$.

Stochastic scheduling problem	Worst-case performance guarantee		Reference
	Arbitrary P_{ij}	$\mathbb{CV}[P_{ij}] \le 1$	
$P\,\|\,\|\,\mathbb{E}\left[\sum w_j C_j\right]$	$\frac{3}{2} + \frac{\Delta}{2} - \frac{\Delta+1}{2m}$	$2 - \frac{1}{m}$	[10]
$R\,\|\,\|\,\mathbb{E}\left[\sum w_j C_j\right]$	$\frac{3}{2} + \frac{\Delta}{2} + \varepsilon$	$2 + \varepsilon$	[12]

3 Techniques for relaxations and scheduling policies

The LP relaxation for the problem with identical parallel machines simply uses variables C_j^{LP} to denote the LP completion time of job j. It can be shown that the following is a valid LP relaxation:

$$\begin{array}{ll} \min & \sum_{j\in J} w_j\, C_j^{\mathrm{LP}} \\ \text{s.t.} & \sum_{j\in W} \mathbb{E}[P_j]\, C_j^{\mathrm{LP}} \ge f(W)\,, \quad \text{for all } W \subseteq J, \end{array}$$

where $f(W) := \frac{1}{2m}\left(\sum_{j\in W}\mathbb{E}[P_j]\right)^2 + \frac{1}{2}\sum_{j\in W}\mathbb{E}[P_j]^2 - \left(\frac{1}{2}-\frac{1}{2m}\right)\sum_{j\in W}\mathbb{V}\mathrm{ar}[P_j]$. These inequalities, for the special case of deterministic processing times where $\mathbb{V}\mathrm{ar}[P_j] = 0$, have been used also before [5]. This LP is a *polymatroid*, and therefore its optimal solution can be computed combinatorially using *Edmonds' greedy algorithm* [3]. The scheduling policy that achieves the performance bound $\frac{3}{2} + \frac{\Delta}{2} - \frac{\Delta+1}{2m}$ is the stochastic variant of *Smith's rule*: greedily schedule the jobs in order of non-increasing ratios $\frac{w_j}{\mathbb{E}[P_j]}$.

For the problem with unrelated machines, we use a *time-indexed LP relaxation*, based on variables x_{ijt} that denote the probability to start job j on machine i at time t. Using these variables, a valid LP relaxation for the problem is as follows:

$$
\begin{array}{lll}
\min & \sum_{j\in J} w_j\, C_j^{\mathrm{LP}} & \\
\text{s.t.} & \sum_{i\in M}\sum_{t\in\mathbb{Z}_{\geq 0}} x_{ijt} = 1 & \text{for all } j \in J,\\
& \sum_{j\in J}\sum_{t=0}^{s} x_{ijt}\, q_{ij\,s-t} \leq 1 & \text{for all } i \in M,\ s \in \mathbb{Z}_{\geq 0},\\
& C_j^{\mathrm{LP}} = \sum_{i\in M}\sum_{t\in\mathbb{Z}_{\geq 0}} x_{ijt}\,(t + \mathbb{E}[P_{ij}]) & \text{for all } j \in J,\\
& x_{ijt} \geq 0 & \text{for all } j \in J,\ i \in M,\ t \in \mathbb{Z}_{\geq 0}.
\end{array}
$$

Note that the variables C_j^{LP} are completely determined by x_{ijt} and only included for convenience. We can show that one can solve this LP with arbitrary precision in polynomial time. The algorithm to translate the LP solution into a scheduling policy is to assign job j to machine i at random, namely with probability $\sum_t x_{ijt}$. The jobs assigned to a given machine i are then again sequenced as in Smith's rule. That policy achieves the performance bound of $\frac{3}{2} + \frac{\Delta}{2} + \varepsilon$.

4 Concluding remarks

Our results can also be generalized to include problems, where jobs have individual release dates r_j. Then, the LPs are simply augmented with release dates, i.e. $C_j^{\mathrm{LP}} \geq r_j$, and also for identical parallel machines the scheduling policy is no longer Smith's rule, but we schedule the jobs in order of non-decreasing LP completion times C_j^{LP}. The performance bounds are equal to $2 + \Delta$ for identical parallel machines [11] and to $2 + \Delta + \varepsilon$ for unrelated machines [12].

It can also be shown that the dependence of the performance bounds on the squared coefficient of variation is asymptotically tight. The most intriguing open problem is to get rid of the dependence of the performance bounds on the coefficient of variation Δ, and obtain constant bounds for arbitrary processing time distributions. Some progress in that direction has very recently been obtained [7].

References

[1] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, M. Sviridenko, Approximation schemes for minimizing average weighted completion time with release dates, *The 40th Annual IEEE Symposium on Foundations of Computer Science*, IEEE, 1999, pp. 32–43.

[2] F. A. Chudak, A min-sum $\frac{3}{2}$-approximation algorithm for scheduling unrelated parallel machines, *Journal of Scheduling*, **2** (1999), 73–77.

[3] J. Edmonds, Submodular functions, matroids and certain polyhedra, *Proceedings of the International Conference on Combinatorics*, Gordon and Breach, 1970, pp. 69–87.

[4] 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.

[5] L. A. Hall, A. S. Schulz, D. B. Shmoys, J. Wein, Scheduling to minimize average completion time: Off-line and on-line approximation algorithms, *Mathematics of Operations Research*, **22** (1997), 513–544.

[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] S. Im, B. Moseley, K. Pruhs, Stochastic scheduling of heavy-tailed jobs, *The 32nd Symposium on Theoretical Aspects of Computer Science*, Leibniz International Proceedings in Informatics, **30** (2015), pp. 474–486.

[8] J. K. Lenstra, A. H. G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, *Annals of Discrete Mathematics*, **1** (1977), 343–362.

[9] R. H. Möhring, F. J. Radermacher, G. Weiss, Stochastic scheduling problems I: General strategies, *ZOR — Zeitschrift für Operations Research*, **28** (1984), 193–260.

[10] R. H. Möhring, A. S. Schulz, M. Uetz, Approximation in stochastic scheduling: The power of LP-based priority policies, *Journal of the ACM*, **46** (1999), 924–942.

[11] A. S. Schulz, Stochastic online scheduling revisited, *The 2nd Conference on Combinatorial Optimization and Applications*, Lecture Notes in Computer Science, **5165** (2008), pp. 448–457.

[12] M. Skutella, M. Sviridenko, M. Uetz, Unrelated machine scheduling with stochastic processing times, *Mathematics of Operations Research*, to appear, `DOI: 10.1287/moor.2015.0757`, 2016.

Extended abstracts

Scheduling data gathering with variable communication speed

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

Keywords: scheduling, data gathering networks, variable communication speed

1 Introduction

Gathering data from remote processors is an important stage of many applications. Running computations in distributed systems requires collecting results obtained by many workers. Wireless sensor networks collecting data find environmental, military, health and home applications [1]. Specific communication protocols have been designed for wireless sensor networks to increase data gathering efficiency [5, 6, 9]. General scheduling algorithms for data gathering were proposed in [2, 3, 4, 7]. It was assumed in these papers that the network parameters, such as the speed of communication and processing, are constant. However, in reality the communication speed often changes because of sharing communication links with other users, maintenance activities etc. Hence, in this work we study scheduling for data gathering networks with variable communication speed.

2 Problem formulation

We analyze a star network consisting of m nodes $P_1, P_2, \dots, P_m$ and a single base station. Node P_i has to transfer data of size α_i directly to the base station, possibly in many separate messages. Only one node can communicate with the base station at a time. We assume the linear model of communication, i.e., communication capabilities of node P_i are characterized by a single parameter called *communication rate* (inverse of speed). Thus, if node P_i communicates with rate C, then it transfers data of size x in time Cx. According to the methodology of *divisible load theory* [8], we assume that data size x is a rational number.

It is assumed that the communication rate of a link between node P_i and the base station changes in negligible time, when another application starts using it, and then remains constant for some period of time. In other words, it is a piecewise constant function of time. Let $t_0 = 0$ be the time when data gathering starts. The communication rates change at n moments $t_j > 0$, $1 \le j \le n$, $t_1 < t_2 < \cdots < t_n$

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

and $t_{n+1} = \infty$. The communication rate of node P_i in interval $[t_j, t_{j+1})$ will be denoted by $C_{i,j}$ for $j = 0, 1, \ldots, n$.

Problem DG-VS (scheduling data gathering with variable communication speed) consists in scheduling the communications between the nodes $P_1, P_2, \ldots, P_m$ and the base station so that the whole data is transferred in the shortest possible time T.

3 Offline algorithm

Theorem 1. *The offline version of DG-VS can be solved in polynomial time.*

Proof. Suppose that $T \in [t_k, t_{k+1})$ for given $k \in \{0, 1, \ldots, n\}$. Then, T can be found by solving the following linear program:

$$\text{minimize} \quad T \tag{1}$$

$$\sum_{j=0}^{k} x_{i,j} = \alpha_i \quad \text{for} \quad i = 1, 2, \ldots, m \tag{2}$$

$$\sum_{i=1}^{m} C_{i,j} x_{i,j} \leq t_{j+1} - t_j \quad \text{for} \quad j = 0, 1, \ldots, k-1 \tag{3}$$

$$\sum_{i=1}^{m} C_{i,k} x_{i,k} \leq T - t_k \tag{4}$$

In the above program, $x_{i,j}$ $(1 \leq i \leq m, 0 \leq j \leq k)$ are rational variables representing the amount of data sent by node P_i in interval $[t_j, t_{j+1})$. We minimize the data gathering completion time T. By constraints (2) each node transfers all its data to the base station. Inequalities (3) and (4) guarantee that the communications fit in the time intervals where they are assigned. Linear program (1)-(4) has $m(k+1)+1 = O(mn)$ variables and $m+k+1 = O(m+n)$ constraints, and hence it can be solved in polynomial time.

In order to solve DG-VS one can use binary search to find the smallest k for which program (1)-(4) has a solution. The number of binary search iterations is $O(\log n)$. The optimum communication schedule can be obtained from the values of variables $x_{i,j}$. Namely, in each interval $[t_j, t_{j+1})$ we schedule consecutively communications from nodes $P_1, P_2, \ldots, P_m$ of sizes $x_{1,j}, x_{2,j}, \ldots, x_{m,j}$ correspondingly, starting at time t_j. Thus, problem DG-VS can be solved in polynomial time. □

4 Online algorithm

Let us assume that although we do not know the exact ranges of communication speeds changes, the relative range of communication rate changes is known. Namely, if $C_i^{\max}$ and $C_i^{\min}$ are the maximum and the minimum communication rate of node P_i, then $\frac{C_i^{\max}}{C_i^{\min}} \leq \Delta$ for some given $\Delta > 1$. Such a situation may arise, e.g., when using a network with *QoS Percentage-Based Policing* [10].

Observation 1. *Any online scheduling strategy for DG-VS which does not introduce idle times in communication is Δ-competitive, since no communication can be more than Δ times slower than in the optimum schedule.*

Observation 2. *If no additional information is given, no online algorithm A consisting in reacting to changing communication speeds can be better than Δ-competitive.*

Proof. Let $m = 2$, $\alpha_1 = \alpha_2 = 1$, $C_{1,0} = C_{2,0} = 1$. We can assume without loss of generality that the first sender chosen by algorithm A is P_1. Now, let $t_1 = 1$, $C_{1,1} = \frac{1}{\Delta}$, $C_{2,1} = \Delta$. The schedule length $T = 1 + \Delta$ obtained by algorithm A is Δ times larger than the optimum schedule length $T^* = 1 + \frac{1}{\Delta}$. □

Since by Observations 1 and 2 it is not possible to construct a better than trivial online algorithm without additional knowledge, let us now assume that the network is homogeneous, i.e. $\alpha_i = \alpha$, $C_i^{\min} = C^{\min}$ and $C_i^{\max} = C^{\max}$ for all i.

Theorem 2. *There exists a $\frac{1+(m-1)\Delta^2}{1+(m-1)\Delta}$-competitive online algorithm solving problem DG-VS for a homogeneous network.*

Proof. Consider algorithm A that always chooses as the sender the node with the smallest current communication rate. Let $\mathcal{S}$ denote the schedule of length T constructed by A and let $\mathcal{S}^*$ be the optimum schedule of length T^*. Let P_i be the last sender in schedule $\mathcal{S}$. The total length of intervals when P_i transfers data in schedule $\mathcal{S}^*$ will be denoted by T_i^*. Let $T_{other}^* = T^* - T_i^*$. Note that it is possible to send data from P_i in schedule $\mathcal{S}$, whenever P_i sends data in schedule $\mathcal{S}^*$. Hence, in the corresponding time intervals the communication in $\mathcal{S}$ is not slower than in $\mathcal{S}^*$ and the size of sent data is at least α. The remaining data, of size at most $(m-1)\alpha$, are sent in $\mathcal{S}$ in time at most ΔT_{other}^*. Thus,

$$T \leq T_i^* + \Delta T_{other}^*. \tag{5}$$

Furthermore, we have $T_{other}^* \leq (m-1)\Delta T_i^*$, and since $T_i^* + T_{other}^* = T^*$, we get $T_{other}^* \leq \frac{T^*(m-1)\Delta}{1+(m-1)\Delta}$. Hence, we obtain from (5) that $\frac{T}{T^*} \leq \frac{1+(m-1)\Delta^2}{1+(m-1)\Delta}$. □

5 Future research

In this work, we analyzed minimizing data gathering time in a network with variable communication speed. We proposed a polynomial-time offline algorithm solving problem DG-VS and a $\frac{1+(m-1)\Delta^2}{1+(m-1)\Delta}$-competitive polynomial-time online algorithm solving DG-VS in a homogeneous network. Future research may concern the construction of non-deterministic online algorithms for DG-VS.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: A survey, *Computer Networks*, **38** (2002), 393–422.

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

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

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

[5] S. C. Ergen, P. Varaiya, TDMA scheduling algorithms for wireless sensor networks, *Wireless Networks*, **16** (2010), 985–997.

[6] S. Kumar, S. Chauhan, A survey on scheduling algorithms for wireless sensor networks, *International Journal of Computer Applications*, **20** (2011), 7–13.

[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. G. Robertazzi, Ten reasons to use divisible load theory, *IEEE Computer*, **36** (2003), 63–68.

[9] L. Shi, A. Fapojuwo, TDMA scheduling with optimized energy efficiency and minimum delay in clustered wireless sensor networks, *IEEE Transactions on Mobile Computing*, **9** (2010), 927–940.

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

Solving a time-dependent scheduling problem by interior point method

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

Wiesław Kurc
Adam Mickiewicz University in Poznań, Poznań, Poland

Keywords: single machine, time-dependent scheduling, total completion time, interior point method

1 Introduction

In many applications one cannot assume that job processing times are known in advance and fixed. Therefore, in modern scheduling theory an important class of scheduling problems compose those with variable job processing times. One of forms of variable processing times is the one in which job processing times are time-dependent, i.e. depend on when the jobs start, [1]. In this talk, we consider a single-machine scheduling problem with time-dependent job processing times that are non-decreasing functions of the job starting times. For this problem, we propose a new algorithm based on *interior point method*, [6].

2 Problem formulation

The problem under consideration can be formulated as follows. A set of jobs $J_0, J_1, \dots, J_n$ has to be processed on a single machine. All jobs are independent, non-preemptable and ready for processing at time 0. The processing time p_j of J_j linearly deteriorates in time, i.e. $p_j = 1 + \alpha_j t$, where deterioration rate $\alpha_j > 0$ and the job starting time $t \geq 0$ for $0 \leq j \leq n$. The aim is to find a schedule with minimal total completion time, $\sum_{j=1}^{n} C_{[j]}$, where $C_{[j]}$ denotes the completion time of the jth job in the schedule. In short, we will call this problem as *problem* (P).

Notice that any instance of problem (P) may be identified with sequence of job deterioration rates $\alpha^0 = (\alpha_0, \alpha_1, \dots, \alpha_n)$ which, in turn, corresponds to a schedule for the problem. Thus, any permutation $\alpha = (\alpha_{[0]}, \alpha_{[1]}, \dots, \alpha_{[n]})$ of α^0 may also be identified with a schedule for (P). Therefore, we will use the same symbol for denoting a permutation of a given sequence of deterioration rates and a schedule corresponding to the sequence.

[*] Speaker, email: `stgawiej@amu.edu.pl`

Given α, we can compute job completion times in the schedule corresponding to this α as follows:

$$C_{[0]} = 1,\ C_{[j]} = C_{[j-1]} + p_{[j]}(C_{[j-1]}) = 1 + a_{[j]}C_{[j-1]} \text{ for } 1 \leq j \leq n,$$

where $a_{[j]} = 1 + \alpha_{[j]}$. Hence, if we denote the sequence of a_j's corresponding to a given α^0 by $a^0 = (a_0, a_1, \ldots, a_n)$, any schedule for (P) may be identified with a permutation a from the set $Perm(a^0)$ of all permutations of a^0. (Since further considerations are limited only to elements of $Perm(a^0)$, we will omit square brackets in indices and write j instead of $[j]$.)

3 Related research

Problem (P) has been introduced in [8], where three basic properties of optimal schedules for (P) have been shown. The *maximum job property* says that in any optimal schedule for (P) as the first job is scheduled the job with the maximal deterioration rate. The *symmetry property* says that optimal schedules for (P) are symmetric starting from the second position, i.e. if $(\alpha_0, \alpha_1, \ldots, \alpha_n)$ is an optimal schedule, then $(\alpha_0, \alpha_n, \ldots, \alpha_1)$ is optimal as well. Finally, the *V-shape property* states that if α is an optimal schedule for problem (P), then for some $0 \leq m \leq n$ it is non-increasing for $0 \leq j \leq m$ and non-decreasing for $m \leq j \leq n$.

The time complexity of problem (P) is still unknown. We conjecture that it is $\mathcal{NP}$-hard in the ordinary sense. Properties presented in [8] decrease the number of possibly optimal schedules for (P) from $O(n!)$ to $O(2^n)$ establishing an upper bound on the complexity. Recently, a stronger necessary condition of optimality for (P), improving the latter bound by factor $O(\frac{1}{\sqrt{n}})$, has been shown in [2].

There are also known some algorithmic results for (P). In [4] have been proposed for (P) two greedy algorithms, based on properties of some functions of deterioration rates called *signatures*. Another algorithm for the problem has been proposed in [7]. Finally, in [5] and [9] have been proposed for (P) two fully polynomial-time approximation schemes (FPTASes).

4 Our results

Our approach to solving problem (P), in preliminary form presented in [3], can be described as follows. Let $C(a)$ be the vector of job completion times, $A(a)$ be an $n \times n$ square matrix with $1's$ on the main diagonal, components $a_j = 1 + \alpha_j$ of the sequence a multiplied by -1 below the main diagonal and equal to 0 otherwise, and $d = (1, 1, \ldots, 1)^\intercal$. Then, an equivalent formulation of problem (P) is as follows:

$$\min W_P(a) = \|C(a)\|_1 \quad \text{subject to} \quad A(a)C(a) = d,$$

where the minimization of $W_P(a)$ is taken over all $a \in Perm(a^0)$.

Since any optimal solution to (P) must be a *V-shaped sequence*, we consider the *polyhedron* of all 2^n such V-sequences for a given a^0. Next, we attach to each vertex of this polyhedron a *permutation matrix* of a special kind and consider the *convex hull* of such permutation matrices, which coincides with n-dimensional polyhedron of all *doubly stochastic matrices* of a special kind. This convex polyhedron, in turn, we use in a new formulation of problem (P), to which we apply the *primal-dual interior point method*, [10].

In order to make our variant of interior point method more computationally efficient, we propose to replace in the interior point method the *Newton method* by an another method, preserving the same size of matrices but without using the *Hessian inverse*. We also propose a further reduction of memory usage by the using of a steepest descent method to the goal function with the barrier and the barrier and penalty components added, respectively.

5 Future research

Our method is a new approach to solving problem (P). However, though preliminary experiments show that it works quite well in practice, provided a good starting point has been selected, some further refinements such as decreasing the cost of numerical inversion of some matrices or coping with the size of Hessian used in computations are needed.

References

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

[2] S. Gawiejnowicz, W. Kurc, A new necessary condition of optimality for a time-dependent scheduling problem, in submission.

[3] S. Gawiejnowicz, W. Kurc, Memory-efficient interior point method for solving a time-dependent scheduling problem, *The 13th Workshop on Advances in Continuous Optimization*, University of Edinburgh, 2015.

[4] 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.

[5] S. Gawiejnowicz, W. Kurc, L. Pankowska, Solving a permutation problem by a fully polynomial time approximation scheme, *Discussiones Mathematicae: Differential Inclusions, Control and Optimization*, **30** (2010), 191–203.

[6] J. Gondzio, Interior point method 25 years later, *European Journal of Operational Research*, **218** (2012), 587–601.

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

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

[9] K. M. 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] M. Ulbrich, S. Ulbrich, L. Vincente, A globally convergent primal-dual interior point filter method for nonlinear programming, *Mathematical Programming, Series A*, **100** (2004), 379–410.

Minmax scheduling with acceptable lead-times: Extensions to position-dependent processing times, due-window and job rejection

Enrique Gerstl
The Hebrew University & Jerusalem College of Technology, Jerusalem, Israel

Baruch Mor
Ariel University, Ariel, Israel

Gur Mosheiov[*]
The Hebrew University, Jerusalem, Israel

Keywords: scheduling, single machine, due-date assignment, position-dependent processing times, minmax, job rejection

1 Introduction

The goal in standard scheduling problems is to find the job schedule that minimizes a certain measure. In *scheduling and due-date assignment problems*, an additional objective is to find the optimal due-dates (see, e.g., Gordon et al. [3]). Scheduling literature contains three main categories of due-date assignment problems: (1) the setting that all the jobs share a common due-date (denoted by CON), (2) the case when due-dates are assigned to jobs as a (mostly linear) function of their processing times (denoted by SLK), (3) the case when due-dates are determined by penalties for exceeding pre-specified deadlines (denoted by DIF). In this talk, we focus our attention on the latter model.

2 Related research

The DIF model was introduced by Seidmann et al. [6]. Their model is based on the assumption that there are specific "lead times that customers consider to be reasonable and expected" [6, p. 394]. As a result, a due-date that exceeds its acceptable lead-time is penalized. Several extensions of the basic DIF model have been published. Shabtay and Steiner [8] considered the case of *job-dependent lead-times*. Shabtay and Steiner [9, 10] and Leyvand et al. [4] studied the DIF model with *controllable job processing times*. Wang et al. [11] focused on a model with a *learning effect* and *deteriorating jobs*. Finally, Mor et al. [5] studied the minmax version of DIF, which is the setting studied in this talk.

[*] Speaker, e-mail: `msomer@huji.ac.il`

In the above papers, with the exception of Wang et al. [11], one assumes constant, i.e., position-independent job processing times. However, in many applications this classical assumption is not valid. Phenomena like *learning*, where processing times decrease when jobs are delayed, or *aging/deterioration*, when processing times increase as a function of the job position or time, are common examples.

3 Our results

The main extension of the DIF model proposed in this talk concerns general position-dependent processing times. In particular, unlike most previously published studies, we do not restrict the job processing times to follow any given function, or even to be monotone, i.e., to reflect learning or aging. To the best of our knowledge, there are no published studies combining the DIF model and general position-dependent processing times.

We extend the DIF model in two directions. First, we extend the DIF model, with general position-dependent job processing times, to a setting of a *Due-Window* for acceptable *Lead-times* that we denote as DWL. The underlying assumption of DWL is that a time interval exists, such that due-dates assigned to be within this interval are not penalized. The additional lower bound on the acceptable lead-time reflects, e.g., the time needed by the customer for preparation of storage space. A due-date assigned to be prior to this lower bound is not acceptable by the customer, and a cost for an early due-date is incurred. The most relevant references for this extension are Gerstl and Mosheiov [1, 2], who recently studied the minmax and minsum versions of DWL with position-independent processing times.

The second extension of DIF considered here is a setting allowing *job rejection*. Indeed, as indicated by a number of researchers in the last decade, in many real-life situations the scheduler may decide to process only a subset of jobs. Each unprocessed ("rejected") job incurs a job-dependent penalty, and the total cost of the rejected jobs becomes a factor in the objective function. We refer the reader to the survey paper on scheduling with job rejection by Shabtay et al. [7].

The extension of DIF to the setting of general position-dependent processing times is shown here to be solved in polynomial time by solving a single linear assignment problem. The extension of DIF to the setting of DWL with position-dependent processing times is shown to remain polynomially solvable. Finally, the problem of DIF with job rejection is proved to be $\mathcal{NP}$-hard. An efficient pseudo-polynomial dynamic programming algorithm is introduced, proving that this extension is $\mathcal{NP}$-hard in the ordinary sense. Numerical tests we conducted show that this algorithm is extremely efficient, and instances with hundreds of jobs are solved in a few seconds.

References

[1] E. Gerstl, G. Mosheiov, Minmax due-date assignment with a time window for acceptable lead-times, *Annals of Operational Research*, **211** (2013), 167–177.

[2] E. Gerstl, G. Mosheiov, Scheduling with a due-window for acceptable lead-times, *Journal of the Operational Research Society*, **66** (2015), 1578–1588.

[3] V. S. Gordon, J-M. Proth, C. Chu, A survey of the state-of-the-art of common due date assignment and scheduling research, *European Journal of Operational Research*, **139** (2002), 1–25.

[4] Y. Leyvand, D. Shabtay, G. Steiner, A unified approach for scheduling with convex resource consumption functions using positional penalties, *European Journal of Operational Research*, **206** (2010), 301–312.

[5] B. Mor, G. Mosheiov, D. Shabtay, A note: Minmax due-date assignment problem with lead-time cost, *Computers & Operations Research*, **40** (2013), 2161–2164.

[6] A. Seidmann, S. S. Panwalkar, M. L. Smith, Optimal assignment of due-dates for a single processor scheduling problem, *International Journal of Production Research*, **19** (1981), 393–399.

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

[8] D. Shabtay, G. Steiner, Two due-date assignment problems in scheduling a single machine, *Operations Research Letters*, **34** (2006), 683–691.

[9] D. Shabtay, G. Steiner, The single-machine earliness-tardiness scheduling problem with due-date assignment and resource-dependent processing times, *Annals of Operations Research*, **159** (2008), 25–40.

[10] D. Shabtay, G. Steiner, Optimal due date assignment in multi-machine scheduling environments, *Journal of Scheduling*, **11** (2008), 217–228.

[11] J-B. Wang, L. Liu, C. Wang, Single machine SLK/DIF due window assignment problem with learning effect and deteriorating jobs, *Applied Mathematical Modeling*, **37** (2013), 8394–8400.

An ”almost-exact” solution to speed scaling scheduling of parallel jobs with preemption

Alexander Kononov[*]
Sobolev Institute of Mathematics, Novosibirsk, Russia

Yulia Kovalenko
Sobolev Institute of Mathematics, Novosibirsk, Russia

Keywords: speed scaling, parallel jobs, approximation algorithms

1 Introduction

In our talk, we consider two basic variants of scheduling *multiprocessor jobs*. We are given a set of parallel jobs, each one specified by its release date, deadline and processing volume, and a set of speed-scalable processors. In the first variant, the processing of job j simultaneously requires precisely $size_j$ processors. In the second variant, the execution of job j simultaneously needs a prespecified subset fix_j of dedicated processors. Note that the parallel execution of parts of the same job is not allowed. Moreover, the execution of each job can be interrupted and resumed without incurring any cost or delay. According to definitions given in literature on scheduling theory, we consider *rigid jobs* and *single mode multiprocessor tasks* (*jobs*), respectively [7].

We consider the standard model of *speed-scaling*, in which if a processor runs at speed s, then the energy consumption is s^α units of energy per time unit, where $\alpha > 1$ is a constant. We assume that if processors execute the same job simultaneously, then all these processors run at the same speed. Each job has to execute a work volume w and since processors may change their speed, a job may be completed faster (or slower) compared to the time w needed for its execution at speed 1. The goal is to find a feasible schedule respecting the release dates and deadlines of all jobs such that the total energy consumption is minimized. We will denote the speed scaling problem with rigid jobs by Π_1 and the speed scaling problem with single mode multiprocessor jobs by Π_2.

2 Related research

For the preemptive single-processor case, Yao et al. [9] proposed an optimal algorithm for finding a feasible schedule with minimum energy consumption. The

[*] Speaker, email: `alvenko@math.nsc.ru`

case when there are available m parallel processors and single-processor jobs has been solved optimally in polynomial time, provided that the preemption and the *migration of jobs* are allowed [1, 3, 5]. We note that the migration of jobs is equivalent to the possibility to execute a parallel job in different modes.

Albers et al. [2] considered the problem on parallel processors in the case when the preemption of the jobs is allowed but not their migration. They proved that it is solvable in polynomial time for instances with agreeable deadlines and unit-work jobs. For general instances with unit-work jobs, they proved that the problem becomes strongly $\mathcal{NP}$-hard and proposed an $(\alpha^\alpha 2^{4\alpha})$-approximation algorithm. For the case where the jobs have arbitrary works, the problem was proved to be $\mathcal{NP}$-hard even for instances with common release dates and common deadlines.

Albers et al. [1] proposed a $2(2-\frac{1}{m})^\alpha$-approximation algorithm for instances with common release dates, or common deadlines, and an $(\alpha^\alpha 2^{4\alpha})$-approximation algorithm for instances with agreeable deadlines. Greiner et al. [8] gave a generic reduction transforming an optimal schedule for the problem on parallel processors with migration to a $B_{\lceil\alpha\rceil}$-approximate solution for the problem on parallel processors with preemptions but without migration, where $B_{\lceil\alpha\rceil}$ is the $\lceil\alpha\rceil$-th *Bell number*. (The latter result holds only for $\alpha \leq m$.) Cohen-Addad et al. [6] showed that the problem without migration is $\mathcal{APX}$-hard, even for jobs with common release date, common deadline and work volumes possess the values 1, 3 or 4.

Bampis et al. [4] studied the heterogeneous preemptive problem on parallel processors where every processor i has a different *speed-to-power function*, $s^{\alpha(i)}$, and both the life interval and the work of jobs are processor-dependent. For the migratory variant, they proposed an algorithm returning a solution within an additive factor of ε far from the optimal solution. The algorithm runs in time polynomial to the size of the instance and to $\frac{1}{\varepsilon}$. For the non-migratory variant, Bampis et al. [4] presented an $(1+\varepsilon)^\alpha \tilde{B}_\alpha$-approximation algorithm, where $\tilde{B}_\alpha$ is the α-th *generalized Bell number*.

To the best of our knowledge, no one considered the speed scaling scheduling of parallel jobs. For more information on scheduling problems with parallel jobs, we refer the reader to the survey book by Drozdowski [7].

3 Our results

We present two algorithms for problems Π_1 and Π_2. We first formulate the problems as a *linear programming* (*LP*) *configuration* with an exponential number of variables and a polynomial number of constraints. Surprisingly, we use the same LP configuration for both problems. First, we prove that for any $\varepsilon > 0$ there is a feasible solution of the LP with energy consumption at most $OPT + \varepsilon$, where

OPT is an optimal solution of the original problem. Then, we consider the dual LP and we show how to use the *ellipsoid algorithm* to it and obtain an optimal solution for the primal LP. For this purpose, we provide two separation oracles, one for the problem Π_1 and one for Π_2, i.e. we present two algorithms which given a solution for the dual LP decides if this solution is feasible or otherwise it identifies a violated constraint. More precisely, we get the following results. Given an instance of Π_1 with m processors, we can solve the separation problem for Π_1 in time polynomial in m, $\frac{1}{\varepsilon}$ and size of the instance. Thus, we have a polynomial time algorithm if m is fixed, and a pseudo-polynomial time algorithm if m is a part of the input. Given an instance of Π_2 with $|fix_j| = 2$ for all jobs, we can solve the separation problem for Π_2 in time polynomial in $\frac{1}{\varepsilon}$ and size of the instance. As we can compute an optimal solution for the dual LP, we can also find an optimal solution for the primal LP by solving it with the variables corresponding to the constraints that were found to be violated during the run of the ellipsoid algorithm and setting all other primal variables to zero. Our two main results can be formulated as follows.

Theorem 1. *A schedule for the speed scaling problem with rigid jobs of energy consumption $OPT + \varepsilon$ can be found in time polynomial in m, $\frac{1}{\varepsilon}$ and size of the instance.*

Recall that if the number of processors used by a job is chosen by the scheduler, and it can be changed at runtime, then such a job is called a *malleable job* [7]. Our algorithm can be generalized for the speed scaling problem with malleable jobs.

Theorem 2. *A schedule for the speed scaling problem with single mode two-processor jobs of energy consumption $OPT + \varepsilon$ can be found in time polynomial in $\frac{1}{\varepsilon}$ and size of the instance.*

In our talk, we assume that a continuous spectrum of processor speeds is available. If only a finite set of discrete speed levels is available, then our algorithm finds an optimal solution. The complexity status of both problems with continuous spectrum of speed is open.

We also prove ordinary $\mathcal{NP}$-hardness of Π_1, when the number of processors is a part of the input and strong $\mathcal{NP}$-hardness of Π_2, when $|fix_j| = 3$ for all jobs.

Acknowledgments

Supported by the RSF grant No 15-11-10009.

References

[1] S. Albers, A. Antoniadis, G. Greiner, On multi-processor speed scaling with migration: Extended abstract, *The 23rd ACM Symposium on Parallelism in Algorithms and Architectures*, ACM, 2011, pp. 279–288.

[2] S. Albers, F. Müller, S. Schmelzer, Speed scaling on parallel processors, *The 19th ACM Symposium on Parallelism in Algorithms and Architectures*, ACM, 2007, pp. 289–298.

[3] E. Angel, E. Bampis, F. Kacem, D. Letsios, Speed scaling on parallel processors with migration, *Lecture Notes in Computer Science*, **7484** (2012), pp. 128–140.

[4] E. Bampis, A. Kononov, D. Letsios, G. Lucarelli, M. Sviridenko, Energy efficient scheduling and routing via randomized rounding, *The 33th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, Leibniz International Proceedings in Informatics, **24** (2013), pp. 449–460.

[5] B. D. Bingham, M. R. Greenstreet, Energy optimal scheduling on multiprocessors with migration, *The IEEE International Symposium on Parallel and Distributed Processing with Applications*, IEEE, 2008, pp. 153–161.

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

[7] M. Drozdowski, *Scheduling for Parallel Processing*, Springer-Verlag, London, 2009.

[8] G. Greiner, T. Nonner, A. Souza, The bell is ringing in speed-scaled multiprocessor scheduling, *The 21st ACM Symposium on Parallelism in Algorithms and Architectures*, ACM, 2009, pp. 11–18.

[9] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced CPU energy, *The 36th Annual Symposium on Foundations of Computer Science*, IEEE, 1995, pp. 374–382.

Workforce planning for cyclic production of multiple parts

Mikhail Y. Kovalyov[*]
National Academy of Sciences of Belarus, Minsk, Belarus

Xavier Delorme
École des Mines de Saint-Étienne, Saint-Étienne, France

Alexandre Dolgui
École des Mines de Nantes, Nantes, France

Keywords: workforce assignment, production line, supply chain

1 Introduction

This presentation is motivated by the problems of *workforce planning* in the production of vehicle engines on a transfer line, see Battaïa et al. [1].

There is given a *paced transfer line* with *m stations* on which vehicle parts of a set $P = \{1, 2, \dots, k\}$ are manufactured. Each part visits the stations in the same order $1, 2, \dots, m$ and fully occupies any station in the time period between its arrival and departure. With a time step C called *cycle time*, a part on station i departs from this station and immediately arrives to station $i+1$, $i = 1, 2, \dots, m-1$, a new part arrives to station 1 and the part at station m leaves the line. Activities taking place between two successive part moves are called (*production*) *cycle*. If part j is handled on station i in a cycle, then the duration of all operations on this part at this station in this cycle is a non-increasing function $p_{ij}(x_{ij})$ of the number of identical workers x_{ij} assigned to station i and part j in this cycle. Workers assigned to different stations perform their operations in parallel. They do not move from one station to another in the same cycle, however, they can move instantly to another station after the cycle has been finished. To be feasible with respect to the cycle time, relation $p_{ij}(x_{ij}) \leq C$ must be satisfied for each station and each part in each cycle. Besides, technological constraints define lower and upper bounds on the values x_{ij} in each cycle: $a_{ij} \leq x_{ij} \leq b_{ij}$, $i = 1, 2, \dots, m$, $j \in P$.

A requirement of the global supply chain is that the production keeps a given proportion of the manufactured parts. To address this issue, the notion of a *Minimal Part Set* (*MPS*) is used. Let the part proportion be given by the numbers $w_1, w_2, \dots, w_k$, where w_j is the percent of parts j to be manufactured, and let GCD denote the greatest common divisor of the numbers $w_1, w_2, \dots, w_k$. Let us denote $o_j = \frac{w_j}{GCD}$, where $j \in P$ and $n = \sum_{j=1}^{k} o_j$. MPS is the multiset that consists of o_j copies of each part j, $j \in P$.

[*] Speaker, email: `kovalyov_my@newman.bas-net.by`

An *MPS sequence* is a part sequence $\pi = (j_1, j_2, \ldots, j_n)$, $j_r \in P$, $r = 1, 2, \ldots, n$, in which each part j occurs o_j times, $j \in P$. According to the MPS sequence π, the parts visit the line cyclically in the order $j_1, j_2, \ldots, j_n, j_1, j_2, \ldots, j_n, \ldots$ In each cycle, each station is assigned a part and this arrangement can be represented by a sequence $\sigma = (j'_1, j'_2, \ldots, j'_m)$, where $j'_h \in P$ is the part assigned to station $h = 1, 2, \ldots, m$. We call such an arrangement a *Station-Part matching* (*SP-matching*). There is a 1-1 correspondence between distinct cycles and distinct SP-matchings.

Given MPS sequence $\pi = (j_1, j_2, \ldots, j_n)$, let's consider a *digraph* $G(\pi)$ with the set of *nodes* $\{j_1, j_2, \ldots, j_n\}$ and the set of arcs $\{(j_1, j_2), (j_2, j_3), \ldots, (j_{n-1}, j_n), (j_n, j_1)\}$. Thus, $G(\pi)$ is the graph called *cycle*. We say that an SP-matching σ is *π-feasible*, if and only if it is a *path* in the graph $G(\pi)$, with possible node repetition if $m > n$. For example, if $m = n + 1$, then $\sigma = (j_2, j_3, \ldots, j_1, j_2)$ is π-feasible, and, if $m = 3$, then $\sigma = (j_6, j_7, j_8)$ is π-feasible. Let $\Phi(\pi)$ denote the set of all π-feasible SP-matchings. We have $|\Phi(\pi)| = n$. Further, let $x_i^{(\sigma)}$ denote the number of workers assigned to station i in the cycle defined by the SP-matching σ, $i = 1, 2, \ldots, m$. The total number of workers used in this cycle is equal to $\sum_{i=1}^{m} x_i^{(\sigma)}$. Denote by x the structure with the entries $x_i^{(\sigma)}$, $i = 1, 2, \ldots, m$, $\sigma \in \Phi(\pi)$. The total number of workers used in the long run of the MPS sequence $\pi = (j_1, j_2, \ldots, j_n)$ is equal to $T(\pi, x) = \max_{\sigma \in \Phi(\pi)} \{\sum_{i=1}^{m} x_i^{(\sigma)}\}$. The problem is to determine an MPS sequence π and the number of workers assigned to each station in each cycle, determined by the structure x, such that the maximum of the total number of workers over all cycles, $T(\pi, x)$, is minimized. We denote this problem as MinMaxSum.

2 Related research

Problem MinMaxSum has much in common with the problem studied by Lee and Vairaktarakis [2]. However, there are two differences. Firstly, the authors assume that the MPS sequence is not repeated cyclically: the line is free before the first part of an MPS sequence arrives to the first station and the production stops when the last part of this sequence departs from the last station. Secondly, they assume that the workforce requirements are fixed, that is, $a_{ij} = b_{ij}$, $i = 1, 2, \ldots, m$, $j \in P$, in our notation. We denote the problem in [2] as Fix. Lee and Vairaktarakis proved that this problem is solvable in $O(n \log n)$ time if $m = 2$, it is $\mathcal{NP}$-hard in the strong sense if $m = 3$, and they suggested an integer linear programming formulation, lower bounds and heuristics.

Problem MinMaxSum with station independent workforce requirements belongs to the class of *Minmaxsum Traveling Salesman Problems* (*TSPs*), which includes Bottleneck TSP, intersects with TSP under Categorization, and it is relevant to the k-Neighbor TSP and Maximum Scatter TSP.

We show that MinMaxSum is equivalent to the same problem without the cycle time constraints $p_{ij}(x_{ij}) \leq C$ and with $a_{ij} = b_{ij} = w_{ij}$ for appropriately defined fixed workforce requirements w_{ij}, $i = 1, 2, \ldots, m$, $j \in P$, propose a brute force algorithm for this problem, prove strong $\mathcal{NP}$-hardness for the case of $m = 3$ and for the case of $m = 4$ and station independent workforce requirements ($w_{ij} = w_j$, $i = 1, 2, \ldots, m$, $j \in P$), and propose an $O(k \log k)$ algorithm for the case $m = 2$ and station independent workforce requirements.

3 Reduction to fixed workforce requirements

Consider an MPS sequence π. Observe that some parts can be missing in an SP-matching $\sigma \in \Phi(\pi)$, but, for any part, there exists an SP-matching $\sigma \in \Phi(\pi)$, in which this part is present on any station. This observation implies two statements: 1) the cycle time constraint and the lower and upper bounds on the number of workers are satisfied if and only if, for each pair (i, j), there exists a number of workers x such that $p_{ij}(x) \leq C$ and $a_{ij} \leq x \leq b_{ij}$, and 2) statement 1) is satisfied, that is, problem MinMaxSum has a solution, if and only if, for each pair (i, j), the optimal number of workers is fixed to be equal to $w_{ij} := \min\{x \mid p_{ij}(x) \leq C, a_{ij} \leq x \leq b_{ij}\}$, assuming that $p_{ij}(b_{ij}) \leq C$ for all i and j. If functions $p_{ij}(x)$ are *invertible*, then $w_{ij} = \max\{a_{ij}, \lceil p_{ij}^{-1}(C) \rceil\}$, otherwise, w_{ij} can be found in $O(\log_2(b_{ij} - a_{ij}))$ time by a bisection search over the range $[a_{ij}, b_{ij}]$. Thus, in $O(mk \log_2 \max_{i,j}\{b_{ij} - a_{ij}\})$ time, MinMaxSum reduces to the same problem with no cycle time constraints and with the fixed workforce requirements w_{ij}, which is to minimize $T(\pi) = \max\left\{F(\sigma) = \sum_{i=1}^{m} w_{i,j'_i} \mid \sigma = (j'_1, j'_2, \ldots, j'_m) \in \Phi(\pi)\right\}$. Assume without loss of generality that $o_1 = \min_{i=1,2,\ldots,k}\{o_i\}$. Denote by Π the set of MPS sequences, in which part 1 is in the first place. Set $\hat{n} = n-1$, $\hat{o}_1 = o_1 - 1$ and $\hat{o}_i = o_i$, $i = 2, 3, \ldots, k$. We have

$$|\Pi| = C_{\hat{n}}^{\hat{o}_1} \cdot C_{\hat{n}-\hat{o}_1}^{\hat{o}_2} \cdots C_{\hat{o}_k}^{\hat{o}_k} = \frac{\hat{n}!}{\hat{o}_1! \hat{o}_2! \cdots \hat{o}_k!} = \frac{(n-1)!}{(o_1 - 1)! o_2! o_3! \cdots o_k!}.$$

MinMaxSum reduces to minimizing $T(\pi)$ for $\pi \in \Pi$. Since $|\Phi(\pi)| = n$, each value $T(\pi)$ can be calculated in $O(mn)$ time. Hence, MinMaxSum can be solved in $O(mn|\Pi|) = O\left(\frac{n!m}{(o_1-1)!o_2!o_3!\cdots o_k!}\right)$ time.

4 Complexity results

Theorem 1. MinMaxSum *is $\mathcal{NP}$-hard in the strong sense for the case $m = 3$ and for the case $m = 4$ and $w_{ij} = w_j$ for $i = 1, 2, \ldots, m$ and $j \in P$.*

We use reductions from Numerical 3-Dimensional Matching and 3-Partition for the first and the second mentioned case, respectively.

Let $m = 2$ and workforce requirements be $w_{ij} = w_j$ for $i = 1, 2$ and $j = 1, 2, \ldots, k$. Assume that part copies are numbered $1, 2, \ldots, n$ and each part copy r is associated with workforce requirement w_r^0 which is equal to the workforce requirement of its part: $w_r^0 = w_j$ if r is a copy of part j. Number part copies in the *Least Workforce Requirement* (*LWR*) order such that $w_1 \leq w_2 \leq \cdots \leq w_n$. We call sequence of part copies $\pi = (1, n, 2, n-1, 3, n-2, 4, \ldots)$, which includes all n copies, *Up-and-Down* sequence.

Theorem 2. *The Up-and-Down sequence is optimal for* MinMaxSum *if* $m = 2$ *and* $w_{ij} = w_j$ *for* $i = 1, 2$ *and* $j \in P$.

The Up-and-Down sequence can be represented using $O(k)$ memory units because copies of the same part appear consecutively in the LWR order. This concise representation of the Up-and-Down sequence can be constructed in $O(k \log k)$ time by sorting parts in the non-decreasing order of their workforce requirements and employing this order in the description of the Up-and-Down sequence of part copies. Thus, MinMaxSum with $m = 2$ and $w_{ij} = w_j$ for $i = 1, 2$ and $j \in P$ can be solved in $O(k \log k)$ time.

5 Future research

In the future, it is interesting to establish computational complexity of the open cases of MinMaxSum with fixed parameters m and/or k, and general or station independent workforce requirements. ILP formulations and (meta)heuristics for the general case are of a practical interest.

References

[1] O. Battaïa, X. Delorme, A. Dolgui, J. Hagemann, A. Horlemann, S. Kovalev, S. Malyutin, Workforce minimization for a mixed-model assembly line in the automotive industry, *International Journal of Production Economics*, **170** (2015), 489–500.

[2] C-Y. Lee, G. L. Vairaktarakis, Workforce planning in mixed model assembly systems, *Operations Research*, **45** (1997), 553–567.

Directed sets, Möbius inversing formula and time-dependent scheduling on precedence-constrained machines

Wiesław Kurc[*]
Adam Mickiewicz University in Poznań, Poznań, Poland

Stanisław Gawiejnowicz
Adam Mickiewicz University in Poznań, Poznań, Poland

Keywords: directed sets, Möbius inversing formula, time-dependent scheduling, total weighted completion time, maximum completion time

1 Introduction

The aim of this talk is to present some preliminary results on scheduling jobs with variable processing times on machines endowed by a partial order. In other words, we consider scheduling problems in which the directed chain of machines typical for such machine systems as flow shop is replaced by a more general directed set. We illustrate the idea by an example in which available machines compose a directed tree. Applying a matrix approach [1], equivalent to *Möbius inversing formula* over incidence algebra [2], we formulate two results concerning scheduling linearly *deteriorating jobs* on such machine systems with the objective to minimize the total weighted completion time or the maximum completion time.

2 Problem formulation

We consider the following problem. The set J of $n+2$ jobs has to be processed on the set M of $m+2$ machines. Jobs are divided into *real jobs* and *artificial jobs*. The processing time of real job $J_i \in J$ varies according to the function $p_i(t) = b_i + \alpha_i t$, where t denotes the starting time of the job, $b_i \geq 0$ and $\alpha_i \geq -1$ for $1 \leq i \leq n$. The processing time of artificial job J_0 is fixed, $p_0(t) = b_0$. Artificial job J_{n+1} has no processing time and its meaning is to simplify the problem formulation. To each job $J_i \in J$, excluding artificial job J_0, there is defined weight ω_{i+1}, indicating the significance of the completion time C_i of job J_i from the perspective of job J_{i+1}.

Jobs $J_1, J_2, \ldots, J_n$ are processed on *real machines* $M_1, M_2, \ldots, M_n$, while artificial jobs J_0 and J_{n+1} are processed on *artificial machines* M_0 and M_{n+1}, respectively. On machines $M_0, M_1, \ldots, M_{n+1}$ there is defined a partial order and all of them are available for processing from time $C_0 \geq 0$.

[*] Speaker, email: `wkurc@amu.edu.pl`

Since jobs $J_1, J_2, \ldots, J_n$ are fully characterized by n triplets $\sigma_i = (b_i, a_i, \omega_i)$, following [1] we use the table

$$\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{n+1}) = \begin{bmatrix} b_0 & b_1 & b_2 & b_n & - \\ - & a_1 & a_2 & a_n & - \\ - & \omega_1 & \omega_2 & \omega_n & \omega_{n+1} \end{bmatrix}$$

in which triplets $\sigma_0 = (b_0, -, -)$ and $\sigma_{n+1} = (-, -, \omega_{n+1})$ correspond to artificial jobs J_0 and J_{n+1}, respectively. (Since any such a table corresponds to a non-delay schedule, we will denote the table and the schedule by the same symbol.)

Given C_0 and σ, job completion times are equal to

$$C_1(\sigma) = C_0 + p_1(C_0) = b_1 + a_1 C_0$$

and

$$C_i(\sigma) = b_i + a_i C_{i-1}(\sigma),$$

where $C_i(\sigma)$ denotes the completion time of job J_i in schedule σ, $C_0(\sigma) \equiv C_0$, $a_i = 1 + \alpha_i$ and $1 \leq i \leq n$. The aim is to find a schedule σ minimizing the total weighted completion time, $\sum_{i=0}^{n} \omega_{i+1} C_i(\sigma)$, or the maximum completion time, $\max\{C_i(\sigma) : 1 \leq i \leq n\}$.

The above problem can be concisely formulated in a matrix form. Let σ° and $Perm(\sigma^\circ)$ denote the table corresponding to the initial order of jobs given in input instance and the set of all permutations of the triplets $\sigma_1, \sigma_2, \ldots, \sigma_n$ in σ°, respectively. Then, the problem under consideration is equivalent to

$$\min \omega^\intercal C(\sigma) = \textstyle\sum_{i=0}^{n} \omega_{i+1} C_i \text{ subject to } A(\sigma)C(\sigma) = b,$$

where $\sigma \in Perm(\sigma^\circ)$, [1].

In order to illustrate our ideas, let us consider the following example. Denote by $M = \{M_i, M_s, M_j, M_k, M_l, M_m\}$ the set of machines endowed with the partial order $M_i \preceq M_k \preceq M_l \preceq M_m$, $M_j \preceq M_k$, and $M_s \preceq M_l$. In other words, M forms a directed tree with the root at M_m and three leaves M_i, M_j and M_s. The table σ° of triplets describing jobs is in the form of

$$\sigma^\circ = \begin{bmatrix} b_i & b_s & b_j & b_k & b_l & b_m & - \\ - & - & - & a_k & a_l & a_m & - \\ - & - & - & \omega_k & \omega_l & \omega_m & \omega_n \end{bmatrix}.$$

Then $C_i(\sigma^\circ) = b_i$, $C_s(\sigma^\circ) = b_s$, $C_j(\sigma^\circ) = b_j$ and $C_k(\sigma^\circ) = C_i(\sigma^\circ) + p_k(C_i(\sigma^\circ)) + C_j(\sigma^\circ) + p_k(C_j(\sigma^\circ)) = a_k C_i(\sigma^\circ) + b_k + a_k C_j(\sigma^\circ) + b_k$, $C_l(\sigma^\circ) = C_k(\sigma^\circ) +$

$p_l(C_k(\sigma^\circ)) + C_s(\sigma^\circ) + p_l(C_s(\sigma^\circ)) = a_lC_k(\sigma^\circ) + b_l + a_lC_s(\sigma^\circ) + b_l$ and, finally, $C_m(\sigma^\circ) = a_mC_l(\sigma^\circ) + b_m$.

Matrix equation $A(\sigma^\circ)C(\sigma^\circ) = b$ is in the form of

$$\begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ -a_k & & -a_k & 1 & & \\ & -a_l & & -a_l & 1 & \\ & & & & -a_m & 1 \end{pmatrix} \begin{pmatrix} C_i \\ C_s \\ C_j \\ C_k \\ C_l \\ C_m \end{pmatrix} = \begin{pmatrix} b_i \\ b_s \\ b_j \\ 2b_k \\ 2b_l \\ b_m \end{pmatrix}.$$

Hence, $C(\sigma^\circ) = A^{-1}(\sigma^\circ)b$ with lower triangular matrix $A^{-1}(\sigma^\circ)$. Therefore,

$$\begin{aligned} \omega^\intercal C(\sigma^\circ) &= b_i\omega_i + b_j\omega_j + \left(a_k\left(b_i + b_j\right) + 2b_k\right)\omega_k + \\ &+ \left(a_ka_l\left(b_i + b_j\right) + 2b_l + a_l\left(2b_k + b_s\right)\right)\omega_l + \\ &+ \left(a_ka_la_m\left(b_i + b_j\right) + 2a_mb_l + b_m + a_la_m\left(2b_k + b_s\right)\right)\omega_m + b_s\omega_s, \end{aligned}$$

since

$$C(\sigma^\circ) = \begin{pmatrix} b_i \\ b_s \\ b_j \\ a_kb_i + a_kb_j + 2b_k \\ a_ka_lb_i + a_ka_lb_j + 2a_lb_k + 2b_l + a_lb_s \\ a_ka_la_mb_i + a_ka_la_mb_j + 2a_la_mb_k + 2a_mb_l + b_m + a_la_mb_s \end{pmatrix}.$$

Job completion times can be computed in a few different ways. If we assume, as above, that in a schedule σ the processing at node k applies the same function $p_k(t)$ to job completion times $C_i(\sigma)$, $C_j(\sigma)$ from the previous stage, then $C_k(\sigma) = (C_i(\sigma) + p_k(C_i(\sigma))) \vartriangle (C_j(\sigma) + p_k(C_j(\sigma)))$, where $\vartriangle \equiv +$. Alternatively, one can assume that $\vartriangle \equiv \max$. Finally, we can assume that $C_k(\sigma) = (C_i(\sigma) + p_i(C_i(\sigma))) \vartriangle (C_j(\sigma) + p_j(C_j(\sigma))) = a_iC_i(\sigma) + a_jC_j(\sigma) + b_i + b_j$, where $\vartriangle \equiv +$. In the latter case, the objective function becomes $\omega^\intercal C(\sigma) = \omega_kC_i(\sigma) + \omega_lC_s(\sigma) + \omega_kC_j(\sigma) + \omega_lC_k(\sigma) + \omega_mC_l(\sigma) + \omega_nC_m(\sigma)$.

3 Our results

Our main results are based on two observations. First, notice that $C(\sigma) = A^{-1}(\sigma)b$ is a form of *Möbius inversing formula* ([2]) over incidence algebra $A(M)$ of all real-valued functions *f* over $M \times M$ such that $f(M_i, M_j) = 0$ unless $M_i \preceq M_j$, where $\preceq$ is a partial order in M. Second, notice that the incidence algebras give us the mathematical background for time-dependent scheduling on machines related by the partial order $\preceq$. For instance, from [2] it follows that $A^{-1}(\sigma)$ exists and is a triangular matrix for *any* partial order on M.

Keeping in mind the observations, our two main results can be stated as follows.

Theorem 1. *Let M be the set of n machines endowed with partial order $\preceq$ such that M is a directed set. Let J be the set of n jobs described by the table σ with $\omega_i = b_i = 1$ for $0 \leq i \leq n+1$, respectively. Moreover, let $\sigma^* \in Perm(\sigma^\circ)$ be an optimal schedule for problem (P) with the $\sum \omega_{j+1} C_j$ criterion. Then elements a_i of σ^* considered along any chain connecting beginning and final elements in M must be V-shaped.*

Theorem 2. *Let assumptions of Theorem 1 be satisfied and let $\sigma^* \in Perm(\sigma^\circ)$ be an optimal schedule for problem (P) with the $C_{\max}$ criterion. Then the elements a_i of σ^* considered along any chain connecting beginning and final elements in M must be non-decreasing.*

4 Future research

In future research we want to extend our ideas to the case of other forms of job processing times and machine precedence constraints.

References

[1] S. Gawiejnowicz, W. Kurc, L. Pankowska, Equivalent time-dependent scheduling problems, *European Journal of Operational Research*, **196** (2009), 919–929.

[2] J. P. S. Kung, G-C. Rota, C. H. Yan, *Combinatorics: The Rota Way*, Cambridge University Press, 2009.

Relocation scheduling with optional recycling operations

Bertrand M-T. Lin[*]
National Chiao Tung University, Hsinchu, Taiwan

Keywords: relocation problem, resource-constrained scheduling, optional recycling operations, dynamic programming, maximum completion time

1 Introduction

In this talk, we introduce a scheduling problem with a new variant of *resource constraints*. The new resource constraints are generalizations of *renewable and non-renewable resources* in the sense that a task when completed will release the resource it has acquired for processing, allowing the amount of released resource to be smaller than, equal to or even larger than the amount of the previously acquired resource. Moreover, in order to release the resource back to the common pool, a recycling operation must be performed. When the amount of the resource is sufficient, some of the recycling operations can be bypassed. The problem is to find a feasible schedule that attains the minimum makespan.

2 Problem formulation

The problem under consideration is formally defined as follows. A set of n jobs $N = \{1, 2, \ldots, n\}$ is to be processed on a single machine. Each job $j \in N$ is characterized by four non-negative integral parameters: (1) resource requirement a_j that is an amount of the resource required to commence the processing of job j, (2) processing time p_j that is the processing time of the regular operation of job j, (3) resource yield b_j that is an amount of the resource that may be returned by job j when it is completed, (4) resource recycling time q_j that is the time required for recycling the b_j units of the resource. A common pool of v_0 units of a single-type resource is given for processing the jobs of N. Job j requires and consumes a_j units of the resource from the resource pool to commence its processing.

Upon its completion, we either perform its recycling operation, taking q_j units of time, or directly proceed to the processing of other jobs. If the recycling operation is carried out, then b_j units of the resource will be produced and deposited into the common pool. The decision consists of two parts: (1) selecting a subset of recycling operations to be performed, and (2) determining a sequence of all regular operations and the selected recycling operations. The derived sequence should

[*] Speaker, email: `bmtlin@mail.nctu.edu.tw`

be feasible subject to the resource constraints in the sense that in the course of execution no job is blocked by insufficiency of the resource. The objective is to minimize the makespan, i.e. the largest completion time of the regular processing operations. We will denote the problem as $1|a_j, b_j, recl|C_{\max}$.

3 Related research

The studied problem is a generalization of the *relocation problem* formulated as a redevelopment project in the east Boston by Kaplan [2]. Taking into account the temporal issues along the planning horizon, Kaplan [2] addressed the reconstruction time lines of the buildings, which allowed to process multiple buildings in parallel if the available resource permitted this. Kononov and Lin [4, 5] investigated two other objective functions. The first attempt to introduce recycling operations was done by Lin and Huang [6]. In this case, a second working crew is available for performing the recycling operations and the two working crews constitute a two-machine flow shop: the first machine is responsible for demolishing the buildings, the second one is used for erecting new buildings. Cheng et al. [1] considered a relocation problem with separate recyclic operations.

4 Our results

We start with the following theorem that links the relocation problem and two-machine flow shop scheduling $F2||C_{\max}$.

Theorem 1. (*Kaplan and Amir* [3]) *Let a_j and b_j be the machine-one and machine-two processing times of job j in the $F2||C_{\max}$ problem. Then, the minimum initial resource level guaranteeing the feasibility of a sequence in the relocation problem is equivalent to the total idle time on machine two of the corresponding job sequence in the $F2||C_{\max}$ problem.*

Lemma 2. *Consider the base relocation problem with only negative jobs, i.e., such that $\delta_j = a_j - b_j < 0$ for all $j \in N$. If the initial resource level v_0 is sufficient for guaranteeing the existence of feasible schedules of the jobs in N, then v_0 is also sufficient for any proper subset of N.*

Returning to the studied problem, we have the following auxiliary result.

Lemma 3. *There is an optimal schedule of $1|a_j, b_j, recl|C_{\max}$ in which (1) the complete jobs precede the partial jobs, (2) the complete jobs are sequenced by Johnson's rule, and (3) the partial jobs can be arranged in arbitrary order.*

To formulate the studied problem, we define binary variable $x_j = 1$ if the recycling operation of job j is performed, and $x_j = 0$ otherwise.

Then, the formulation of the problem as an integer program is as follows:

$$\text{Minimize} \sum_{j=1}^{n} x_j q_j \tag{1}$$

subject to

$$v_0 + \sum_{i=1}^{j-1} x_i \delta_i \geq x_j a_j, \quad 1 \leq j \leq n, \tag{2}$$

$$v_0 + \sum_{i=1}^{n} x_i \delta_i \geq \sum_{i=1}^{n} a_i (1 - x_i), \quad 1 \leq j \leq n, \tag{3}$$

$$x_j \in \{0, 1\}, \quad 1 \leq j \leq n. \tag{4}$$

The left hand side of constraints (4) and (5) gives the resource levels at the completion of jobs in certain positions. Constraints (2) ensure the feasibility of each complete job, and constraints (3) confine the feasibility of the partial jobs.

The problem is $\mathcal{NP}$-hard, since it is a generalization of the Knapsack problem. However, it still allows the development of pseudo-polynomial algorithm by dynamic programing (DP). The design of our DP algorithm differs from most of other ones, in which feasible schedules are constructed by recursion from feasible partial schedules. Namely, infeasibility in our design is allowed during the construction course of partial schedules. State $f(j, \tau, \eta)$ is defined for the first j jobs, $1, 2, \ldots, j$, subject to the conditions that the resource level at the completion of all complete jobs is exactly τ and that the total resource requirement of the partial jobs is exactly η. Many (partial) schedules may correspond to state $f(j, \tau, \eta)$. Define $f(j, \tau, \eta)$ as the maximum total length of rejected recycling operations among the schedules corresponding to state (j, τ, η).

Dynamic Programming Algorithm for problem $1|a_j, b_j, recl|C_{\max}$

Initialization: For $j = 0$ to n, $\tau = 0$ to $v_0 + \sum_{i \in N^+} \delta_i$, $\eta = 0$ to $\sum_{i \in N} a_i$

$$f(j, \tau, \eta) = \begin{cases} 0, & \text{if } j = \tau = \eta = 0; \\ -\infty, & \text{otherwise.} \end{cases}$$

Recursion: For $j = 0$ to n, $\tau = 0$ to $v_0 + \sum_{i \in N^+} \delta_i$, $\eta = 0$ to $\sum_{i \in N} a_i$

$$f(j, \tau, \eta) = \begin{cases} \max \left\{ f(j-1, \tau - \delta_j, \eta), f(j-1, \tau, \eta - a_j) + q_j \right\}, & \text{if } \tau - \delta_j \geq a_j; \\ f(j-1, \tau, \eta - a_j) + q_j, & \text{otherwise.} \end{cases}$$

Goal: Find $\max \left\{ f(n, \tau, \eta) : 0 \leq \tau \leq v_0 + \sum_{j \in N^+} \delta_j, 0 \leq \eta \leq \tau \right\}$.

In recursion, condition $\tau - \delta_j \geq a_j$ examines whether or not job j is eligible to be scheduled as the last complete job. If job j is eligible, we can schedule it either complete or partial. The inequality $\eta \leq \tau$ in the goal function is to ensure the feasibility of the partial jobs in the schedules corresponding to the states defined by n. The overall computing time is $O\Big(n(v_0 + \sum_{j\in N^+} \delta_j) \sum_{j\in N} a_j\Big)$, which is pseudo-polynomial in terms of the input length, confirming that the $1|a_j, b_j, recl|C_{\max}$ problem cannot be strongly $\mathcal{NP}$-hard.

References

[1] T-C. E. Cheng, B. M-T. Lin, H-L. Huang, Makespan minimization in the relocation problem with separate resource recycling operations, *Computers & Operations Research*, **412** (2011), 4536–4544.

[2] E. H. Kaplan, Relocation models for public housing redevelopment programs, *Planning & Design*, **13** (1986), 5–19.

[3] E. H. Kaplan, A. Amir, A fast feasibility test for relocation problems, *European Journal Operational Research*, **35** (1988), 201–205.

[4] A. V. Kononov, B. M-T. Lin, On the relocation problems with multiple identical working crews, *Discrete Optimization*, **3** (2006), 368–381.

[5] A. V. Kononov, B. M-T. Lin, The relocation problem to minimize the weighted completion time, *Journal of Scheduling*, **13** (2010), 123–129.

[6] B. M-T. Lin, H-L. Huang, On the relocation problem with a second working crew for resource recycling, *International Journal of Systems Science*, **37** (2006), 27–34.

Financial scheduling with time-dependent resource consumption

Krzysztof M. Ocetkiewicz[*]
Gdańsk University of Technology, Gdańsk, Poland

Michał Małafiejski
Gdańsk University of Technology, Gdańsk, Poland

Keywords: financial scheduling, non-renewable resources, time-dependent scheduling, dynamic programming, approximation schemes

1 Introduction

Consider the following scenario. After a period of careless spending, a person finds oneself with a number of loans to repay. Each loan has a known characteristic expressed by due date or interest rate. Fortunately, the person is employed, so there are a number of known money gains. Now, arises the question: it is possible to repay all the loans? If it is, how to repay them with the least amount of money?

In this talk, we investigate the following problem. We are given a single processor and a number of jobs with zero processing times. The execution of each job requires a certain amount of additional, non-renewable resource. This requirement is time-dependent, i.e., for every job a function of time describes the resource requirement to process the job at the given moment. The resource is obtained in a number of gains. Each gain is described by the moment the gain occurs and the amount of resource gained. There is no limited capacity for storing the resource and it does not deteriorate with time. All information is known in advance. We start processing at time $t_0 = 0$ and with no resource.

This problem models the mentioned above situation in the following way. We (the processor) are facing a number of debts to repay (the jobs). Usually, the number of bank transfers that we can issue at any time is not a limiting factor, so completing such jobs requires virtually no time. On the other hand, enough cash, i.e. resource, is needed to make payments. Finally, the income, i.e. resource gains, increases cash reserves at specified moments.

[*] Speaker, e-mail: `Krzysztof.Ocetkiewicz@eti.pg.gda.pl`

2 Problem formulation

There is given one processor and n jobs $J = \{J_1, J_2, \ldots, J_n\}$ with zero processing times. The completion of each job requires consuming a given amount of an additional resource, described by function

$$g_i(s_i) = \begin{cases} a_i, \text{ if } s_i \leq d_i, \\ a_i + b_i, \text{ otherwise,} \end{cases}$$

where s_i is a job's starting time and $a_i \in \mathbb{Z}_+$, $b_i \in \mathbb{Z}_+$ for $i = 1, 2, \ldots, n$. Each job is completed immediately after being started, i.e., $s_i = C_i$ for every job. The amount of available resource is 0 at the schedule's start and increases in moments t_i by $G_i \in \mathbb{Z}_+$ for $i = 1, 2, \ldots k$ and $t_1 < t_2 < \ldots < t_k$. The goal is to check whether there exists a feasible schedule such that all jobs are completed. Denote the problem by $1|NR, ZET, g_i \in \{a_i, a_i + b_i\}|-$. This problem answers the question whether one is able to repay all loans, when missing a due date results in a one-time penalty. We will also extend the above problem to $1|NR, ZET, g_i \in \{a_i, a_i + b_i\} | \sum g_j$, where the $\sum g_j$ denotes the criterion of minimizing the total amount of resource spent to complete all jobs. This, in turn, can be understood as the minimization of the total cost of repaying all loans. Finally, we substitute the step debt growth rate with linear one, obtaining resource requirement function in the form of

$$g_i(s_i) = a_i + b_i s_i.$$

Again, the goal is to answer whether there exists a feasible schedule such that all jobs are completed. We denote the problem by $1|NR, ZET, g_i = a_i + b_i s_i|-$. This model relates to the case when interest rates are enlarging the debt every month.

3 Related research

The problems with *non-renewable resources* are sometimes referred to as *financial scheduling*, since the resources often represent the amounts of available money, fuel or other valuable (and thus limited) commodities. Such problems received some attention in the last years from both theoretical (see [1, 2] for numerous complexity results) and practical point of view (pseudo-polynomial algorithms and approximation schemes [4, 5]). Time dependent resources in different context were considered in [3], where resources are renewable but the available amounts change with time. In other models, available non-renewable resource is used to decrease processing time of time-dependent jobs, [6].

4 Our results

Theorem 1. *The problem* $1|NR, ZET, g_i \in \{a_i, a_i + b_i\}|-$ *is* $\mathcal{NP}$*-complete.*

Proof. We proceed by a reduction from the SUBSET SUM problem, in short *SS*: there is given a sequence $Z = \{z_1, z_2, \ldots, z_n\}$, where $z_i \in \mathbb{Z}_+$ for $1 \le i \le n$ and a number S; is there a subset $Z' \subset Z$ such that $\sum_{z_j \in Z'} z_j = S$? For a given instance I of *SS*, we build an instance I' of $1|NR, ZET, g_i \in \{a_i, a_i + b_i\}|-$ as follows: $a_i = b_i = z_i$, $d_i = 2$ for every item z_i in the *SS* problem, $k = 2$, $t_1 = 1$, $G_1 = S$, $t_2 = 3$, $G_2 = 2(\sum_{i=1}^{n} z_i - S)$. If there exists a positive answer to I, then one can schedule jobs corresponding to elements in the Z' set at the moment 1 and the remaining jobs at the moment 3. On the other hand, I' has a positive answer only if jobs scheduled up to the moment 2 will consume exactly S units of the resource, which means that there exists also a positive answer to I. □

For solving this problem, we will employ a dynamic programming algorithm. Without loss of generality, we can reorder all jobs such that $d_1 \le d_2 \le \ldots \le d_n$. If two jobs have the same due date, we put them in arbitrary order. Let $D[i, P]$ be equal to the smallest amount of resource that must be spent to complete in time jobs from the set $Z_i = \{J_1, J_2, \ldots, J_i\}$ up to the moment d_i, while gathering total penalty equal to $P = \sum_{j \in ZP_i} b_j$, where ZP_i is a subset of Z_i containing the penalized jobs. If a given value of P cannot be obtained, we set $D[i, P] = +\infty$. Clearly, $D[0, 0] = 0$, $D[i, 0] = \sum_{j=1,2,\ldots,i} a_j$ and $D[0, p] = +\infty$ for $p > 0$. We will call the algorithm as *Algorithm PP*.

Lemma 2. *Let* S_i *be the total amount of resources available at the moment* d_i*, i.e.* $S_i = \sum_{j: t_j \le d_i} G_j$*. Then*

$$D[i+1, P] = \min \begin{cases} a_{i+1} + D[i, P], & \text{if } a_{i+1} \le S_{i+1}, \\ D[i, P - b_{i+1}], & \text{if } P \ge b_{i+1}, \\ +\infty & \text{otherwise.} \end{cases} \tag{1}$$

In view of Lemma 2, all we need to do is to find the smallest $p \in \{0, 1, \ldots, P\}$ such that $D[n, p] < +\infty$ and $\sum_{i=1}^{n} a_i + p \le S_n$, where $P = \sum_{i=1}^{n} b_i$ is the largest value of attainable penalty.

Lemma 3. *Let* $b > 0$*,* $\varepsilon > 0$*. Transform the instance* I *of our problem, obtaining instance* I'*, in such a way that:* $R = \frac{\varepsilon b}{n}$*,* $n' = n$*,* $k' = k$*,* $a'_i = a_i$*,* $g'_j = g_j$*,* $d'_i = d_i$*,* $G'_j = G_j$*,* $b'_i = \left\lceil \frac{b_i}{R} \right\rceil$ *for* $i = 1, 2, \ldots, n$*,* $j = 1, \ldots, k$*. Let OPT be the value of the optimal solution to* I*. Applying Algorithm PP with* I' *and* $P \ge OPT$*, we can either find* $(1 + \varepsilon)$*-approximate solution to instance* I *or ensure that* $OPT < b$*.*

On the basis of Lemma 3, one can construct a fully-polynomial approximation scheme (an FPTAS) for the problem, which iteratively invokes Algorithm *PP*. Each invocation will either produce the required result or will decrease the upper bound for optimal solution value.

Theorem 4. *The problem* $1|NR, ZET, g_i = a_i + b_i s_i|-$ *is strongly* $\mathcal{NP}$*-complete.*

Proof (*Sketch*). We proceed by a reduction from the 3-Partition problem. For a given instance I of 3-Partition, we build an instance I' of $1|NR, ZET, g_i = a_i + b_i s_i|-$ as follows: $n = 3m, k = m, t_i = i-1, G_i = iS, g_j(t) = z_j + z_j t$ for $1 \le i \le m$, $1 \le j \le 3m$. If there exists a positive answer to I, then one can schedule jobs corresponding to elements in the Z_i set at the moments $i - 1$ obtaining a feasible schedule. On the other hand, there exists a valid schedule only if at every moment t_i for $1 \le i \le m$ jobs consuming exactly iS units of resource are scheduled, i.e., there exists a set of three jobs, J_a, J_b and J_c, such that $i(z_a + z_b + z_c) = iS$. □

5 Future research

An interesting question arises what happens to the problem when common (unit) or arbitrary processing times of jobs are allowed.

References

[1] E. R. Gafarov, A. A. Lazarev, F. Werner, Single machine scheduling problems with financial resource constraints: Some complexity results and properties, *Mathematical Social Sciences*, **62** (2011), 7–13.

[2] E. R. Gafarov, A. A. Lazarev, F. Werner, A note on the paper ”Single machine scheduling problems with financial resource constraints: Some complexity results and properties” by E.R. Gafarov et al., *Mathematical Social Sciences*, **65** (2013), 232.

[3] T. Tautenhahn, G. J. Woeginger, Unit-time scheduling problems with time dependent resources, *Computing*, **58** (1997), 97–111.

[4] P. Győrgyi, T. Kis, Aproximation schemes for single machine scheduling with non-renewable resource constraints, *Journal of Scheduling*, **17** (2014), 135–144.

[5] P. Győrgyi, T. Kis, Approximability of scheduling problems with resource consuming jobs, *Annals of Operations Research*, **235** (2015), 319–336.

[6] C-M. Wei, J-B. Wang, P. Ji, Single-machine scheduling with time-and-resource-dependent processing times, *Applied Mathematical Modelling*, **36** (2012), 792–798.

Scheduling on identical parallel machines with controllable processing times to minimize the makespan

Daniel Oron
The University of Sydney Business School, Sydney, Australia

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

George Steiner
McMaster University, Hamilton, Ontario, Canada

Keywords: scheduling, parallel machines, controllable processing times, total weighted resource consumption, maximum completion time

1 Introduction

Scheduling with *controllable processing times* has been the focus of extensive research for the last three decades, see Shabtay and Steiner [5] for a survey. In contrast to the common assumption in deterministic scheduling saying that job processing times are constant values, in scheduling with controllable processing times the job durations are controllable through the allocation of resources to job operations. In this talk, we consider a problem of this type.

2 Problem formulation

We focus on the following scheduling problem with controllable processing times. A set of n jobs $J = \{J_1, J_2, ..., J_n\}$ has to processed on a set of m identical machines $M = \{M_1, M_2, ..., M_m\}$ working in parallel. Job preemption is not allowed. The job processing time, $p_j(u_j)$, is a convex function of the amount of a *non-renewable resource*, u_j, that is allocated to the processing operation, i.e., $p_j(u_j) = (\frac{w_j}{u_j})^k$, where w_j is a positive parameter representing the *workload* of job J_j and k is a positive constant. A solution S for our scheduling problem is defined by a partition τ of set J into m subsets $J_{M_1}, J_{M_2}, ..., J_{M_m}$, where J_{M_i} is the set of jobs to be processed on machine M_i, $i = 1, 2, \ldots, m$, and by a resource allocation vector $\mathbf{u} = (u_1, u_2, \ldots, u_n)$. The quality of a solution is measured by two criteria: the first is the makespan criterion given by

$$C_{\max} = \max_{i=1,2,...,m} \left\{ \sum_{J_j \in J_{M_i}} p_j(u_j) \right\} = \max_{i=1,2,...,m} \left\{ \sum_{J_j \in J_{M_i}} \left(\frac{w_j}{u_j} \right)^k \right\} \qquad (1)$$

and the second criterion is the total weighted resource consumption given by

$$U_v = \sum_{j=1}^{n} v_j u_j, \tag{2}$$

where v_j is the cost of one unit of resource allocated to the processing operation of job J_j for $j = 1, 2, \ldots, n$. We focus on the following four different problem variations, where $\overline{U}$ and K are given parameters: (P1) finding a solution $S = (\tau, \mathbf{u})$ minimizing $C_{\max} + U_v$, (P2) finding a solution $S = (\tau, \mathbf{u})$ minimizing $C_{\max}$ subject to $U_v \leq \overline{U}$, (P3) finding a solution $S = (\tau, \mathbf{u})$ minimizing U_v subject to $C_{\max} \leq K$, (P4) identifying a *Pareto-optimal* solution for each Pareto-optimal point, where a solution S with $C_{\max}(S) = K$ and $U_v(S) = \overline{U}$ is called Pareto-optimal if there does not exist another solution S' such that $C_{\max}(S') \leq K$ and $U_v(S') \leq \overline{U}$, with at least one of these inequalities being strict.

3 Related research

Shabtay and Kaspi [6] studied the special of the (P2) problem variation, where $v_j = 1$ for $j = 1, 2, \ldots, n$ and (i) provided a method to obtain the optimal resource allocation for problem (P2) as a function of τ, (ii) use the result in (i) to reduce this special case of (P2) to a purely combinatorial problem, and (iii) show that the reduced combinatorial problem is $\mathcal{NP}$-hard when $m = 2$ by reducing the Partition problem to it. However, the analysis in [6] has some drawbacks. The first is that the analysis is restricted to problem (P2) with $v_j = 1$ for $j = 1, 2, \ldots, n$. The second is that it includes complexity results only for the case where $m = 2$. Finally, it does not provide any practical tool to solve the problem. Our aim in this talk is to bridge these gaps through the analysis of problems (P1)–(P4).

4 Our results

The following lemma presents the optimal resource allocation strategy for (P2) as a function of partition τ.

Lemma 1. *The optimal resource allocation strategy for* (*P2*) *as a function of* τ *is to assign to any job* $J_j \in J_{M_i}$

$$r_j^* = v_j u_j^* = (\theta_j)^{\frac{k}{k+1}} \left(\sum_{J_l \in J_{M_i}} (\theta_l)^{\frac{k}{k+1}} \right)^{\frac{1}{k}} \frac{\overline{U}}{w'_G}$$

units of resource, where $w'_G = \sum_{i=1}^{m} \left(\sum_{J_j \in J_{M_i}} (\theta_j)^{\frac{k}{k+1}} \right)^{\frac{k+1}{k}} = \sum_{i=1}^{m} \left(\sum_{J_j \in J_{M_i}} a_j \right)^{k'}$,

$k' = \frac{k+1}{k} > 1$, $\theta_j = v_j w_j$ *and* $a_j = \left(\theta_j\right)^{\frac{k}{k+1}}$ *for* $j = 1, 2, \ldots, n$.

Based on Lemma 1, we show that the minimal makespan value, considered as a function of τ, can be computed by $C_{\max}(\tau, \mathbf{u}^*(\tau)) = \left(\frac{w'_G}{\overline{U}}\right)^k$. Since both $\overline{U}$ and k are constant parameters, problem (P2) reduces to the following Weighted Workload Partition (WWP) problem: given a set $A = \{a_1, a_2, \ldots, a_n\}$ and a parameter k, find a partition τ of set A into m subsets $J_{M_1}, J_{M_2}, \ldots, J_{M_m}$ such that

$$f(\tau) = w'_G = \sum_{i=1}^{m} \left(\sum_{J_j \in J_{M_i}} a_j \right)^{k'} \tag{3}$$

is minimized.

We also show that remaining variations of our problem, i.e., (P1), (P2) and(P4), reduce to the WWP problem as well, and prove the following complexity result by a reduction from the 3-Partition problem.

Theorem 2. *The decision version of the WWP problem is strongly $\mathcal{NP}$-complete.*

We briefly explain how we can provide a fully-polynomial approximation scheme (an FPTAS) for solving the WWP problem with a fixed number of machines, regardless of the fact that we could not provide a pseudo-polynomial time algorithm for this problem, as the states of the corresponding dynamic programming (DP) algorithm have to include non-integer values, even if the input of the entire instance is restricted to integer values. The fact that an FPTAS can be constructed follows from the fact that a DP algorithm can be constructed for the scaled and rounded version of the problem. By using the appropriate scaling parameter, we can actually ensure that the algorithm runs in polynomial time. Following the method presented in Ibarra and Kim [2], we scale the a_j parameters by a constant K and then round down the resulting scaled parameters. Let $\overline{a}_j = \left\lfloor \frac{a_j}{K} \right\rfloor$ be the scaled and rounded a_j value for $j = 1, 2, \ldots, n$, and let WWP(S& R) be the scaled and rounded version of the WWP problem. Accordingly, our objective in WWP(S& R) is to find a partition τ of set $A = \{1, 2, \ldots, n\}$ into m subsets $\tau_1, \tau_2, \ldots, \tau_m$ such that $\overline{f}(\tau) = \sum_{i=1}^{m} (\sum_{j \in \tau_i} \overline{a}_j)^{k'}$ is minimized. In order to present an FPTAS for the WWP problem, we provide a state space generation algorithm that optimally solve the WWP(S& R) in $O\left(nm\overline{Q}^m\right)$ time, where $\overline{Q} = \sum_{j=1}^{n} \overline{a}_j$. Then, we prove the following theorem.

Theorem 3. *Solving the WWP(S&R) problem with $K = \frac{\log(1+\varepsilon)\Sigma_{j=1}^{n} a_j}{k'mn}$ and $\varepsilon \leq \frac{2}{3}$ yields an FPTAS for the WWP problem.*

The longest processing time (LPT) heuristic is commonly used for solving the strongly $\mathcal{NP}$-hard $P\,||\,C_{\max}$ problem. The heuristic begins by renumbering the n jobs in a non-increasing order of processing times such that $p_1 \geq p_2 \geq \cdots \geq p_n$. Then, for $j = 1, 2, \ldots, n$, the algorithm assigns job J_j to the least loaded machine so far. Graham [1] proved that the LPT heuristic provides $(\frac{4}{3} - \frac{1}{3m})$-approximation algorithm for the $P\,||\,C_{\max}$ problem. A straightforward modification of the LPT heuristic to solve the WWP problem would be to apply it with the a_j parameters replacing the p_j parameters. We prove the following theorem.

Theorem 4. *The modified LPT heuristic is $h(m-\bar{v},k',r^*)$-approximation algorithm for the WWP, where $h(m-\bar{v},k',r^*) = \frac{(m-\bar{v}-r^*)(m-\bar{v}+r^*+1)^{k'}+r^*(r^*+1)^{k'}}{(m-\bar{v})(m-\bar{v}+1)^{k'}}$, with $D(v) = \sum\limits_{j=v+1}^{n} a_j$, $\bar{v} = \sup\{J_1, J_2, \ldots, J_n : a_j \geq \frac{D(j)}{m-j+1}\}$, $A(\bar{v}) = \sum\limits_{j=\bar{v}+1}^{n} \frac{a_j}{m-\bar{v}}$, and r^* is the r value that maximizes*

$$g(r) = \left(\frac{A(\bar{v})}{m-\bar{v}+1}\right)^{k'} \left((m-\bar{v}-r)(m-\bar{v}+r+1)^{k'} + r(r+1)^{k'}\right).$$

References

[1] R. L. Graham, Bounds for certain multiprocessor anomalies, *Bell System Technology Journal*, **45** (1966), 1563–1581.

[2] O. H. Ibarra, C. E. Kim, Fast approximation algorithms for the knapsack and the sum of subset problems, *Journal of the ACM*, **22** (1975), 463–468.

[3] D. Shabtay, Single and a two-resource allocation algorithms for minimizing the maximal lateness in a single machine scheduling problem, *Computers & Operations Research*, **31** (2004), 1303–1315.

[4] D. Shabtay, M. Kaspi, Minimizing the total weighted flow time in a single machine with controllable processing times, *Computers & Operations Research*, **31** (2004), 2279–2289.

[5] D. Shabtay, G. Steiner, A survey of scheduling with controllable processing times, *Discrete Applied Mathematics*, **155** (2006), 1643–1666.

[6] D. Shabtay, M. Kaspi, Parallel machine scheduling with a convex resource consumption function, *European Journal of Operations Research*, **173** (2006), 92–107.

Precedence constrained position-dependent scheduling on parallel machines via schedule transformations

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

Keywords: scheduling, position-dependent jobs, precedence constraints, polynomial algorithms, maximum completion time, total weighted completion time

1 Introduction

In many scheduling problems one cannot assume that job processing times are fixed. There are two main groups of scheduling problems with variable job processing times which are considered in literature: *time-dependent scheduling*, where the processing time of a job depends on its starting time (for a good review see [2]) and *position-dependent scheduling*, where the processing time of a job depends on its position in a schedule (for a survey see [1]).

The majority of papers on scheduling problems with variable processing times entail two limitations. First, the results are mainly connected to scheduling on one machine (see, e.g., [7]) or on parallel machines but with empty precedence constraints among jobs (see, e.g., [5]). Second, the most of results related to scheduling problems with variable job processing times concern particular cases, not general ones. Examples of recent general results are presented in [3].

2 Problem formulation

The group of problems to be presented during the talk can be formulated as follows. We are given a number of parallel identical machines and n non-preemptable jobs with some precedence constraints. Job processing times are variable and depend on the position r of a job in a schedule. This dependency is described by a positive and non-increasing discrete function φ of argument r. In other words, the processing time of the j-th job on r-th position in a schedule is equal to $p_{j,r} = \varphi(r)$, where $1 \leq r, j \leq n$. The aim is to find a schedule with minimal maximum completion time or minimal total weighted completion time. Using the commonly known scheduling notation (for a brief description see [2]), we denote these problems by $\mathrm{P}|\mathrm{prec}, p_{j,r} = \varphi(r)|f$, where $f \in \{C_{\max}, \sum w_i C_i\}$.

[*] Speaker, email: `bap@amu.edu.pl`

Special cases of these problems, when $\varphi(r) = 1$ for every r, are the problems of scheduling jobs with unit job processing times, $P|\text{prec}, p_{j,r} = 1|f$. Therefore, a natural question is whether some known algorithms for the latter problem can be applied to their counterparts with job processing times described by the φ function. We give an answer to this question applying a new methodology of transforming schedules. We also specify certain conditions, under which the $P|\text{prec}, p_{j,r} = \varphi(r)|f$ problems can be solved by appropriately modified algorithms for unit job processing times.

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 $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. Moreover, let Φ be a function defined as $\Phi(k) = \sum_{i=1}^{k} \varphi(i)$, where $k \in \mathbb{N}$. Notice that $\Phi(0) = 0$ and that the Φ function is a bijection between its domain and image.

A schedule of n jobs is called a *φ-natural schedule*, if there is at least one job, such that the starting time of this job is equal to 0 and for every job both its starting and its completion time are the values of the Φ function at some points. If every job started at time $\Phi(k)$ is completed at time $\Phi(k+1)$, then the schedule is called a *φ-simple schedule*. If $\varphi(r) = 1$ for every r, then every φ-natural schedule is called a *natural schedule* and, consequently, every φ-simple schedule is called a *simple schedule*. Notice, that every simple schedule is a schedule of jobs with unit processing times.

A schedule is called a *continuous schedule*, if every machine starts the processing of jobs at the moment 0 and none of the machines is idle before finishing all the jobs assigned to it.

Lemma 1. *Let there be given a number of identical parallel machines able to execute each of available jobs. If an algorithm A generates a continuous simple schedule for job processing times in the form of $p_{j,r} = 1$, then algorithm A generates a continuous φ-simple schedule for job processing times in the form of $p_{j,r} = \varphi(r)$.*

The above lemma is used in proofs of the following theorems.

Theorem 2. *Let φ be a positive non-increasing discrete function and let I be arbitrary instance of the* $P|\text{prec}, p_{j,r} = 1|f$ *problem, where $f \in \{C_{max}, \sum w_i C_i, \sum C_i\}$. If algorithm A generates an optimal schedule for I and it is a continuous simple schedule, then algorithm A generates also an optimal schedule for the corresponding instance of the* $P|\text{prec}, p_{j,r} = \varphi(r)|f$ *problem and it is a continuous φ-simple schedule.*

Theorem 3. *If the φ function is positive and non-increasing with respect to r, then Hu's algorithm* [4] *solves the* $P|\text{in-tree}, p_{j,r} = \varphi(r)|C_{\max}$ *problem.*

We will present and comment several examples of applications of the above theorems. Moreover, we will present some examples showing that the assumptions of our theorems are reasonably strong. In particular, we will show some counterexamples to several hypotheses, where the assumptions are weakened.

4 Future research

The results can be a base for further research on the problem of applying known algorithms to more general cases. It is an open question, whether we can construct transformations such as presented here for other classes of schedules.

References

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

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

[3] S. Gawiejnowicz, A. Kononov, Isomorphic scheduling problems, *Annals of Operations Research*, **213** (2014), 131–145.

[4] T-C. Hu, Parallel sequencing and assembly line problems, *Operations Research*, **9** (1961), 841–848.

[5] 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.

[6] B. Przybylski, Precedence constrained parallel-machine scheduling of position-dependent jobs, submitted.

[7] J-B. Wang, J-J. Wang, Single-machine scheduling problems with precedence constraints and simple linear deterioration, *Applied Mathematical Modelling*, **39** (2014), 1172–1182.

Single machine scheduling subject to a generalized linear cumulative effect

Kabir Rustogi
University of Greenwich, London, United Kingdom

Vitaly A. Strusevich[*]
University of Greenwich, London, United Kingdom

Keywords: single machine, maintenance, approximation schemes, half-product

1 Introduction

This talk is devoted to a single machine scheduling model, in which actual processing times of the jobs are not constant but are subject to a special effect. Each job $j \in N = \{1, 2, \ldots, n\}$ is associated with an integer p_j that is called its ”normal” processing time. This value can be understood as the actual processing duration of job j, provided that the machine is in some initial state, and may change due to a certain effect that extends a traditional *cumulative effect*. Under the cumulative effect the actual processing time of job j depends on p_j and on the sum of normal processing times of all jobs sequenced earlier; see, e.g., [6]. In our model, the actual processing time of a job depends on the sum of certain parameters, other than normal processing times, associated with previously scheduled jobs. For the introduced model, we solve the problem of minimizing the makespan, with and without precedence constraints. We also consider a situation when a maintenance activity is included into a schedule and develop a fast fully polynomial-time approximation scheme for this case.

2 Problem formulation

Suppose that the jobs of set $N = \{1, 2, \ldots, n\}$ are processed on a single machine in accordance with a sequence $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$. Let $p_j(\pi; r)$ denote the actual processing time of job $j = \pi(r)$ scheduled in the r-th position of π. Under the most studied cumulative effect, introduced in [6], we have that

$$p_j(\pi; r) = p_j \left(1 + b \sum_{h=1}^{r-1} p_{\pi(h)}\right)^A, \qquad (1)$$

[*] Speaker, email: `V.Strusevich@greenwich.ac.uk`

where A is a given constant, and b is either equal to 1 or to -1. A drawback of the model with a cumulative effect (1) is that normally no convincing practical motivation of the model is given.

In this talk, we study a cumulative effect that arises when a job $j \in N$ is associated not only with the normal processing time p_j but also with two additional parameters, b_j and $q_j > 0$, so that

$$p_j(\pi; r) = p_j \left(1 + b_j \sum_{h=1}^{r-1} q_{\pi(h)} \right). \tag{2}$$

For illustration of our model, suppose that a floor sanding machine is used to treat floors in several rooms. The normal time p_j is the time requirement for sanding floors in room j, provided that the new sanding belt/disk is used. The value of q_j can be seen as the amount generated saw dust or an appropriately measured wear of the sanding belt/disk, which depends on the area of room j and the initial quality of the floor in the room but not explicitly on the time of treatment. The rate parameter b_j captures the fact that for some rooms the effect is less noticeable.

Let P_r (respectively, Q_r) denote the sum of the p_j values (respectively, q_j values) associated with the jobs scheduled prior to job $\pi(r)$. Denote the problem of minimizing the makespan $C_{\max}$ under the effect (1) by $1 \left| p_j (1 + bP_r)^A \right| C_{\max}$ and its counterpart under the effect (2) by $1 \left| p_j \left(1 + b_j Q_r\right) \right| C_{\max}$.

3 Our results

By reducing problem $1 \left| p_j \left(1 + b_j Q_r\right) \right| C_{\max}$ to the classical scheduling problem $1 | | \sum w_j C_j$ of minimizing the sum of weighted completion times, we prove the following result.

Theorem 1. *For problem $1 \left| p_j \left(1 + b_j Q_r\right) \right| C_{\max}$, an optimal schedule can be found in $O(n \log n)$ time by sorting the jobs in non-increasing order of the ratios $\frac{p_j b_j}{q_j}$.*

We also study a version of problem $1 \left| p_j \left(1 + b_j Q_r\right) \right| C_{\max}$ in which precedence constraints are imposed on the set of jobs. These precedence constrains are given in a form of a acyclic directed graph, with the nodes representing the jobs. Provided that the digraph is series-parallel, we denote the problems under effects (1) and (2) by $1 \left| p_j (1 + bP_r)^A, SP - prec \right| C_{\max}$ and $1 \left| p_j \left(1 + b_j Q_r\right), SP - prec \right| C_{\max}$, respectively. We refer the reader to [3] for a range of results on single machine scheduling with series-parallel precedence constraints and various effects, including problems $1 \left| p_j (1 + P_r), SP - prec \right| C_{\max}$ and $1 \left| p_j (1 + P_r)^2, SP - prec \right| C_{\max}$.

Extending the example given above, problem $1\left|p_j\left(1+b_jQ_r\right), SP-prec\right| C_{\max}$ can arise if precedence constraints occur due to a particular physical layout of the building in which the rooms to be sanded are located.

Using the theory of minimizing *priority-generating functions* over series-parallel precedence constraints, which is described in detail in [7], we prove that for problem $1\left|p_j\left(1+b_jQ_r\right), SP-prec\right| C_{\max}$ the objective function is priority-generating and, therefore, the problem is solvable in $O\left(n\log n\right)$ time.

For the model with $b_j>0$, the actual processing times grow, and a maintenance period (MP) can be introduced into a schedule. During an MP no processing takes place. The duration of an MP is either a constant or depends on its start time τ. In this talk, we address problem $1\left|p_j\left(1+b_jQ_r\right), MP\left(\lambda\right)\right| C_{\max}$, where $MP\left(\lambda\right)$ means that the duration of the introduced single MP is $\lambda\tau+\mu$, for given constants λ and μ; in particular, $MP\left(0\right)$ corresponds to the MP of duration μ. Problem $1\left|p_j\left(1+bP_r\right), MP\left(\lambda\right)\right| C_{\max}$ is studied in [4]. For problem $1\left|p_j\left(1+b_jQ_r\right), MP\left(\lambda\right)\right| C_{\max}$, we additionally allow that the MP does not restore the processing conditions back to the initial state, so that for job j scheduled after the MP its normal processing time changes from p_j to σp_j, where $\sigma\geq 1$.

We demonstrate that problem $1\left|p_j\left(1+b_jQ_r\right), MP\left(\lambda\right)\right| C_{\max}$ can be reformulated as the problem of Boolean programming of minimizing a function $F(\mathbf{x})=H(\mathbf{x})+K$, where its variable part $H(\mathbf{x})$ is a special non-separable quadratic function, known as the *half-product* [1]. The problem of minimizing function $H(\mathbf{x})$ is $\mathcal{NP}$-hard in the ordinary sense, and the best known fully polynomial-time approximation scheme (FPTAS) requires $O\left(\frac{n^2}{\varepsilon}\right)$ time [2]. However, an FPTAS for minimizing the function $H(\mathbf{x})$ does not necessarily behave as an FPTAS for minimizing the function $F(\mathbf{x})=H(\mathbf{x})+K$ with an additive constant; see [2] and [5]. For the problem of minimizing $F(\mathbf{x})$ the following approach is outlined in [2]. Let LB and UB be the lower and upper bounds on the optimal value of $F(\mathbf{x}^*)$.

Theorem 2. *For the problem of minimizing function $F(\mathbf{x})$, if the ratio $\frac{UB}{LB}$ is bounded from above by some positive γ, then there exists an algorithm that delivers a solution $\mathbf{x}^0$ such that $F(\mathbf{x}^0)-LB\leq\varepsilon LB$ in $O(\gamma\frac{n^2}{\varepsilon})$ time.*

If the value of γ is bounded from above by a constant, then the algorithm from Theorem 2 behaves as an FPTAS which requires $O\left(\frac{n^2}{\varepsilon}\right)$ time.

To be able to apply Theorem 2 we do the following: (i) reformulate problem $1\left|p_j\left(1+b_jQ_r\right), MP(\lambda)\right| C_{\max}$ as the problem of minimizing a special form of a half-product related function $F(\mathbf{x})$, known as a *symmetric quadratic function* [5]; (ii) prove that the objective function is convex; (iii) determine the lower bound $LB\leq F(\mathbf{x}^*)$ by solving the continuous relaxation of the problem in $O\left(n^2\right)$ time, which is done by reducing the relaxed problem to finding a flow in a special network that minimizes a convex quadratic cost function; (iv) round the solution to the continuous relaxation to find an upper bound UB such that there holds $LB \leq F(\mathbf{x}^*) \leq UB \leq 4LB$. Thus, Theorem 2 holds with $\gamma = 4$, and hence problem $1 \left| p_j \left(1 + b_j Q_r\right), MP(\lambda) \right| C_{\max}$ admits an FPTAS that requires $O\left(\frac{n^2}{\varepsilon}\right)$ time.

References

[1] T. Badics, E. Boros, Minimization of half-products, *Mathematics of Operations Research*, **33** (1998), 649–660.

[2] E. Erel, J. B. Ghosh, FPTAS for half-products minimization with scheduling applications, *Discrete Applied Mathematics*, **156** (2008), 3046–3056.

[3] V. S. Gordon, C. N. Potts, V. A. Strusevich, J. D. Whitehead, Single machine scheduling models with deterioration and learning: handling precedence constraints via priority generation, *Journal of Scheduling*, **11** (2008), 357–370.

[4] H. Kellerer, K. Rustogi, V. A. Strusevich, Approximation schemes for scheduling on a single machine subject to cumulative deterioration and maintenance, *Journal of Scheduling*, **16** (2013), 675–683.

[5] H. Kellerer, V. A.Strusevich, The symmetric quadratic knapsack problem: approximation and scheduling applications, *4OR*, **10** (2012), 111–161.

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

[7] V. S. Tanaev, V. S. Gordon, Y. M. Shafransky, *Scheduling Theory: Single-Stage Systems*, Kluwer, Dordrecht, 1994.

An approach for proving the NP-hardness of optimization problems with hard computable objectives

Yakov M. Shafransky[*]
The United Institute of Informatics Problems of the NAS Belarus, Minsk, Belarus

T-C. Edwin Cheng
The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

C-T. Daniel Ng
The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

Keywords: combinatorial optimization, computational complexity, scheduling

1 Introduction

We consider a general *combinatorial optimization problem* P_{opt} in the following setting, which is consistent with the definitions given in standard texts such as Garey and Johnson [2], Papadimitriou and Steiglitz [4], Papadimitriou [3], and Schrijver [5]. (We confine our consideration to minimization problems although all the results are also valid for maximization problems.) There is given a set D_P of problem instances, a finite set $X(I)$ of feasible elements for each $I \in D_P$, and a real-valued objective function $F(I,x)$, $x \in X(I)$. The aim is to find $x^* \in X(I)$ such that $F(I,x^*) = \min\{F(I,x)|x \in X(I)\}$ for the given instance $I \in D_P$ if $X(I) \neq \emptyset$ or report that $X(I) = \emptyset$.

The elements x^* are called the *optimal elements*.

Hereinafter, x^* means either $\arg\min\{F(I,x)|x \in X(I)\}$ if $X(I) \neq \emptyset$ or $X(I) = \emptyset$.

We provide a new reduction-based approach for proving the $\mathcal{NP}$-hardness of optimization problems and establish that it includes the "classical" approach as a special case. We apply our alternative approach to prove the $\mathcal{NP}$-hardness of a problem that defies the classical approach. Besides, we construct a special case of the problem with the property that finding an optimal element takes polynomial time despite that computing the objective function values is $\mathcal{NP}$-hard.

2 Our results

Following the conventional notion (see, e.g., [2]), we say that problem P_{opt} is $\mathcal{NP}$-hard if there is a *Turing reduction* from any problem in $\mathcal{NP}$ to P_{opt}.

[*] Speaker, email: `shafr-04@yandex.ru`

Consider the following problem associated with P_{opt}:

Standard Decision Problem
Instance: $I \in D_P$ and a number y.
Output: "yes" if there is an $x^0 \in X(I)$ such that $F(I, x^0) \le y$; "no" otherwise.

In particular, if $X(I) = \emptyset$, then the output is "no".

To show the $\mathcal{NP}$-hardness of an optimization problem using the "classical" approach, one constructs a Standard Decision Problem that corresponds to the original problem P_{opt} and proves the $\mathcal{NP}$-hardness of the former problem. We refer to this procedure as the *standard scheme* for proving the $\mathcal{NP}$-hardness of an optimization problem.

Since $x^0 \in X(I)$ such that $F(I, x^0) \le y$ exists if and only if $F(I, x^*) \le y$, the validity of the standard scheme is based on the following Turing reduction of the Standard Decision Problem to the optimization problem.

Step (a). Solve the optimization problem P_{opt}.
Step (b). Given x^* (found in Step (a)), calculate $F(I, x^*)$ (if $X(I) \ne \emptyset$).
Step (c). Compare $F(I, x^*)$ and number y. Return "yes" if $F(I, x^*) \le y$; otherwise (if $F(I, x^*) > y$ or if $X(I) = \emptyset$), return "no".

If both Step (a) and Step (b) can be executed in polynomial time, then the Standard Decision Problem is polynomially solvable. If the Standard Decision Problem is proved to be $\mathcal{NP}$-hard and Step (b) can be executed in polynomial time, then Step (a) cannot be executed in polynomial time (unless $\mathcal{P} = \mathcal{NP}$). The latter fact allows us to conclude that the optimization problem is $\mathcal{NP}$-hard. On the other hand, if we have no polynomial algorithms to implement Step (b), then we can say nothing about the $\mathcal{NP}$-hardness of problem P_{opt}.

Thus, the standard scheme works correctly if and only if Step (b) can be executed in time polynomial of the input size of problem P_{opt}.

Following [2], we denote by $Length[I]$ the length of a reasonable encoding of instance I.

By and large, the standard scheme works quite well. However, it cannot cope with some odd cases where, e.g., given $x^* \in X(I)$ for P_{opt} problem, the value of $F(I, x^*)$ cannot be computed in time polynomial of $Length[I]$.

It may be argued that such odd cases need no additional research because the $\mathcal{NP}$-hardness of computing $F(I, x^*)$ implies the $\mathcal{NP}$-hardness of searching for x^*. To counter this argument, we present a polynomially solvable optimization problem with an objective function that cannot be calculated in polynomial time if $\mathcal{P} \ne \mathcal{NP}$. To construct an example of such a problem, we formulate a scheduling problem involving parallel machines and jobs with start-time and position dependent processing times.

It seems that some researchers bear in mind the standard scheme in defining special classes of optimization problems. Vazirani [6, Appendix A] considers a class of $\mathcal{NP}$-optimization problems consisting of all optimization problems with polynomially computable objective functions. The same restriction is used by Ausiello et al. [1, Chapter 1] to define the class $\mathcal{NPO}$ of so-called *constructive problems*.

We do not confine our discussion to problems with polynomially computable objective functions. So, the problems under consideration are in general outside the class $\mathcal{NPO}$ and they are not $\mathcal{NP}$-optimization problems.

To outline the basic idea of our approach to proving the $\mathcal{NP}$-hardness of optimization problems, we first analyze the standard scheme from the following point of view. The Standard Decision Problem is formulated in such a way that it has answer ”yes” if and only if $F(I, x^*) \leq y$ for a given number y. If we have the optimal element x^*, then we can easily check whether $F(I, x^*) \leq y$ holds (if $F(I, x^*)$ is easily computable). On the other hand, if we do not have x^*, then the Standard Decision Problem should be $\mathcal{NP}$-hard, which implies the $\mathcal{NP}$-hardness of the corresponding optimization problem.

We develop our approach for analyzing problems with hard computable objective functions. This means that we should avoid the calculation of the objective function values in our process of proving $\mathcal{NP}$-hardness. In the standard scheme, we use a property of the problem instance I to provide the correctness of the inequality $F(I, x^*) \leq y$. The particular instance I may have this property (then the answer is ”yes” for the Standard Decision Problem) or it may not have the property (then the answer is ”no”). Note that the checking of the property for I should be done without knowing the element x^* for instance I.

The main idea of our approach is to replace the property based on the inequality $F(I, x^*) \leq y$ with a different property that does not involve the calculation of the objective function values. The new property should possess two features. It should be easily checkable if we are given an optimal element x^* and its checking should be an $\mathcal{NP}$-hard problem if we have instance I and do not have x^*. Then the $\mathcal{NP}$-hardness of the optimization problem will follow immediately from an argument similar to that used in case of the standard scheme.

The idea looks quite simple and almost evident upon its discovery. As with any discovery, the most challenging aspect is to stumble upon an innovative insight into a problem, which may take a long time to emerge.

The list of our results is as follows. We formulate our alternative approach to proving the $\mathcal{NP}$-hardness of optimization problems and show that the classical approach is a special case of it. We formulate and analize a scheduling problem with start-time and position dependent processing times of jobs (Problem 1) and propose a simple but non-polynomial algorithm to solve it. We show that the objective function values of Problem 1 cannot be computed in polynomial time if $\mathcal{P} \neq \mathcal{NP}$. Besides, we construct a polynomially solvable special case of Problem 1 with the same property of having a hard computable objective function. We use the alternative approach to show the $\mathcal{NP}$-hardness of Problem 1.

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer-Verlag, Berlin-Heidelberg, 1999.

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

[3] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, MA, 1995.

[4] C. H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[5] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, Springer-Verlag, Berlin-Heidelberg, 2003.

[6] V. V. Vazirani, *Approximation Algorithms*, Springer-Verlag, Berlin-Heidelberg, 2001.

Indexes

Index of authors

Index of keywords