Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

UAM
Adam Mickiewicz
University
Poznań

THE FOURTH INTERNATIONAL WORKSHOP ON DYNAMIC SCHEDULING PROBLEMS

ZURICH UNIVERSITY OF APPLIED SCIENCES (ZHAW)
SCHOOL OF ENGINEERING
JUNE 5TH–6TH, 2023, WINTERTHUR, SWITZERLAND

EXTENDED ABSTRACTS

IWDSP
WINTERTHUR 2023

Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

UAM

Adam Mickiewicz
University
Poznań

THE FOURTH INTERNATIONAL WORKSHOP ON DYNAMIC SCHEDULING PROBLEMS

ZURICH UNIVERSITY OF APPLIED SCIENCES (ZHAW)
SCHOOL OF ENGINEERING
JUNE 5TH–6TH, 2023, WINTERTHUR, SWITZERLAND

EXTENDED ABSTRACTS

IWDSP
WINTERTHUR 2023

This book contains extended abstracts of the plenary lecture and papers presented at the Fourth International Workshop on Dynamic Scheduling Problems, June 5th–6th, 2023, Winterthur, Switzerland.

Cover design by Bartłomiej Przybylski
Edited by Stanisław Gawiejnowicz

ISBN: 978-83-962157-0-3 (printed version)
ISBN: 978-83-962157-1-0 (eBook)
ISBN: 978-83-962157-2-7 (printed version + eBook)
DOI: 10.14708/isbn.978-83-962157-1-0

Publisher:

Polish Mathematical Society
Śniadeckich 8, 00-956 Warsaw
Poland

Welcome to IWDSP 2023

Dear participant,

on behalf of the Program and Local Committees, we are pleased to welcome you to IWDSP 2023, the Fourth International Workshop on Dynamic Scheduling Problems, and to the ZHAW Institute of Data Analysis and Process Design, Zurich University of Applied Sciences, Winterthur, Switzerland, and the Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland, which are the hosts of this event.

The IWDSP 2023 workshop is the fourth event in the series started in 2016 with IWDSP 2016 and continued in 2018 with IWDSP 2018 and in 2021 with IWDSP 2021. These three workshops attracted the authors from Australia, Belarus, Belgium, Canada, Egypt, France, Germany, India, Israel, Italy, the Netherlands, Poland, P. R. China, Russian Federation, Taiwan, the United Kingdom and the United States.

Books of extended abstracts of the papers presented by the authors on the IWDSP 2016, IWDSP 2018 and IWDSP 2021 workshops are available at the Web sites `https://iwdsp2016.wmi.amu.edu.pl`, `https://iwdsp2018.wmi.amu.edu.pl` and `https://iwdsp2021.wmi.amu.edu.pl`, respectively. Selected revised papers presented at the IWDSP 2018 workshop have also been published in a special issue (see issue no. 6 of volume 23 of the Journal of Scheduling, available at `https://link.springer.com/journal/10951/volumes-and-issues/23-6`). A similar issue of the journal devoted to the IWDSP 2021 workshop will be published soon.

IWDSP 2023, similarly as the previous three workshops in the series, focuses on dynamic scheduling problems defined by parameters whose values are varying in time and which often appear in applications. Therefore, main topics related to the workshop scope include

- *scheduling with variable job processing*, e.g. controllable job processing times, time-dependent job processing times, position-dependent job processing times, resource-dependent job processing times, discrete-continuous scheduling;
- *scheduling with various factors affecting job execution*, e.g. aging, alteration, deterioration, learning, shortening;
- *scheduling on variable speed machines*, e.g. energy-efficient scheduling, scheduling under time-of-use electricity tariffs, scheduling with rate-modifying activities;

- *scheduling problems with constraints on machine availability*, e.g. scheduling with maintenance activities, scheduling on machines with non-availability periods;
- *scheduling under uncertainty*, e.g. robust scheduling, stochastic scheduling, scheduling over scenarios, scheduling with explorable uncertainty;
- *scheduling with a partial, changing in time, data on jobs or machines*, e.g. online scheduling, semi-online scheduling;
- *scheduling in non-classic models of job preemption*, e.g. malleable task scheduling, scheduling pliable jobs, scheduling splittable jobs;
- other *scheduling problems with job or machine parameters changing in time*, e.g. scheduling in data gathering networks, scheduling in health care systems, etc.

The aim of the IWDSP 2023 workshop is to present the recent research in these important domains of scheduling theory.

The Program Committee, supported by the members of the Advisory Committee and external reviewers, selected for presentation at the IWDSP 2023 workshop papers submitted by the authors from Australia, Brazil, Colombia, France, Germany, Israel, Italy, Poland, P. R. China, Switzerland and the United States. These papers, together with a plenary lecture on scheduling over scenarios, allowed the committee to prepare an attractive program of the event.

We wish you a fruitful workshop, expressing the hope that you will find IWDSP 2023 stimulating for your further research.

Stanisław Gawiejnowicz
The Chair of the Program Committee
The Co-chair of the Local Committee
Helmut A. Sedding
The Co-chair of the Local Committee

Committees

Committees

Program Committee

Stanisław GAWIEJNOWICZ (Chair), Adam Mickiewicz University, Poznań, Poland

Gur MOSHEIOV, Hebrew University, Jerusalem, Israel

Advisory Committee

Alessandro AGNETIS, University of Siena, Siena, Italy

Jose M. FRAMINAN TORRES, University of Sevilla, Sevilla, Spain

Hans KELLERER, University of Graz, Graz, Austria

Chung-Lun LI, The Hong Kong Polytechnic University, Hong Kong, People's Republic of China

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

Viswanath NAGARAJAN, University of Michigan, Ann Arbor, USA

Daniel ORON, University of Sydney, Sydney, Australia

Panos M. PARDALOS, University of Florida, Gainesville, USA

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

Suresh P. SETHI, The University of Texas at Dallas, Dallas, USA

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

Zhiyi TAN, Zhejiang University, Hangzhou, People's Republic of China

Local Committee

Stanisław GAWIEJNOWICZ (co-Chair), Adam Mickiewicz University, Poznań, Poland

Andreas KLINKERT, Zurich University of Applied Sciences, Winterthur, Switzerland

Bartłomiej PRZYBYLSKI, Adam Mickiewicz University, Poznań, Poland

Helmut A. SEDDING (co-Chair), Zurich University of Applied Sciences, Winterthur, Switzerland

Contents

Venue

Venue

IWDSP 2023 takes place in ZHAW's TP building on the upper floor in room TP404. This building is in the west of the ZHAW campus in Technikumstrasse 9, 8400 Winterthur, Switzerland.

The historical physics building (TP) was finished in 1960 as part of the *Technikum Winterthur*, which was funded in 1874. It was the first of its kind in Switzerland and served as a role model for educating machine and electrical technicians. It extended its scope to chemists, precision engineers and railway administrators. In 1901, Albert Einstein teaches in Winterthur for two months on a temporary contract, teaching mathematics and descriptive geometry. In 1998, the school was joined with other institutions as a university of applied sciences. Since 2007 it is named *Zürcher Hochschule für Angewandte Wissenschaften* (ZHAW), enrolling over 14 000 students and 3 500 full time employees.

The oldest institute of the university, founded 22 years ago on May 7, 2001, is the *Institute of Data Analysis and Process Design*. Its research focus lies in machine learning and statistics, finance, transport, aviation, and production operations research. It is now proud to host the workshop, organized together with the Adam Mickiewicz University, Poznań, Poland.

An old school pitched floor lecture hall houses the talks. During the breaks in the upper floor hall we enjoy a wonderful view on the old Technikum building of 1879.

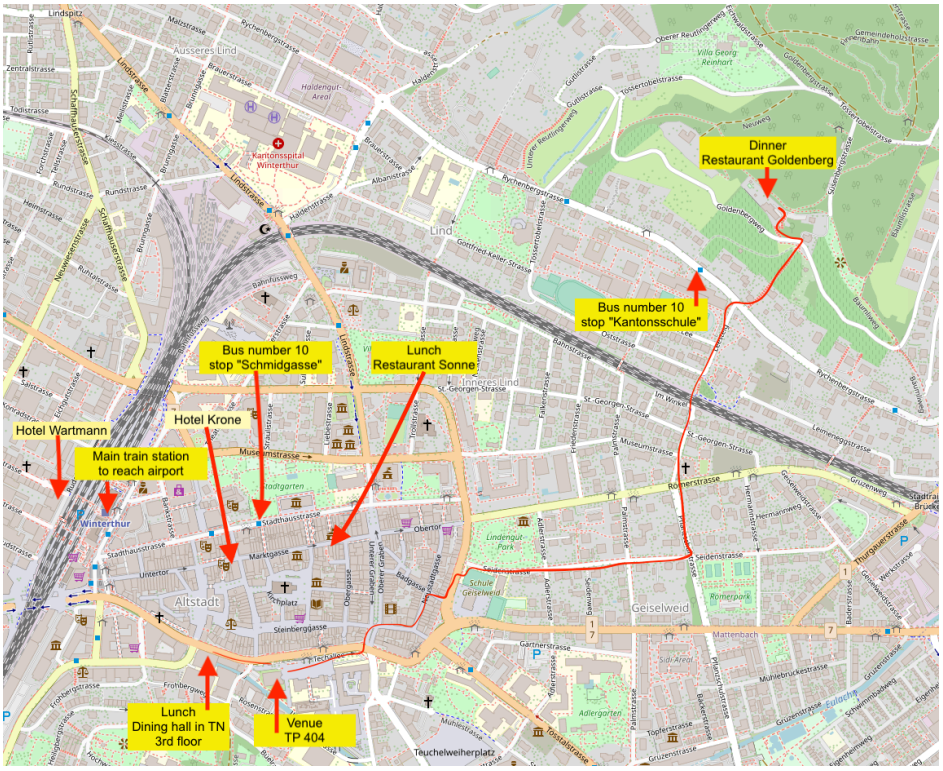
Lunch

Lunch is served in the dining hall *Mensa Eulachpassage* on the third floor of the newly erected TN building located at Technikumstrasse 71.

Dinner

Monday's dinner is located at *Restaurant Goldenberg* in a former mansion in late neoclassical architecture built in 1928/29 and renovated in 2010. The terrace provides a magnificent view on Winterthur's old town in the south east and the Alpine mountains in the south west. We meet in the rear area for an Apéro, unless the weather lets us retreat inside. Traditional Swiss dinner is served in a panoramic chamber.

The restaurant is reached from the venue by a walk of two kilometers length and fifty meters height difference. A shortcut is offered by bus line number 10 to Oberwinterthur from station Schmidgasse to bus stop Kantonsschule.



The city of Winterthur

Map image source and copyright: OpenStreetMap contributors

Programme

Monday, June 5th, 2023

08:00 – 08:30	Registration
08:30 – 09:00	Opening
09:00 – 10:20	**Session no. 1**
	Speakers: Dvir Shabtay (online), Johnson Phosavanh
	Chair: Stanisław Gawiejnowicz
10:20 – 10:40	Coffee break
10:40 – 12:00	**Session no. 2**
	Speakers: Arthur Kramer, Nir Halman
	Chair: Gur Mosheiov
12:00 – 14:00	Lunch break
14:00 – 15:20	**Plenary lecture**
	Speaker: Leen Stougie
	Chair: Helmut A. Sedding
15:20 – 15:40	Coffee break
15:40 – 17:00	**Session no. 3**
	Speakers: Stanisław Gawiejnowicz,
	~~Christoph Damerius~~ (cancelled)
	Chair: Leen Stougie
19:00 – 21:30	Conference dinner

Tuesday, June 6th, 2023

08:30 – 9:00	Registration
09:00 – 10:20	**Session no. 4**
	Speakers: Joanna Berlińska, Baruch Mor
	Chair: Gur Mosheiov
10:20 – 10:40	Coffee break
10:40 – 12:00	**Session no. 5**
	Speakers: Yenny A. Paredes-Astudillo, Gur Mosheiov
	Chair: Stanisław Gawiejnowicz
12:00 – 14:00	Lunch break
14:00 – 15:20	**Session no. 6**
	Speakers: Stanisław Gawiejnowicz, Helmut A. Sedding
	Chair: Nir Halman
15:20 – 15:40	Coffee break
15:40 – 16:00	Closing

Plenary lecture

DOI: 10.14708/isbn.978-83-962157-1-0p21-22
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

Scheduling over scenarios

Leen Stougie*
CWI and Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

Keywords: scheduling over scenarios, robust scheduling, stochastic scheduling

Apart from some exceptions, modeling *uncertainty* makes problems harder. *Scenarios* are basic ingredients of optimization problems under uncertainty. Often they are specified only implicitly, e.g. as ranges over parameter values leading to robust optimization. Scenarios also appear as mass points of discrete probability distributions, or as samples for approximating continuous distributions (see Kleywegt et al. [4]). Using machine learning scenarios that occur more or less frequently can be predicted. Anyway, if in all these varieties scenarios are specified completely and individually, then the input size increases significantly, and therefore the computational complexity of the problems may not increase, like is the case with stochastic linear optimization (see Dyer and Stougie [2]).

In this lecture we explore how introduction of fully specified scenarios may alter the complexity of problems. We do so at the hand of studying basic scheduling problems. For a very recent survey on scheduling under scenarios, we refer to Shabtay and Gilenson [5]. We study a natural version that received relatively little attention so far (see [5]). We are given a set J of jobs, each with its processing time, a set of parallel identical machines and a set of K scenarios, where each scenario is specified as a subset of jobs in J that must be executed if that scenario occurs. The goal is to find an assignment of jobs to the machines that is the same for all scenarios, i.e., if a job does not occur in a scenario it is simply skipped. We consider the classical problems of minimizing the makespan in Feuerstein et al. [3] and minimizing the sum of the jobs' completion times in Bosman et al. [1]. For each we consider the objectives of minimizing the maximum over all scenarios (robust version) and minimizing the average of makespans of all scenarios (stochastic version).

We show that the presence of scenarios may increase the complexity of the problems significantly. As a dramatic example, I mention the ordinary,

*Speaker, e-mail: Leen.Stougie@cwi.nl

single scenario, version of the makespan problem with all processing times equal to 1, which is a trivial problem. With scenarios the robust version of the problem with only two machines becomes inapproximable in polynomial time within ratio 2 unless $\mathcal{P} = \mathcal{NP}$. This is even more surprising once one realizes that putting all jobs on one machine already yields a 2-approximation.

Similar results hold for the sum of completion times problem. As for the makespan problem, the leap in complexity requires that the number of scenarios is part of the input. However, I will give an example of a scheduling problem minimizing total completion time that is in $\mathcal{P}$ in its ordinary version and becomes $\mathcal{NP}$-hard already for 3 scenarios. These results mostly show the mysterious role that scenarios play in the complexity of combinatorial optimization problems.

This lecture is based on work in collaboration with Thomas Bosman, Martijn van Ee (Marine Academy), Esteban Feuerstein (UBuenos Aires), Csanad Imreh, Alberto Marchetti-Spaccamela (La Sapienza Rome), Frans Schalekamp (Cornell), Rene Sitters (VU/CWI), Martin Skutella (TU Berlin), Suzanne van der Ster (Ahold) and Anke van Zuylen (Cornell).

References

[1] T. Bosman, M. van Ee, E. Ergen, C. Imreh, A. Marchetti-Spaccamela, M. Skutella, L. Stougie, Minimizing the sum of completion times over scenarios, forthcoming.

[2] M. E. Dyer, L. Stougie, Computational complexity of stochastic programming problems, *Mathematical Programming, Series A*, **106** (2006), 423–432, doi:10.1007/s10107-005-0597-0.

[3] E. Feuerstein, A. Marchetti-Spaccamela, F. Schalekamp, R. Sitters, S. van der Ster, L. Stougie, A. van Zuylen, Minimizing worst-case and average-case makespan over scenarios, *Journal of Scheduling*, **20** (2017), 545–555, doi:10.1007/s10951-016-0484-y.

[4] A. J. Kleywegt, A. Shapiro, T. Homem-de-Mello, The sample average approximation method for stochastic discrete optimization, *SIAM Journal on Optimization*, **12** (2002), 479–502, doi:10.1137/S1052623499363220.

[5] D. Shabtay, M. Gilenson, A state-of-the-art survey on multi-scenario scheduling, *European Journal of the Operational Research*, **310** (2023), 3–23, doi:10.1016/j.ejor.2022.11.014.

Extended abstracts

DOI: 10.14708/isbn.978-83-962157-1-0p25-30
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

Scheduling data gathering with variable communication speed and startup times

Joanna Berlińska*
Adam Mickiewicz University, Poznań, Poland

Keywords: scheduling, data gathering networks, variable communication speed, startup times

1 Introduction

A *data gathering network* is a computer system comprising a set of worker nodes and a base station. The workers obtain some datasets and pass them to the base station for analysis, merging and storing. An example data gathering network is a distributed computer system, where the results of computations obtained on different machines have to be gathered on a single server. Data gathering wireless sensor networks find a wide range of applications in military systems, environment monitoring and healthcare.

Early works on scheduling in data gathering networks (Choi and Robertazzi [7], Moges and Robertazzi [12]) considered problems where only the total size of data that should be gathered was given. The goal was to compute the amounts of data that individual worker nodes should collect, and to organize the communication in the network, so as to minimize the time of data gathering. Later research concentrated on the case where the amounts of data collected by the worker nodes were fixed. Scheduling algorithms were proposed for various types of data gathering networks, including systems with limited base station memory (Berlińska [2]), dataset due dates (Berlińska [4]), local computations (Berlińska and Przybylski [6]) and data compression (Berlińska [1], Li and Luo [9], Luo et al. [11, 10]).

All the aforementioned articles share the assumption that the parameters of a network are constant. However, in real networks, the communication rate may change with time, due to maintenance activities or sharing the links with other users and applications. Data gathering networks with such a variable communication speed were studied by Berlińska [3] and Berlińska and Mor [5]. In [3], the analyzed problem was to transfer the data from the workers to the base station in the shortest possible time.

*Speaker, e-mail: Joanna.Berlinska@amu.edu.pl

It was shown that if communication preemptions are allowed, the problem can be solved in polynomial time, but in the non-preemptive scenario, it becomes strongly $\mathcal{NP}$-hard. In [5], preemptive scheduling was studied, but the time required by the base station to process the obtained data was taken into account in addition to the communication time. This resulted, again, in a strongly $\mathcal{NP}$-hard problem.

In this work, we go back to analyzing networks where the data processing time is negligible. However, we use a more realistic communication model, where preemptions are possible, but sending a message from a given worker to the base station requires a constant startup time. Thus, each preemption incurs an additional time cost.

2 Problem formulation

The analyzed data gathering network consists of m worker nodes $P_1, \ldots, P_m$ and a base station P_0. Each worker P_i holds dataset D_i of size α_i, which has to be sent to the base station. At most one node can communicate with the base station at a time. The communication speed of node P_i changes depending on the corresponding link being used by other applications. A link will be called *loaded* if it is used by background communications at a given time, and *free* in the opposite case. When the link between P_i and P_0 is free, its speed is equal to $\frac{1}{c_i}$, for $i = 1, \ldots, m$. A loaded link becomes δ times slower, for some fixed rational $\delta > 1$. Moreover, starting a communication between P_i and P_0 requires a constant startup time s. Thus, sending one message containing data of size α (where $0 < \alpha \leq \alpha_i$) between P_i and P_0 takes time $s + c_i\alpha$ when the corresponding link is free, and $s + \delta c_i\alpha$ when it is loaded.

The communication speed changes of the link between P_i and P_0 are described by a set of n_i disjoint time intervals $[t'_{i,j}, t''_{i,j})$ (where $j = 1, \ldots, n_i$, $t''_{i,j} < t'_{i,j+1}$ for $j < n_i$) in which the link is loaded. The total number of loaded intervals in the whole network is $n_1 + \cdots + n_m = n$.

Our goal is to minimize the total time T needed to pass all the data to the base station, also called the makespan.

3 Results

The following result is obtained by a pseudopolynomial reduction from the strongly $\mathcal{NP}$-complete 3-PARTITION problem (Garey and Johnson [8]).

Proposition 1. *The problem of makespan minimization in data gathering networks with variable communication speed and non-zero startup times is strongly $\mathcal{NP}$-hard.*

The first step towards constructing algorithms for solving our problem could be identifying the potential preemption points. Natural candidates are the moments when the communication speed of some link changes or when a link speed change is expected after s units of time. However, the following proposition shows that investigating such moments only is not sufficient.

Proposition 2. *Constructing an optimum schedule for the analyzed problem may require preempting a dataset transfer at a time t such that $t \neq \tau - ks$ for any integer k and any τ which is a moment when some link speed changes.*

Sketch of proof. Let $m = 3$, $s = 2$, $\delta = 4$, $c_1 = c_2 = c_3 = 1$, $\alpha_1 = \alpha_2 = 3$, $\alpha_3 = 2$. Suppose the first link is never loaded, the second link is loaded in the intervals $[0, 3)$ and $[9, \infty)$, and the third link is loaded in the intervals $[0, 11)$ and $[13, \infty)$. A schedule of length 16 can be built by sending dataset D_1 in the intervals $[0, 4)$ and $[13, 16)$, dataset D_2 in the interval $[4, 9)$, and dataset D_3 in the interval $[9, 13)$. In this schedule, a preemption is made at time 4, while all numbers of the form $\tau - ks$ are odd integers. We prove that no schedule of length not exceeding 16 can be built if there is no preemption at time 4. □

The above result shows that constructing an exact algorithm for our problem is a challenge. Moreover, communication speed changes are often unpredictable in practice. Therefore, we propose the following heuristic scheduling rules that can be used as online algorithms.

- OnRate – a preemption is made if a loaded link is currently used, and another link that is available (i.e., such that the corresponding dataset is not yet fully transferred) becomes free. If there are several free links available, the dataset with the smallest size remaining to be sent and a free link is chosen for transfer. Similarly, if only loaded links can be used, the dataset with the smallest remaining size is selected.

- OnRateB – this algorithm works like OnRate, with the exception that if several free links or only loaded links are available, the dataset with the largest remaining size is chosen.

- OnRateS – this algorithm works similarly to OnRate, but a preemption is only made if a loaded link is currently used, another link becomes free, and the time remaining to transfer the current dataset over the loaded link is larger than s. This behavior is meant to avoid introducing preemptions (and in consequence, additional startup times) when only a short time is required to finish the current communication.

- OnRateBS – this algorithm works like OnRateS, with the exception that if several free links or only loaded links are available, the dataset with the largest remaining size is chosen.

- OnRateN – no preemptions are made. When a communication finishes and some free links are available, the dataset with the smallest remaining size and a free link is selected. If only loaded links can be used, the dataset with the smallest remaining size is chosen.

- OnRateBN – this algorithm works like OnRateN, with the exception that if several free links or only loaded links are available, the dataset with the largest remaining size is selected.

We tested our algorithms on instances with $\delta = 2$, $s \in [0, 2]$, $c_i = 1$, $m \in [10, 50]$, and dataset sizes chosen randomly from the interval $[1, 20]$. Communication speed changes were generated as in [3, 5]. Namely, we used parameters $F, L \in \{1, 5\}$ to control the lengths of the free and loaded intervals of all links. The length of the first free interval of link i was selected randomly from the range $[0, \frac{F}{m} \sum_{i=1}^{m} c_i \alpha_i]$. Then, the length of the first loaded interval was chosen randomly from the range $[0, \frac{L}{m} \sum_{i=1}^{m} c_i \alpha_i]$. The lengths of further free and loaded intervals were similarly selected until reaching a certain time horizon.

We compared the makespans obtained by our algorithms with the lower bound LB defined as follows. First, we computed the minimum time LB_1 required for sending all datasets in the case when $s = 0$, using the algorithm proposed in [3]. Then, we set $LB_2 = \max_{i=1}^{m}\{T(i)\}$, where $T(i)$ is the time needed to transfer dataset D_i in one message, starting at time 0. Furthemore, we computed $LB_3 = mS + \sum_{i=1}^{m} c_i \alpha_i$, which is the time required to send all datasets under the assumption that all the links are always free. Finally, we set $LB = \max\{LB_1, LB_2, LB_3\}$.

The results of our computational experiments lead us to the following conclusions.

- All the algorithms obtain better results for large values of m than for the small ones.

- When s is small or when $F = L = 5$ (which means that the communication speeds change rarely), the best results are obtained by OnRateB and OnRateBS. Thus, sending larger datasets first seems to be beneficial in these cases.

- When s is large and $F = 1$, $L = 5$ or $F = 5$, $L = 1$, the best results are obtained by algorithms OnRateS and OnRateBS, which deliver similar results. Hence, avoiding too many startup times is in this case more important than prioritizing large or small datasets.

- The non-preemptive algorithms strongly outperform the preemptive ones when s is very large and $F = L = 1$, which means that the communication speeds change very often. Algorithm OnRateBN is usually a little better than OnRateN.

4 Future research

Among the proposed algorithms, OnRateS and OnRateBS aim at balancing the profits gained by preempting communications using loaded links and the costs of additional startup times. However, these two heuristics perform much more similar to OnRate and OnRateB than to OnRateN and OnRateBN, even when s is large. Therefore, in the future, we want to construct new algorithms that will handle large startup times better than OnRateS and OnRateBS, and deliver high quality schedules for a wide range of s values.

References

[1] J. Berlińska, Scheduling for data gathering networks with data compression, *European Journal of Operational Research*, **246** (2015), 744–749, doi:10.1016/j.ejor.2015.05.026.

[2] J. Berlińska, Heuristics for scheduling data gathering with limited base station memory, *Annals of Operations Research*, **285** (2020), 149–159, doi:10.1007/s10479-019-03185-3.

[3] J. Berlińska, Scheduling in data gathering networks with background communications, *Journal of Scheduling*, **23** (2020), 681–691, doi:10.1007/s10951-020-00648-5.

[4] J. Berlińska, A comparison of priority rules for minimizing the maximum lateness in tree data gathering networks, *Engineering Optimization*, **54** (2022), 218–231, doi:10.1080/0305215X.2020.1861263.

[5] J. Berlińska, B. Mor, Scheduling data gathering with background communications and a processing stage, *Computers & Industrial Engineering*, **171** (2022), 108467, doi:10.1016/j.cie.2022.108467.

[6] J. Berlińska, B. Przybylski, Scheduling for gathering multitype data with local computations, *European Journal of Operational Research*, **294** (2021), 453–459, doi:10.1016/j.ejor.2021.01.043.

[7] K. Choi, T. G. Robertazzi, Divisible load scheduling in wireless sensor networks with information utility, *IEEE International Performance Computing and Communications Conference 2008: IPCCC 2008*, pp. 9–17, doi:10.1109/PCCC.2008.4745126.

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

[9] C. Li, W. Luo, Exact and approximation algorithms for minimizing energy in wireless sensor data gathering network with data compression, *American Journal of Mathematical and Management Sciences*, **41** (2022), 305–315, doi:10.1080/01966324.2021.1960226.

[10] W. Luo, B. Gu, G. Lin, Communication scheduling in data gathering networks of heterogeneous sensors with data compression: Algorithms and empirical experiments, *European Journal of Operational Research*, **271** (2018), 462–473, doi:10.1016/j.ejor.2018.05.047.

[11] W. Luo, Y. Xu, B. Gu, W. Tong, R. Goebel, G. Lin, Algorithms for communication scheduling in data gathering network with data compression, *Algorithmica*, **80** (2018), 3158–3176, doi:10.1007/s00453-017-0373-6.

[12] 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, doi:10.1109/TAES.2006.1603426.

DOI: 10.14708/isbn.978-83-962157-1-0p31-37
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

Scheduling with a limited testing budget[‡]

Christoph Damerius*
Universität Hamburg, Hamburg, Germany

Peter Kling
Universität Hamburg, Hamburg, Germany

Minming Li
City University of Hong Kong, Hong Kong, P. R. China

Chenyang Xu
East China Normal University, Shanghai, P. R. China

Ruilong Zhang
University at Buffalo, Buffalo, USA

Keywords: total completion time, makespan, LP rounding, competitive analysis, approximation algorithms, $\mathcal{NP}$-hardness, PTAS

1 Introduction

With increased interest in applying scheduling algorithms to solve real-life problems, many models and methods have been addressing the *uncertainty* in the scheduling community. Several elegant models that capture uncertainty have been studied in the past two decades, most of which fall under the umbrella of the research on *robust optimization* or *stochastic optimization*. In those settings, the uncertainty is usually described by the input. In robust optimization, the input consists of several scenarios, while the input is sampled from a known distribution in stochastic optimization. In some practical cases, we can gain additional information about the input by paying extra costs, e.g., money, time, energy, or memory. This model is also known as *explorable uncertainty*, which aims to study the trade-offs between the exploration cost and the quality of a solution.

An intriguing scheduling model for explorable uncertainty was proposed by Dürr et al. [5] under the name of *scheduling with testing*. In their model, each job has an upper limit on its processing time that can be decreased

[‡]Cancelled after the beginning of workshop at the request of the tentative speaker.
*Tentative speaker, e-mail: damerius@informatik.uni-hamburg.de

to a (possibly unknown) lower limit by some preliminary action (testing). In their case, a testing operation requires one unit of time. They investigate non-preemptive single-machine schedules to minimize the total completion time or makespan. Dufossé et al. [4] considered a variation, where jobs have processing times p or $p+x$ for a fixed p and x, but are initially hidden. By testing jobs, their actual processing time can be revealed. They developed algorithms for the adaptive and non-adaptive versions of the problem. In the adaptive version, a the algorithm may see the processing time of the jobs tested before, in order to adapt its testing behavior. Later, Albers and Eckl [2, 3] extended Dürr's model to non-uniform testing times under preemptive and non-preemptive versions for multiple machines.

In this paper, we extend their problem to the *budget* setting where, instead of one unit of time, each test consumes a job-specific testing cost and one requires that the total testing cost cannot exceed a given budget. We consider the offline variant, when the lower processing time is known, and the oblivious variant, when the lower processing time is unknown. The objective is to minimize the total completion time or makespan on a single machine. Further, we differentiate between the uniform and non-uniform (job-specific) testing costs.

2 Problem formulation

An instance to *Scheduling with a Limited Testing Budget* (*SLTB*) is a 5-tuple $\mathcal{I} = (J, (p_j^\wedge)_{j\in J}, (p_j^\vee)_{j\in J}, (c_j)_{j\in J}, B)$. $J = [n]$ denotes a *set of n jobs*. We denote by $p_j^\vee \geq 0$ the *lower processing time*, by $p_j^\wedge \geq p_j^\vee$ the *upper limit on the processing time* and $c_j \geq 0$ the *testing cost* of job j, respectively. Additionally, a total amount of *budget* $B \geq 0$ is given.

In the offline versions of the problem, $\mathcal{I}$ is completely known beforehand. In the oblivious version, the lower processing times are only revealed once we have decided which jobs to test, while the remaining information is known a priori. Further, we only consider non-preemptive and, w.l.o.g., busy schedules on a single machine. Let C_j be the completion time for job $j \in J$ for such a schedule. We want to minimize the makespan $\max_{j\in J}\{C_j\}$ or total completion time $\sum_{j\in J} C_j$. The problems are referred to as $SLTB_{TC}$ and $SLTB_M$, respectively.

Each job can be either tested or untested, and use $p_j^\vee$ or $p_j^\wedge$ processing time, respectively. Testing a job will consume c_j units of the budget. Once the set of tested jobs is known, it is trivial to construct an optimal schedule that minimizes one of the above objectives. For makespan minimization, we schedule the jobs in any order. For total completion time minimization, we

use the SPT (shortest processing time first) rule. Hence we can express a feasible schedule as a subset $J_\vee \subseteq J$ of jobs to test, such that $\sum_{j \in J_\vee} c_j \leq B$.

3 Our results

We study the offline and oblivious variants of $\mathrm{SLTB_{TC}}$ and $\mathrm{SLTB_{M}}$.

For $\mathrm{SLTB_{TC}}$ we show that the problem is $\mathcal{NP}$-hard, even when all the lower processing times are 0, by a reduction from the PARTITION problem. Furthermore, we design a PTAS for the general case and an FPTAS for the case of uniform lower processing times. For the oblivious setting, we give a $(4+\epsilon)$-competitive deterministic algorithm. This is almost tight, since we show that no deterministic algorithm is better than 4-competitive.

For $\mathrm{SLTB_{M}}$, we derive our results from a connection between our problem and the 0-1 KNAPSACK problem. We prove that the offline setting is $\mathcal{NP}$-hard and admits an FPTAS. For the oblivious setting, a competitive ratio of $2+\epsilon$ is obtained. We complement this by a lower bound of 2.

3.1 $\mathcal{NP}$-hardness

We apply a reduction from the PARTITION problem. Let $U = \{u_1, \ldots, u_n\}$ be a PARTITION instance and $U_{\text{half}} := \frac{1}{2}\sum_{j=1}^{n} u_j$. We construct a $\mathrm{SLTB_{TC}}$ instance by creating a job pair j_1, j_2 for each $u_j \in U$. We then set the job parameters such that $c_{j_1} - c_{j_2} = u_j$ and $p^{\wedge}_{j_1} - p^{\wedge}_{j_2} = \frac{u_j}{j}$ for each $j \in [n]$.

To control how these jobs are scheduled, we make sure that, when we take jobs from two different pairs (j_1, j_2), (j'_1, j'_2) with $j < j'$, the difference between the processing times of j_i and $j'_{i'}$ $(i, i' \in \{1, 2\})$ is large compared to the u_j. Hence, whenever exactly one job from each pair is tested, the other job will be scheduled in the j'th position in the schedule. (We can ignore the tested jobs since they have zero processing time.) It can be easily shown that the contribution of a job j to the total completion time of a schedule is $i \cdot p_j$, if j is scheduled as the ith-last job in the schedule.

Assume that we test exactly one job from each pair of jobs (j_1, j_2), $j \in [n]$. Then we have two options. By testing a job j_1, we schedule job j_2 into position j in the schedule. We pay an additional cost of u_j, but the contribution to the total completion time is reduced by $(p^{\wedge}_{j_1} - p^{\wedge}_{j_2}) \cdot j = (\frac{u_j}{j}) \cdot j = u_j$. This means that the total completion time is minimized if the budget used for testing is maximized. If we test all cheap jobs $(j_2)_{j \in [n]}$, the total testing cost is $\sum_{j=1}^{n} c_{j_2}$. If we pick all expensive jobs

$(j_1)_{j\in[n]}$, we require a budget of $\sum_{j=1}^{n} c_{j_2} + 2U_{\text{half}}$. By setting the budget to $B = \sum_{j=1}^{n} c_{j_2} + U_{\text{half}}$, we force an optimal solution for SLTB_{TC} to select jobs with total testing cost of exactly B and thus indirectly find a partition.

Lastly, we show that schedules that do not test exactly one job from each pair either violate the budget constraint or their objective value is larger than the objective value of the schedule corresponding to a partition.

3.2 PTAS for offline version

Our results are based on an integer linear programming (ILP) formulation for offline SLTB_{TC}. Essentially, our ILP contains variables $x_{j,i,t}$ that dictate whether job $j \in J$ should be scheduled in position $i \in I$, where t controls whether the job is scheduled as tested or untested job. The ILP is conceptually similar to the classical matching ILP on bipartite graphs (see, e.g., Ahuja et al. [1]), with jobs and positions representing the two disjoint independent sets of the bipartition. A matching would then describe an assignment of jobs to positions. However, there are two main differences. First, we have two variables per pair of job and position (distinguished by t). This translates to each job-position pair having two edges that connect them in the (multi-)graph. Second, the total cost of tested jobs is restricted by the budget B, introducing dependencies when selecting edges.

Our approach combines a rounding scheme of the ILP with an exploitation of the cost structure of the problem. We relax the ILP to an LP by allowing the variables $x_{j,i,t}$ to take on fractional values between 0 and 1. We start with an optimal LP solution and then continue with our rounding scheme, which consists of two phases. In the first phase, we round the solution such that all fractional variables correspond to the edges of a single cycle in the graph. These variables are hard to round directly without overusing the budget. Here we start the second rounding phase. We relax some of the constraints in the LP to be able to continue the rounding process. Specifically, we allow certain positions to schedule two jobs (we call these positions *crowded*). We repeatedly cut up the cycle (that becomes a path after one cut) at carefully chosen positions. After each cut, we can continue the rounding process. This is repeated until we end up with an integral (but infeasible) solution that has some crowded positions. The cut position selection in this rounding phase is specifically tailored to control where crowded positions can appear in the integral solution.

This allows us to bound the increase in the objective in the last step: Here, we move some jobs from crowded positions to different positions to make this solution feasible. To do this, we exploit the cost structure

of SLTB_{TC}. Essentially, when we move jobs to different positions, we guarantee that no job is moved too far from its current position. This way, its contribution to the objective will not increase too much.

3.3 Oblivious SLTB_{TC}

We start by considering the uniform case (all jobs have the same testing cost) to build some intuition. The problem is then equivalent to minimize total completion times by testing at most some k jobs. For the worst-case analysis, we can then assume that each job j tested by our algorithm has $p_j^\vee = p_j^\wedge$. In contrast, we get that $p_j^\vee = 0$ for all all jobs tested by an optimal solution. Thus, from this perspective, regardless of which jobs we test, our total completion time remains unchanged, but the optimum depends on our tested jobs because the adversary can only let the job j that is not tested by our algorithm have $p_j^\vee = 0$.

The problem is then essentially equivalent to the following optimization problem: Given a set of jobs J with all lower processing times being 0, the goal is to test k jobs such that the total completion time of all remaining jobs is minimized. For this much easier problem, it is easy to see that the best strategy is selecting the k jobs with the largest upper processing time.

Our algorithm for the non-uniform case builds upon this understanding of worst-case instances. For a given instance $\mathcal{I}$, it constructs an auxiliary instance $\tilde{\mathcal{I}}$ where all lower processing times are zero. It then solves this instance optimally or approximately, and returns the resulting solution. We then show that $ALG(\mathcal{I}) \leq 2ALG(\tilde{\mathcal{I}}) + 2OPT(\mathcal{I})$. Because we can use the PTAS from the offline model for $\widetilde{\mathcal{I}}$, this then implies a $(4+2\epsilon)$-competitive algorithm (if all jobs have the same testing cost, then $\widetilde{\mathcal{I}}$ can be solved optimally, and we get a 4-competitive algorithm instead).

The lower bound is shown by a hard instance $\mathcal{I} = (J, \mathbf{1}, (p_j^\vee)_{j\in J}, \mathbf{1}, \frac{n}{2})$, where the adversary always lets our tested jobs have lower processing time 1 and the processing time of any other job be 0. Then, any deterministic algorithm has objective value $\frac{n(n+1)}{2}$, while an optimal solution can achieve a total completion time of $\frac{n(n+2)}{8}$, implying our lower bound 4.

3.4 Offline and oblivious SLTB_{M}

The offline SLTB problem under makespan minimization is closely related to the classical 0-1 KNAPSACK problem. This problem aims to select a subset of items such that (i) the total weight of the selected items does not exceed a given capacity; (ii) the total value of the selected items is maximized. To see the connection, consider the testing cost of each job

as the weight of each item and the profit of testing a job $(p_j^\wedge - p_j^\vee)$ as the value of an item. Then we build on the algorithmic idea of the knapsack dynamic programming and design an FPTAS for the offline setting.

We use the same framework as the total completion time minimization model for the oblivious setting and obtain a $(2+\epsilon)$-competitive algorithm. The ratio becomes better here since, for the makespan objective, we have $ALG(\mathcal{I}) \leq ALG(\tilde{\mathcal{I}}) + OPT(\mathcal{I})$, saving a factor of 2. The lower bound proof is also based on the same hard instance $\mathcal{I} = (J, \mathbf{1}, (p_j)_{j\in J}, \mathbf{1}, \frac{n}{2})$. Any deterministic algorithm's makespan is n, while the optimum is $\frac{n}{2}$, giving a lower bound of 2.

4 Future research

Our results open promising avenues for future research. For the setting where we minimize the total completion time, it remains open whether $\mathcal{NP}$-hardness holds for uniform testing cost. Also, while our LP-rounding-based PTAS achieves the best possible approximation, it remains open whether there is a faster, combinatorial algorithm. It would also be natural to consider the multiple machines case.

Another exciting direction of future research is the following *bipartite matching with testing* problem that generalizes our problem, arising from the graph-theoretic perspective: Consider a bipartite graph $G := (L \cup R, E)$ in which each edge $e \in E$ has a cost c_e that can be reduced to $\check{c}_e$ via a testing operation. Given the possibility to test edges before adding them to the matching, we seek a min-cost perfect matching that respects a given testing budget.

References

[1] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, *Network Flows – Theory, Algorithms and Applications*, Prentice-Hall, 1993.

[2] S. Albers, A. Eckl, Explorable uncertainty in scheduling with non-uniform testing times, *CoRR*, abs/2009.13316, 2020; `https://arxiv.org/abs/2009.13316`, arXiv:2009.13316.

[3] S. Albers, A. Eckl, Scheduling with testing on multiple identical parallel machines, *CoRR*, abs/2105.02052, 2021; `https://arxiv.org/abs/2105.02052`, arXiv:2105.02052.

[4] F. Dufossé, Ch. Dürr, N. Nadal, D. Trystram, O. C. Vásquez, Scheduling with a processing time oracle. *CoRR*, abs/2005.03394, 2020; `https://arxiv.org/abs/2005.03394`, arXiv:2005.03394.

[5] Ch. Dürr, T. Erlebach, N. Megow, J. Meißner, An adversarial model for scheduling with testing, *Algorithmica*, **82** (2020), 3630–3675, doi:10.1007/s00453-020-00742-2.

DOI: 10.14708/isbn.978-83-962157-1-0p39-44
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

New results on scheduling time-dependent jobs under precedence constraints

Stanisław Gawiejnowicz*
Adam Mickiewicz University, Poznań, Poland

Keywords: scheduling, single machine, deteriorating jobs, shortening jobs, precedence constraints, algorithms, time complexity

1 Introduction

Scheduling problems with variable job processing times constitute an important part of scheduling theory. In the most popular model of variable processing times, called *time-dependent scheduling*, one assumes that the processing time of each job is described by a function of the starting time of the job. If the function is non-decreasing (non-increasing), we deal with *deteriorating* jobs (*shortening* jobs). Though the first papers on the subject were published about 45 years ago, time-dependent scheduling problems still are studied due to many important applications. We refer the reader to review by Gawiejnowicz [8] for introductory review of this domain, and to monographs by Strusevich and Rustogi [14] and Gawiejnowicz [9] for more detailed discussion of the subject.

Among various groups of time-dependent scheduling problems, those with non-empty precedence constraints seem to be relatively unexplored. In this talk, we summarize the present state of the art in that domain, and discuss some conjectures related to single machine scheduling problems with time-dependent job processing times and arbitrary precedence constraints.

2 Problem formulation

We consider the following general time-dependent scheduling problem. We are given a set of n jobs with variable, time-dependent processing times. The processing time of job J_j is in the form of $p_j = f_j(t)$, where $t \geq 0$ denotes the starting time of J_j, $1 \leq j \leq n$. There are defined precedence constraints on the jobs, described by a directed, acyclic graph $G(V, A)$,

*Speaker, e-mail: stgawiej@amu.edu.pl

where $V = \{1, 2, \ldots, n\}$ and $A \subseteq V \times V$. The aim is to find a schedule which is feasible with respect to the precedence constraints and minimizes a criterion function F. In the three-field notation, the problem can be denoted as $1|p_j = f_j(t), prec|F$. If functions f_j are proportional, proportionally-linear or linear functions of t, we write that job J_j is a proportional, a linearly-proportional or a linear job, respectively.

3 Related research

A few groups of results on scheduling time-dependent jobs under precedence constraints are known. Below we review only the most important cases, for a detailed discussion on time-dependent scheduling under precedence constraints, we refer the reader to [9, Chap. 18].

We begin with results concerning proportional jobs, when $f_j(t) = b_j t$ for $1 \leq j \leq n$, and special forms of precedence constraints such as the weak and strong chains, introduced by Dror et al [5]. Let us recall that a chain of jobs is called a *strong* (a *weak*) *chain*, if between jobs of the chain no job (any number of jobs) from another chain can be inserted.

Wang et al [18] proved that single machine scheduling problem with strong chains, $1|p_j = b_j t, s\text{-}chains|\sum w_j C_j^2$, can be solved in $O(n \log n)$ time by scheduling jobs in non-decreasing order of some ratios. The same authors claimed that the single machine scheduling problem with weak chains, $1|p_j = b_j t, w\text{-}chains|\sum w_j C_j^2$, can be solved in $O(n^2)$ time.

Applying the same approach, Duan et al [6] obtained similar results for problem $1|p_j = b_j t, w\text{-}chains|\sum w_j C_j^k$, where $k \in \mathbb{Z}_+$. Their results were generalized by Wang et al [17] for problems $1|p_j = b_j t, w\text{-}chains|\sum w_j W_j^k$ and $1|p_j = b_j t, s\text{-}chains|\sum w_j W_j^k$, where $W_j := C_j - p_j$ denotes the *waiting time* of the jth job and $k \in \mathbb{Z}_+$.

When job precedence constraints are in the form of a *series-parallel graph* (Valdes et al [15]), Wang et al [16] and Wang et al [17] claimed that problems $1|p_j = b_j t, ser\text{-}par|\sum w_j C_j$ and $1|p_j = b_j t, ser\text{-}par|\sum w_j W_j^k$, respectively, can both be solved in $O(n \log n)$ time.

A separate group of results is related to arbitrary precedence constraints. The most general problem with the precedence constraints is problem $1|p_j = b_j t, prec|f_{\max}$, where one assumes that for job J_j there is defined a non-decreasing cost function f_j that specifies a cost $f_j(C_j)$ that can be computed in a constant time and has to be paid at the completion time of the job, $1 \leq j \leq n$. The problem can be solved by appropriate modification of Lawler's algorithm for problem $1||f_{\max}$ (Lawler [13]): as

far as there are jobs to be scheduled, from jobs without successors we choose the job that will cause the smallest cost in the given position and we schedule it as the last one. Gawiejnowicz [9, Chap. 18] showed how to solve problem $1|p_j = b_j t, prec|f_{\max}$ in $O(n^2)$ time using appropriately modified Lawler's algorithm. Dębczyński and Gawiejnowicz [4, Sect. 3] showed that a single machine scheduling problem with variable job processing times can be solved in polynomial time by Lawler's algorithm, if for the problem two properties hold: *schedule non-idleness*, saying that any feasible schedule does not include idle-times, and *the maximum completion time invariantness*, stating the completion time of the last scheduled job is the same for all feasible schedules. The importance of the two properties was later confirmed by Arigliano et al [1], and Chen and Yuan [2].

The next group of results concerns linear jobs, i.e. when $f_j(t) = a_j + b_j t$ or $f_j(t) = a_j - b_j t$ for $1 \leq j \leq n$. Gawiejnowicz [9, Chap. 18] presented optimal algorithms for linear jobs, $F = C_{\max}$ and precedence constraints in the form of a set of chains, a tree, a forest or a series-parallel graph. The first three cases can be solved in $O(n \log n)$ time, while the latter one can be solved either in $O(n \log n)$ time if *decomposition tree* of the precedence graph is known, or in $O(n^2)$ time otherwise.

The case of $F = \sum C_j$ is more difficult than the case of $F = C_{\max}$, since time complexity of problem $1|p_j = a_j + b_j t| \sum C_j$ is still unknown (Gawiejnowicz [8, Sect. 7.1]). Nevertheless, some special cases are known. Strusevich and Rustogi [14, Chap. 8] proved that problem $1|p_j = a_j + bt, ser\text{-}par| \sum C_j$ is solvable in $O(n \log n)$ time.

Relatively few results are known for shortening jobs. Gao et al [7] applied the approach used for solving problems with linear jobs and $F = C_{\max}$ criterion to problem $1|p_j = b_j(1 - bt), chains| \sum w_j C_j$ and claimed that the latter problem can be solved in $O(n \log n)$ time. Gawiejnowicz et al [12] proved that the case of linearly shortening jobs, $F = C_{\max}$ and precedence constraints in the form of a set of chains or a tree can be solved in $O(n \log n)$ time, while the case of a series-parallel directed acyclic graph can be solved either in $O(n \log n)$ time if decomposition tree of the graph is known, or in $O(n^2)$ time otherwise.

Some authors considered scheduling problems with other forms of variable processing times and non-empty precedence constraints. For example, Dębczyński [3] proved that a few single machine scheduling problems with k-partite precedence constraints and *mixed* variable processing times can be solved in $O(n^2)$ time.

4 Our results

Our contribution is three-fold. First, we show that some of results discussed in Sect. 3 can be obtained in a simpler way. Next, we show that for each feasible schedule for problem $1|p_j = a_j + b_j t|C_{\max}$ there exists a feasible schedule for a single machine scheduling problem with fixed job processing times and another criterion and vice versa. We also show that the values of optimality criteria for these two problems are related. Finally, based on the equivalence and applying some ideas introduced by Gawiejnowicz et al [10, 11], we formulate a conjecture on the time complexity of problem $1|p_j = a_j + b_j t, prec|C_{\max}$ and some other conjectures concerning time complexity of a few other time-dependent scheduling problems.

5 Future research

In future research, it would be interesting to consider special cases of the problem mentioned earlier, e.g. when some jobs have the same processing times or when some additional conditions hold, e.g. job processing time and job weight are agreeable. Another possibility is to consider selected parallel- or dedicated-machine counterparts of single-machine problems with precedence constraints.

References

[1] A. Arigliano, G. Ghiani, A. Grieco, E. Guerriero, Single-machine time-dependent scheduling problems with fixed rate-modifying activities and resumable jobs, *4OR*, **15** (2017), 201–215, doi:10.1007/s10288-016-0333-z.

[2] R-B. Chen, J-J. Yuan, Single-machine scheduling of proportional-linearly deteriorating jobs with positional due indices, *4OR*, **18** (2020), 177–196, doi:10.1007/s10288-019-00410-4.

[3] M. Dębczyński, Maximum cost scheduling of jobs with mixed variable processing times and k-partite precedence constraints, *Optimization Letters*, **8** (2014), 395–400, doi:10.1007/s11590-012-0582-5.

[4] M. Dębczyński, S. Gawiejnowicz, Scheduling jobs with mixed processing times, arbitrary precedence constraints and maximum cost criterion, *Computers & Industrial Engineering*, **64** (2013), 273–279, doi:10.1016/j.cie.2012.10.010.

[5] M. Dror, W. Kubiak, P. Dell'Olmo, Scheduling chains to minimize mean flow time, *Information Processing Letters*, **61** (1997), 297–301, doi:10.1016/S0020-0190(97)00037-9.

[6] H-L. Duan, W. Wang, Y-B. Wu, Scheduling deteriorating jobs with chain constraints and a power function of job completion times, *Journal of Industrial and Production Engineering*, **31** (2014), 128–133, doi:10.1080/21681015.2014.908501.

[7] W-J. Gao, X. Huang, J-B. Wang, Single-machine scheduling with precedence constraints and decreasing start-time dependent processing times, *International Journal of Advanced Manufacturing Technology*, **46** (2010), 291–299, doi:10.1007/s00170-009-2089-5.

[8] S. Gawiejnowicz, A review of four decades of time-dependent scheduling: main results, new topics, and open problems, *Journal of Scheduling*, **23** (2020), 3–47, doi:10.1007/s10951-019-00630-w.

[9] S. Gawiejnowicz, *Models and Algorithms of Time-Dependent Scheduling*, Springer, Berlin-Heidelberg, 2020, doi:10.1007/978-3-662-59362-2.

[10] S. Gawiejnowicz, W. Kurc, L. Pankowska, Equivalent time-dependent scheduling problems, *European Journal of Operational Research*, **196** (2009), 919–929, doi:10.1016/j.ejor.2008.04.040.

[11] S. Gawiejnowicz, W. Kurc, L. Pankowska, Conjugate time-dependent scheduling problems, *Journal of Scheduling*, **12** (2009), 543–553, doi:10.1007/s10951-009-0121-0.

[12] S. Gawiejnowicz, T-C. Lai, M-H. Chiang, Scheduling linearly shortening jobs under precedence constraints, *Applied Mathematical Modelling*, **35** (2011), 2005–2015, doi:10.1016/j.apm.2010.11.012.

[13] E. L. Lawler, Optimal sequencing of a single machine subject to precedence constraints, *Management Science*, **19** (1973), 544–546, doi:10.1287/mnsc.19.5.544.

[14] V. A. Strusevich, K. Rustogi, *Scheduling with Time-Changing Effects and Rate-Modifying Activities*, Springer, Cham, 2017, doi:10.1007/978-3-319-39574-6.

[15] J. Valdes, R. E. Tarjan, E. L. Lawler, The recognition of series-parallel digraphs, *SIAM Journal on Computing*, **11** (1982), 289–313, doi:10.1137/0211023.

[16] J-B. Wang, C-T. Ng, T-C. E. Cheng, Single-machine scheduling with deteriorating jobs under a series-parallel graph constraint, *Computers & Operations Research*, **35** (2008), 2684–2693, doi:10.1016/j.cor.2006.12.026.

[17] J-B. Wang, J-J. Wang, Single-machine scheduling problems with precedence constraints and simple-linear deterioration, *Applied Mathematical Modelling*, **39** (2015), 1172–1182, doi:10.1016/j.apm.2014.07.028.

[18] J-B. Wang, J-J. Wang, P. Ji, Scheduling jobs with chain precedence constraints and deteriorating jobs, *Journal of the Operations Research Society*, **62** (2011), 1765–1770, doi:10.1057/jors.2010.120.

DOI: 10.14708/isbn.978-83-962157-1-0p45-51
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

An FPTAS and an SFPTAS for maximizing the weighted number of just-in-time jobs in a proportionate flow shop system

Stanisław Gawiejnowicz*
Adam Mickiewicz University, Poznań, Poland

Nir Halman
Bar-Ilan University, Ramat Gan, Israel

Keywords: scheduling, proportionate flow shop, just-in-time jobs, FPTAS, SFPTAS, monotone dynamic programming, pseudo-polynomial time algorithms

1 Introduction

We consider the following scheduling problem. A set J of n jobs has to be processed on m machines of a *proportionate flow shop* system. All jobs are available at time 0 and job preemption is not allowed. Job processing times are machine-independent, i.e. irrespectively on which machine job $J_j \in J$ is processed, it is defined by a processing time p_j, a due date d_j and a reward w_j, where $1 \leq j \leq n$. We assume that all these parameters are given positive integers.

For a given schedule, the completion time of job J_j on machine i and its completion time on the last machine are denoted as C_{ij} and $C_j = C_{mj}$, respectively, where $1 \leq j \leq n$ and $1 \leq i \leq m$. If job J_j is completed exactly at its due date, i.e., $C_j = d_j$, then a reward w_j is collected and job J_j is called a *just-in-time* (*JIT*) *job*.

Let $E \cup T = J$, $E \cap T = \emptyset$, be a partition of the jobs into the set E of jobs completed just in time and the set T of jobs completed before or after their deadlines in a given schedule. The goal is to find a schedule with a maximal weighted number of jobs in the set E, i.e.,

$$\max \sum_{J_j \in E} w_j.$$

*Speaker, e-mail: stgawiej@amu.edu.pl

This problem, in the three-field notation denoted as $PFm||\sum w_j$, is a generalization of the single machine scheduling problem $1||\sum w_j$, considered for the first time by Lann and Mosheiov [8], who proved that the single machine problem can be solved in $O(n^2)$ time. For the generalized problem, $PFm||\sum w_j$, we present a fully polynomial-time approximation scheme (an FPTAS) and a strongly-polynomial FPTAS (an SFPTAS) for maximizing the weighted number of just-in-time jobs, faster by a factor of n, compared to the state-of-the-art FPTAS known in the literature.

2 Related research

Choi and Yoon [3], applying a reduction from the PARTITION problem, proved that two-machine flow shop problem $F2||\sum w_j$ is weakly $\mathcal{NP}$-hard. For this problem, Shabtay and Bensoussan [11] introduced a pseudo-polynomial time algorithm, running in $O(n^2\sum_{j=1}^{n} p_j)$ time. The same authors, based on their pseudo-polynomial time algorithm, proposed for problem $PF2||\sum w_j$ an FPTAS running in $O(\frac{n^4}{\varepsilon}\log(\frac{n}{\varepsilon}))$ time.

Shabtay [10, Thm. 2], applying a reduction from the PARTITION problem, showed that proportionate flow shop problem $PFm||\sum w_j$ is $\mathcal{NP}$-hard already for $m = 2$ machines, gave an $O(n^3\sum_{j=1}^{n} p_j)$ pseudo-polynomial time algorithm [10, Thm. 3] and designed a tailor-made strongly-polynomial FPTAS for problem $PF2||\sum w_j$ with running time of $O(\frac{n^4}{\varepsilon})$ [10, Thm. 4]. Elalouf et al [4] proposed a faster FPTAS for problem $PF2||\sum w_j$ running in $O(\frac{n^3}{\varepsilon})$ time. A bicriterion variant of problem $PF2||\sum w_j$ was considered by Shabtay et al [12], while a distributive version of problem $PFm||\sum w_j$ was addressed by Shabtay [9]. Gerstl et al [6] gave an alternate DP formulation for problem $PF2||\sum w_j$ that runs faster than the one of Shabtay [11] by a factor of n. However, the authors left open the natural question of whether this improved DP formulation can lead to an improved FPTAS for this problem or not. We answer this question in the affirmative.

3 Our results

We formulate problem $PFm||\sum w_j$ as a monotone dynamic program (DP) that falls into the FPTAS frameworks of Alon and Halman [1, 2]. This gives rise to two different approximation schemes for problem $PFm||\sum w_j$, an FPTAS and an SFPTAS, that run in time cubic on n and linear in $\frac{1}{\varepsilon}$, up to log terms, i.e., faster by a factor of n, up to log terms, than the tailor-made FPTAS of Shabtay [10]. Moreover, applying the above

frameworks makes superflous the statement of the algorithm, as well as its running time and approximation error analyses.

3.1 Structural properties

Gerstl et al [6] proved the following three properties of problem $PFm||\sum w_j$.

Property 1. *There exists an optimal permutation schedule for problem* $PFm||\sum w_j$.

Property 2. *There exists an optimal schedule for problem* $PFm||\sum w_j$ *such that the JIT jobs are scheduled according to an EDD* (*Earliest Due-Date*) *order.*

Property 3. *There exists an optimal schedule for problem* $PFm||\sum w_j$ *such that the jobs are scheduled on the first machine with no idle time between consecutive jobs.*

Now, we overview the improved algorithm by Gerstl et al [6]. Based on Property 2, they first assumed that the jobs are sorted according to the EDD order. Next, they designed a DP algorithm with a 3-dimensional state variable as follows.

Let $f(j,k,C_{1k})$ be the maximum weighted number of JIT jobs that can be obtained by assigning the remaining $n-j$ jobs $J_{j+1},\ldots,J_n$, given that the last JIT job is J_k, $k \le j$, and that its completion time on machine 1 is C_{1k}. Then, since J_k is a JIT job, $C_k = d_k$. The decision made for each state variable is whether to schedule the next job, J_{j+1}, as a JIT job if it is possible, or to make it a tardy job. The DP recursion reads as follows:

$$f(j,k,C_{1k}) = \begin{cases} f(j+1,k,C_{1k}), \text{ if } \max\{d_k + p_{j+1}, C_{1k} + mp_{j+1}\} > d_{j+1}, \\ \max\{f(j+1,k,C_{1k}), f(j+1,j+1,C_{1k}+p_{j+1}) + w_{j+1}\}, \text{ otherwise.} \end{cases} \tag{1}$$

Note that the term $\max\{d_k + p_{j+1}, C_{1k} + mp_{j+1}\}$ is the earliest date job J_{j+1} can complete on the last machine: $d_k + p_{j+1}$ is the sum of the processing time of job J_{j+1} and the completion time of the last JIT job on the last machine, while $C_{1k} + mp_{j+1}$ is the time it takes to complete processing job J_{j+1} assuming (i) it starts to be processed on the first machine at time C_{1k} and (ii) all remaining machines are available.

The boundary conditions (when all n jobs are already scheduled) are:

$$f(n,k,C_{1k}) = 0, \qquad C_{1k} = 1,\ldots,\sum_{j=1}^{n} p_j,\ k = 1,\ldots,n. \tag{2}$$

The optimal solution is given by $f(0,0,0)$. Using Property 3, an upper bound on C_{1k} is $\sum_{j=1}^{n} p_j$. This implies that the number of states, and therefore – the running time of DP (1)-(2), is $O(n^2 \sum_{j=1}^{n} p_j)$.

3.2 An FPTAS and an SFPTAS

In this section, we apply the frameworks of Alon and Halman [1, 2] to derive two FPTASes for the problem under consideration. For the ease of presentation, we cite from the concise summary of these papers as given in Apps. A-B in Gawiejnowicz et al [5].

First, applying the framework of Alon and Halman [1] for monotone DPs, we give our FPTAS. To do so, we reformulate the problem based on the DP recurrence (1)-(2), in which we switch from the $f(j,k,C_{1k})$ notation to the one of one-dimensional functions $z_{j,k}(C_{1k}) \equiv f(n-j,k,C_{1k})$, and from backward DP recursion to a forward one. We denote by $z_{j,k}(C_{1k})$ the maximum weighted number of JIT jobs that can be obtained by assigning the remaining j jobs $J_{n-j+1},\ldots,J_n$, given that the last JIT job is J_k, $k \leq n-j$, and that its completion time on machine 1 is C_{1k}. (Since J_k is a JIT job, $C_k = d_k$.) The decision made for each state variable is whether to schedule the next job, J_{n-j+1}, as a JIT job, if possible, or make it a tardy job. The recurrence is:

$$z_{j,k}(C_{1k}) = \begin{cases} z_{j-1,k}(C_{1k}), \text{ if } \max\{d_k + p_{n-j+1}, C_{1k} + mp_{n-j+1}\} > d_{n-j+1}, \\ \max\{z_{j-1,k}(C_{1k}), z_{j-1,n-j+1}(C_{1k} + p_{n-j+1}) + w_{n-j+1}\}, \quad \text{otherwise.} \end{cases} \tag{3}$$

The boundary conditions are:

$$z_{0,k}(\cdot) \equiv 0, \qquad k = 1,\ldots,n. \tag{4}$$

The optimal solution is given by $z_{n,0}(0)$. We note that the functions $z_{j,k}(\cdot)$ are monotone non-increasing, since the later a JIT job J_k completes its processing on the first machine, the less reward we may collect from the remaining jobs $J_{n-j+1},\ldots,J_n$. (A formal proof can be obtained via a simple induction on the recurrence (3)-(4)). The DP formulation (3)-(4) is therefore a *monotone DP*, admitting FPTASes, whenever the DP satisfies some additional technical conditions (see, e.g., Halman et al [7], Alon and Halman [1, 2]). The reason for which we switch from the $f(j,k,C_{1k})$ notation used in DP formulation (1)-(2) to the one of $z_{j,k}(C_{1k})$ of DP recurrence (3)-(4) and to forward DP recursions is that we want to have a clear correspondence to the notation used by Alon and Halman [1, 2].

In order to show that DP formulation (3)-(4) fits into the framework, first we show that it can be seen as a special case of DP formulation (12) given in Gawiejnowicz et al [5].

Next, we set a bound U_z on the ratio between the maximal value of function $z_{j,k}(\cdot)$ and its minimal non-zero value,

$$U_z \leq \frac{\sum_{j=1}^{n} w_j}{\min_{j=1,\ldots,n}\{w_j\}} \leq \sum_{j=1}^{n} w_j,$$

and a bound U_S on the cardinality of the state space,

$$U_S = \sum_{j=1}^{n} p_j.$$

Finally, in the full paper we show that DP formulation (3)-(4) satisfies Conditions A.1-A.4 in Gawiejnowicz et al [5, App. A].

Hence, applying Theorem A.1 in Gawiejnowicz et al [5, App. A] with parameter value set to $\tau_f = O(n^2)$, we get the following result.

Theorem 1. *There exists an FPTAS for maximizing the weighted number of just-in-time jobs on proportionate flow shop problem* $PFm||\sum w_j$ *that runs in time*

$$O\left(\frac{n^3}{\varepsilon}\log\frac{n\log\sum_{j=1}^{n} w_j}{\varepsilon}\log\sum_{j=1}^{n} w_j \log\sum_{j=1}^{n} p_j\right).$$

Note that unlike the approximation scheme designed in Shabtay [10], whose running time depends solely on n and $\frac{1}{\varepsilon}$, and that is therefore an SFPTAS, the running time of the approximation scheme stated in Theorem 1 depends polynomially on the binary encoding of the numbers in the problem instance and thus it is not an SFPTAS.

Now, using the very recent framework of Alon and Halman [2] for the design of SFPTASes for monotone DPs, and using the same DP recurrence (3)-(4), we give an SFPTAS for the considered problem. This SFPTAS framework requires that a monotone DP can be cast as (12) in Gawiejnowcz et al [5] and satisfies Conditions B.1-B.6 as stated by [5, App. B].

Recall from the discussion above that DP formulation (3)-(4) is a special case of DP formulation (12) in Gawiejnowicz et al [5]. We note that Condition B.2 is not satisfied by DP formulation (3)-(4): an upper bound on the value that $z_{j,k}$ can achieve is

$$UB = nw_{\max},$$

where $w_{\max} = \max_{j=1,\ldots,n}\{w_j\}$, and a lower bound on every non-zero feasible solution is $w_{\min} = \min_{j=1,\ldots,n}\{w_j\}$. However, the ratio

$$\frac{nw_{\max}}{w_{\min}}$$

is not guaranteed to be strongly-polynomially bounded. Therefore, we will apply the SFPTAS framework on a relaxed problem. In the relaxed problem, we drop every job J_j with $w_j \leq \frac{\varepsilon}{2n} w_{\max}$. Since

$$LB = w_{\max}$$

is a lower bound on the optimal solution, and we give up on a total profit of no more than $\frac{\varepsilon}{2} LB$ (as there are at most n jobs), the optimal solution of the relaxed problem serves as a $\frac{\varepsilon}{2}$-approximation of the original one. Therefore, an SFPTAS that $\frac{\varepsilon}{3}$-approximates the relaxed problem will serve as an SFPTAS for the original problem.

In the full paper we show that DP (3)-(4) for the relaxed problem satisfies the aforementioned six conditions B.1-B.6. Therefore, applying Theorem B.1 in Gawiejnowicz et al [5], we obtain the following result.

Theorem 2. *There exists an SFPTAS for maximizing the weighted number of just-in-time jobs on proportionate flow shop problem* $PFm||\sum w_j$ *that runs in time*

$$O\left(\frac{n^3}{\varepsilon}\log^2\frac{n\log\frac{n}{\varepsilon}}{\varepsilon}\log\frac{n}{\varepsilon}\right).$$

4 Future research

Future research may concern the existence of an SFPTAS with quadratic running time dependency on n and linear on $\frac{1}{\varepsilon}$, up to log terms. Another interesting topic is to give other example problems for which faster pseudo-polynomial time algorithms give rise to faster FPTASes.

References

[1] T. Alon, N. Halman, Automatic generation of FPTASes for stochastic monotone dynamic programs made easier, *SIAM Journal on Discrete Mathematics*, **35** (2021), 2679–2722, doi:10.1137/19M1308633.

[2] T. Alon, N. Halman, Strongly polynomial FPTASes for monotone dynamic programs, *Algorithmica*, **84** (2022), 1–35, doi:10.1007/s00453-022-00954-8.

[3] B-C. Choi, S-H. Yoon, Maximizing the weighted number of just-in-time jobs in flow shop scheduling, *Journal of Scheduling*, **10** (2007), 237–243, doi:10.1007/s10951-007-0030-z.

[4] A. Elalouf, E. Levner, H. Tang, An improved FPTAS for maximizing the weighted number of just-in-time jobs in a two-machine flow shop problem, *Journal of Scheduling*, **16** (2013), 429–435, doi:10.1007/s10951-013-0320-6.

[5] S. Gawiejnowicz, N. Halman, H. Kellerer, Knapsack problems with position-dependent item weights or profits, *Annals of Operations Research*, **2023**, doi:10.1007/s10479-023-05265-x.

[6] E. Gerstl, B. Mor, G. Mosheiov, A note: Maximizing the weighted number of just-in-time jobs in a proportionate flow shop, *Information Processing Letters*, **115** (2015), 159–162, doi:https://doi.org/10.1016/j.ipl.2014.09.004.

[7] N. Halman, D. Klabjan, C-L. Li, J. Orlin, D. Simchi-Levi, Fully polynomial time approximation schemes for stochastic dynamic programs, *SIAM Journal on Discrete Mathematics*, **28** (2014), 1725–1796, doi:10.1137/130925153.

[8] A. Lann, G. Mosheiov, Single machine scheduling to minimize the number of early/tardy jobs, *Computers & Operations Research*, **23** (1996), 964–974, doi:10.1016/0305-0548(95)00078-X.

[9] D. Shabtay, Maximizing the weighted number of just-in-time jobs in a distributed flow-shop scheduling system, *Naval Research Logistics*, 2023, doi:10.1002/nav.22097.

[10] D. Shabtay, The just-in-time scheduling problem in a flow-shop scheduling system, *European Journal of Operational Research*, **216** (2012), 521–532, doi:10.1016/j.ejor.2011.07.053.

[11] D. Shabtay, Y. Bensoussan, Maximizing the weighted number of just-in-time jobs in several two-machine systems, *Journal of Scheduling*, **15** (2012), 39–47, doi:10.1007/s10951-010-0204-y.

[12] D. Shabtay, Y. Bensoussan, M. Kaspi, A bicriteria approach to maximize the weighted number of just-in-time jobs and to minimize the total resource consumption cost in a two-machine flow-shop scheduling system, *International Journal of Production Economics*, **136** (2012), 67–74, doi:10.1016/j.ijpe.2011.09.011.

DOI: 10.14708/isbn.978-83-962157-1-0p53-60
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

A faster FPTAS for proportionate flow shop scheduling with step-deteriorating processing times

Nir Halman*
Bar-Ilan University, Ramat Gan, Israel

Keywords: scheduling, proportionate flow shop, step-deteriorating processing times, FPTAS, SFPTAS, monotone dynamic programming, pseudo-polynomial time algorithms

1 Introduction

We consider proportionate flow shop scheduling problems with step-deteriorating processing times that are studied by Shabtay and Mor [6]. There are n independent non-preemptive jobs that are available at time zero to be processed on a set of m machines in a flow shop scheduling system. In such a system, each job in the shop must follow the same route through the machines in increasing order of their indices.

While job processing times are machine independent (and as such the problem is a proportionate flow shop problem), they depend on their start time (Gawiejnowicz [3]). Each job J_j, $j = 1, \ldots, n$ has

(i) *normal processing time* a_j,

(ii) *deterioration penalty* b_j, and

(iii) *deterioration date* δ_j, such that if the job starts to be processed on the first machine on time $t > \delta_j$, then its (time-dependent) *actual* processing time, denoted by $p_j(t)$, is increased by the deterioration penalty to the *late* processing time $l_j := a_j + b_j$.

It is known that there exists an optimal permutation schedule $\sigma = (\sigma(1), \ldots, \sigma(n))$ such that all jobs are processed in the same sequence on each one of the machines. Therefore, the starting time of job J_j on the first machine is $S_{\sigma(j)} = \sum_{i=1}^{j-1} p_{\sigma(i)}(S_{\sigma(i)})$. We assume that the actual

*Speaker, e-mail: halman@biu.ac.il

processing time of job $J_{\sigma(j)}$ on any one of the m machines is the following step function:

$$p_{\sigma(j)} = \begin{cases} a_{\sigma(j)}, & \text{if } S_{\sigma(j)} \leq \delta_{\sigma(j)}, \\ l_{\sigma(j)}, & \text{otherwise.} \end{cases} \tag{1}$$

Whenever the actual processing time of a job j is equal to a_j, the job is called *early* and whenever it is equal to $l_{\sigma(j)}$ it is called *tardy*. Shabtay and Mor [6] study two criterion functions (i) the completion time of the entire schedule, denoted by $C_{\max}(\sigma)$, and (ii) the total machine completion time (a.k.a. the total load), denoted by $TL(\sigma)$. The goal is to determine the schedules σ_C and σ_{TL} that minimize the makespan and total load, respectively, i.e.,

$$\sigma_C = \arg\min_{\sigma} \{C_{\max}(\sigma)\}, \qquad \sigma_{TL} = \arg\min_{\sigma} \{TL(\sigma)\}.$$

Shabtay and Mor [6, Thm. 3] unify the two criterion functions and design for the unified problem a tailor-made strongly-polynomial FPTAS with running time of $O(\frac{n^4}{\epsilon})$. In this paper we reformulate the unified problem as a certain *monotone dynamic program* (DP) that falls into the FPTAS frameworks of Alon and Halman [1, 2], thus giving for it two different FPTASes.

While we use general frameworks, our fastest FPTAS runs in time cubic on n and linear in $\frac{1}{\epsilon}$, up to log terms, i.e., faster by a factor of n, up to log terms, than tailor-made FPTAS of [6]. Moreover, applying existing FPTAS frameworks (as opposed to designing tailor-made FPTASes) makes the statement of the algorithm, as well as its running time and approximation error analyses – all superfluous.

2 Structural properties of optimal schedule

Let

$$\begin{aligned} Z(\sigma) := \quad & \alpha \max_{j\in\{1,\ldots,n\}} \{p_{\sigma(j)}\} + \sum_{j=1}^{n} p_{\sigma(j)} = \\ & \alpha \max\left\{ \max_{J_j\in\mathcal{E}(\sigma)} \{a_j\}, \max_{J_j\in\mathcal{T}(\sigma)} \{a_j + b_j\} \right\} + \sum_{J_j\in\mathcal{T}(\sigma)} b_j + A_\Sigma, \end{aligned} \tag{2}$$

where $\mathcal{E}(\sigma)$ is the set of early jobs in σ, $\mathcal{T}(\sigma)$ is the set of its tardy jobs, $A_\Sigma = \sum_{j=1}^{n} a_j$ and $\alpha \in \mathbb{R}^+$ is a parameter.

Shabtay and Mor [6, Eqs. (10)-(13)] show that both the makespan and total load problems can be expressed as the unified problem with criterion $Z(\sigma)$ defined by Eq. (2) in the following way: when $\alpha = m - 1$, then

problem with this criterion is equivalent to the makespan problem, and it is equivalent to the total load problem, whenever $\alpha = \frac{m(m-1)}{2}$.

Shabtay and Mor [6, Lem. 3-4] further show that there exists an optimal permutation schedule σ for minimizing the criterion defined by Eq. (2) where no tardy job precedes an early job and the early jobs are scheduled in non-decreasing order of $a_j + \delta_j$, i.e., according to the *Extended Earliest Deteriorating Date* (*EEDD*) *rule*. They denote by $\tau = (\mathcal{E}, \mathcal{T})$ a partition of the sets of jobs to early and tardy jobs, and by σ_τ a schedule corresponding to partition τ that satisfies the properties above, i.e., in σ_τ the jobs in $\mathcal{E}$ are scheduled first and according to the EEDD rule and then the jobs in $\mathcal{T}$ in an arbitrary order.

In the rest of this section we assume without loss of generality that the entire set of jobs is numbered according to the EEDD rule. It is easy to check whether $\tau = (\mathcal{E} = \{J_1, \ldots, J_n\}, \mathcal{T} = \emptyset)$ is a feasible (and therefore also optimal) solution: all we need to do is to verify that the following n inequalities

$$\sum_{i=1}^{j-1} a_i \leq \delta_j, \quad j = 1, \ldots n$$

hold. If this is indeed the case, then the optimal schedule is to process the jobs in increasing order of their indices to get the minimal value of

$$Z(\sigma_\tau) = \alpha a_{\max} + A_\Sigma,$$

where $a_{\max} = \max_{j=1,\ldots,n}\{a_j\}$. Therefore, hereafter we assume that in any feasible solution $\mathcal{T} \neq \emptyset$.

Shabtay and Mor [6] next decompose the solution space of the problem into $n+1$ auxiliary subproblems, each of which corresponding to a different value of

$$p_{\max}(\sigma_\tau) := \max\left\{\max_{J_j \in \mathcal{E}(\sigma)}\{a_j\}, \max_{J_j \in \mathcal{T}(\sigma)}\{a_j + b_j\}\right\}$$

in Eq. (2). Note that $p_{\max}(\sigma_\tau) \in \{l_0, l_1, \ldots, l_n\}$, where $l_0 := a_{\max}$. Following this decomposition, in the h-auxiliary problem, $h = 1, \ldots, n$, the goal is to find a job partition which minimizes the criterion in Eq. (2) subject to the condition that job J_h is both tardy and has the maximal processing time among all jobs, making $J_h \in \mathcal{T}$ and $p_{\max}(\sigma_\tau) = l_h$.

In the 0-auxiliary problem, there is at least one tardy job, but $p_{\max}(\sigma_\tau) = l_0$. Denote by $Z_h(\sigma)$ the value of an h-auxiliary problem over schedule σ. Note that by Eq. (2) we get that

$$Z_h(\sigma) := \alpha l_h + \sum_{J_j \in \mathcal{T}(\sigma)} b_j + A_\Sigma \leq (\alpha + n) l_h. \tag{3}$$

Denote by $Z_h^\star$ the optimal solution value of the h-auxiliary problem (setting $Z_h^\star := \infty$ if there is no feasible solution) and by $Z^\star$ the optimal solution value of the problem, we get that $Z^\star = \min_{h=0,1,\ldots,n}\{Z_h^\star\}$. Therefore, having FPTASes to each one of the auxiliary problems yields an FPTAS to our original problem. By designing $O(\frac{n^3}{\epsilon})$ time FPTASes for the auxiliary problems, Shabtay and Mor [6, Thm. 3] get an $O(\frac{n^4}{\epsilon})$ time FPTAS for the original problem.

3 Two FPTASes as monotone dynamic programs

In this section, we formulate the h-auxiliary problem as a certain monotone DP with a forward Bellman recursion. For this to work out, we first renumber the entire set of jobs according to the *reversed* EEDD rule, i.e., in non-increasing order of $a_j + \delta_j$. Monotone DPs are known to admit FPTASes, whenever they satisfy some additional technical conditions (see, e.g., Halman et al. [5], Alon and Halman [1, 2]).

Assume first that $h \geq 1$, i.e., job h has to be scheduled as a tardy job and its actual processing time is maximal among the ones of all jobs. Let $z_j(s)$ be the value of criterion $Z(\sigma)$ in Eq. (2), considering only jobs $1, \ldots, j$ and starting at time s. The goal is to calculate $z_n(0)$. We set the boundary condition to be

$$z_0(s) = \alpha l_h + A_\Sigma, \quad s \geq 0. \tag{4}$$

The Bellman recursion reads

$$z_j(s) = \begin{cases} z_{j-1}(s) + b_j, & j = h, \\ z_{j-1}(s) + b_j, & s > \delta_j \text{ and } l_j \leq l_h, \\ \infty, & s > \delta_j \text{ and } l_j > l_h, \\ z_{j-1}(s + a_j), & s \leq \delta_j \text{ and } l_j > l_h, \\ \min\{z_{j-1}(s) + b_j, z_{j-1}(s + a_j)\}, & s \leq \delta_j \text{ and } l_j \leq l_h. \end{cases} \tag{5}$$

We next explain the Bellman recursion. Recall that in the h-auxiliary problem (with $h \geq 1$) job h has to be scheduled as a tardy job and its actual processing time is maximal among the ones of all jobs. The first line ensures that the actual processing time of job h is indeed l_h. The second and third lines deal with the case of job $j \neq h$ whose deterioration date has already passed. If its late processing time is not greater than the one of job h, then in the third line we will schedule it as a tardy job. Otherwise, it is not possible to schedule it as either an early or tardy job, so in the fourth line we return the value of ∞ as the value of an infeasible solution. The last two lines deal with the case of a job $j \neq h$, whose deterioration date

has not been reached yet. If its late processing time is greater than the one of job h, then in the fifth line we will schedule it an early job. Otherwise, it is possible to schedule it as either an early or tardy job.

We next assume that $h = 0$, i.e., $p_{\max}(\sigma_\tau) = l_0$ is realized by an early job. Let $j^\star$ be the index of that job. We define the boundary condition $z_0(\cdot)$ as above, and let the Bellman recursion to be as in Eq. (5) with its first line in the form of

$$z_{j-1}(s) + b_j, \qquad j = h,$$

being replaced by the following two lines:

$$\begin{array}{ll} z_{j-1}(s + a_j), & j = j^\star \text{ and } s \leq \delta_{j^\star}, \\ \infty, & j = j^\star \text{ and } s > \delta_{j^\star}. \end{array}$$

In the remaining of this section, we will apply the framework of Alon and Halman [1] to derive an FPTAS for DP (4)-(5). For the ease of presentation, instead of citing directly from papers by Alon and Halman [1, 2], we will cite from the concise summary of these papers as given in Gawiejnowicz et al. [4, Apps. A-B].

In order to show that DP formulation (4)-(5) fits into the framework, first we shall show that DP formulation (4)-(5) can be seen as a special case of DP formulation [4, Eq. (12)] in the following sense:

(i) we set the level index t to be the index j and therefore the number of levels is $T = n$, i.e., the number of jobs to schedule;

(ii) we set the other index i to be fixed to 1, therefore, we get that $m = 1$ and for the ease of notation we will drop in the sequel the second index from the $z_{t,i}$ notation in [4, Eq. (12)];

(iii) we set the state variable $I_{t,i}$ to be s, i.e., the current starting time of the schedule;

(iv) for every level t, we set the additional information $A_{t,1}(s)$ to be the conditions stated in (5), which determine the values of $f_{t,i}$ in each case;

(v) when considering level t in [4, Eq. (12)], instead of using all previously-calculated $\{z_r\}_{r<t}$, we use only z_{t-1};

(vi) we set the boundary functions to be $f_{0,1} \equiv \alpha l_h + A_\Sigma$.

Thus, by (i)-(vi), we conclude that DP formulation (4)-(5) is indeed a special case of DP formulation [4, Eq. (12)].

Next, we shall set a bound U_z on the ratio between the maximal value of function $z_j(\cdot)$ and its minimal non-zero value:

$$U_z \leq \frac{(\alpha+n)l_h}{\alpha l_h + A_\Sigma} \leq \frac{n}{\alpha} \leq n, \tag{6}$$

see (3)-(4), and a bound U_S on the cardinality of the state space:

$$U_S = nl_h.$$

In the full paper, we show that DP formulation (4)-(5) satisfies Conditions A.1-A.4 as stated in [4, App. A]. Hence, applying [4, Thm. A.1] with parameter value set to $\tau_f = O(n)$, to each one of the $n+1$ auxiliary problems, we get the following result.

Theorem 1. *There exists an FPTAS for proportionate flow shop scheduling with step-deteriorating processing times in the form of Eq. (1) that runs in time*

$$O\left(\frac{n^3}{\epsilon}\log\frac{n\log n}{\epsilon}\log n\log\left(n\max_{h=0,\ldots,n}\{l_h\}\right)\right).$$

Note that unlike the FPTAS designed in [6], whose running time depends solely on n and $\frac{1}{\epsilon}$, and that is therefore strongly polynomial in the instance size (SFPTAS), the running time stated in Theorem 1 depends polynomially on the binary encoding of the numbers in the problem instance and is thus non-strongly polynomial.

Now, we describe an SFPTAS using the very recent framework of Alon and Halman [2] for the design of SFPTASes for monotone DPs, and using the same DP recurrence (4)-(5). This SFPTAS framework requires that a monotone DP can be cast as [4, Eq. (12)] and satisfies Conditions B.1-B.6 as stated in [4, App. B].

Recall from the previous discussion that DP formulation (4)-(5) is a special case of DP formulation [4, Eq. (12)]. In the full paper, we show that DP (4)-(5) satisfies the six Conditions B.1-B.6.

By the calculus of sets of change points (Gawiejnowicz et al. [4, Prp.B.1(1, 4, 5)]), the set of change points $\mathcal{C}_{\bar{z}_j}$ (see [4, Def. B.1]) can be calculated in $\tau_j^{\mathcal{C}} = O(|\mathcal{C}_{\bar{z}_j}|)$ time, $|\mathcal{C}_{\bar{z}_j}| = O(|\mathcal{C}_{\tilde{z}_j}|) = O(\frac{n\log U_z}{\epsilon}) = O(\frac{n\log n}{\epsilon})$ for $1 \leq j \leq n$ (see Alon and Halman [2, Prop. 3.7] with parameter K as set in [2, Alg. 4] to $\sqrt[n+1]{1+\epsilon}$), and $|\mathcal{C}_{\bar{z}_0}| = O(1)$, so $\tau_{\mathcal{C}} = O(\frac{n^2\log n}{\epsilon})$. Therefore, Conditions B.5 and B.6(ii) with the parameter setting $a = 2$ hold, so $U_{\mathcal{C}}$, as

defined in [4, App. B] is in $O\left(\frac{n \log n}{\epsilon}\right)$. Therefore, applying [4, Thm. B.1] to each one of the $n+1$ auxiliary problems, we obtain the following result closing the paper.

Theorem 2. *There exists an FPTAS for proportionate flow shop scheduling with step-deteriorating processing times in the form of Eq. (1) that runs in time*

$$O\left(\frac{n^3}{\epsilon} \log^2 \frac{n \log n}{\epsilon} \log n\right).$$

4 Further research

We reformulated two proportionate flow shop scheduling problems with step-deteriorating processing times as a monotone DP that falls into the recent SFPTAS framework of [2], thus giving for it an SFPTAS with running time of $O\left(\frac{n^3}{\epsilon} \log^2 \frac{n \log n}{\epsilon} \log n\right)$. This gives a speedup of the running time of the SFPTAS of [6] by a factor of n, up to log terms. Doing so, we answer in the affirmative an open question raised by Shabtay and Mor [6]. We pose as an open question the existence of an SFPTAS with quadratic running time dependency on n and linear on $\frac{1}{\epsilon}$, up to log terms.

References

[1] T. Alon, N. Halman, Automatic generation of FPTASes for stochastic monotone dynamic programs made easier, *SIAM Journal on Discrete Mathematics*, **35** (2021), 2679–2722, doi:10.1137/19M1308633.

[2] T. Alon, N. Halman, Strongly polynomial FPTASes for monotone dynamic programs, *Algorithmica*, **84** (2022), 2785–2819, doi:10.1007/s00453-022-00954-8.

[3] S. Gawiejnowicz, *Models and Algorithms of Time-Dependent Scheduling*, Springer, Berlin-Heidelberg, 2020, doi:rg/10.1007/978-3-662-59362-2.

[4] S. Gawiejnowicz, N. Halman, H. Kellerer, Knapsack problems with position-dependent item weights or profits, *Annals of Operations Research*, 2023, doi:10.1007/s10479-023-05265-x.

[5] N. Halman, D. Klabjan, C-L. Li, J. Orlin, D. Simchi-Levi, Fully polynomial time approximation schemes for stochastic dynamic programs, *SIAM Journal on Discrete Mathematics*, **28** (2014), 1725–1796, doi:10.1137/130925153.

[6] D. Shabtay, B. Mor, Exact algorithms and approximation schemes for proportionate flow shop scheduling with step-deteriorating processing times, *Journal of Scheduling*, 2022, doi:10.1007/s10951-022-00766-2.

DOI: 10.14708/isbn.978-83-962157-1-0p61-67
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

A scenario-based planning approach for dynamically scheduled patients

Arthur Kramer*
École des Mines de Saint-Étienne, Saint-Étienne, France
CNRS, LIMOS, Aubièrre, France
Henri Fayol Institute, Saint-Étienne, France

Thiago Alves de Queiroz
Federal University of Catalão, Catalão, Brazil

Manuel Iori
University of Modena and Reggio Emilia, Reggio Emilia, Italy

Yong-Hong Kuo
The University of Hong Kong, Hong Kong, P. R. China

Keywords: dynamic scheduling, healthcare, emergency department, scenario-based planning-approach

1 Introduction

Overcrowding is an issue commonly faced by *Emergency Departments* (*ED*) and that has been widely reported in the literature. Increasing the ED capacity by, for example, hiring additional physicians and nurses, is one way to tackle the overcrowding problem. However, this type of solution may not be feasible in practice due to its financial impact. Consequently, managers are interested in investigating other options to improve patient flow to reduce overcrowding. One way could be optimizing processes and properly using the available resources.

A variety of optimization problems is devoted to applications in the healthcare area, including scheduling problems. In our work, we study the *ED patient scheduling problem*. More specifically, we focus on the weighted tardiness minimization by optimizing the assignment of patients to doctors over a time horizon. This problem is characterized as dynamic, because no information about the patients (e.g., arrival times, service times, and urgency levels) is known before their arrival in the ED, and

*Speaker, e-mail: arthur.kramer@emse.fr

stochastic, because some information is uncertain (e.g., service times). Thus, the problem we face is an *online stochastic optimization problem*. Our objective is to propose an algorithm that could help managers to decide which patient to serve each time a doctor is idle. Because many EDs adopt a common practice of prioritizing patients according to their urgency level, the proposed algorithm would be an alternative to this practice.

We present a *Scenario-Based Planning Approach* ($SBPA$) to solve the problem of dynamically scheduling patients to doctors under uncertainties in the context of EDs. For a complete and detailed view of our propositions, we address the reader to our recent paper (Quieroz et al. [6]).

2 Related research

In Di Somma et al. [1], the authors provide a discussion about the overcrowding problem that is commonly faced by EDs worldwide. A comprehensive review of the contributions of operations research to EDs patient flow optimization can be found in the paper by Saghafian et al. [7].

For what concerns dynamic scheduling in EDs, despite its relevance, the number of studies addressing patient scheduling is relatively limited. A general overview of the dynamic scheduling problem is provided by Ouelhadj and Petrovic [5]. The authors emphasized that dynamic scheduling approaches can be classified as

(i) reactive,

(ii) predictive-reactive, and

(iii) robust proactive.

They highlighted predictive-reactive as the most common category in the literature. According to this approach, an initial schedule is built and then adjusted (reoptimized) once new information is available. The main drawback of this strategy is that uncertainties are not considered during the decision process. In turn, *online anticipatory algorithms* (also known as *lookahead algorithms*) consider sample scenarios of possible future realizations in order to minimize the effect of uncertainties.

Many examples of the successful application of scenario-based algorithms to solve optimization problems can be found in the literature for, e.g., the vehicle routing problem (Hvattum et al. [3]), the ship routing and scheduling problems (Tirado et al. [9]), the dynamic dial-a-ride problem (Schilde et al. [8]) and the resource allocation problem (Duma and Aringhieri [2]).

3 Problem formulation

We consider the scheduling problem of assigning patients to doctors in the ED context. In this problem, we are given a set $J = \{1, \ldots, n\}$ of patients to be served by a set $M = \{1, \ldots, m\}$ of doctors within the given time horizon which starts at time zero and ends at time T. We assume that doctors work in parallel and have the same efficiency. Each doctor can visit at most one patient at a time. Each patient $j \in J$ has an arrival time (release date) r_j, an expected service time p_j^a, a priority weight (urgency level) w_j, and a target due time d_j for the beginning of the visit.

When the visit begins after the target due time, then a tardiness $T_j = \max\{0; s_j - d_j\}$, where s_j is the starting time of the visit of patient j, is considered. The expected service time of a patient is estimated upon the patient arrival at the ED and is based on her/his urgency level. However, the realized service time, denoted as p_j^e, is known only after the service is finished and may be different from the expected service time p_j^a.

Under the notation, the *Dynamic Scheduling of Patients with identical parallel Doctors, release dates, and non-deterministic service times* problem ($DSPD$) looks for a schedule of all patients that minimizes the total weighted tardiness ($\sum_{j \in J} w_j T_j$).

In the DSPD problem, the status of the system is impacted by two types of dynamic events that may occur. The first event regards the arrival of patients when there is a doctor that is idle. The second event occurs when a doctor finishes visiting a patient and there is at least one other patient on the waiting list. Thus, each time any of these events occur a decision should be taken.

4 Our results

We propose a *Scenario-Based Planning Approach* ($SBPA$) to solve the DSPD. Differently from reoptimization heuristics, the SBPA considers sample scenarios of the future to make decisions each time a new event occurs. These scenarios consider fictive patients that may arrive in the ED. We recall that in the DSPD two types of events may occur:

(i) a patient arrives at the ED and there is at least one doctor available;

(ii) a doctor finishes visiting a patient and there is at least one patient in the waiting list, i.e., a doctor can start a new visit.

In the SBPA, which can be classified as an event-driven algorithm, when a new event occurs all information about the current state of the system

(current schedule, patients waiting and being visited, doctors available, etc.) is collected. Then, past and in-progress decisions are fixed and decisions about the future are taken/updated. In order to take new decisions, we optimize each scenario and evaluate the solutions from each scenario. So, we decide, with a *consensus function*, which decisions to take.

In the SBPA, each time an event occurs, a smaller deterministic scheduling problem is solved for each scenario considered. In order to solve these deterministic problems, we use a *Variable Neighborhood Search* (VNS). We address the reader to the work by Lan et al. [4] for a recent survey on the application of VNS to healthcare-related problems.

Another important component of the SBPA is the consensus function that is used to calculate the score of the obtained solutions and to identify the most common decisions. Our consensus function calculates, for each solution/scenario, the number of times the pair (j, o) appears in all other solutions/scenarios. The pair (j, o) represents the assignment of a real patient j to a doctor o. Thus, the decisions associated with the solution with the highest score are adopted.

Our proposed SBPA is shown in Algorithm 1. The current schedule x of patients contains the fixed past and in-progress decisions and the new patients that are ready with regard to the current time t. That is, x takes into account the real patients. Then, a set Ω of scenarios is considered, where each scenario $x_\omega \in \Omega$ contains the real patients in x plus additional fictive patients that may arrive in the ED in the next t_{SH} time instants. Each scenario is then solved with the VNS. The score of each solution is calculated by the consensus function and the current solution x is updated with the decisions obtained from the solution x_ω having the highest score.

Algorithm 1 SBPA

Input: Set Ω of scenarios with fictive patients; t_{SH}; t_{wait}; $x \leftarrow \emptyset$
Output: Schedule x with taken decisions
while there is an event **do**
    $t \leftarrow$ time at which the event happens
    $x \leftarrow$ add the newly revealed patients
    for all scenario $\omega \in \Omega$ **do**
        $x_\omega \leftarrow x \cup \{$ fictive patients j of scenario ω with $r_j \in [t, t + t_{SH}]\}$
        Optimize x_ω with the VNS
        Update x_ω by replacing each fictive patient by an idle time t_{wait}
    $x_{best(\omega)} \leftarrow$ solution x_ω with the highest consensus function score
    Update x with the decisions in $x_{best(\omega)}$

The proposed methods have been coded in the C++. The numerical experiments were carried on a computer with Intel Xeon E3-1245v5 3.50 GHz, with 32 GB of RAM, running under Linux Ubuntu 16.04.7 LTS. The SBPA was tested on a set of 186 realistic instances (see [6] for details), derived from a dataset from an emergency department located in Italy.

We performed several experiments to evaluate the proposed SBPA, to assess the impact of the number of doctors and of the presence of *early information* (EI) in the solution quality.

In the first experiment, we varied the number of available doctors in the set 2, 3, 4, 5, 6 and we compared the performance of SBPA with a re-optimization method, named REO-VNS, that do not consider uncertainties during the decision process. In Table 1, we present the obtained results in terms of running time (t_{run}) and weighted tardiness (wT) for each number of doctors considered. The first two lines represent the aggregated results for all 186 instances; the third compares the wT obtained by REO-VNS and SBPA; the remaining lines show the percentage reduction in terms of time and wT, when the number of doctors varies for the SBPA.

Table 1. Evaluating the number of doctors: SBPA vs. REO-VNS

Method	2 doctors		3 doctors		4 doctors		5 doctors		6 doctors	
	t_{run}	wT	t_{run}	wT	t_{run}	wT	t_{run}	wT	t_{run}	wT
REO-VNS	2.33	1191.42	0.76	87.84	0.10	3.17	0.07	1.31	0.07	0.56
SBPA	592.12	1175.55	478.11	81.19	238.41	3.17	235.32	1.23	231.36	0.56
Red. SBPA/REO-VNS	–	1.33	–	7.57	–	0.00	–	6.11	–	0.00
Red. SBPA 2 doctors (%)	–	–	19.25	93.09	59.74	99.73	60.26	99.90	60.93	99.95
Red. SBPA 3 doctors (%)	–	–	–	–	50.13	96.10	50.78	98.49	51.61	99.31
Red. SBPA 4 doctors (%)	–	–	–	–	–	–	1.30	61.20	2.96	82.33
Red. SBPA 5 doctors (%)	–	–	–	–	–	–	–	–	1.68	54.47

From Table 1, we can observe REO-VNS performs much better than SBPA in terms of execution time. This behavior is expected once the SBPA solves one problem for each scenario each time a new event occurs. However, in terms of solution quality, it can be noticed that on average the solutions obtained with SBPA are up to 7.57% better than the ones obtained with REO-VNS. This result is due the fact that SBPA takes advantage of the scenarios to make decisions. It can also be observed that considering three doctors instead of only two would allow reducing the wT from 1175.55 to 81.19.

We also performed some experiments considering that some data is obtained in advance, i.e., the arrival of some patients is known some minutes in advance due to, e.g., a phone call during the ambulance transportation. In our experiments, we consider that 10%, 15%, 20%, or 30% of the patients

of each group to have information revealed t_{call} minutes in advance (see [6] for details). Thus, in Table 2 we report the results obtained with SBPA in the presence of EI.

Table 2. Evaluating the presence of early information

Method	no EI		10% of EI		15% of EI		20% of EI		30% of EI	
	t_{run}	wT	t_{run}	wT	t_{run}	wT	t_{run}	wT	t_{run}	wT
SBPA	478.11	81.19	468.73	81.19	464.63	79.50	461.18	79.50	454.39	77.88
Red. SBPA no EI (%)	–	–	1.96	0.00	2.82	2.08	3.54	2.08	4.96	4.08
Red. SBPA 10% (%)	–	–	–	–	0.87	2.08	1.61	2.08	3.06	4.08
Red. SBPA 15% (%)	–	–	–	–	–	–	0.74	0.00	2.20	2.04
Red. SBPA 20% (%)	–	–	–	–	–	–	–	–	1.47	2.04

We notice that, overall, having EI helps EDs in reducing the total weighted tardiness. The reduction in terms of wT is at most 4.08% when comparing the cases without early information with the case of knowing 30% of the arrivals in advance. The more the percentage of patients having EI, the higher the reduction is. Thus, the possibility of having early information is beneficial to the ED as it allows a reduction in the wT.

5 Future research

As future research directions, we identify the proposal of other solution strategies that better exploit the information from the scenarios, e.g., branch-and-regret heuristics. Another recommendation is to improve the VNS by proposing and testing new neighborhood structures. Concerning the problem itself, the consideration of additional characteristics is worth investigating, e.g., preemption.

Acknowledgements

This research was funded by the Research Grants Council of Hong Kong (grant no. ECS 27200419), the National Council of Technological and Scientific Development of Brazil (CNPq - grants no. 405369/2021-2 and 311185/2020-7), the State of Goiás Research Foundation (FAPEG), and the Institute for Information Sciences and Technologies (INS2I) of the French National Centre for Scientific Research (CNRS).

References

[1] S. Di Somma, L. Paladino, L. Vaughan, I. Lalle, L. Magrini, M. Magnanti, Overcrowding in emergency department: an international issue, *Internal and Emergency Medicine*, **10** (2015), 171–175, doi:10.1007/s11739-014-1154-8.

[2] D. Duma, R. Aringhieri, Real-time resource allocation in the emergency department: a case study, *Omega*, **117** (2023), 102844, doi:10.1016/j.omega.2023.102844.

[3] L. M. Hvattum, A. Løkketangen, G. Laporte, A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems, *Networks*, **49** (2007), 330–340, doi:10.1002/net.20182.

[4] S. Lan, W. Fan, S. Yang, P. M. Pardalos, N. Mladenović, A survey on the applications of variable neighborhood search algorithm in healthcare management, *Annals of Mathematics and Artificial Intelligence*, **89** (2021), 741–775, doi:10.1007/s10472-021-09727-5.

[5] D. Ouelhadj, S. Petrovic, A survey of dynamic scheduling in manufacturing systems, *Journal of Scheduling*, **12** (2009), 417–431, doi:10.1007/s10951-008-0090-8.

[6] T. A. Queiroz, M. Iori, A. Kramer, Y. H. Kuo, Dynamic scheduling of patients in emergency departments, *European Journal of Operational Research*, **310** (2023), 100–116, doi:10.1016/j.ejor.2023.03.004.

[7] S. Saghafian, G. Austin, S. J. Traub, Operations research/ management contributions to emergency department patient flow optimization: review and research prospects, *IIE Transactions on Healthcare Systems Engineering*, **5** (2015), 101–123, doi:10.1080/19488300.2015.1017676.

[8] M. Schilde, K. F. Doerner, R. F. Hartl, Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem, *European Journal of Operational Research*, **238** (2014), 18–30, doi:10.1016/j.ejor.2014.03.005.

[9] G. Tirado, L. M. Hvattum, K. Fagerholt, J. F. Cordeau, Heuristics for dynamic and stochastic routing in industrial shipping, *Computers & Operations Research*, **40** (2013), 253–263, doi:10.1016/j.cor.2012.06.011.

DOI: 10.14708/isbn.978-83-962157-1-0p69-74
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

Single machine lot scheduling with fixed maintenance activity

Baruch Mor*
Ariel University, Ariel, Israel

Gur Mosheiov
The Hebrew University, Jerusalem, Israel

Dana Shapira
Ariel University, Ariel, Israel

Keywords: lot scheduling, single machine, maintenance activities, order rejection, dynamic programming, total weighted completion time

1 Introduction

We study a single machine *lot scheduling* problem. In this setting, the producer receives orders of different sizes, which are processed in lots. The total size of the orders assigned to a specific lot cannot exceed its capacity. Order splitting is permitted. The processing times of the lots are identical. Moreover, we assume that a fixed maintenance activity is performed, and during the maintenance time, no production is feasible. The objective function is minimum total weighted completion times of the orders.

We also study an extension allowing order rejection. Thus, the scheduler has the option to process only a subset of the orders. The other orders are rejected, and the scheduler is penalized accordingly. The objective function remains total weighted completion time, subject to an upper bound on the total permitted rejection cost.

For the above $\mathcal{NP}$-hard problems, we introduce pseudo-polynomial dynamic programming solution algorithms. Large-size instances are shown to be solved efficiently.

2 Related research

In recent years, the topic of lot scheduling has become popular among scheduling researchers. In this class of problems, the producer receives

*Speaker, e-mail: baruchm@g.ariel.ac.il

orders of different sizes processed in lots. The total size of the orders assigned to a specific lot cannot exceed its capacity, and a single order must be processed in at most two consecutive lots. Also, the processing time of the orders assigned to a given lot is clearly that of the lot. There are numerous real-life applications for this setting, and a typical one is a setting in which a number of products need to go through a heating process in a burn-in industrial oven of a given size.

Many classical combinations of scheduling measures and machine settings have been considered in the recently published papers dealing with lot scheduling. The first relevant paper on the setting of lot scheduling studied here is that of Hou et al. [3]. They studied the problem of minimizing total completion time on a single machine, given

(i) identical lots (both in terms of size and processing time) and

(ii) allowing splitting, i.e., a setting in which an order can be split between consecutive lots.

They introduced a polynomial time solution for this case.

Yang et al. [19] focused on the same setting, assuming that splitting is not allowed. They proved that this problem is $\mathcal{NP}$-hard in the strong sense and introduced and tested a binary integer programming model and four heuristics. The heuristic based on the binary integer program with a (reasonable) bound on the running time was proved to provide the best results. Zhang et al. [20] extended the problem to that of minimizing the total weighted completion time and total weighted discounted completion time. Given that order splitting is allowed, they proved that the policy of weighted shortest processing time first (WSPT) is optimal for both cases.

Mor et al. [10] considered the measure of a minimum number of tardy orders (with order splitting), a minimum weighted number of tardy orders (with order splitting), and a minimum number of tardy orders (assuming no splitting). The first problem was shown to be polynomially solvable, the second was proved to be $\mathcal{NP}$-hard, a pseudo-polynomial algorithm was introduced and tested, and the third was proved to be strongly $\mathcal{NP}$-hard, and a heuristic was proposed and tested.

Mor et al. [11] investigated lot scheduling problems with the option of order rejection, assuming an upper bound on the maximal permitted rejection cost. Pseudo-polynomial dynamic programming algorithms were introduced for three $\mathcal{NP}$-hard problems: minimum makespan, minimum total completion time, and minimum total weighted completion time.

Mor [5] studied lot scheduling with position-dependent lot processing times. The scheduling measure was minimum total weighted completion

time and three special cases: minimum makespan, minimum total completion time, and a minimum linear combination of both. All problems were shown to have polynomial time solutions.

Mor and Mosheiov [8] studied lot scheduling with *Just-In-Time* objectives, assuming a common due date and a common due window, which are decision variables. Polynomial-time dynamic programming algorithms were introduced when the scheduling measures consist of order earliness and tardiness and the due-date (-window) cost, and order-splitting is allowed.

Nurit et al. [16] studied lot scheduling on identical parallel machines and provided exact and heuristic algorithms.

In this paper, we return to the problem of minimizing the total weighted completion time. As in most of the above-mentioned models, we assume lots have identical sizes and processing times. Order-splitting is permitted. We extend the setting in two directions. First, we assume that a fixed maintenance activity must be performed. During the maintenance time, no production is feasible. Also, no splitting over the maintenance is allowed, i.e., an order cannot start before the maintenance and be completed after it.

Scheduling a maintenance activity has numerous applications, and many models have been published (assuming different machine settings, scheduling measures, and assuming different assumptions on the timing and the impact of the maintenance); for details, we refer the reader to the comprehensive survey in the book of Strusevich and Rustogi [18].

The second extension refers to the option of order rejection. In this setting, the scheduler can process only a subset of the orders. The other orders are outsourced (or even totally rejected), and the scheduler is penalized accordingly. The objective function remains the total weighted completion time, subject to the very natural assumption of an upper bound on the total permitted rejection cost. We refer the reader to the survey paper of Shabtay et al. [17] for an extensive review of scheduling with job/order rejection. The popularity of this topic among scheduling researchers is reflected in the following list of papers published in the last two years: Mosheiov et al. [15], Mor et al. [9], Mor and Mosheiov [7], Atsmony and Mosheiov [1], Hermelin et al. ([2], Mor and Shabtay [12], Mor and Shapira [13, 14], Koulamas and Kyparisis [4] and Mor [6].

We note that all input data, i.e., the sizes and processing times of the lots, the size of the orders, the order-dependent rejection costs, and the upper bound on the total permitted rejection cost, belong to the set of positive integers.

3 Our results

For both studied problems, based on the fact that even the special case assuming lots of unit size 1 and identical unit weights is $\mathcal{NP}$-hard, we prove that they are $\mathcal{NP}$-hard, introduce for them pseudo-polynomial dynamic programming algorithms, and consequently establish that they are $\mathcal{NP}$-hard in the ordinary sense. We also report results of an extensive numerical study that supports our claim that the suggested DP algorithms are capable of solving large-size instances efficiently.

4 Future research

Modifying the current setting to the case of a flexible maintenance activity (which must be completed before a given upper bound) or to more general machine settings (parallel machines or shops) are challenging topics for future research.

References

[1] M. Atsmony, G. Mosheiov, A greedy heuristic for solving scheduling problems with bounded rejection cost, *Computers & Operations Research*, **144** (2022), 105827, doi:10.1016/j.cor.2022.105827.

[2] D. Hermelin, D. Shabtay, C. Zelig, M. Pinedo, A general scheme for solving a large set of scheduling problems with rejection in FPT time, *Journal of Scheduling*, **25** (2022), 229–255, doi:10.1007/s10951-022-00731-z.

[3] Y-T. Hou, D-L. Yang, W-H. Kuo, Lot scheduling on a single machine, *Information Processing Letters*, **114** (2014), 718–722, doi:10.1016/j.ipl.2014.06.016.

[4] C. Koulamas, G. Kyparisis, Two-stage no-wait proportionate flow shop scheduling with minimal service time variation and optional job rejection, *European Journal of Operational Research*, **305** (2023), 608–616, doi:10.1016/j.ejor.2022.06.025.

[5] B. Mor, Single-machine lot scheduling with variable lot processing times, *Engineering Optimization*, **53** (2023), 321–334, doi:10.1080/0305215X.2020.1722119.

[6] B. Mor, Single machine scheduling problems involving job-dependent step-deterioration dates and job rejection, *Operational Research*, **23** (2023), 10, doi:10.1007/s12351-023-00754-0.

[7] B. Mor, G. Mosheiov, A note: flowshop scheduling with linear deterioration and job-rejection, *4OR*, **19** (2021), 103–111, doi:10.1007/s10288-020-00436-z.

[8] B. Mor, G. Mosheiov, A note on the single machine CON and CONW problems with lot scheduling, *Journal of Combinatorial Optimization*, **42** (2021), 327–338, doi:10.1007/s10878-021-00709-1.

[9] B. Mor, G. Mosheiov, D. Shapira, Flowshop scheduling with learning effect and job rejection, *Journal of Scheduling*, **23** (2020), 631–641, doi:10.1007/s10951-019-00612-y.

[10] B. Mor, G. Mosheiov, D. Shapira, Lot scheduling on a single machine to minimize the (weighted) number of tardy orders, *Information Processing Letters*, **164** (2020), 106009, doi:10.1016/j.ipl.2020.106009.

[11] B. Mor, G. Mosheiov, D. Shapira, Single machine lot scheduling with optional job-rejection, *Journal of Combinatorial Optimization*, **41** (2021), 1–11, doi:10.1007/s10878-020-00651-8.

[12] B. Mor, D. Shabtay, Single-machine scheduling with total late work and job rejection, *Computers & Industrial Engineering*, **169** (2022), 108168, doi:10.1016/j.cie.2022.108168.

[13] B. Mor, D. Shapira, Minsum scheduling with acceptable lead-times and optional job rejection, *Optimization Letters*, **16** (2022), 1073–1091, doi:10.1007/s11590-021-01763-8.

[14] B. Mor, D. Shapira, Single machine scheduling with non-availability interval and optional job rejection, *Journal of Combinatorial Optimization*, **44** (2022), 480–497, doi:10.1007/s10878-022-00845-2.

[15] G. Mosheiov, A. Sarig, V. Strusevich, Minmax scheduling and due-window assignment with position-dependent processing times and job rejection, *4OR*, **18** (2020), 439–456, doi:10.1007/s10288-019-00418-w.

[16] B. Nurit, B. Mor, Y. Schlissel, D. Shapira, Lot scheduling involving completion time problems on identical parallel machines, *Operational Research*, **23** (2023), 12, doi:10.1007/s12351-023-00744-2.

[17] D. Shabtay, N. Gaspar, M. Kaspi, A survey on offline scheduling with rejection, *Journal of Scheduling*, **16** (2013), 3–28, doi:10.1007/s10951-012-0303-z.

[18] V. A. Strusevich, K. Rustogi, *Scheduling with Time-Changing Effects and Rate-Modifying Activities*, Springer, Cham, 2017, doi:10.1007/978-3-319-39574-6.

[19] D-L. Yang, Y-T. Hou, W-H. Kuo, A note on a single-machine lot scheduling problem with indivisible orders, *Computers & Operations Research*, **79** (2017), 34–38, doi:10.1016/j.cor.2016.10.004.

[20] E. Zhang, M. Liu, F. Zheng, Y. Xu, Single machine lot scheduling to minimize the total weighted (discounted) completion time, *Information Processing Letters*, **142** (2019), 46–51, doi:10.1016/j.ipl.2018.10.002.

DOI: 10.14708/isbn.978-83-962157-1-0p75-79
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

Polynomial-time solutions for minimizing total load on unrelated machines with position-dependent processing times and rate-modifying activities

Baruch Mor
Ariel University, Ariel, Israel

Gur Mosheiov*
The Hebrew University, Jerusalem, Israel

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

Keywords: scheduling, , parallel machines, position-dependent processing times, rate-modifying activities, total load

1 Introduction

The scheduling problem we study consists of the following features:

(i) The machine setting: parallel unrelated machines.

(ii) The job processing times are position-dependent in the most general way (no monotonicity or any specific function are assumed).

(iii) An optional rate-modifying (maintenance) activity, which is machine-dependent, may be performed by the scheduler on each machine.

(iv) The objective function is minimum total load (i.e. the sum of the completion times of the last processed jobs on all machines).

This setting combines two popular topics in scheduling research in recent years. First, the assumption of variable processing times (and in particular of position-dependent processing times) was shown to be valid in many real-life applications. We refer the reader to the book of Gawiejnowicz [2] for an extensive survey on this topic. A second, very practical topic, is that of scheduling a rate-modifying activity. This topic, which has been studied

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

by numerous researchers as well, focuses on the option of scheduling a maintenance activity, that improves the performance of the processor and, consequently, the processing times of the following jobs are reduced. An extensive survey of this topic can be found in the book of Strusevich and Rustogi [15].

2 Related research

The objective function we consider, minimum total load, has been rarely studied. This measure, which is equivalent to the sum of the total processing times on all the machines, becomes relevant when the cost of the system is a function of the busy-time of the machines. Some of the references dealing with scheduling problems with total load minimization are: Mosheiov [10] who studied the problem of minimizing total load on parallel identical machines with time-dependent processing times (linear deterioration), Mosheiov [11] who focused the case of position-based deterioration, Yu et al. [16] who solved a more general case in which the job processing times are both position- and machine-dependent, Fiszman and Mosheiov [1] who focused on minimum total load on a proportionate flowshop with position-dependent processing time and job-rejection, Mor et al. [8] who studied the problem on a proportionate flowshop with bounded job rejection, Mor and Mosheiov [7] who studied the problem on parallel identical machines with linear deterioration, and Mor and Mosheiov [9] who solved the problem of minimizing total load on a flowshop with linear deterioration and job rejection.

3 Our results

In the classical setting of parallel identical machines, total load is clearly independent of both the allocation of jobs to the machines and the job-sequences on the machines. The problem becomes harder when the machine setting is that of parallel unrelated. We further extend the setting to that of general position-dependent processing times, and furthermore, we consider the option of performing a rate-modifying maintenance activity on each machine. For a given number of machines, this general setting is shown to be solved in polynomial time in the number of jobs.

We then study a number of related problems. First, we focus on the special case of job-deterioration. The assumption in this case is that the job processing times increase (in the most general way) as a function of the job-

position. Thus, there are no specific functions that reflect the deterioration type. For the numerous models and applications of scheduling with job-deterioration, we refer the reader again to the book of Gawiejnowicz [2]. A much more efficient solution algorithm is introduced in this paper for the setting of general job-deterioration.

An extension of the basic setting studied here (unrelated machines, position-dependent job processing times, optional rate-modifying activity on each machine, minimizing total load) to the case in which optional job-rejection is considered, is studied next. In this setting, the scheduler may process only a subset of the jobs, and the remaining jobs are rejected (e.g., outsourced or totally rejected). The topic of scheduling with job-rejection becomes popular among researchers in recent years, mainly due to its practicality: in many real-life situations the schedule either cannot or does not want to process all the jobs and reject some of them. For a survey on scheduling with job-rejection, we refer the reader to the survey paper of Shabtay et al. [13].

Another extension of the basic setting we study, is the setting where the job processing times are controllable through an allocation of a limited resource; see the survey paper of Shabtay and Steiner [14]. The actual processing time of a job is a function of both the basic workload and the amount of the allocated resource. The most popular model of scheduling with controllable processing times is that of convex resource consumption function, see Monma et al. [4]. We extend the model studied here (unrelated machines, a rate modifying activity and an objective function of minimum total load) to the setting of controllable processing times with position-dependent workloads. Oron [12] studied a similar setting (with no maintenance activities), where the machine setting is parallel identical and the objective function is minimum total completion. Other recent studies focusing on position-dependent workloads are Lu and Liu [3] and Mor [5, 6].

References

[1] S. Fiszman, G. Mosheiov, Minimizing total load on a proportionate flowshop with position-dependent processing times and job-rejection, *Information Processing Letters*, **132** (2018), 39–43, doi:10.1016/j.ipl.2017.12.004.

[2] S. Gawiejnowicz, *Models and Algorithms of Time-Dependent Scheduling*, Springer, Berlin Heidelberg, 2020, doi:10.1007/978-3-662-59362-2.

[3] Y-Y. Lu, J-Y. Liu, A note on resource allocation scheduling with position-dependent workloads, *Engineering Optimization*, **50** (2018), 1810–1827, doi:10.1080/0305215X.2017.1414207.

[4] C. L. Monma, A. Schrijver, M. Todd, V-K. Wei, Convex resource allocation problems on directed acyclic graphs: duality, complexity, special cases, and extensions, *Mathematics of Operations Research*, **15** (1990), 736–748, doi:10.1287/moor.15.4.736.

[5] B. Mor, Single-machine minmax common due-window assignment and scheduling problems with convex resource allocation, *Engineering Optimization*, **51** (2019), 1251–1267, doi:10.1080/0305215X.2018.1519557.

[6] B. Mor, Minmax common flow-allowance problems with convex resource allocation and position-dependent workloads, *Journal of Combinatorial Optimization*, **43** (2022), 79–97, doi:10.1007/s10878-021-00746-w.

[7] B. Mor, G. Mosheiov, Minimizing total load on parallel machines with linear deterioration, *Optimization Letters*, **14** (2020), 771–779, doi:10.1007/s11590-019-01526-6.

[8] B. Mor, G. Mosheiov, D. Shapira, Flowshop scheduling with learning effect and job rejection, *Journal of Scheduling*, **23** (2020), 631–641, doi:10.1007/s10951-019-00612-y.

[9] B. Mor, G. Mosheiov, A note: flowshop scheduling with linear deterioration and job-rejection, *4OR*, **19** (2021), 103–111, doi:10.1007/s10288-020-00436-z.

[10] G. Mosheiov, Multi-machine scheduling with linear deterioration, *Infor*, **36** (1998), 205–214, doi:10.1080/03155986.1998.11732359.

[11] G. Mosheiov, A note: multi-machine scheduling with general position-based deterioration to minimize total load, *International Journal of Production Economics*, **135** (2012), 523–535, doi:10.1016/j.ijpe.2011.09.005.

[12] D. Oron, Scheduling controllable processing time jobs with position-dependent workloads, *International Journal of Production Economics*, **173** (2016), 153–160, doi:10.1016/j.ijpe.2015.12.014.

[13] D. Shabtay, N. Gaspar, M. Kaspi, A survey on offline scheduling with rejection, *Journal of Scheduling*, **16** (2013), 3–28, doi:10.1007/s10951-012-0303-z.

[14] D. Shabtay, G. Steiner, A survey of scheduling with controllable processing times, *Discrete Applied Mathematics*, **155** (2007), 1643–1666, doi:10.1016/j.dam.2007.02.003.

[15] V. A. Strusevich, K. Rustogi, *Scheduling with Time-Changing Effects and Rate-Modifying Activities*, Springer, Cham, 2017, doi:10.1007/978-3-319-39574-6.

[16] X. Yu, Y. Zhang, K. Huang, Multi-machine scheduling with general position-based deterioration to minimize total load revisited, *Information Processing Letters*, **114** (2014), 399–404, doi:10.1016/j.ipl.2014.02.009.

DOI: 10.14708/isbn.978-83-962157-1-0p81-88
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

Approach to integrate the learning and fatigue effect into the flow shop scheduling problem

Yenny A. Paredes-Astudillo*
INSA Lyon, Villeurbanne, France
Universidad de La Sabana, Chía, Colombia

Valérie Botta-Genoulaz
INSA Lyon, Villeurbanne, France

Jairo R. Montoya-Torres
Universidad de La Sabana, Chía, Colombia

Martha Patricia Caro
Pontificia Universidad Javeriana, Bogotá, Colombia

Keywords: scheduling, flow shop, learning effect, fatigue effect

1 Introduction

Scheduling problems have gained attention since the 1950s. The classical theory has worked on the basic hypothesis that job processing time are constant, or that uncertainty can be dealt using *stochastic* or *fuzzy* approaches. These approaches are far from reality, especially when the resources are humans. Therefore, in recent years, the consideration of humans as a flexible and complex resource has gained importance. It is true that production systems have been transformed with the arrival of technology, but humans continue to be the most common resource in several areas because of their cognitive and physical skills that machines cannot yet emulate economically (Sgarbossa et al. [8]).

In the case of manual operations, job processing times may change because of the learning or deterioration effects (Gawiejnowicz [3]). Recently, scheduling problems with *learning effect* have attracted attention, in particular the study of the single machine system (Paredes-Astudillo et al. [7]). However, the flow shop scheduling problem (FSSP) is attractive in industrial contexts, as it is a common configuration in real manufacturing systems such as textile, footwear, automotive, picking process, or luxury

*Speaker, e-mail: yenny.paredes-astudillo@insa-lyon.fr

industries, etc., where the worker's performance has a direct effect on productivity levels. In this type of scenario, it is evident that the way jobs are sequenced can directly reinforce or weaken the learning or deteriorating process (for example, fatigue in humans). Consequently, obtaining a suitable job scheduling becomes relevant.

The FSSP with learning and deterioration effects belongs to the family of scheduling problems with time-changing effects, for which the search for a solution requires a significant computational effort (Agnetis et al. [1], Mosheiov [5]). The problem is known to be $\mathcal{NP}$-hard (Wang and Xia [10]). This has motivated studies focusing mainly on the analysis of the problem complexity and the design of algorithms to resolve it, rather than the integration of human characteristics into the problem. The latter requires interdisciplinary work to develop accurate modelling of human factors, allowing the operational research community to integrate human phenomena into production systems (Calzavara et al. [2], Sgarbossa et al. [8]).

The objective of this work is twofold. On the one hand, we first describe a real problem and how approaching it as a dynamic scheduling problem helps us understand and find ways to solve it. On the other hand, once we identify theoretical approaches for modeling the effect of learning and fatigue, we try to find the right model for the case study and use the human factors tools to derive the parameters. The case study is based on a picking line, where each picking zone represents a workstation, such as in the flow shop configuration (details are given in Sect. 2.1).

2 Methodology

This section presents each phase of the proposed methodology to integrate the learning and fatigue effects into the FSSP.

2.1 Case study

The problem takes place in a picking line of a Colombian dry food company. This line has 14 picking zones linked by a conveyor. All orders have the same routing. This type of picking system involves two main elements: the first is the worker who is in each picking zone, while the second is a container that follows the routing, visiting all picking zones and filling all the SKUs (Stock Keeping Units) that belong to its order. The company has more than 1000 SKUs, and the processing time for each container in each picking zone varies depending on the SKUs needed in the order. For

the purposes of this study, orders shall be referred to as jobs. The company aims to obtain the sequencing of jobs and the breaks regime, balancing economic objectives (i.e., makespan) and social objectives (i.e., maximal difference of accumulated fatigue level among operators).

2.2 FSSP

Formally speaking, the process under study can be modelled as a FSSP where a set of n independent jobs must be processed by a set of m resources (workers), in order to minimize the makespan. Each worker can process one job at a given time, all resources process jobs in the same sequence and preemption of a job is not allowed. Let $\overline{p}_{ij}$ be the normal (baseline) processing time of job j by the ith resource. All workers are available in their workstation at the beginning and have a 100% productivity rate. The scheduling is performed through the jobs permutation sequence.

2.3 Learning model

A total of 12 workers signed an informed consent and participated in this phase (4 women and 8 men). Half of them have internal working experience of less than 6 months, while the others have an experience of more than 6 months. To check if the picking worker is impacted by a learning effect (mainly related to the SKUs position), an experimental test was designed. It was conducted in the training laboratory for the picking operations of the company under study (which has similar characteristics to the real operation). The test consists in processing a typical order (with 21 SKUs), repeating this operation 6 times per worker.

To calculate the learning rate of each worker, the two intervals method was used (Tilindis and Kleiza [9]). In the framework of scheduling problem, the autonomous learning or learning-by-doing is defined for several models. In this study, the most accepted and commonly used models, from those referenced by Paredes-Astudillo et al. [7], were selected to fit the experimental data.

2.4 Fatigue model

A total of 14 picking workers (one per each picking zone) participated in this phase. The subjects (7 women and 7 men) were between 18 and 34 years old ($26.21 y.o. \pm 5.23$). Participants agreed to take part in the study by signing an informed consent.

Muscular activity was registered with eight (8) Surface Electromyography Sensors (SX230, BioMetrics Ltd., Uk). Sensors were placed in 8 muscles involved in the picking operation for 40 minutes of real work. We adapted from Glock et al. [4] the fatigue-recovery model to define the accumulated fatigue level per worker. Similarly, a *Rest Allowance* (RA) model is adopted for determining the location and length of breaks.

2.5 FSSP with learning effects

In this phase of the study, fatigue and learning should be incorporated simultaneously, as phenomena that affect the job processing time. The first conceptual models dealing with FSSP with a learning effect for minimization of makespan are proposed. We intend to propose a bi-criteria model, including makespan minimization (economic objective) and minimization of the maximum difference of accumulated fatigue level among workers (social objective). Since the FSSP problem belongs to the class of $\mathcal{NP}$-hard problems, we solve it by implementing a constructive heuristic method such as the *Nawaz-Enscore-Ham algorithm* (NEH), improved by a Simulated Annealing algorithm (SA) and connected with a simulation technique.

3 Our results

In this section, we present the results obtained up to now in our study.

3.1 FSSP results

In Paredes-Astudillo et al. [6] we present the mathematical model, using Mixed-Integer Linear Programming (MILP) of a typical flow shop scheduling problem with makespan minimization. For small problem instances, the Glpk solver allows us to find the optimal solution. This model is the basis for the subsequent incorporation of the effects of learning and fatigue.

3.2 Learning model results

Using the two intervals method (Glock et al. [4]), it was possible to calculate the Learning Rate (LR) from the experimental data. Our results are similar to the LR reported in the literature, which indicates that for manual labor, the rate is nearly 80%. The experimental data were fitted to nine learning equations collected by Paredes-Astudillo et al. [7].

The performance comparison with the MAPE, showed that in 92% of cases, three models (Eqs. (1),(2), and (3)) showed a superior fit. They are:

- *Position-Based Learning Effect Model* (*P-LE*)

$$p_{ijr} = \bar{p}_{ij} r^{\alpha}, \tag{1}$$

- *Sum-of-Processing-Time-Based Learning Effect Model* (*ST-LE*)

$$p_{ijr} = \left(1 + \theta \sum_{k=1}^{r-1} p_{ijk}\right)^{\alpha} \bar{p}_{ij} \tag{2}$$

and

- *DeJong's Learning Effect Model* (*DJ-LE*)

$$p_{ijr} = \left(M + (1 - M) r^{\alpha}\right) \bar{p}_{ij}, \tag{3}$$

where α is the learning index ($\alpha < 0$), θ is a conversion factor (e.g., $\frac{1}{60}$ to convert hours to minutes) and M represents the *factor of incompressibility* ($M = 0.5$ in our case). These findings indicate that the learning effect of workers is not represented by a single model; on the contrary, different models of learning effect may coexist.

3.3 Fatigue model results

During the experimental sessions, we measured muscular activity of 8 muscles during the execution of picking operations. The signals were processed and normalized.

We managed to achieve an indicator of fatigue of 26.80% of the total of the 1082 SKUs registered in the warehouse. Due to the SKU rotation and based on the Pareto principle (80/20 rule), it is possible to speak of a representation closer to 80% of the total of picking operation.

3.4 FSSP with learning and fatigue effect results

During the initial phase, the problem was approached conceptually. In Paredes-Astudillo et al. [6] mathematical models of FSSP with learning effect and minimizing the makespan were stated.

Four models for calculating the learning effect mentioned in the literature were considered (Eqs. (1),(2),(4), and (5)). For small problem instances, Glpk and Bonmin solvers were used to solve linear (with Eqs. (1) and (4)) and non-linear models (with Eqs. (2) and (5)), respectively, where

- *Truncated Position-Based Learning Effect* (*TP-LE*)

$$p_{ijr} = \bar{p}_{ij} \max\left\{r^{\alpha}, \beta\right\}, \tag{4}$$

- *Truncated Sum-of-Processing-Time-Based Learning Effect* (*TST-LE*)

$$p_{ijr} = \bar{p}_{ij} \max\left\{\left((1+\theta\sum_{k=1}^{r-1} p_{ijk}\right)^{\alpha}, \beta\right\} \tag{5}$$

and β is a control parameter with $0 < \beta < 1$.

The results and the literature support the claim that this problem, in the case of three or more production resources with makespan minimization, belongs to the class of $\mathcal{NP}$-hard problems.

Considering large problem instances, we applied the Johnson's Rule (JS) for the problem with 2-resources and the NEH algorithm. We also developed a SA algorithm, where the initial solution is obtained using the NEH and the Adjacent Pairwise Interchange (API) and Non-Adjacent Pairwise Interchange (NAPI) local search operators were compared.

Computational experiments were performed using benchmark datasets for job processing times, α and β. With a total of 155.520 executions, we concluded that the NEH algorithm achieves outstanding results and that the SA works better for models with faster learning rate. In general terms, the SA + API shows improved performance compared to SA + NAPI.

4 Future research

The presented ongoing research includes the mathematical model of the FSSP with fatigue and a description of an algorithm for solving benchmark and case study instances. Thereafter, we addressed the FSSP with learning and fatigue effects, considering the minimization of the makespan and the maximum difference of accumulated fatigue level among workers.

In future research, the SA algorithm will be used, and a simulation technique to identify a good solution for both objectives. For the case study, the plan is to provide a breaks pattern for the workers which will enable them to recover physically.

With our work, we combine human factor modeling and operational research in a real-world scenario, enabling the company to maintain productivity standards and improve the well-being of workers.

Acknowledgements

The work is funded under research grants INGPhD-45-2021 and INGPHD-51-2022 from Universidad de La Sabana, Colombia and from the Eiffel Excellence Scholarship PhD stream, awarded to the first author by the French Ministry of Europe and Foreign Affairs.

The authors thank the Center for Ergonomics Studies of Pontificia Universidad Javeriana, the company and the workers who voluntary participated in this study.

References

[1] A. Agnetis, J-C. Billaut, S. Gawiejnowicz, D. Pacciarelli, A. Soukhal, *Multiagent Scheduling: Models and Algorithms*, Springer, Berlin-Heidelberg, 2014, doi:10.1007/978-3-642-41880-8.

[2] M. Calzavara, A. Persona, F. Sgarbossa, V. Visentin, A model for rest allowance estimation to improve tasks assignment to operators, *International Journal of Production Research*, **57** (2019), 948–962, doi:10.1080/00207543.2018.1497816.

[3] S. Gawiejnowicz, *Models and Algorithms of Time-Dependent Scheduling*, Springer, Berlin-Heidelberg, 2020, doi:doi:10.1007/978-3-662-59362-2.

[4] C. H. Glock, E. H. Grosse, T. Kim, W. P. Neumann, A. Sobhani, An integrated cost and worker fatigue evaluation model of a packaging process, *International Journal of Production Economics*, **207** (2019), 107–124, doi:10.1016/j.ijpe.2018.09.022.

[5] G. Mosheiov, Scheduling problems with a learning effect, *European Journal of Operational Research*, **132** (2001), 687–693, doi:10.1016/S0377-2217(00)00175-2.

[6] Y. A. Paredes-Astudillo, V. Botta-Genoulaz, J. R. Montoya-Torres, Comparing linear and non-linear modelling approaches of learning effects in 2-stage flow-shop scheduling problems, *IFAC-PapersOnLine*, **55** (2022), 842–847, doi:10.1016/j.ifacol.2022.09.517.

[7] Y. A. Paredes-Astudillo, J. R. Montoya-Torres, V. Botta-Genoulaz, Taxonomy of scheduling problems with learning and deterioration effects, *Algorithms*, **15** (2022), 439, doi:10.3390/a15110439.

[8] F. Sgarbossa, E. H. Grosse, W. P. Neumann, D. Battini, C. H. Glock, Human factors in production and logistics systems of the future, *Annual Reviews in Control*, **49** (2020), 295–305, doi:10.1016/j.arcontrol.2020.04.007.

[9] J. Tilindis, V. Kleiza, Learning curve parameter estimation beyond traditional statistics, *Applied Mathematical Modelling*, **45** (2017), 768–783, doi:10.1016/j.apm.2017.01.025.

[10] J. Wang, Z. Xia, Flow-shop scheduling with a learning effect, *Journal of the Operational Research Society*, **56** (2005), 1325–1330, doi:10.1057/palgrave.jors.2601856.

DOI: 10.14708/isbn.978-83-962157-1-0p89-94
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

Single-machine scheduling with two competing agents and a rate-modifying activity with weighted due-date related functions

Johnson Phosavanh*
The University of Sydney Business School, Sydney, Australia

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

Keywords: rate-modifying activities, competing agents, due-dates, dynamic programming

1 Introduction

In this study, we first consider solving the problem of minimizing the total weighted late work for a single agent with a rate-modifying activity, then we extend the results to the two-agent case. Additionally, we consider the two-agent problem involving the weighted number of tardy jobs, and finally, we consider the asymmetric problem of minimizing the total weighted late work with a bound on the weighted number of late jobs. All four problems considered in this study are binary $\mathcal{NP}$-hard, and we provide efficient pseudopolynomial-time dynamic programs to solve them. In addition to this, we conduct a comprehensive numerical analysis of the algorithms and provide numerical examples to demonstrate the complexity of the problem.

2 Related research

Two-agent scheduling problems in the form where one agent has an objective function, and the other has a constraint with an upper bound were first studied by Agnetis et al. [2]. The objective and constraint functions considered in this study were the maximum of regular functions, total completion time, the weighted completion time, and the number of tardy jobs. In this pioneering study, the authors provided polynomial-time algorithms that are used to obtain optimal schedules for all problems except for problems involving the total completion time and weighted

*Speaker, e-mail: johnson.phosavanh@sydney.edu.au

completion time, which were shown to be binary $\mathcal{NP}$-hard. More generally, multiagent scheduling problems on a single machine have been studied extensively, culminating in the book by Agnetis et al. [1] and the survey paper by Gonzalez and Framinan [9]. More recent work on two-agent scheduling has focused on solving problems with specific job characteristics or more complex job configurations. For example, two-agent single-machine scheduling problems have been studied under the assumption of equal job processing times by Oron et al. [8], and even with the added complexity of job rejection when minimizing the total late work, a problem which was established to be binary $\mathcal{NP}$-hard by Freud and Mosheiov [3].

The concept of *rate-modifying activities* (*RMAs*) was first proposed by Lee and Leon in a study on single-agent, single-machine scheduling problems (Lee and Leon [5]). The authors assumed that the rate-modifying activity was an optional task that, when completed, decreases the processing time of all subsequently scheduled jobs. More recently, a book on time-varying processing times was published with a significant portion dedicated to problems involving rate-modifying activities (Strusevich and Rustogi [11]). Similar to two-agent single-machine problems, total late work has only recently been studied with a rate-modifying activity where it is established that the problem is binary $\mathcal{NP}$-hard (Mosheiov and Oron [7]).

To tie these two concepts together, we will consider two scheduling criteria: the weighted number of tardy jobs and the total weighted late work. Both single-agent problems involving these functions have been shown to be binary $\mathcal{NP}$-hard (Potts and Van Wassenhove [10], Hariri et al. [4]). Recently, these functions have also been studied in the multiagent setting. For instance, the two-agent problem of minimizing the total weighted late work with a bound on the total completion time and the multiagent problem of finding Pareto optimal schedules for m agents were also binary $\mathcal{NP}$-hard (Zhang [12], Li and Yuan [6]).

3 Problem formulation

We consider the case where two agents, A and B, each have a set of jobs, denoted $\mathcal{J}^A = \{J_1^A, \ldots, J_{n_A}^A\}$ and $\mathcal{J}^B = \{J_1^B, \ldots, J_{n_B}^B\}$, respectively. Without loss of generality, assume that the objective is associated with agent A and the constraint is applied to agent B. Each job J_j^k has a regular processing time denoted p_j^k, a reduced processing time denoted q_j^k, only realized after the RMA has been completed, a due date denoted d_j^k and a positive weight w_j^k. The optional RMA takes T units of time to complete.

For a given schedule, we denote C_j^k to be the completion time of job J_j^k.

We will employ the same notation established by Agnetis et al. [2] to characterize competing two-agent problems, $1 \mid \text{RMA} \mid f_A : f_B \leq Q$, where f_A is the functional objective applied to agent A, and f_B is the scheduling criterion applied to agent B, which has an upper bound of Q (Agnetis et al. [2]). In this study, we will consider two different scheduling criteria: the weighted number of tardy jobs $\sum_{j=1}^{n_k} w_j^k U_j^k$, where $U_j^k = \mathbb{1}_{\{C_j^k > d_j^k\}}$ is an indicator if job J_j^k is completed late, and the total weighted late work $Y_w^k := \sum_{j=1}^{n_k} w_j^k \min\{\max\{C_j^k - d_j^k, 0\}, q_j^k\}$, where we have q_j^k instead of the traditional p_j^k as it can be shown that any job that cannot be started by its due date will be completed with the effect of the RMA. It is important to note that each of these functions are calculated with values specific to agent k only, $k \in \{A, B\}$, however, they are influenced by the presence of the other agent's jobs.

Furthermore, to simplify the notation, we also define $n := n_A + n_B$ to be the total number of jobs in the problem and $P := \sum_{k \in \{A,B\}} \sum_{j=1}^{n_k} p_j^k$ to be the sum of the regular processing times of all the jobs in the problem.

4 Our results

We first study the single-agent problem with RMA on minimizing the total weighted late work. Clearly, this problem is binary $\mathcal{NP}$-hard as the non-RMA version is already binary $\mathcal{NP}$-hard. Using the properties of an optimal job schedule derived using switching arguments, we propose a pseudo-polynomial dynamic programming algorithm to solve the problem. Following this, we generalize our result to the two-agent problem, and to complement this, we also consider the weighted number of late jobs as an additional scheduling criterion. Here, we study three additional problems, two where the objective and constraint functions are the same, and the third where we minimize the total weighted late work with a bound on the weighted number of late jobs. Table 1 summarizes the results of the four problems considered in our study.

To conclude our study, we conduct an extensive numerical analysis of the algorithms under a wide parameter setting. We also provide numerical examples for $1 \mid \text{RMA} \mid Y_w^A : Y_w^B \leq Q$. Fig. 1 displays optimal job schedules for $1 \mid \text{RMA} \mid Y_w^A : Y_w^B \leq Q$ as the bound on the constraint Q is varied for the data in Table 2 and setting $T = 10$. Note that the data is sorted in EDD order for each agent. The lines on the graph show the realized total weighted late work for each agent and correspond to the left y-axis,

Table 1. Summary of running times for problems studied

Problem	**Running time**
$1 \mid \text{RMA} \mid Y_w$	$\mathcal{O}(n^3P)$
$1 \mid \text{RMA} \mid \sum w_j^A U_j^A : \sum w_j^B U_j^B \leq Q$	$\mathcal{O}(nPQ)$
$1 \mid \text{RMA} \mid Y_w^A : \sum w_j^B U_j^B \leq Q$	$\mathcal{O}(nn_A^2PQ)$
$1 \mid \text{RMA} \mid Y_w^A : Y_w^B \leq Q$	$\mathcal{O}(n^3PQ)$

whereas the shaded rectangular boxes indicate the order in which jobs are completed in the optimal job schedule, with the job position given on the right y-axis. Also, note that jobs that are skipped in the job schedule have been omitted from the figure.

Table 2. Jobs dataset

	1	2	3	4	5
p_j^A	4	5	4	10	20
q_j^A	2	4	3	7	8
w_j^A	2	7	6	3	9
d_j^A	12	15	20	37	40
p_j^B	5	5	5	20	15
q_j^B	4	4	5	3	8
w_j^B	2	7	9	2	10
d_j^B	8	9	16	29	51

Fig. 1 displays some interesting properties of optimal job schedules for $1 \mid \text{RMA} \mid Y_w^A : Y_w^B \leq Q$. Firstly, as expected, as Q is increased and the bound on the constraint is loosened, more agent B jobs are delayed and skipped, and more agent A jobs are allowed to be completed earlier in the job schedule. Additionally, we see that there exist optimal job schedules where jobs are not completed in EDD order. For example, when $Q = 5$, job J_2^B is completed before J_1^B, and job J_5^A is also completed before J_4^A. Finally, we observe that the position of the RMA is not fixed and is hard to predict. For example, at around $Q = 80$, the RMA switches from the fourth task completed to the first task completed, then back to the fourth task completed.

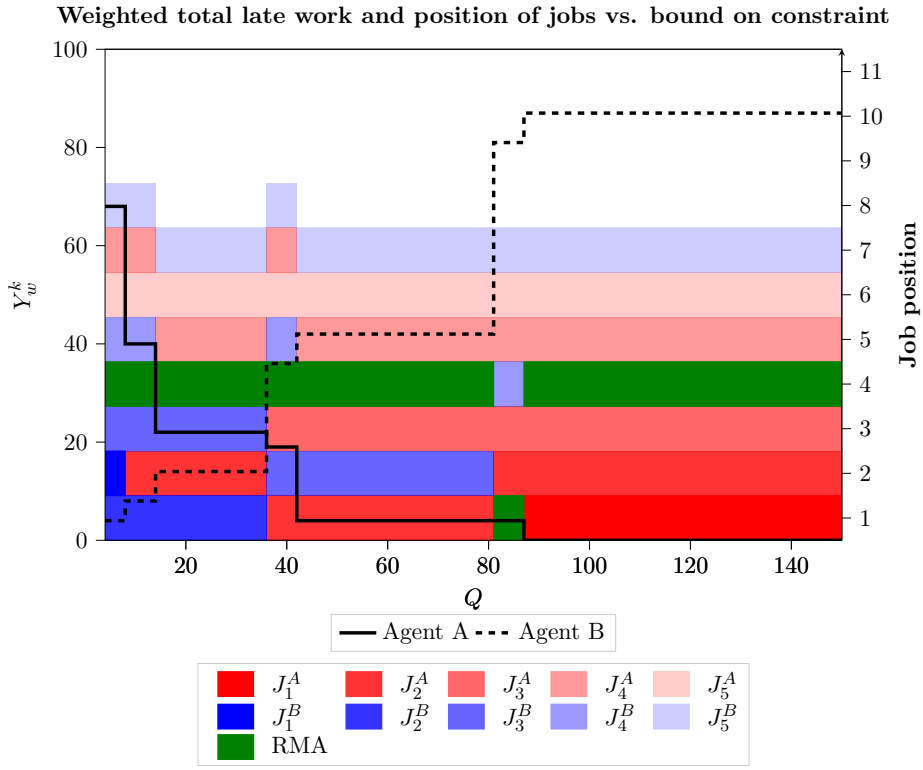


Figure 1. Solving $1 \mid \text{RMA} \mid Y_w^A : Y_w^B \leq Q$ for different values of Q with $T = 10$ and data from Table 2

5 Future research

In the future, this problem may be studied with additional assumptions and different machine specifications, such as equal job processing times or job rejection.

References

[1] A. Agnetis, J-C. Billaut, S. Gawiejnowicz, D. Pacciarelli, A. Soukhal, *Multiagent Scheduling: Models and Algorithms*, Springer, Berlin-Heidelberg, 2014, doi:10.1007/978-3-642-41880-8.

[2] A. Agnetis, P. Mirchandani, D. Pacciarelli, A. Pacifici, Scheduling problems with two competing agents, *Operations Research*, **52** (2004), 229–242, doi:10.1287/opre.1030.0092.

[3] D. Freud, G. Mosheiov, Scheduling with competing agents, total late work and job rejection, *Computers & Operations Research*, **133** (2021), 105329, doi:10.1016/j.cor.2021.105329.

[4] A. M. A. Hariri, C. N. Potts, L. N. Van Wassenhove, Single machine scheduling to minimize total weighted late work, *ORSA Journal on Computing*, **7** (1995), 23–242, doi:10.1287/ijoc.7.2.232.

[5] C-Y. Lee, V. Leon, Machine scheduling with a rate-modifying activity, *European Journal of Operational Research*, **128** (2011), 119–128, doi:10.1016/S0377-2217(99)00066-1.

[6] S. Li, J. Yuan, Single-machine scheduling with multi-agents to minimize total weighted late work, *Journal of Scheduling*, **23** (2020), 497–512, doi:10.1007/s10951-020-00646-7.

[7] G. Mosheiov, D. Oron, A note on scheduling a rate modifying activity to minimize total late work, *Computers & Industrial Engineering*, **154** (2021), 107138, doi:10.1016/j.cie.2021.107138.

[8] D. Oron, D. Shabtay, G. Steiner, Single machine scheduling with two competing agents and equal job processing times, *European Journal of Operational Research*, **244** (2015), 86–99, doi:10.1016/j.ejor.2015.01.003.

[9] P. Perez-Gonzalez, J. Framinan, A common framework and taxonomy for multicriteria scheduling problem with interfering and competing jobs: Multi-agent scheduling problems, *European Journal of Operational Research*, **235** (2014), 1–16, doi:10.1016/j.ejor.2013.09.017.

[10] C. N. Potts, L. N. Van Wassenhove, Algorithms for scheduling a single machine to minimize the weighted number of late jobs, *Management Science*, **34** (1988), 843–858, doi:10.1287/mnsc.34.7.843.

[11] V. A. Strusevich, K. Rustogi, *Scheduling with Time-Changing Effects and Rate-Modifying Activities*, Springer, Cham, 2017, doi:10.1007/978-3-319-39574-6.

[12] X. Zhang, Two competitive agents to minimize the weighted total late work and the total completion time, *Applied Mathematics and Computation*, **406** (2021), 126286, doi:10.1016/j.amc.2021.126286.

DOI: 10.14708/isbn.978-83-962157-1-0p95-100
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

Lot sizing and scheduling of injection molding machines with setup resources and demand uncertainty

Helmut Sedding*
ZHAW School of Engineering, Winterthur, Switzerland

Maxim Seidel
ZHAW School of Engineering, Winterthur, Switzerland

Keywords: scheduling, lot sizing, robust optimization

1 Introduction

Lot sizing and scheduling determines production quantities and their allocation on available resources to meet demands over time, represented by time periods $\mathcal{T}$. In this paper, we introduce a model suited for production planning of an injection molding factory of plastic parts.

In the factory, several injection molding machines are available. A machine needs a product-specific tool to produce plastic parts. The tool determines the number of produced parts per time unit, which is independent of the machine. During production, a machine mostly runs unsupervised. During setup, the machine cannot produce. Setup of a different tool requires manual labor. The number of setup workers is limited. Hence a production plan needs to respect this capacity. Every product has a recurring but variable demand. Moreover, the demand is uncertain and the actual demand is known only shortly before dispatching the produce. The production plan should satisfy the demands and anticipate demand changes while respecting all production constraints.

The problem setting studied seems not to be covered in the literature. We are only aware of work that partially covers our practical problem setting, but not its entirety. In essence, the studied problem is related to other lot sizing and scheduling problems. For recent literature reviews, we refer the reader to Copil et al. [12] and Wörbelauer et al. [25].

*Speaker, e-mail: helmut.sedding@zhaw.ch

In this work, we introduce a mathematical model for lot sizing and scheduling on identical machines while respecting setup times that are product-dependent and sequence-independent with a limited number of setup resources. By applying a rolling horizon planning procedure, the plan can adapt to changing parameters. Demand changes are anticipated by robust planning.

2 Related research

Our model is close to *Discrete Lot Sizing and Scheduling* (*DLS*), described in Fleischmann [15] and first considered in Lasdon and Terjung [20]. DLS's main feature is that a time period is used to produce exactly one type of product or none, the assumption of "all-or-nothing" by Schrage [23]. Relevant to our model is the DLS model in Campbell [9] with multiple machines, introducing setup time as a fraction of a period. This model can also limit the number of a product's parallel executions.

We consider product-dependent and sequence-independent setup times. For a single resource, this is studied in Cattrysse et al. [10]. In the application area of injection molding, such setup times are described in Ibarra-Rojas et al. [18], Ríos-Solís et al. [22], and Servantes-Sanmiguel et al. [11]. Other available references on injection molding planning consider the more complex case of sequence-dependent setup times, e.g. Lin et al. [21], and Ghosh and Nagi [17].

Considering a limited number of workers available to set up the machines is barely covered in [25], even in scheduling without lot-sizing. In the literature reviews in Allahverdi et al. [4], Allahverdi [2], there is no research found about sequence-independent parallel machine scheduling; this literature gap is also underlined in Allahverdi et al. [3]. Only recently there appear references on limited setup resources (see, e.g., Fanjul-Peyro [14], Yepes-Borrero et al. [27, 26]).

Robust optimization models are increasingly used in the last years (see, e.g., Bertsimas et al. [5], Gabrel et al. [16], Sözüer and Thiele [24]), also in lot-sizing and scheduling. For example, Alem et al. [1] and Curcio et al. [13] use a robust optimization model to respond to uncertain demands in a *General Lot-sizing and Scheduling* (*GLS*) *problem*. Without the scheduling aspect, robust lot-sizing is discussed by Bertsimas and Thiele [8] and Klabjan et al. [19].

3 Our results

We give a mathematical programming formulation of our model, describe a rolling horizon planning procedure to dynamically react on changing settings, and, since demands can be uncertain in our case, we introduce a robust model with a budgeted uncertainty, building upon results in Bertsimas and Sim [6, 7], and Bertsimas and Thiele [8]. Numerical tests evaluate the usability of our model. Let us briefly describe our model and approach to handle demand uncertainty.

Given is a set of time periods $\mathcal{T}$, a subset of dispatch times $\mathcal{H} \subseteq \mathcal{T}$, and a number of identical machines on which a set of products $\mathcal{J}$ can be produced. A positive demand value $d_{j,h}$ states how many items of a product $j \in \mathcal{J}$ need to be completed until dispatch time $h \in \mathcal{H}$. The objective is to minimize the total deficit penalty, which involves, for each dispatch time, the difference between cumulative demand and cumulative produce. We need to decide which products are produced on the available machines in each time period $t \in \mathcal{T}$. This requires setting up a product-specific tool, which takes s_j time periods and requires a setup worker. For this, only a limited number w of setup workers is available.

To incorporate demand uncertainty, let $\widehat{d}_{j,h} \geq 0$ denote the maximum deviation of the (mean) demand $d_{j,h}$ of product $j \in \mathcal{J}$ at dispatch time $h \in \mathcal{H}$. Then, the uncertain demand lies in $d_{j,h} \pm \widehat{d}_{j,h}\, \xi_{j,h}$ with random variable $\xi_{j,h} \in [-1, 1]$. For each product $j \in \mathcal{J}$, the production planner may choose a budget of uncertainty $\Gamma_j \geq 0$, which imposes $\sum_{h \in \mathcal{H}} |\xi_{j,h}| \leq \Gamma_j$. Hence, it specifies how many dispatches can be fulfilled safely in the worst case. It is then possible to introduce a robust counterpart of the model by applying the method described in Bertsimas and Sim [6].

References

[1] D. Alem, E. Curcio, P. Amorim, B. Almada-Lobo, A computational study of the general lot-sizing and scheduling model under demand uncertainty via robust and stochastic approaches, *Computers & Operations Research*, **90** (2018), 125–141, doi:10.1016/j.cor.2017.09.005.

[2] A. Allahverdi, The third comprehensive survey on scheduling problems with setup times/costs, *European Journal of Operational Research*, **246** (2015), 345–378, doi:10.1016/j.ejor.2015.04.004.

[3] A. Allahverdi, J. N. D. Gupta, T. Aldowaisan, A review of scheduling research involving setup considerations, *Omega*, **27** (1999), 219–239, doi:10.1016/S0305-0483(98)00042-5.

[4] A. Allahverdi, C-T. Ng, T-C. E. Cheng, M. Y. Kovalyov, A survey of scheduling problems with setup times or costs, *European Journal of Operational Research*, **187** (2008), 985–1032, doi:10.1016/j.ejor.2006.06.060.

[5] D. Bertsimas, D. B. Brown, C. Caramanis, Theory and applications of robust optimization, *SIAM Review*, **53** (2011), 464–501, doi:10.1137/080734510.

[6] D. Bertsimas, M. Sim, Robust discrete optimization and network flows, *Mathematical Programming*, **98** (2003), 49–71, doi:10.1007/s10107-003-0396-4.

[7] D. Bertsimas, M. Sim, The price of robustness, *Operations Research*, **52** (2004), 35–53, doi:10.1287/opre.1030.0065.

[8] D. Bertsimas, A. Thiele, A robust optimization approach to inventory theory, *Operations Research*, **54** (2006), 150–168, doi:10.1287/opre.1050.0238.

[9] G. M. Campbell, Using short-term dedication for scheduling multiple products on parallel machines, *Production and Operations Management*, **1** (1992), 295–307, doi:10.1111/j.1937-5956.1992.tb00361.x.

[10] D. Cattrysse, M. Salomon, R. Kuik, L. N. Van Wassenhove, A dual ascent and column generation heuristic for the discrete lotsizing and scheduling problem with setup times, *Management Science*, **39** (1993), 477–486, doi:10.1287/mnsc.39.4.477.

[11] K. I. Cervantes-Sanmiguel, M. Judith Vargas-Flores, O. J. Ibarra-Rojas, A two-stage sequential approach for scheduling with lot-sizing decisions in the context of plastic injection systems, *Computers & Industrial Engineering*, **151** (2021), 106969, doi:10.1016/j.cie.2020.106969.

[12] K. Copil, M. Wörbelauer, H. Meyr, H. Tempelmeier, Simultaneous lotsizing and scheduling problems: a classification and review of models, *OR Spectrum*, **39** (2017), 1–64, doi:10.1007/s00291-015-0429-4.

[13] E. Curcio, P. Amorim, Q. Zhang, B. Almada-Lobo, Adaptation and approximate strategies for solving the lot-sizing and scheduling problem under multistage demand uncertainty, *International Journal of Production Economics*, **202** (2018), 81–96, doi:10.1016/j.ijpe.2018.04.012.

[14] L. Fanjul-Peyro, Models and an exact method for the unrelated parallel machine scheduling problem with setups and resources, *Expert Systems with Applications*, **X** (2020), 100022, doi:10.1016/j.eswax.2020.100022.

[15] B. Fleischmann, The discrete lot-sizing and scheduling problem, *European Journal of Operational Research*, **44** (1990), 337–348, doi:10.1016/0377-2217(90)90245-7.

[16] V. Gabrel, C. Murat, A. Thiele, Recent advances in robust optimization: an overview, *European Journal of Operational Research*, **235** (2014), 471–483, doi:10.1016/j.ejor.2013.09.036.

[17] S. Ghosh Dastidar, R. Nagi, Scheduling injection molding operations with multiple resource constraints and sequence dependent setup times and costs, *Computers & Operations Research*, **32** (2005), 2987–3005, doi:10.1016/j.cor.2004.04.012.

[18] O. J. Ibarra-Rojas, R. Z. Ríos-Mercado, Y. A. Rios-Solis, M. A. Saucedo-Espinosa, A decomposition approach for the piece–mold–machine manufacturing problem, *International Journal of Production Economics*, **134** (2011), 255–261, doi:10.1016/j.ijpe.2011.07.006.

[19] D. Klabjan, D. Simchi-Levi, M. Song, Robust stochastic lot-sizing by means of histograms, *Production and Operations Management*, **22** (2013), 691–710, doi:10.1111/j.1937-5956.2012.01420.x.

[20] L. S. Lasdon, R. C. Terjung, An efficient algorithm for multi-item scheduling, *Operations Research*, **19** (1971), 946–969, doi:10.1287/opre.19.4.946.

[21] C. K-Y. Lin, C-L. Wong, Y-C. Yeung, Heuristic approaches for a scheduling problem in the plastic molding department of an audio company, *Journal of Heuristics*, **8** (2002), 515–540, doi:10.1023/A:1016588608032.

[22] Y. Á. Ríos-Solís, O. J. Ibarra-Rojas, M. Cabo, E. Possani, A heuristic based on mathematical programming for a lot-sizing and scheduling

problem in mold-injection production, *European Journal of Operational Research*, **284** (2020), 861–873, doi:10.1016/j.ejor.2020.01.016.

[23] L. Schrage, The multiproduct lot scheduling problem, in: M. A. H. Dempster, J. K. Lenstra, A. H. G. Rinnooy Kan (eds.), *Deterministic and Stochastic Scheduling*, pp. 233–244, Springer, Dordrecht, 1982.

[24] S. Sözüer, A. C. Thiele, The state of robust optimization, in: M. Doumpos, C. Zopounidis, E. Grigoroudis (eds.), *Robustness Analysis in Decision Aiding, Optimization, and Analytics*, pp. 89–112, Springer, Cham, 2016, doi:10.1007/978-3-319-33121-8_5.

[25] M. Wörbelauer, H. Meyr, B. Almada-Lobo, Simultaneous lotsizing and scheduling considering secondary resources: a general model, literature review and classification, *OR Spectrum*, **41** (2019), 1–43, doi:10.1007/s00291-018-0536-0.

[26] J. C. Yepes-Borrero, F. Perea, R. Ruiz, F. Villa, Bi-objective parallel-machine scheduling with additional resources during setups, *European Journal of Operational Research*, **292** (2021), 443–455, doi:10.1016/j.ejor.2020.10.052.

[27] J. C. Yepes-Borrero, F. Villa, F. Perea, J. P. Caballero-Villalobos, GRASP algorithm for the unrelated parallel-machine scheduling problem with setup times and additional resources, *Expert Systems with Applications*, **141** (2020), 112959, doi:10.1016/j.eswa.2019.112959.

DOI: 10.14708/isbn.978-83-962157-1-0p101-108
The Fourth International Workshop on Dynamic Scheduling Problems
ZHAW School of Engineering, Winterthur, Switzerland, June 5th – 6th, 2023

Minimizing the total operations rejection cost plus makespan value in a two-machine flow shop scheduling problem[‡]

Enrique Gerstl
School of Business Administration, The Hebrew University, Jerusalem, Israel

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

Keywords: flow shop, rejection, $\mathcal{NP}$-hardness, approximation algorithms

1 Introduction

One of the assumptions in classical flow shop scheduling problems is that all operations have to be processed in the shop. However, in heavily loaded flow shop systems, accepting all operations in the shop may lead to high job completion times, and thus to a poor *quality of service* (QoS) and future abandonment of customers. One of the most common ways of dealing with such a problem is to reject a subset of the operations. This, on the one hand, has the advantage of lowering the load on the production system. On the other hand, rejecting an operation has its own costs. In this paper, we focus on such a two-machine flow shop scheduling problem, where the scheduler can reject a subset of the operations at a certain cost.

2 Problem formulation

Our problem can be formulated as follows. A set of n jobs, $\mathcal{J} = \{J_1, \ldots, J_n\}$, is available to be processed at time 0 on two machines in a flow shop scheduling system. Let O_{ij} be the operation in which job J_j is to be processed on machine M_i, where $i = 1, 2$ and $j = 1, \ldots, n$, and let $\mathcal{O} = \{O_{ij} | i = 1, 2 \text{ and } j = 1, \ldots, n\}$ be the set of all operations. The processing time of operation O_{ij} is p_{ij}, and the scheduler can decide whether to process this operation

[‡]Presented online
*Speaker, e-mail: dvirs@bgu.ac.il

in the shop or to reject it at a cost of e_{ij}. If operation O_{1j} is rejected, it is executed by the subcontractor prior to time 0, and therefore O_{2j} is available at time zero to be processed on M_2.

A solution S for our scheduling problem with rejection consists of two sets of decisions: *Partitioning* and *Scheduling*. The first requires a decision to be made on the partitioning of set $\mathcal{O}$ into two subsets: $\mathcal{O}_A$ and $\mathcal{O}_R$, corresponding to the set of accepted and the set of rejected operations, respectively. The second set of decisions pertains to the scheduling of the jobs in $\mathcal{O}_A$ on the machines.

Given such a schedule, let C_{ij} be the completion time of operation O_{ij}, and let $C_{\max}(\mathcal{O}_A) = \max_{O_{ij}\in\mathcal{O}_A}\{C_{ij}\}$ be the makespan value. Our objective is to find a solution S minimizing the sum of the makespan and the total rejection cost.

Following the three-field notation for scheduling problems, we refer to the problem as $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij}\in\mathcal{O}_R} e_{ij}$.

3 Related research

Lee and Choi [4] were the first to study the $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij}\in\mathcal{O}_R} e_{ij}$ problem, focusing on the special case where $e_{ij} = \alpha_i p_{ij}$ for $i = 1, 2$ and $j = 1, \ldots, n$. They proved that the corresponding problem is $\mathcal{NP}$-hard when either

(i) $\alpha_1 < 1$ and $\alpha_2 \geq 1$ or

(ii) $\alpha_1 < 1$, $\alpha_2 < 1$ and $\alpha_1 + \alpha_2 \geq 1$.

However, they showed that when either

(iii) $\alpha_1 + \alpha_2 \leq 1$ or

(iv) $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$

the problem can be solved in polynomial time.

Lee and Choi [4] also constructed a greedy algorithm that has a $(5-\sqrt{5})/2$ approximation ratio when (i) holds, and a $(1+\sqrt{5})/2$ approximation ratio when (ii) holds.

Jiang *et al.* [2] provided a tighter analysis of Lee and Choi's greedy algorithm, showing that it actually yields a $\frac{5}{4}$ approximation ratio if (i) hold, and $\frac{4}{3}$ approximation ratio if (ii) holds. They also modified the algorithm to one that provides a $\frac{5}{4}$ approximation ratio if (ii) holds. Gau and Lu [1] showed that the $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij}\in\mathcal{O}_R} \alpha_i p_{ij}$ problem

is solvable in pseudo-polynomial time, and convert it to a *fully polynomial-time approximation scheme* (*FPTAS*).

As the literature is restricted to the special case where $e_{ij} = \alpha_i p_{ij}$ for $i = 1, 2$ and $j = 1, \ldots, n$, our main aim in this paper is to analyze the general case where the rejection costs are arbitrary.

4 Our results

When $e_{ij} \to \infty$ for any $O_{ij} \in \mathcal{O}$, our problem reduces to the classical $F2\,||\,C_{\max}$ problem that can be solved in $O(n \log n)$ time by applying Johnson's algorithm, [3].

Algorithm 1. *Johnson's algorithm for solving the* $F2\,||\,C_{\max}$ *problem.*

Step 1 [Partitioning]*: Partition* $\mathcal{J}$ *as follows:* $\mathcal{J}_1 = \{J_j | p_{1j} \leq p_{2j}\}$ *and* $\mathcal{J}_2 = \{J_j | p_{1j} > p_{2j}\}$.

Step 2 [Sequencing]*: Construct an optimal permutation schedule,* $\sigma^*_{\mathcal{J}}$*, as follows: Schedule first the jobs in* $\mathcal{J}_1$ *in a non-decreasing order of* p_{1j}*. Then schedule the jobs of in* $\mathcal{J}_2$ *in a non-increasing order of* p_{2j}.

Step 3 [Scheduling]*: For* $j = 1, \ldots, n$*, compute* $C_{i,\sigma^*_{\mathcal{J}}(j)}$ *by* $C_{1,\sigma^*_{\mathcal{J}}(j)} = \sum_{l=1}^{j} p_{1,\sigma^*_{\mathcal{J}}(l)}$ *and* $C_{2,\sigma^*_{\mathcal{J}}(j)} = \max\{C_{2,\sigma^*_{\mathcal{J}}(j-1)}, C_{1,\sigma^*_{\mathcal{J}}(j)}\} + p_{2,\sigma^*_{\mathcal{J}}(j)}$*, where* $C_{i,\sigma^*_{\mathcal{J}}(0)} = 0$*, by definition, for* $i = 1, 2$*. Schedule* $J_{\sigma^*_{\mathcal{J}}(j)}$ *on* M_i $(i = 1, 2)$ *during time interval* $(C_{i,\sigma^*_{\mathcal{J}}(j)} - p_{i,\sigma^*_{\mathcal{J}}(j)}, C_{i,\sigma^*_{\mathcal{J}}(j)}]$.

We call any schedule that is constructed using Algorithm 1 a *Johnson schedule,* and the corresponding job permutation, $\sigma^*_{\mathcal{J}}$, a *Johnson permutation.* Consider now an alternative optimal schedule (which we call *modified Johnson schedule*), which takes the Johnson schedule as an input, and modifies it by shifting the operations on M_2 to the right as much as possible without changing the makespan value.

We next show that if the partitioning of set $\mathcal{O}$ into $\mathcal{O}_A$ and $\mathcal{O}_R$ is given, then we can easily obtain the optimal scheduling decision of set $\mathcal{O}_A$ on the two machines by using Johnson's algorithm. To do so, given the partition of set $\mathcal{O}$ into $\mathcal{O}_A$ and $\mathcal{O}_R$, we further partition set $\mathcal{J}$ into the following four subsets: $\mathcal{J}^{(1)} = \{J_j \in \mathcal{J} \mid O_{1j} \in \mathcal{O}_R \text{ and } O_{2j} \in \mathcal{O}_A\}$; $\mathcal{J}^{(2)} = \{J_j \in \mathcal{J} \mid O_{1j} \in \mathcal{O}_A \text{ and } O_{2j} \in \mathcal{O}_A\}$; $\mathcal{J}^{(3)} = \{J_j \in \mathcal{J} \mid O_{1j} \in \mathcal{O}_A \text{ and } O_{2j} \in \mathcal{O}_R\}$; and $\mathcal{J}^{(4)} = \{J_j \in \mathcal{J} \mid O_{1j} \in \mathcal{O}_R \text{ and } O_{2j} \in \mathcal{O}_R\}$. Note that we can view any $O_{ij} \in \mathcal{O}_R$ as an operation that is scheduled in the shop, but requires zero processing time. Therefore, based on Johnson's algorithm, the following corollary holds:

Corollary 1. *Given a partition of set $\mathcal{O}$ into $\mathcal{O}_A$ and $\mathcal{O}_R$, the optimal schedule, $\sigma^*_{\mathcal{O}_A}$, can be computed in $O(n \log n)$ time by: scheduling first the jobs in $\mathcal{J}^{(1)}$ in an arbitrary order; scheduling next the jobs in $\mathcal{J}^{(2)}$ which are ordered according to Johnson's rule with respect to the original processing times (p_{1j} and p_{2j}); and lastly scheduling the jobs in $\mathcal{J}^{(3)}$ in an arbitrary order.*

Given a partition of set $\mathcal{O}$ into $\mathcal{O}_A$ and $\mathcal{O}_R$, we next present a method to easily compute the minimal makespan value for such a partition, given the minimal makespan value that corresponds to a *partial* modified Johnson schedule where only the jobs in $\mathcal{J}^{(2)}$ are scheduled. The method is based on ideas that were already presented in Gau and Lu [1].

Consider a *partial* modified Johnson schedule that includes only the jobs in $\mathcal{J}^{(2)}$, and let s_i and c_i be the start and completion times of this schedule on machine M_i, where $i = 1, 2$. Note that $c_2 \geq c_1$ in such a *partial* modified Johnson schedule, i.e., the makespan value is equal to the completion time on M_2. Moreover, let $x = c_1 - s_2$ and $r = c_2 - c_1$. Based on Corollary 1, to compute the makespan value of a *complete* modified Johnson schedule, we need to include the jobs in $\mathcal{J}^{(1)}$ at the beginning of the schedule and the jobs in $\mathcal{J}^{(3)}$ at the end of the schedule. Now, let $a = \sum_{J_j \in \mathcal{J}^{(3)}} p_{1j}$ and $b = \sum_{J_j \in \mathcal{J}^{(1)}} p_{2j}$.

Lemma 1. *The makespan value of a complete modified Johnson schedule can be computed by $C_{\max} = c_2 + \max\{0, a - r, b - s_2\}$.*

The following theorem is included without a proof.

Theorem 2. *The $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{1j} \in \mathcal{O}_R} e_{1j}$ problem is $\mathcal{NP}$-hard even when either (i) $p_{2j} = p$ and $e_{2j} \to \infty$ for $j = 1, \ldots, n$; or (ii) $p_{1j} = p$ and $e_{1j} \to \infty$ for $j = 1, \ldots, n$.*

4.1 Linear programming formulation

Hereafter, we assume (without loss of generality) that the jobs are renumbered according to Johnson permutation, $\sigma^*_{\mathcal{J}} = (1, \ldots, n)$. We next present an *Integer Linear Programming* (*ILP*) formulation for problem $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij} \in \mathcal{O}_R} e_{ij}$. To do so, let x_{ij} be a binary decision variable that equals 1 if we assign job J_j to set $\mathcal{J}^{(i)}$ and equals 0 otherwise ($i = 1, \ldots, 4$ and $j = 1, \ldots, n$). Obviously, we can assign job J_j to only one of the sets, and therefore we include the following set of constraints:

$$\sum_{i=1}^{4} x_{ij} = 1 \text{ for } j = 1, \ldots, n. \tag{1}$$

For $i = 1, 2$, let c_i, and t_i be continuous decision variables representing the start and completion time of a *partial* modified Johnson schedule that includes only the jobs in $\mathcal{J}^{(2)}$ on machine M_i $(i = 1, 2)$. Moreover, let $x = c_1 - s_2$ and $r = c_2 - t_1$. We have that

$$c_1 = \sum_{j=1}^{n} p_{1j} x_{2j}, \tag{2}$$

and that

$$c_2 \geq \sum_{j=1}^{k} p_{1j} x_{2j} + \sum_{j=k}^{n} p_{2j} x_{2j} \text{ for } k = 1, \ldots, n. \tag{3}$$

The set of constraints in (3) is based on computing the value of the longest path in a graph consist of n paths, where each path in the graph provides a lower bound on the makespan value of the jobs in $\mathcal{J}^{(2)}$. To compute c'_i, which is the completion time of the entire modified Johnson schedule on machine M_i, where $i = 1, 2$, we need to include the jobs in $\mathcal{J}^{(1)}$ at the beginning of the schedule, and the jobs in $\mathcal{J}^{(3)}$ at the end of the schedule. To do so, we include the constraints that

$$a = \sum_{j=1}^{n} p_{1j} x_{3j};\ b = \sum_{j=1}^{n} p_{2j} x_{1j};\ r = c_2 - c_1; \tag{4}$$

and in a modified Johnson schedule, the constraint that

$$s_2 = c_2 - \sum_{j=1}^{n} p_{2j} x_{2j}. \tag{5}$$

Since $C_{\max} = \max\{c'_1, c'_2\} = c_2 + \max\{0, a - r, b - s_2\}$, we include the constraints that

$$C_{\max} \geq c_2 + a - r;\ C_{\max} \geq c_2 + b - s_2; \text{ and that } C_{\max} \geq c_2. \tag{6}$$

The objective is to minimize

$$C_{\max} + \sum_{j=1}^{n} e_{1j}(x_{1j} + x_{4j}) + \sum_{j=1}^{n} e_{2j}(x_{3j} + x_{4j}). \tag{7}$$

Concluding, the *ILP* formulation for problem $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{1j} \in \mathcal{O}_R} e_{1j}$ consists of minimizing the cost function in (7) subject to the conditions in (1)-(6).

4.2 Approximation algorithms

It is easy to see that if $p_{ij} \leq e_{ij}$ then there exists an optimal schedule in which $O_{ij} \in \mathcal{O}_A$. Based on this observation, we suggest a naive $O(n \log n)$ time heuristic algorithm, **H1**, that starts by setting $\mathcal{O}_A = \{O_{ij} \in \mathcal{O} \mid$

$p_{ij} \leq e_{ij}\}$, and $\mathcal{O}_R = \mathcal{O} \setminus \mathcal{O}_A$. We then define $\mathcal{J}^{(1)} = \{J_j \in \mathcal{J} \mid O_{1j} \in \mathcal{O}_R$ and $O_{2j} \in \mathcal{O}_A\}$; $\mathcal{J}^{(2)} = \{J_j \in \mathcal{J} \mid O_{1j} \in \mathcal{O}_A$ and $O_{2j} \in \mathcal{O}_A\}$; $\mathcal{J}^{(3)} = \{J_j \in \mathcal{J} \mid O_{1j} \in \mathcal{O}_A$ and $O_{2j} \in \mathcal{O}_R\}$ and $\mathcal{J}^{(4)} = \{J_j \in \mathcal{J} \mid O_{1j} \in \mathcal{O}_R$ and $O_{2j} \in \mathcal{O}_R\}$. Finally, apply Corollary 1 to schedule the operations.

Algorithm **H1** was based on the observation that if $p_{ij} \leq e_{ij}$, there exists an optimal schedule in which $O_{ij} \in \mathcal{O}_A$. However, this does not necessarily imply that an operation O_{ij} that satisfies $p_{ij} > e_{ij}$ should be included in set $\mathcal{O}_R$, as suggested by **H1**. Therefore, we suggest using an alternative heuristic algorithm, **H2**, that is based on a greedy approach. Algorithm **H2** starts by applying Algorithm 1 with $\mathcal{O}_A = \mathcal{O}$ to obtain $\sigma^*_{\mathcal{J}}$. We then renumber the jobs according to $\sigma^*_{\mathcal{J}}$, and process to apply an iterative greedy algorithm. In iteration j ($j \in \{1, \ldots, n\}$) of the algorithm, we make a greedy decision regarding which set ($\mathcal{J}^{(i)}$ for $i = 1, 2, 3, 4$) to assign job J_j.

Theorem 3. *Algorithms **H1** and **H2** are 2-approximation algorithms for the* $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij} \in \mathcal{O}_R} e_{ij}$ *problem, and the approximation ratio is tight.*

4.3 Pseudo-polynomial-time algorithm

We next show that the same ideas used by Gau and Lu [1] to solve the $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij} \in \mathcal{O}_R} \alpha_i p_{ij}$ problem can also lead to a pseudo-polynomial time algorithm for the more general $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij} \in \mathcal{O}_R} e_{ij}$ problem. To do so, consider a partial partition of job set $\mathcal{J}_j = \{J_1, \ldots, J_j\}$ into the four possible sets of jobs, $\mathcal{J}_j^{(1)}, \mathcal{J}_j^{(2)}, \mathcal{J}_j^{(3)}$ and $\mathcal{J}_j^{(4)}$. We represent such a feasible partition by the state $[E, l, x, r, a, b, \{\mathcal{J}_j^{(i)} | i = 1, 2, 3, 4\}]$, where $a = \sum_{J_j \in \mathcal{J}_j^{(3)}} p_{1j}$, $b = \sum_{J_j \in \mathcal{J}_j^{(1)}} p_{2j}$ and $E = \sum_{J_j \in \mathcal{J}_j^{(1)}} e_{1j} + \sum_{J_j \in \mathcal{J}_j^{(3)}} e_{2j} + \sum_{J_j \in \mathcal{J}_j^{(4)}} (e_{1j} + e_{2j})$. In this partition, the *partial* modified Johnson schedule, which includes only the jobs in $\mathcal{J}_j^{(2)}$, has (*i*) a start time of l on M_2; (*ii*) a gap of x between the finish time on M_1 and the start time on M_2; and (*iii*) a gap of r between the finish times on M_2 and M_1.

Lemma 4. *Let* $[E, l, x, r, a, b, \overline{\mathcal{J}}_j]$ *and* $[E', l, x, r, a, b, \overline{\mathcal{J}}'_j]$ *be two feasible states representing feasible partial partitions of job set* $\mathcal{J}_j$. *If* $E \leq E'$ *then the partial partition represented by* $[E', l, x, r, a, b, \overline{\mathcal{J}}'_j]$ *is dominated by the partial partition represented by* $[E, l, x, r, a, b, \overline{\mathcal{J}}_j]$.

The implication of Lemma 4 is that any state that is dominated by another state can be eliminated from a state construction procedure that

extends feasible partial solutions to complete ones. Let set $\mathcal{L}_j$ include all possible non-dominated feasible states for the problem on job set $\mathcal{J}_j$, where $j = 1, \ldots, n$, and let $P_i = \sum_{j=1}^{n} p_{ij}$ for $i = 1, 2$. The following theorem plays an important rule in our algorithm:

Theorem 5. *Given $\mathcal{L}_j$, one can compute $\mathcal{L}_{j-1}$ in $O((P_1)^2(P_2)^2 \min\{P_1, P_2\})$ time.*

We solve the $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij}\in\mathcal{O}_R} e_{ij}$ problem by using the result in Theorem 5, to compute $\mathcal{L}_j$ for $j = 1, \ldots, n$ starting from $\mathcal{L}_0 = \{[E, l, x, r, a, b, \overline{\mathcal{J}}_j] = [0, 0, 0, 0, 0, 0, \{\emptyset, \emptyset, \emptyset, \emptyset\}]\}$. Using this method, we were able to solve the problem in $O(n(P_1)^2(P_2)^2 \min\{P_1, P_2\})$ time.

Gau and Lu [1] converted their pseudo-polynomial time algorithm for the $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij}\in\mathcal{O}_R} \alpha_i p_{ij}$ problem into an FPTAS using the scaling and rounding technique, [5]. To implement this method, one needs to derive a lower bound, LB, on the objective value that satisfies the condition that the ratio between the running time of the pseudo-polynomial time algorithm and LB is a polynomial function of n. Unfortunately, such a lower bound is not available for the $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij}\in\mathcal{O}_R} e_{ij}$ problem. We overcome the above difficulty by dividing the solution space into $O(n)$ subproblems. For each subproblem, we provided a lower bound on the solution value that satisfies the condition that the ratio between the running time of the pseudo-polynomial time algorithm that solves the subproblem and the lower bound value is a polynomial function of n. This enable us to construct an FPTAS for each of the subproblems, separately. Then, we showed that picking the best solution out of the $O(n)$ approximate solutions for the subproblems yields an FPTAS for the general $F2\,|rej(O_{ij})|\,C_{\max}(\mathcal{O}_A) + \sum_{O_{ij}\in\mathcal{O}_R} e_{ij}$ problem.

5 Future research

The following research questions are still open regarding the problem we study and can be considered in future research:

(i) Can we provide approximation algorithms running in polynomial time and having an approximation ratio which is smaller than 2?

(ii) Can we improve the time complexity required by the pseudo-polynomial time algorithm or the FPTAS?

(iii) Can we provide an efficient algorithm (e.g., an FPT algorithm) to solve the case of bounded number of different processing times on both machines?

References

[1] Q. Gau, X. Lu, Two-machine flow shop scheduling with individual operation's rejection, *Asia-Pacific Journal of Operational Research*, **31** (2014), 1450002, doi:10.1142/S021759591450002X.

[2] X. Jiang, A. Zhang, Y. Chen, G. Chen, K. Lee, An improved algorithm for a two-stage production scheduling problem with an outsourcing option, *Theoretical Computer Science*, **876** (2021), 59–69, doi:10.1016/j.tcs.2021.05.022.

[3] S. M. Johnson, Optimal two and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly*, **1** (1954), 61–67, doi:10.1002/nav.3800010110.

[4] K. Lee, B. C. Choi, Two-stage production scheduling with an outsourcing option, *European Journal of Operation Research*, **213** (2011), 489–497, doi:10.1016/j.ejor.2011.03.037.

[5] S. Sahni, Algorithms for scheduling independent tasks, *Journal of the ACM*, **23** (1976), 116–127, doi:10.1145/321921.321934.

Indexes

Index of authors

Index of keywords