PMS 20 GNT 22

18th International Workshop on Project Management and Scheduling

6-8 April 2022 Ghent, Belgium

VLERICK BUSINESS SCHOOL

GHENT UNIVERSITY

https://pms2022.sciencesconf.org/

Book of Extended Abstracts

Mario Vanhoucke

Ghent, 11/04/2022

Foreword

Ghent April, 2022.

PMS is an international workshop series devoted to Project Management and Scheduling. It was inaugurated by the European Working Group on Project Management and Scheduling (EURO - EWG Project management and scheduling), originally coordinated by Prof. Jan Węglarz (Poznan University of Technology, Poland) and now coordinated by Prof. Erik Demeulemeester (KU Leuven, Belgium) and Prof. Joanna Józefowska (Poznan University of Technology, Poland).

The EWG decided to organise a workshop every two years. The workshops provide an ideal opportunity to discuss recent and important issues in the field of project management (planning, scheduling, control) and machine scheduling (single and parallel machine problems, flow shop, job shop, etc.).

The first workshop was held in Lisbon in July 1988. The successor workshops were held in Compiègne (1990), Como (1992), Leuven (1994), Poznan (1996), Istanbul (1998), Osnabrück (2000), Valencia (2002), Nancy (2004), Poznan (2006), Istanbul (2008), Tours (2010), Leuven (2012), Munich (2014), Valencia (2016), Rome (2018) and Toulouse (2021 - online edition due to COVID-19).

The PMS 2022 workshop should have been held at the Vlerick Business School, located in Ghent, Belgium. The Vlerick Business School Campus Ghent is located in the historic city centre of Ghent, and is situated on the renovated premises of the Major Seminary, which was an institution for the training of Catholic clergy for the diocese of Ghent. Due to the COVID-19 coronavirus pandemic, we were unfortunately forced to organise the PMS 2022 workshop in a virtual way. The PMS 2022 workshop is co-organised by Vlerick Business School and Ghent University under the scientific supervision of the EURO EWG-PMS International committee.

The Workshop covers the following but non-exhaustive list of project management and scheduling areas:

- Project Management: Network modelling, Project scheduling, Resource management, Due- date management, Project risk management, Project scheduling under uncertainty, Proactive/reactive project scheduling, Multi-criteria project scheduling, Applications, Software
- Machine Scheduling: Shop scheduling, Scheduling with additional constraints, Machine assignment and scheduling, Flexible/robust scheduling, Grid scheduling, Multi-criteria scheduling, Applications, Software.

Methodological/theoretical papers related to Operational Research, Artificial Intelligence/ Machine Learning models, exact and heuristic algorithms for scheduling problems were presented, as well as papers dealing with data-driven approaches, practical applications and industrial case studies.

Overall, 49 extended abstracts were received and 42 extended abstracts were accepted after a peer-review process, and 105 participants registered to the workshop. A total of 20 nationalities are represented among the participants, as displayed below:

France	31	Colombia	2
Germany	20	Norway	2
Belgium	17	Switzerland	2
Italy	4	USA	2
Turkey	3	UK	2
Isreal	3	Hungary	1
Poland	3	India	1
China	3	Canada	1
Spain	3	Philippines	1
Australia	2	**# Countries**	**20**
Portugal	2	**# Participants**	**105**

Not less than 21 extended abstracts applied to the Best Student Paper Award and 6 finalists were selected to present their work in 2 dedicated sessions. Prizes were awarded by EURO. Congratulations to Hendrik Weber (First Prize), Tom Portoleau (Second Prize) and Jakob Snauwaert (Third Prize).

We had the pleasure to listen to plenary talks by Vincent T'Kindt and Öncü Hazir.

You will find in these proceedings :

- The member list of the Organisation Committee,
- The member list of the Program Committee,
- The finalists and the winners of the Best Student Paper Award,
- The conference program,
- The plenary talk abstracts and a short bio of each plenary speaker,
- The extended abstracts,
- The list of participants,
- The list of sponsors.

We warmly thank all the participants and the international program committee who greatly contributed to the large success of the workshop!

The PMS 2022 organising Committee.

Organising committee

pms2022@sciencesconf.org

Mario Vanhoucke. Ghent University (Workshop chair)
José Coelho. Ghent University
Annelies Martens. Ghent University
Tom Servranckx. Ghent University
Gaëtane Beernaert. Vlerick Business School

International Program Committee

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

Best student paper award

The following extended abstracts were finalists of the best student paper award:

- Generation and Characterization of Real-World Instances for the Flexible Resource-Constrained Multi-Project Scheduling Problem
  Hendrik Weber, Robert Brachmann, Rainer Kolisch

- Large neighborhood search for a multi-mode resource constrained scheduling problem with resource leveling objective
  Tom Portoleau, Christian Artigues, Tamara Borreguero Sanchidrian, Alvaro Garcia Sanchez, Miguel Ortega Mier, Pierre Lopez

- New empirical and artificial data instances for the multi-skilled resource-constrained project scheduling problem
  Jakob Snauwaert, Mario Vanhoucke

- Solving the Assembly Line Balancing Problem with LocalSolver
  Léa Blaise, Thierry Benoist, Christian Artigues

- A branch and bound approach for stochastic 2-machine flow shop scheduling with rework
  Lei Liu, Marcello Urgo

- Total Core Idle Time minimization for the permutation flowshop scheduling problem
  Paula Sanchez-de los Reyes, Paz Perez-Gonzalez

The jury is made of the whole International Program Committee, except those involved in the phD thesis.

- The first prize (500€) was awarded by EURO to Hendrik Weber (Technical University of Munich, Germany)
- The second prize (300€) was awarded by EURO to Tom Portoleau (LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France)
- The third prize was awarded to (200€) was awarded by EURO to Jakob Snauwaert (Ghent University, Belgium)

DAY 1 - Wednesday 6 April 2022

Welcome Session at 8.30 AM

https://vlerick.zoom.us/j/81322496972?pwd=Mmdwb3Y3QXlXRWFJYlJJZCt5L2kwZz09

Student Award 1

(Same ZOOM link as Welcome Session)

9.00 AM - 10.30 AM

Chair: Erik Demeulemeester

Generation and Characterization of Real-World Instances for the Flexible Resource-Constrained Multi-Project Scheduling Problem
Hendrik Weber, Robert Brachmann, Rainer Kolisch

Large neighborhood search for a multi-mode resource constrained scheduling problem with resource leveling objective
Tom Portoleau, Christian Artigues, Tamara Borreguero Sanchidrian, Alvaro Garcia Sanchez, Miguel Ortega Mier, Pierre Lopez

New empirical and artificial data instances for the multi-skilled resource-constrained project scheduling problem
Jakob Snauwaert, Mario Vanhoucke

Project Scheduling Track

https://vlerick.zoom.us/j/83725972479?pwd=ZTMvTjBBaE1leU5za2xFSlJ1N1JYZz09

10.50 AM - 12.10 PM
PS 1: RCPSP | Chair: José Coelho

Carbon footprint aware resource constrained project scheduling problem in manufacturing
Humyun Rahman, Tom Servranckx, Ripon Chakrabortty, Mario Vanhoucke, Sondoss El Sawah

Problem-specific Priority Rules for Resource-Constrained Project Scheduling Problem with Alternative Subgraphs
Rojin Nekoueian, Tom Servranckx, Mario Vanhoucke

Assembly Line Performance Analysis Based on Aircraft Preliminary Design: a Scheduling Approach
Anouck Chan, Stéphanie Roussel, Thomas Polacsek

Heuristic solution approaches to the multi-project scheduling problem
Dries Bredael, Mario Vanhoucke

1.40 PM - 3.00 PM
PS 2: Extensions | Chair: Jürgen Zimmerman

A Novel Continuous-Time Mixed-Integer Linear Programming Model for the Multi-Mode Resource-Constrained Project Scheduling Problem
Nicklas Klein

A relaxation-based generation scheme for the RCPSP/max,pi
Mareike Karnebogen, Jürgen Zimmermann

A time-based schedule-generation scheme for project scheduling with storage resources
Mario Christian Sillus, Christoph Schwindt

A fix-and-optimize heuristic for the resource renting problem
Max Reinke, Jürgen Zimmermann

3.20 PM - 4.00 PM
PS 3: Methodologies | Chair: Avraham Shtub

Early-Stage Prediction of Project Duration - Machine Learning Approach vs. Traditional Approach
Itai Lishner, Avraham Shtub

A Method to Find Criticalities in Project Networks with Feeding Precedence Relations
Lucio Bianco, Massimiliano Caramia, Stefano Giordani, Alessio Salvatore

Machine Scheduling Track

https://vlerick.zoom.us/j/81295231269?pwd=aWFuWVJOd1lXald4V2g5MGFseFErdz09

10.50 AM - 12.10 PM
MS 1: Job scheduling | Chair: Vincent T'kindt

Learning based heuristics for scheduling jobs with release dates on a single machine to minimize the sum of completion times
Axel Parmentier, Vincent T'kindt

Sequencing two classes of jobs on a machine with an external no-idle constraint
Alessandro Agnetis, Marco Pranzo

Application of Quantum Approximate Optimization Algorithm to Job Shop Scheduling Problem
Tomasz Pecyna, Krzysztof Kurowski, Rafal Rozycki, Grzegorz Waligora, Jan Weglarz

Extending Smith's Rule with Task Mandatory Parts and Release Dates
Camille Bonnin, Margaux Nattaf, Arnaud Malapert, Marie-Laure Espinouse

1.40 PM - 3.00 PM
MS 2: Parallel machine scheduling | Chair: Stéphane Dauzere-Peres

Insights and results for the offline and online weighted capacitated parallel machine scheduling problem
Izack Cohen, Ilan Cohen, Iyar Zaks

Maximal slacks between lower bounds of the makespan on parallel processors
Claire Hanen, Jacques Carlier

An Inclusion-Exclusion based general exponential-time algorithm for the solution of unrelated parallel machine scheduling problems
Olivier Ploton, Vincent T'kindt

Aggregation techniques for a scheduling model on parallel machines in the photolithography area of the semiconductor manufacturing industry
Jeremy Berthier, Stéphane Dauzere-Peres, Claude Yugma

3.20 PM - 4.00 PM
MS 3: Robust/stochastic scheduling | Chair: Christian Artigues

Robust scheduling within SNCF railway maintenance centers
Rahman Torba, Stéphane Dauzere-Peres, Claude Yugma, Cédric Gallais, François Ramond

Two-stage stochastic/robust scheduling using permutable operation groups
Louis Riviere, Christian Artigues, Hélène Fargier

Plenary Session 1

(Same ZOOM link as Welcome Session)

4.00 PM - 5.00 PM

Chair: Erwin Pesch

The Fairy Tale of Scheduling and the Enchanted Combinatoric
Vincent T'Kindt

DAY 2 - Thursday 7 April 2022

Student Award 2

https://vlerick.zoom.us/j/88051222725?pwd=RythWUdPYzN2Y2tuOUFYKzdBbytPUT09

9.00 AM - 10.30 AM
Chair: Joanna Józefowska

Solving the Assembly Line Balancing Problem with LocalSolver
Léa Blaise, Thierry Benoist, Christian Artigues

A branch and bound approach for stochastic 2-machine flow shop scheduling with rework
Lei Liu, Marcello Urgo

Total Core Idle Time minimization for the permutation flowshop scheduling problem
Paula Sanchez-de los Reyes, Paz Perez-Gonzalez

Project Scheduling Track

https://vlerick.zoom.us/j/83565550863?pwd=emJjbU43bGcwRmljeHROdTF5MzQxZz09

10.50 AM - 12.10 PM
PS 4: Risk | Chair: Massimiliano Caramia

Budget allocation in risk prevention and risk protection considering risk interdependency
Xin Guan, Tom Servranckx, Mario Vanhoucke

Using schedule risk analysis with resource constraints for project control
Jie Song, Annelies Martens, Mario Vanhoucke

A comparative analysis for bounding the project completion time distribution in stochastic project networks
Forough Vaseghi, Annelies Martens, Mario Vanhoucke

A comparison of two project forecasting methods using risk models: Structural Equation Modeling and Bayesian Networks
Izel Unsal Altuncan, Mario Vanhoucke, Annelies Martens

1.40 PM - 3.00 PM
PS 5: Applications | Chair: Rainer Kolisch

Human-centered interactions for project scheduling decision-aid in space industry
Hugo Chevroton, Cyril Briand, Philippe Truillet, Melody Mailliez, Céline Lemercier

Project Planning for Engineering Automotive Production Systems
Maximilian Kolter, Martin Grunow, Rainer Kolisch, Thomas Stäblein

Comparative Study of Two Machine Learning Tasks in Project Scheduling
Weikang Guo, Mario Vanhoucke, José Coelho

Automated design of priority rules for the RCPSP via efficient genetic programming approach
Jingyu Luo, Mario Vanhoucke, José Coelho

Machine Scheduling Track

https://vlerick.zoom.us/j/82147276729?pwd=enBVVzlmeXdleHQycTBwY0l3Yi90dz09

10.50 AM - 12.10 PM
MS 4: Flexible scheduling | Chair: Alessandro Agnetis

Just-In-Time Flexible Job Shop with Stochastic Processing Times
Camilo Rodriguez-Espinosa, Eliana María González-Neira

Logic-based Benders Decomposition for preemptive Flexible Job-Shop Scheduling
Carla Juvin, Laurent Houssin, Pierre Lopez

On the relevance of the makespan service level for the flexible job shop scheduling problem under uncertainty
Mario Flores Gomez, Stéphane Dauzere-Peres, Valeria Borodin

Disjunctive graph model for flexible job-shop schedulingproblem with transportation and limited buffer space
Lucas Berterottière, Claude Yugma, Stéphane Dauzere-Peres

1.40 PM - 3.00 PM
MS 5: Extensions | Chair: Sigrid Knust

Heuristic Parameter Estimation by Machine Learning
Aykut Uzunoglu

Operating rooms scheduling with a shared resource: a red-blue knapsack modeling approach
Federico Della Croce, Andrea Grosso, Vincent T'kindt

Valid inequalities for the dynamic asset protection problem
Quentin Pena, Aziz Moukrim, Mehdi Serairi

A realistic hybrid flow shop scheduling problem with availability restrictions, priorities, and machine qualifications
Christin Schumacher, Dominik Mäckel

Plenary Session 2

(Same ZOOM link as Student Award 2)

3.20 PM - 4.20 PM
Chair: Christoph Schwindt

Project Modeling and Planning under Uncertainty: Last 20 years and Future Perspectives
Öncü Hazir

Closing Session

(Same ZOOM link as Student Award 2)

Plenary Talk

The Fairy Tale of Scheduling and the Enchanted Combinatoric

Vincent T'Kindt (Polytechnic University of Tours)

Time: 6 April 2022, 4.00 PM - 5.00 PM

Chair: Erwin Pesch

This talk is about the relationship between Scheduling (the nice guy) and Combinatoric (the bad guy). As scheduling researchers our challenges are, roughly speaking, to study how scheduling problems can be efficiently solved by computers. But then comes the Combinatoric, bringing to us difficulties, making us upset about not being able to easily solve our scheduling problems. Understanding why Combinatoric complicates our life is challenging but part of the game: there is no way to solve efficiently a scheduling problem without understanding what makes it difficult and without using appropriate algorithms.

I will give a feedback on my own experience, using some machine scheduling problems as illustrative examples. Notably, I will mainly focus on the exact and heuristic solution of scheduling problems by algorithms relying on mathematical programming, making also an outing in the lands of theoretical computer science and machine learning. I will conclude by evoking some research topics which could be used to write the next chapters of the fairy tale.

Keywords: Machine scheduling; Mathematical Programming ; Exponential-time algorithmsKeywords : Machine scheduling; Mathematical Programming ; Exponential-time algorithms

Vincent T'kindt is full professor in computer science at the Polytechnique College of the University of Tours (France). He got his Ph.D. in Computer Science in 1999 from the University of Tours for and his habilitation (HDR) in 2005. From 2007 to 2020, he headed the scheduling group (25 people) of the Laboratory of Applied and Theoretical Computer Science (LIFAT) of the University of Tours.

His initial research works were focused on multi-objective optimisation and scheduling theory, topic on which he published three books. Along the years, he has been interested in many topics related to optimisation with most of the time applications to scheduling theory: e.g. mathematical programming to solve large-size problems, exponential-time algorithms or even recently the interplay of machine learning and operational research. His researches have led to three books, more than 50 papers in international peer-reviewed journals and about 200 communications in conferences.

Project Modeling and Planning under Uncertainty: Last 20 years and Future Perspectives

Öncü Hazir (Rennes Business School)

Time: 7 April 2022, 3.20 PM - 4.20 PM

Chair: Christoph Schwindt

In this speech, a classification for major sources of uncertainty in projects will be presented and the approaches adopted in the literature for mitigating the impact of uncertainty will be discussed. A special emphasis will be given to robust optimisation and its applications in project planning. The progress in academic knowledge in this area over the last 20 years will be summarised. Mathematical models will be presented; the extensions and application areas will be discussed.

The analysis might serve as a useful basis for investigating and modelling the characteristics of various sources of uncertainty and their impacts on achieving project targets. The discussion can support researchers to identify the research gaps in developing project plans under uncertainty. Regarding research directions, the discussion will highlight some interesting topics that require studies combining project planning and control, data analytics, risk management, decision support systems.

Keywords: Project Planning, Scheduling, Risk Management, Robust Optimisation, Data Analytics, Decision Support Systems

Öncü Hazır is an associate professor at Rennes School of Business. He received his BS degree in Industrial Engineering and MBA degree from the Middle East Technical University, Ankara, Turkey. He completed his Ph.D. at the Department of Business Administration in Bilkent University in 2008. His dissertation was on project scheduling. He

worked as a post-doctorate researcher in Laboratoire d'Informatique de Paris 6 and Ecole Nationale Supérieure des Mines de Saint-Etienne and as an associate professor at TED University. He participated in various research projects in project planning and control, machine scheduling, and assembly line balancing and published several articles in operations research and management journals. He is a co-author of the recently published textbook, “An Introduction to Project Modelling and Planning”.

An Inclusion-Exclusion based general exponential-time algorithm for the solution of unrelated parallel machine scheduling problems

Olivier Ploton[1], Vincent T'kindt[1]

Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée
(LIFAT, EA 6300), ERL CNRS 7002 ROOT, Tours, France
{olivier.ploton,vincent.tkindt}@univ-tours.fr

Keywords: parallel machine scheduling, exponential algorithms, Inclusion-Exclusion.

1 Introduction

In this paper we cope with unrelated parallel machine scheduling problems. There are n jobs to be scheduled on m unrelated parallel machines. Each job i, when assigned to machine j, is defined by a processing time p_{ij}, a release time r_{ij}, and a deadline $\tilde{d}_{ij}$. In a schedule, each job i is associated with an elementary cost f_{ij}, computed as a function of its completion time C_i. The aim is to minimize the objective function, defined either as the maximum or as the sum of elementary costs. These problems, denoted by $R|r_{ij}, \tilde{d}_{ij}|f_{\max}$ and $R|r_{ij}, \tilde{d}_{ij}|\sum f_{ij}$ using the notation of Graham et al. (1979), are strongly NP-hard.

We are interested in the exact solution of these problems, with the best possible worst case space and time complexity bounds. In this work we focus on theoretical results rather than experimental ones. To evaluate complexity, we take the number n of jobs as the size of an instance. The measure $||\mathcal{I}||$ of an instance $\mathcal{I}$ is essentially the sum $\sum_{i,j} p_{ij}$ of processing times.

We restrict to regular and polynomial elementary cost functions, i.e. each function $C \mapsto f_{ij}(C)$ is non-decreasing w.r.t. C, and $(\forall C, f_{ij}(C) \leq P(C))$ for some polynomial P. This requirement is met by classical cost functions, as the completion time C_i, the lateness L_{ij}, the tardiness T_{ij}, the tardiness indicator U_{ij}, and any combination of these functions with positive weights.

Numerous particular cases of parallel machine scheduling problems have been studied, but there are few results in the general case. Jansen et al. (2013) provide a general worst-case time and space bound in $O^*(2^{O(n)}||\mathcal{I}||^{O(1)}) = O^*(c^n||\mathcal{I}||^{O(1)})$ for some c. Lente et al. (2014) describe an algorithm solving the $P||f_{\max}$ and $P||\sum f$ problems with $O^*(3^n)$ space and time worst-case complexity bounds. Our main contribution is to show that the very general $R|r_{ij}, \tilde{d}_{ij}|f_{\max}$ and $R|r_{ij}, \tilde{d}_{ij}|\sum f_{ij}$ problems can be solved with a worst-case time complexity bound in $O^*(2^n||\mathcal{I}||^{O(1)})$, and using only pseudopolynomial space.

The algorithms we describe use Inclusion-Exclusion, along with dynamic programming. The Inclusion-Exclusion formula is interesting because it allows to decide whether or not a problem admits a solution without ever building an explicit one. This technique, described in Fomin and Kratsch (2010), has been applied to NP-hard graph problems as the Hamiltonian path (Bax 1993, Karp 1982), Steiner trees and perfect matchings (Nederlof 2013). It has not been used widely in scheduling, but in Karp (1982), and in Nederlof (2008).

2 From the counting problem to an explicit optimal solution

The problem we cope with consists in minimizing a regular objective function $\gamma = \max_i f_{i,j(i)}$ or $\gamma = \sum_i f_{i,j(i)}$, where $j(i)$ is the machine assigned to job i. As γ is reg-

ular, we can restrict to semi-active schedules, where each job starts as soon as possible. Such a schedule S can be represented as a list of job lists, one per machine: $S = ((i_{11}...i_{1\ell_1})\dots(i_{m1}...i_{m\ell_m}))$. Moreover, it is useful to introduce release time bounds $\boldsymbol{B} = (B_1 \dots B_m)$, before which machines are not available. This way, a schedule can be decomposed into a prefix and a suffix, and the completions of the prefix are the release bounds of the suffix, as shown in Figure 1.

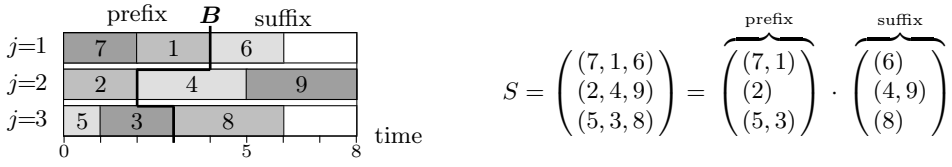


Fig. 1. Gantt chart of a semi-active schedule

Given a job set I, a threshold objective value ε and a vector of release bounds $\boldsymbol{B}$, the counting problem consists in computing the number $N(I, \boldsymbol{B}, \varepsilon)$ of schedules using (all) jobs of I, starting from $\boldsymbol{B}$, and with objective at most ε. The decision problem consists in testing if there exists such a schedule, i.e. if $N(I, \boldsymbol{B}, \varepsilon) > 0$. If we know how to compute N, we can build an optimal solution using a polynomial number of calls to N. First, we can compute the optimum objective value $\gamma^{opt} = \min\{\varepsilon \in \mathbb{N} \mid N(I, \mathbf{0}, \varepsilon) > 0\}$. Then, given a prefix π, we can use N as an oracle to check whether this prefix can be completed into an optimal solution. It can be completed when $N(I \setminus \pi, \boldsymbol{C}(\pi), \gamma^{opt}) > 0$ for a max-type objective, and when $N(I \setminus \pi, \boldsymbol{C}(\pi), \gamma^{opt} - \gamma(\pi)) > 0$, for a sum-type objective, with $\boldsymbol{C}(\pi)$ being the completions of π on each machine. Finally, we can build, step by step, an optimal solution (Algorithm 1).

Function *OptimalSolution*:
  $\pi \leftarrow$ empty prefix
  repeat n times
    find $i \notin \pi$ such that ($\pi' = \pi$ followed by i) can be completed into an optimal solution
    $\pi \leftarrow \pi'$
  return π

Algorithm 1: Computation of an optimal solution

3 The Inclusion-Exclusion technique

As explained by Fomin and Kratsch (2010), using the Inclusion-Exclusion formula consists in relaxing the problem by allowing schedules with duplicated or missing jobs. In order to reduce the m-machine problem into m single-machine problems in sections 4 and 5, we shall further relax the problem this way: given a schedule $S = ((i_{11}...i_{1\ell_1})\dots(i_{m1}...i_{m\ell_m}))$, we only require at most n jobs per machine, i.e. $\ell_1 \leq n, ..., \ell_m \leq n$, instead of requiring globally n jobs, i.e. $\ell_1 + ... + \ell_m = n$. So, a relaxed schedule is a list of m *independent* sub-lists of at most n jobs, with no correlation between them and no extra constraints.

The Inclusion-Exclusion formula computes the number N' of relaxed schedules using *all* jobs, given the numbers of relaxed schedules using *only* jobs of X, for each job subset X. It states:

$$N'(I, \boldsymbol{B}, \varepsilon) = \sum_{X \subset I} (-1)^{|I|-|X|} N_X(\boldsymbol{B}, \varepsilon) \tag{1}$$

where $N_X(\boldsymbol{B},\varepsilon)$ is the number of relaxed schedules starting from $\boldsymbol{B}$, using only jobs of X and with objective at most ε. Notice that $N'(I,\boldsymbol{B},\varepsilon)$ is not exactly the number $N(I,\boldsymbol{B},\varepsilon)$ of solutions, but, from a relaxed schedule using all jobs, we can derive a potentially better solution by removing duplicated jobs. So, $N'(I,\boldsymbol{B},\varepsilon){>}0 \iff N(I,\boldsymbol{B},\varepsilon){>}0$ and we may safely replace N with N'. We are now about to compute each $N_X(\boldsymbol{B},\varepsilon)$ in sections 4 and 5.

4 The relaxed counting problem for maximum-type objective functions

In order to compute $N_X(\boldsymbol{B},\varepsilon)$, we define $M_{X,j,\varepsilon}[b,\ell]$ as the number of relaxed schedules using only jobs of X, starting at release time bound b on the single machine j, with at most ℓ jobs, and with objective bounded by ε. The maximum objective is globally bounded by ε iff it is bounded by ε on each machine, so we have:

$$N_X(\boldsymbol{B},\varepsilon) = \prod_{j=1}^{m} M_{X,j,\varepsilon}[B_j,n] \tag{2}$$

We compute $M_{X,j,\varepsilon}[b,\ell]$ by dynamic programming. An empty job sequence counts as one solution. A non-empty job sequence $(i_1,i_2,...,i_h)$ can be decomposed into a head job $i=i_1$ and a tail $(i_2,...,i_h)$. The release bound of the tail is the completion C_i of the head job i. We derive:

$$M_{X,j,\varepsilon}[b,0] = 1 \tag{3a}$$

$$M_{X,j,\varepsilon}[b,\ell] = 1 + \sum_{i\in X,\ C_i\le\tilde{d}_{ij},\ f_{ij}(C_i)\le\varepsilon} M_{X,j,\varepsilon}[C_i,\ell{-}1] \qquad \text{for } \ell>0 \tag{3b}$$

$$\text{where } C_i = \max(b,r_{ij}) + p_{ij} \tag{3c}$$

We now focus on worst-case time and space complexities:

- As $B_j = O(||\mathcal{I}||)$, each computation of $M_{X,j,\varepsilon}[B_j,n]$ is in $O^*(||\mathcal{I}||)$ time and space.
- By (1), each decision step $N'(I,\boldsymbol{B},\varepsilon){>}0$ is in $O^*(2^n||\mathcal{I}||)$ time and $O^*(||\mathcal{I}||)$ space.
- There are $O(\log\gamma^{opt}){=}O^*(1)$ decision steps to compute γ^{opt} by dichotomy and $O(n){=}O^*(1)$ decision steps to compute a solution, the overall algorithm complexity is in $O^*(2^n||\mathcal{I}||)$ time and $O^*(||\mathcal{I}||)$ space.

5 The relaxed counting problem for sum-type objective functions

We redefine $N_X(\boldsymbol{B},\varepsilon)$ as the number of relaxed schedules with objective *exactly* ε, and we adapt the rationale. we define $M_{X,j}[b,\ell,\varepsilon]$ as the number of relaxed schedules using only jobs of X, starting at release time bound b on the single machine j, with at most ℓ jobs, and with objective *equal to* ε. The global objective ε is split between the machines, so we derive this convolution formula:

$$N_X(\boldsymbol{B},\varepsilon) = \sum_{\varepsilon_1+...+\varepsilon_m=\varepsilon} \prod_{j=1}^{m} M_{X,j}[B_j,n,\varepsilon_j] \tag{4}$$

We compute $M_{X,j}[b,\ell,\varepsilon]$ by dynamic programming. An empty job sequence counts as one solution if $\varepsilon=0$, no solution otherwise, which we write $\mathbb{1}_{\varepsilon=0}$. We derive:

$$M_{X,j}[b,0,\varepsilon] = \mathbb{1}_{\varepsilon=0} \tag{5a}$$

$$M_{X,j}[b,\ell,\varepsilon] = \mathbb{1}_{\varepsilon=0} + \sum_{i\in X,\ C_i\le\tilde{d}_{ij},\ f_{ij}(C_i)\le\varepsilon} M_{X,j}[C_i,\ell{-}1,\varepsilon{-}f_{ij}(C_i)] \qquad \text{for } \ell>0 \tag{5b}$$

$$\text{where } C_i = \max(b,r_{ij}) + p_{ij} \tag{5c}$$

To exploit the dynamic programming equation (5b), in which ε varies, and the fast convolution operator implementing formula (4) (Knuth 1997), we determine an upper bound $\varepsilon_{\max}$ (in $O(\gamma^{opt})$), and we compute together, as a vector, all $N_X(\boldsymbol{B}, \varepsilon)$ for all $\varepsilon \leq \varepsilon_{\max}$. We call this vector $\boldsymbol{N}_X(\boldsymbol{B})$. We apply the Inclusion-Exclusion formula (1) vectorially to compute a vector $\boldsymbol{N}'(I, \boldsymbol{B})$. There exists a schedule with objective at most $\varepsilon_{\max}$ when $\boldsymbol{N}'(I, \boldsymbol{B})$ has a non-null component. We derive these worst-case time and space complexities:

- By dynamic programming and convolution, each computation of $\boldsymbol{N}_X(\boldsymbol{B})$ is in $O^*(||\mathcal{I}||\gamma^{opt})$ time and space.
- Each computation of $\boldsymbol{N}'(I, \boldsymbol{B})$ and thus each decision step is in $O^*(2^n||\mathcal{I}||\gamma^{opt})$ time and $O^*(||\mathcal{I}||\gamma^{opt})$ space.
- There are $O^*(1)$ decision steps to compute a solution, the overall algorithm complexity is in $O^*(2^n||\mathcal{I}||\gamma^{opt})$ time and $O^*(||\mathcal{I}||\gamma^{opt})$ space.

6 Conclusions

In this paper we consider the problem of scheduling a set of jobs on unrelated parallel machines in the presence of job release dates and deadlines, and we deal with the minimization of any general regular, either maximum or sum, objective function. We describe a generic exact exponential algorithm, based on Inclusion-Exclusion and dynamic programming, solving the $R|r_{ij}, \tilde{d}_{ij}|f_{\max}$ problem in $O^*(2^n||\mathcal{I}||)$ time and $O^*(||\mathcal{I}||)$ space, and solving the $R|r_{ij}, \tilde{d}_{ij}|\sum f_{ij}$ problem in $O^*(2^n||\mathcal{I}||\gamma^{opt})$ time and $O^*(||\mathcal{I}||\gamma^{opt})$ space, where γ^{opt} is the optimum objective value, polynomial in $||\mathcal{I}||$. The strength of this algorithm is to manage a wide class of parallel machine scheduling problems, and to achieve, on a theoretical point on view, moderate exponential-time and pseudopolynomial space worst-case complexity bounds. While not the fastest in practice compared to specialized algorithms, this generic algorithm enhances the state-of-the-art theoretical worst-case complexity bounds of several particular parallel machine scheduling problems.

References

Bax E.T., 1993, "Inclusion and exclusion algorithm for the Hamiltonian path problem", *Information Processing Letters*, Vol 17(4), pp 203–207.

Fomin F.V., D. Kratsch, 2010, "Exact exponential algorithms", Springer.

Graham R.L., E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, 1979, "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey", *Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications*, Vol 5, pp 287–326.

Jansen K., F. Land, K. Land, 2013, "Bounding the Running Time of Algorithms for Scheduling and Packing Problems", *Algorithms and Data Structures - 13th International Symposium*, pp 439–450.

Karp R.M., 1982, "Dynamic Processing meets the principle of inclusion and exclusion", *Operational Research Letters*, Vol 1(2), pp 49–51.

Knuth, D., 1997, "The Art of Computer Programming", *Vol 2, p 305*, ISBN 0-201-89684-2.

Lenté C., Liedloff M., Soukhal A., T'Kindt V., 2014, "Exponential Algorithms for Scheduling Problems", https://hal.archives-ouvertes.fr/hal-00944382.

Nederlof J., 2008, "Inclusion-exclusion for hard problems", *Master Thesis*, Utrecht University.

Nederlof J., 2013, "Fast Polynomial-Space Algorithms Using Inclusion-Exclusion", *Algorithmica*, Vol 65, pp 868–884.

Early-Stage Prediction of Project Duration -

Machine Learning Approach vs. Traditional Approach

Itai Lishner, Avraham Shtub

Faculty of Industrial Engineering and Management,
Technion—Israel Institute of Technology, Haifa, Israel
e-mail: itailishner@hotmail.com

Keywords: Prediction, Machine Learning, Project Management, Gantt, Scheduling.

1. Introduction

Predicting the duration of a project is one of the most challenging tasks a project manager ought to handle. The uniqueness of each project, the unknown that accompanies the project life cycle and the risks that might come true, make the prediction task difficult and sometimes seem impossible. An early-stage prediction of project duration is by nature more difficult than prediction at other points in the project life cycle, as there is much information that is still unknown and will only be revealed during the execution of the project. Nevertheless, the need for a prediction of the project's duration in its early stages is essential for many reasons (Majid 2006, Alami, 2016), such as project portfolio planning, product roadmap planning, resources planning, budget planning, cash flow, customer satisfaction, and more. One of the most common methods to predict a project duration is using a Gantt chart (Gantt 1903); several prediction methods had been developed based on the Gantt chart, such as the Critical Path Method (CPM) and the PERT (Program Evaluation and Review Technique) (Wiest 1964, Petersen 1991). The simplicity of the Gantt chart makes it a very popular tool (Wilson 2003), but the fact that the Gantt chart cannot foresee changes in activities or resources, do not take into account the uncertainty in project execution and the fact that it's subject to estimation errors (König 2005, Hill et al. 2000, Josephs and Hahn 1995, Moore and Healy 2008, Burt and Kemp 1994), prevent it from being a good prediction tool in the early stages of the project. As using the Gantt chart alone is not good enough for predicting the project end date in its early stage, it was suggested (White and Hassan 2019, König et al. 2015) that to predict a project outcome, one should rely on information learned from past projects. The prediction can be done by creating a regression model that foresees the gap between the Gantt chart prediction and the actual end date of the project, based on information from similar projects performed in the past. The information extracted from past projects can be a single feature such as "estimated duration" or it can include additional features, such as "Project type", "Project risks", etc. These features add more information which potentially can increase prediction accuracy, but also make the regression calculation more complex to solve. The use of machine learning techniques allows one to create and solve a regression model for multi-feature regression without needing to handle the complexity of the calculation. This study presents a comparison between three types of early-stage prediction of a project duration: Traditional Gantt chart using CPM, Single Feature Regression (SFR) and Machine learning Multi-Features Regression (MFR). The comparison is based on data from 26 different projects executed in the same organization between the years 2018–2021.

2. Prediction models

2.1. Gantt chart and CPM

The Gantt chart was developed at the beginning of the 20th century by Henry L. Gantt (Gantt, 1903); it's a graphic diagram that consists of 2 axes; the vertical axis represents the project's activities and the horizontal axis represents time. The activities are presented in a bar format according to the start time and end time when the length of the bar represents the activity duration (Wilson, 2003). It is easy to see each activity's duration, when it begins and ends, the precedence relations between the tasks, where the *slack* is in the project, and it can also show the activities which

are on the critical path. To predict the project duration using the Gantt chart, one should point to the date on which the last activity on the critical path is expected to end. The Gantt chart may vary and be updated several times during the project life cycle; each update may cause an updated prediction of the duration of the project. The prediction of a project duration using the Gantt chart is based on activity-level time estimation and the precedence relationships among the activities in the project network schedule; therefore the prediction results are subject to estimation errors and changes in activities during the project life cycle. Applying the CPM methodology (Wiest 1964) on the Gantt chart allows one to get a clear date prediction on when the project will end. This is done by identifying the activities on the critical path. The critical path is the path that has the longest time of all paths starting from the first activity to the last activity in the project (Zareei, 2018). In order to determine the critical path, five parameters shall be defined for each activity in the Gantt chart: *D[i]* - the duration of activity *i, ES[i]* - the earliest possible start time of activity *i, EF[i]*- the earliest possible finish time of activity *i, LS[i]* - the latest possible start of activity *i, LF[i]* - the latest possible finish time of activity *i*. Calculating *ES[i]* and *EF[i]* for each activity is being done by forward pass algorithm calculation as specified in equation *1*.

ES[0] = 0.
For j = 1,2,3, …, n (where n is the last activity)
ES[j] = maximum{EF[m]}(where m represents all preceding activities of j)
EF[j] = ES[j] + D[j] (1)

Calculating *LS[i]* and *LF[i]* for each activity is done by backward pass algorithm calculation as specified in equation *2*.

Let LS[n] equal the minimal completion time of the project (LS[n]>=ES[n]).
For j = n-1, n-2, …, 0
LF[j] = minimum {LS[k]} (where k represents activities in which activity j is their predecessor)
LS[j] = LF[j] – D[j] (2)

Having the values of *ES[i], EF[i],LS[i]* and *LF[i]* for each activity allows one to calculate the activity *slack* time; *slack* is the maximum time interval that the activity can be delayed without delaying the project end time and it is calculated according to equation *3*. A *slack* equal to 0 means that any delay in the activity will cause a delay in the project end date; therefore an activity with no *slack* is a critical activity located on the critical path.

$$Slack[i]=LS[i]-ES[i]=LF[i]-EF[i] \quad (3)$$

2.2. Single-feature regression (SFR)

The use of past project information may help to improve prediction accuracy. Many projects start with the estimation of the project ending time when the project is still in its beginning stage, and the actual ending time is revealed when the project ends. The use of a data set that contains the project's estimated end date and the actual end date of the projects, allows one to create a regression model that describes the relation between predicted duration and the actual duration. Linear regression has been chosen to be used to describe the relation between the two features. The outcome is the SFR model which allows predicting the project duration according to the Gantt chart prediction as an input. This basic information about the projects is usually available for organizations and therefore the SFR can be implemented in many organizations. Equation 4 describes the SFR model. $T_{prediction}$ is the outcome of the SFR model—the predicted duration time of the project; T_{Gantt} is the input which is the Gantt chart prediction duration; *a* is the slope of the linear line and *b* is the intercept.

$$T_{prediction} = \mathrm{b} + \mathrm{a} \cdot T_{Gantt} \quad (4)$$

2.3. Multi-feature regression (MLR)

Adding additional information may improve the prediction accuracy. In addition to the Gantt chart prediction information used for the SFR model, we collected more features of each of the projects in the data set. These features will be used to build a machine learning-based regression model aimed to improve the predicting accuracy vs. the SFR. The additional features are not always available when retrospecting projects, so in order to use the MLR model, an organization needs to adopt a method of recording these features and build a data set that will be used for building the MLR model. The MLR model is based on a neural network model with Multi-Layer Perceptron (MLP) (Berlinetal 2009, López-Martínetal 2013) The model consists of an input layer, output layer and two hidden layers. The output of the MLR model is the predicted duration of the project and the

input for the model is 10 project features: *Estimated duration, Project type, Product type, Stability of the project scope, Degree of uncertainties or risks for project delay, Importance of time, Importance of cost, Experience with the technology in the project, Level of details in the project plan, Sub-contractor dependency*. A detailed description of each feature will be explained in the data set section.

3. Method and results

In order to compare the performance of the three early-stage project predicting methods, we used a data set of 26 projects from the same organization. The organization is a high-tech company developing multi-disciplinary products, which include hardware and software integrated into a variety of products. The project types can be R&D, operational and logistic projects; each project may include a combination of these types. All of the projects used in this study were executed during the years 2018 to 2021. The approach we used to compare the three prediction methods included randomly dividing the projects into two groups. The first group includes 80% of the projects and is used for training the machine learning and regression models. The second group includes the rest of the 20% of the projects and is used as a validation set for testing and comparing all three prediction methods. The random partition for the training set and validation set was performed three times and yielded three different validation sets A, B and C. This had been done in order to reduce the bias that a single test may create. The accuracy of the prediction was determined by calculating the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE) (Willmott and Matsuura 2005) and the Mean Absolute Percent Error (MAPE) (Goodwin and Lawton 1999) which is described by equations 5, 6 and 7 respectively.

$$MAE = \frac{\sum_{i=1}^{n} |T\_true_i - T\_prediction_i|}{n} \quad (5)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (T\_true_i - T\_prediction_i)^2}{n}} \quad (6)$$

$$MAPE = \frac{\sum_{i=1}^{n} |T\_true_i - T\_prediction_i|}{n \cdot |T\_true_i|} \times 100\% \quad (7)$$

Where T_true_i represents the true duration of the project *i*, $T\_prediction_i$ represents the predicted duration of the project, *i* and *n* represent the number of projects in the data set. A lower value of MAE, RMSE and MAPE indicates a lower error which means better prediction accuracy. The use of several types of error measurements, such as MSE, RMSE and MAPE allows one to have different points of view on the accuracy. In the MSE, all the individual differences are weighted equally in the average and it is considered to be more accurate than the RMSE, when comparing errors (Willmott and Matsuura 2005). The RMSE gives a relatively high weight to large errors which gives a good indication when a large error exists. The use of MAE and the RMSE together allows one to diagnose the variation in the errors; the RMSE must be bigger or equal to the MAE; if the RMSE is equal to the MAE, then all the errors are of the same magnitude. As the difference between the RMSE and MAE is greater, the greater the variance in the individual errors in the sample. The units of both MAE and RMSE are the same time units as the project duration so it's easier to understand the estimation delay in terms of time. The MAPE is related to the percentage of the error from the true value, which allows one to also get a relative point of view on the results. The results of the comparison are summarized in Table 1. It reveals that the least accurate method for early-stage project duration predicting is the Gantt chart; the SFR accuracy was better than the Gantt chart and the MFR prediction was the most accurate. The table includes the MAE, RMSE and MAPE of the three methods for each one of the three validation sets A, B and C. As can be seen in Table 1, in all three validation sets the MFR prediction is about twice more accurate than the Gantt chart. The SFR is also consistently more accurate than the Gantt chart, but less accurate than the MLR.

Table 1. Comparison of Gannt chart, SFR and MLR

Method	Validation Set	MAE	RMSE	MAPE
Gantt Chart	A	7.17	9.11	28.34%
SFR	A	6.46	7.11	36.58%
MLR	A	3.6	4.62	14.64%
Gantt Chart	B	5.89	7.85	29.56%
SFR	B	8.04	10.52	35.61%
MLR	B	3.62	3.77	18.92%

Gantt Chart	C	12.77	14.29	36.98%
SFR	C	6.97	7.11	25.76%
MLR	C	6.26	7.53	17.33%

4. Conclusions

The use case presented in the study shows two prediction methods that produce more accurate predictions than the traditional Gantt chart prediction. The SFR and the MLR is based on records from past projects, the SFR used a single feature that is usually available for organizations and therefore the SFR can be implemented in many organizations. The MLR model prediction is more accurate than the SFR model, but required more data that is not always available in organizations. The study also shows that even a relatively low amount of recorded projects, 26 in the presented case study, is enough to significantly improve the prediction accuracy over the traditional Gantt chart. The SFR and the MLR are relatively easy to implement and can help organizations to improve their project duration prediction in the very early stages of the project. Future research can apply these methods to more projects from different organizations to gain more statistical data and find statistically how much accuracy the SFR and the MLR can achieve.

References

Alami, A. , 2016. Why do information technology projects fail. Procedia Comput. Sci. 100.

Berlin, S., Raz, T., Glezer, C. and Zviran, M., 2009. Comparison of estimation methods of cost and duration in IT projects. Information and software technology, 51(4), pp.738-748.

Burt, C.D. and Kemp, S., 1994. Construction of activity duration and time management potential. Applied Cognitive Psychology, 8(2), pp.155-168.

Gantt, H.L., 1903. A graphical daily balance in manufacture.

Goodwin, P. and Lawton, R., 1999. On the asymmetry of the symmetric MAPE. International journal of forecasting, 15(4), pp.405-408.

Hill, J., Thomas, L.C. and Allen, D.E., 2000. Experts' estimates of task durations in software development projects. International journal of project management, 18(1), pp.13-21.

Josephs, R.A. and Hahn, E.D., 1995. Bias and accuracy in estimates of task duration. Organizational Behavior and Human Decision Processes, 61(2), pp.202-213.

König, C.J., 2005. Anchors distort estimates of expected duration. Psychological Reports, 96(2), pp.253-256.

König, C.J., Wirz, A., Thomas, K.E. and Weidmann, R.Z., 2015. The effects of previous misestimation of task duration on estimating future task duration. Current Psychology, 34(1).

Lishner, I. and Shtub, A., 2019. Measuring the success of Lean and Agile projects: Are cost, time, scope and quality equally important?. The Journal of Modern Project Management, 7(1).

López-Martín, C., Chavoya, A. and Meda-Campaña, M.E., 2013, December. Use of a feedforward neural network for predicting the development duration of software projects. In 2013 12th International Conference on Machine Learning and Applications (Vol. 2, pp. 15)

Moore, D.A. and Healy, P.J., 2008. The trouble with overconfidence. Psychological review, 115(2), p.502.

Majid, I. 2006. “Causes and effect of delays in Aceh construction industry.”
Master of Science thesis, Dept. of Civil Engineering, Univ. Technology Malaysia.

Petersen, P.B., 1991. “The evolution of the Gantt chart and its relevance today”. Journal of Managerial Issues, Vol. 3, no.2, pp.131-155.

White, R.W. and Hassan Awadallah, A., 2019, January. Task duration estimation. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (pp. 636-644).

Wiest, J. D. ,1964. Some properties of schedules for large projects with limited resources. Operations research, 12(3), 395-418.

Willmott, C.J. and Matsuura, K., 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Climate research, 30(1), pp.79-82.

Wilson, J.M., 2003. Gantt charts: A centenary appreciation. European Journal of Operational Research, Vol 149, no. 2, pp.430-437.

Zareei, S. (2018). Project scheduling for constructing biogas plant using critical path method. *Renewable and Sustainable Energy Reviews, 81*, 756–759.

Solving the Assembly Line Balancing Problem with LocalSolver

Léa Blaise[1,2], Thierry Benoist[2] and Christian Artigues[1]

[1] LAAS-CNRS, Université de Toulouse, CNRS, INP, Toulouse, France
[2] LocalSolver, 24 Avenue Hoche, Paris, France
lblaise@localsolver.com

Keywords: local search, ejection chains, packing, solver.

1 Introduction and context

This paper introduces two algorithms implemented in LocalSolver, yielding great results on problems that present an ordering structure or a packing structure, such as the Assembly Line Balancing Problem.

LocalSolver is a global mathematical programming solver, whose goal is to offer a model-and-run approach to optimization problems, including combinatorial, continuous, and mixed problems, and to offer high quality solutions in short running times, even on large instances. It allows OR practitioners to focus on the modeling of the problem using a simple formalism, and then to defer its actual resolution to a solver based on efficient and reliable optimization techniques, including local search, but also linear, non-linear, and constraint programming. The local search algorithms implemented in LocalSolver are described in Gardi *et al.* (2014).

We focus here on the Assembly Line Balancing Problem, and we show how the algorithms implemented in LocalSolver produce excellent results on this problem. The Assembly Line Balancing Problem is described as follows. We consider a set of n tasks, of fixed duration, that are partially ordered by precedence relations. The problem consists in assigning the tasks to n workstations, while ensuring that the total duration of a workstation's tasks does not exceed the cycle time c. The precedence relations between the tasks impose that a task can be assigned to the same workstation as its predecessors, or to a later workstation, but cannot be assigned to an earlier workstation. The objective is to minimize the number of workstations used.

The LocalSolver model for the Assembly Line Balancing Problem is straightforward, written using set variables. In LocalSolver's formalism, a set variable of domain n is a decision variable whose value can be any (unordered) subset of $\{0, ..., n-1\}$. Each workstation S is then modeled using a set variable of domain n, whose value is equal to the set of tasks it contains. Since each task must be assigned to exactly one workstation, we constrain the set variables to form a partition of $\{0, ..., n-1\}$.

2 Greedy algorithm

In this Section, we introduce a greedy algorithm, working both as an initialization algorithm and as a "destroy and repair" local move, taking the capacity and precedence constraints into account. The implementation of this algorithm being as generic as possible, it is not only applicable to the Assembly Line Balancing Problem, but to any problem presenting an ordering structure and a packing structure on its set variables.

2.1 Building an initial feasible solution

In the LocalSolver model for the Assembly Line Balancing Problem, the precedence constraint between two tasks `t1` and `t2` is written:

```
constraint find(stations, t1) <= find(stations, t2);          (1)
```

The `find` operator takes two arguments: an array of set variables of domain n and an integer expression between 0 and $n-1$, and returns the index of the set variable which contains the requested element (or -1 if none of the set variables contain the element). The constraint written above then reads "The chosen workstation for task `t1` must be lower or equal to the chosen workstation for task `t2`".

When such precedence constraints are detected in the model, an ordering on the set variables can be deduced. We can then build an initial solution verifying the precedence constraints between the tasks, as well as the capacity constraints on the workstations. The initialization algorithm is as follows. Let $S_0, ..., S_{n-1}$ be the set variables of the model (in increasing order). Each set variable is initially empty. An element $0 \leq t \leq n-1$ (representing a task in the case of the Assembly Line Balancing Problem) is said to be eligible for insertion into a set variable S if all of its predecessors have already been assigned to a set variable, and if its weight (the task's duration) is lower than the remaining space in S (its capacity minus the total weight of the elements it already contains). Let E be the set of eligible elements. The first set to be filled is $S = S_0$. While the set of eligible elements is non empty, an element $t \in E$ is randomly chosen, and inserted into S. The set of eligible elements is then updated accordingly. When there is no eligible element anymore, we move on to the next set variable, until every element has been assigned to a set variable. The complexity of this algorithm is $O(n^2)$.

This algorithm allows LocalSolver to immediately obtain a first feasible solution for any instance of the Assembly Line Balancing Problem. This is particularly useful on the large instances, for which finding a feasible solution from a random assignment of the set variables' elements could take several seconds.

2.2 "Destroy and repair" local move

The greedy algorithm described in 2.1 can be adapted to be integrated into LocalSolver's local search component as a "destroy and repair" local move. Indeed, we can choose to apply it to fill a subset of the model's set variables rather than all of them. For example, in the case of the Assembly Line Balancing Problem, we can choose to apply it not to distribute the whole set of tasks into the different workstations, but to reorganize the tasks already assigned to a subset of workstations only. When applying this local move, the current solution is re-optimized by destroying and then rebuilding part of the solution.

To further diversify the explored solutions, the algorithm can either by applied by increasing order of the set variables (as in the initialization algorithm described in 2.1), or by decreasing order of the set variables.

3 Packing move based on ejection chains

In this Section, we describe another local move implemented inside LocalSolver's local search component, which makes use of the packing structure detected in the model (the total duration of a workstation's tasks must be lower than the cycle time). This local move is based on ejection chains: it consists in a series of element movements from one set variable to another. The move is similar to one of the algorithms described by Capua *et al.* (2018) to solve the Bin Packing Problem with Conflicts. The authors also describe a local

move based on ejection chains, with several differences. In their algorithm, all the sets and elements of the problem are considered, and a random ordering of the sets is imposed (an element can only be moved from its current set to another set of higher index). On the contrary, our local move only focuses on a subset of unordered set variables.

The local move's procedure is as follows. We start by choosing a random subset of the model's set variables. Let S be the set variable with the smallest weight among all chosen sets. An element $t \in S$ is randomly chosen to be ejected from S. The goal of the local move is to reorganize the other selected set variables' elements, so that t can be inserted into one of them. If t can be inserted into a new set variable $S^\star$ without violating the capacity constraint on $S^\star$, t is assigned to $S^\star$. The move is successful. If not, we try to swap it with another, strictly smaller element. In order to do this, we consider the smallest element $t' \notin S$ that can be replaced by t. If no such element exists, or if the weight of t' is larger than that of t, the local move results in failure, and we revert back to the previous solution. Otherwise, t' is ejected from its current set variable S', so that t can be inserted in its place. The same procedure is then repeated, with t' as the new current element, until the move ends in either success or failure. Since the weights of the ejected elements are strictly decreasing, the move ends in at most N steps, where N is the total number of elements in the selected subset of set variables. The complexity of each step is $O(N)$.

Figure 1 illustrates the local move's procedure on a small example with eight elements distributed into five set variables. The first element to be moved is 7, then 0, then 4, and finally 6.

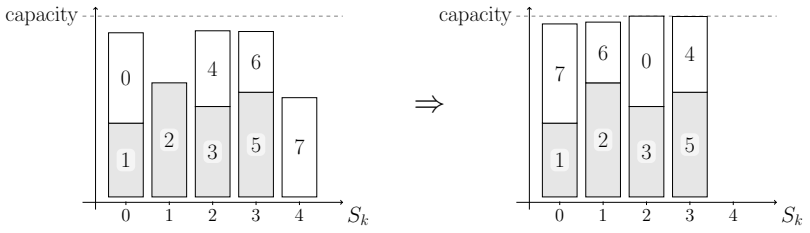


Fig. 1. Example – Solutions before (left) and after the local move (right)

This local move is particularly effective on the most combinatorial instances, in which each set variable contains very few elements. Indeed, if the smallest set variable selected contains only one element, and if the move is successful, the solution is improved (assuming that the number of used set variables is to be minimized). However, it is also useful on other types of instances, since it increases the weight gaps between the different set variables, which makes it easier to find improvements in the following iterations.

This local move can be applied to any problem presenting a packing structure. It improves LocalSolver's performance on various problems, such as the Bin Packing Problem, and the Assembly Line Balancing Problem.

4 Numerical results

We compare the performance of LocalSolver 10.5, CP Optimizer 20.1.0, and Gurobi 9.1, on the Assembly Line Balancing Problem, in 60 and 600 seconds of running time. The models we use to evaluate the performance of CP Optimizer and Gurobi are respectively given by Laborie (2020) and Pastor *et al.* (2007). We use the instances of the "very large" dataset proposed by Otto *et al.* (2013). The dataset includes 525 instances of 1000 tasks

to assign. The metric used is the gap to the best known solution, equal to the minimum between the best known solution given in (Otto *et al.* 2013) and the value yielded by LocalSolver after 10 minutes of running time. Indeed, LocalSolver improves the result given by the authors of the article on 59% of the instances. In Table 1, we give the percentage of feasible instances, the percentage of instances for which each solver reaches a solution within 1% of the best known solution, as well as the average gap.

Table 1. Comparison of LocalSolver's, CP Optimizer's, and Gurobi's performance – 1000 tasks

	LocalSolver		CP Optimizer		Gurobi	
	60s	600s	60s	600s	60s	600s
Feasible instances	100%	100%	100%	100%	0%	0%
Instances w. gap $< 1\%$	95%	99%	59%	64%	0%	0%
Average gap	0.4%	0.1%	2.1%	1.7%	–	–

We can see that LocalSolver's performance is significantly better than that of the other two solvers. The performance gap between LocalSolver and CP Optimizer is particularly striking on the most combinatorial instances, which are considered to be the most difficult.

The algorithms we describe here are however not dedicated to the Assembly Line Balancing problem, and can improve LocalSolver's performance on other problems as well. For example, the integration of the local move described in 3 into LocalSolver's local search enables it to get strictly better results on 58% of the very hard Bin Packing instances proposed by Gschwind and Irnich (2016), lowering the average gap to the best known lower bound from 0.44% to 0.36%.

5 Conclusion

In this paper, we considered a family of problems presenting a packing structure, such as the Bin Packing Problem, and an ordering structure, such as the Assembly Line Balancing Problem. We introduced a greedy algorithm, making use of both kinds of structures to build feasible solutions. We showed how it can be used both as an initialization algorithm and as a "destroy and repair" local move. We also introduced a packing move based on ejection chains, particularly efficient on the most combinatorial packing instances. Their integration into LocalSolver enables it to obtain great results on the targeted problems.

References

Gardi F., T. Benoist, J. Darlay, B. Estellon, and R. Megel, 2014, Mathematical Programming Solver Based on Local Search, Wiley.

Alena Otto and Christian Otto and Armin Scholl. Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing, 2013, *European Journal of Operational Research*.

Timo Gschwind and Stefan Irnich. Dual inequalities for stabilized column generation revisited, 2016, *INFORMS Journal of Computing*.

Renatha Capua, Yuri Frota, Luiz Satoru Ochi, and Thibaut Vidal. A study on exponential-size neighborhoods for the bin packing problem with conflicts, 2018, *Journal of Heuristics*.

Philippe Laborie. Solving the Simple Assembly Line Balancing Problem with CP Optimizer, 2020, https://www.linkedin.com/pulse/solving-simple-assembly-line-balancing-problem-cp-philippe-laborie/.

Pastor, Rafael and Ferrer, Laia and García, Alberto. Evaluating optimization models to solve SALBP, 2007, *Lecture Notes in Computer Science*.

A time-based schedule-generation scheme for project scheduling with storage resources

Mario C. Sillus and Christoph Schwindt

Clausthal University of Technology, Germany
{mario.christian.sillus,christoph.schwindt}@tu-clausthal.de

Keywords: schedule-generation scheme, storage resources, dynamic release dates, generalized precedence relations.

1 Introduction

Storage resources model material stocks or liquid funds, which are depleted and replenished at the occurrence of certain events during the execution of a project. Both renewable and nonrenewable resources are special cases of storage resources. Project scheduling subject to storage-resource constraints and generalized precedence relations consists in sequencing the events in such a way that the inventories of the storage resources evolve within given bounds and prescribed minimum and maximum time lags between the events are met.

The problem was introduced by Neumann and Schwindt (2002), who addressed structural issues, generated benchmark data sets, and devised a branch-and-bound algorithm enumerating disjunctions of precedence relations. Laborie (2003) presents consistency tests and an effective constrained-based branch-and-bound algorithm solving all remaining open instances. Carlier *et al.* (2009) consider a special case with a single resource of infinite capacity. They report on complexity results and propose polynomial-time algorithms to compute an optimal schedule for a given sequence of events. Carlier *et al.* (2018) propose tight lower bounds for the instances of Neumann and Schwindt.

In this paper, we present a time-based schedule-generation scheme that is intended to serve as a building block for metaheuristic schedule-improvement procedures for large problem instances. The method decodes an event list into a feasible schedule by iteratively resolving resource conflicts.

The remainder of this paper is organized as follows. In Sect. 2 we provide a conceptual model formulation and briefly review structural properties of the problem. The basic schedule-generation scheme and several enhancements are developed in Sect. 3. In Sect. 4 we report on computational results obtained on a standard data set from literature.

2 Problem statement

We consider a project employing several storage resources $k \in \mathcal{R}$. During project execution, the inventory level of each resource k must remain within a given nonempty interval $[\underline{R}_k, \overline{R}_k]$ with $\underline{R}_k \in \mathbb{Z} \cup \{-\infty\}$ and $\overline{R}_k \in \mathbb{Z} \cup \{\infty\}$. The inventory levels of resources $k \in \mathcal{R}$ change upon occurrences of events $i = 1, \ldots, n$, which typically coincide with project milestones or starts and completions of project activities. Upon occurrence of event i, the inventory levels of resources k change by $r_{ik} \in \mathbb{Z}$ units. We say that i replenishes the inventory of k if $r_{ik} > 0$ and i depletes the inventory of k if $r_{ik} < 0$. The set V of all events also contains two fictitious events $i = 0$ and $i = n + 1$ standing for the project start and termination, respectively, where r_{0k} is the opening stock level of resource k and without loss of generality $r_{(n+1)k} = 0$. For certain pairs $(i, j) \in A$ of events $i, j \in V$, a minimum time lag $\delta_{ij} \in \mathbb{Z}$ between the occurrences of i and j is prescribed. If $\delta_{ij} < 0$, the value $-\delta_{ij}$ can be viewed as maximum time lag between events j and i. The project scheduling problem $(PSc|temp|C_{max})$ under consideration consists in assigning occurrence times $t_i \geq 0$ to all events $i \in V$ in such a way that (1) a given regular (*i. e.*, componentwise nondecreasing)

objective function f in vector $\boldsymbol{t} = (t_i)_{i \in V}$ is minimized, (2) the resource constraints arising from the storage resources and (3) the generalized precedence relations defined by the time lags are satisfied, and (4) the project is started at time 0.

$$(PSc|temp|f) \quad \begin{cases} \text{Min.} & f(\boldsymbol{t}) & & (1) \\ \text{s. t.} & \underline{R}_k \leq \sum\limits_{i \in V : t_i \leq t_j} r_{ik} \leq \overline{R}_k & (j \in V; k \in \mathcal{R}) & (2) \\ & t_j - t_i \geq \delta_{ij} & ((i,j) \in A) & (3) \\ & t_0 = 0 & & (4) \end{cases}$$

Problem $PSc|temp|f$ generalizes the classical problem $PS|temp|f$ with renewable resources, whose feasibility variant is known to be strongly NP-hard. In difference to the case of renewable resources, finding a feasible schedule for the more general problem with storage resources remains NP-hard even if $|\mathcal{R}| = 1$ and $\delta_{ij} > 0$ for all $(i,j) \in A$.

3 Schedule-generation scheme

Let Π be the set of all precedence-feasible permutations π on set V, *i. e.*, the set of all event lists $\pi = (0, i_1, \ldots, i_n, n+1)$ with $\lambda < \mu$ if $d_{ij} \geq 0$ and $d_{ji} < 0$ for $i = i_\lambda$ and $j = i_\mu$. By $\mathcal{S}'$ we denote the set $\mathcal{S}$ of all feasible schedules $\boldsymbol{t}$ plus an infeasible schedule $\boldsymbol{t}^\infty$ serving to indicate that no feasible schedule could be found. Basically, a schedule-generation scheme (SGS) is a mapping $\sigma : \Pi \to \mathcal{S}'$ assigning a schedule $\boldsymbol{t}' \in \mathcal{S}'$ to each permutation $\pi \in \Pi$.

3.1 Basic scheme

Traditional schedule-generation schemes, like the serial SGS for problems with renewable resources, schedule the activities one by one in the order given by permutation π. In each iteration, the respective partial schedule represents a feasible solution to the problem defined on the set of activities scheduled thus far. Deadlocks caused by maximum time lags are resolved using unscheduling techniques. Given that preserving the feasibility of a partial schedule would generally require the simultaneous addition of several events and that finding such a set constitutes an NP-hard problem, we opt for a different approach, which draws from an enumeration scheme with dynamic activity release dates devised by Fest *et al.* (1998) for the project duration problem $PS|temp|C_{max}$. The basic idea consists in first relaxing the resource constraints and then iteratively resolving inventory shortfalls or excesses by defining release dates δ_{0j} for appropriate events j, which are determined by their position in π. The principle to (pre-)select the events to be postponed via a permutation can be interpreted as a linear preselective strategy (Stork 2001).

Our time-based SGS is displayed in Algorithm 1. Let $D = (d_{ij})_{i,j \in V}$ denote the distance matrix containing the transitive time lags d_{ij} implied by prescribed time lags δ_{ij}. In each iteration of the SGS, we consider the current earliest schedule $\boldsymbol{et} = (d_{0i})_{i \in V}$ with $t_0 = 0$ and satisfying all time lags δ_{ij} for $(i,j) \in A$ and release dates δ_{0j} introduced so far. Schedule $\boldsymbol{et}$ is then scanned for the earliest time t at which a resource conflict occurs, *i. e.*, $r_k(\boldsymbol{et}, t) := \sum_{i \in V : et_i \leq t} r_{ik} \notin [\underline{R}_k, \overline{R}_k]$ for some k. If $\underline{R}_k \leq r_k(\boldsymbol{et}, t) \leq \overline{R}_k$ for all $k \in \mathcal{R}$ and all $t \geq 0$, the SGS is terminated by returning feasible schedule $\boldsymbol{et}$. Otherwise, we compute the set $\mathcal{C}$ of all events $j \in V$ that contributed to the resource conflict on k. To resolve the conflict, at least one event $j \in \mathcal{C}$ has to be delayed to the earliest occurrence time $et_i > t$ of an event i with the opposite resource requirement, *i. e.*, $r_{ik} \cdot r_{jk} < 0$. Event j can be synchronized with event i precisely if $d_{ji} \leq 0$. The event that is actually deferred is the last element $j \in \mathcal{C}$ on list π possessing such a partner event i. If no such event j exists, the SGS is stopped and infeasible schedule $\boldsymbol{t}^\infty$ is returned. Otherwise, the new release date δ_{0j} is set to $t_j^* := \min\{et_i \mid i \in V,\ et_i > t,\ r_{ik} \cdot r_{jk} < 0,\ d_{ji} \leq 0\}$. Finally, δ_{0j} is added to distance matrix D by updating $d_{0l} := \max\{d_{0l}, t_j^* + d_{jl}\}$ for all events $l \in V$.

Algorithm 1 Basic schedule-generation scheme SGS(π)

1: compute distance matrix $D = (d_{ij})_{i,j \in V}$ of time lags $(\delta_{ij})_{(i,j) \in A}$;
2: **loop**
3: set earliest schedule $\boldsymbol{et} := (d_{0i})_{i \in V}$;
4: determine earliest time t at which a resource conflict occurs on some resource $k \in \mathcal{R}$;
5: **if** $t = \infty$ **then return** $\boldsymbol{et}$; ($*$ feasible schedule has been generated $*$)
6: **if** $r_k(\boldsymbol{et}, t) < \underline{R}_k$ **then** set conflict set $\mathcal{C} := \{j \in V \mid et_j \leq t,\ r_{jk} < 0\}$;
7: **else** set conflict set $\mathcal{C} := \{j \in V \mid et_j \leq t,\ r_{jk} > 0\}$;
8: determine last event $j \in \mathcal{C}$ on π with $t_j^* := \min_{i \in V}\{et_i \mid et_i > t,\ r_{ik} \cdot r_{jk} < 0,\ d_{ji} \leq 0\} < \infty$;
9: **if** there is no such event j **then return** $\boldsymbol{t}^\infty$; ($*$ no feasible schedule could be found $*$)
10: **else** put $d_{0l} := \max\{d_{0l}, t_j^* + d_{jl}\}$ for all $l \in V$; ($*$ add $\delta_{0j} = t_j^*$ and update D $*$)

3.2 Expansions

E1: Randomization. Given that we add release dates $\delta_{0j} = t_j^*$ and not precedence relations $\delta_{ij} = 0$, the SGS can encounter a so-called leapfrogging phenomenon that may cause cycling in an infinite loop of mutual shifting among three or more events. To get out of leapfrogging, it proves expedient to introduce a pinch of randomness when selecting event j. More precisely, let $\mathcal{C}' := \{j \in \mathcal{C} \mid t_j^* < \infty,\ (d_{jj'} < 0) \vee (d_{j'j} \geq 0) \text{ for all } j' \in \mathcal{C}\}$ be the set of candidates to deferment. Starting with the last $j \in \mathcal{C}'$ in π, we accept j as the event to be delayed with probability $p < 1$. If j was rejected, we recursively proceed with the preceding $j \in \mathcal{C}'$ in π until some $j \in \mathcal{C}'$ was accepted, where we return to the last $j \in \mathcal{C}'$ in π if the first event $j \in \mathcal{C}'$ was rejected. The number of iterations follows geometric distribution $Geo(p)$. Consequently, it suffices to draw a random number z from distribution $Geo(p)$ and to select the mth element $j \in \mathcal{C}'$ in π with $m := |\mathcal{C}'| - (z-1) \mod |\mathcal{C}'|$.

E2: Preservation of feasible initial inventories. If the problem instance is feasible and no deadlines $-d_{j0} < \infty$ are imposed on the occurrence of events $j \in V$, Algorithm 1 can only be quitted without feasible schedule if the initial inventory level $r_k(\boldsymbol{et}, 0)$ is not within the bounds $\underline{R}_k$ and $\overline{R}_k$. The reason is that due to constraint $t_0 = 0$, conflicts at time $t = 0$ may become unsolvable if $r_{0k} \notin [\underline{R}_k, \overline{R}_k]$. If such a resource k with infeasible opening stock exists, we should avoid right-shifting any event l with $et_l = 0$ while removing an infeasibility at time $t > 0$. When performing the update of distance matrix D, delaying event j to time t_j^* leads to an increase of d_{0l} exactly if $t_j^* + d_{jl} > et_l$. Accordingly, we impose the additional condition $t_j^* \leq \min\{-d_{jl} \mid l \in V : et_l = 0\}$ on the selection of event j, provided that at least one event from set $\mathcal{C}'$ satisfies this condition.

E3: Schedule contraction. Despite the additional condition introduced in expansion E2, we may still encounter situations in which there does not exist any $j \in \mathcal{C}'$ when dealing with a conflict at time $t = 0$. Since in general $r_{0k} \neq 0$, it might be useful to synchronize $j = 0$ with the next appropriate event i for solving the conflict. In the basic SGS, however, condition $d_{ji} \leq 0$ prevents $j = 0$ from being moved because $et_i = d_{0i} > 0$ contradicts $d_{ji} = d_{0i} \leq 0$. Instead of delaying $j = 0$, the schedule contraction technique performs an equivalent relative movement between events by left-shifting event i and further events l. To this end, we put $d_{0l} := \max\{\hat{d}_{0l}, d_{0l} - et_i\}$ for all $l \in V$, where $\hat{d}_{0l}$ stands for the initial earliest occurrence times computed on line 1 of Algorithm 1. Schedule contraction allows us to entirely avoid premature terminations of the SGS.

E4: Postprocessing. Randomizing the selection of event j reliably prevents long leapfrogging phases. Nevertheless, leapfrogging cannot be avoided completely, and thus the schedule $\boldsymbol{et}$ yielded by the SGS often fails to be quasiactive (Neumann *et al.* 2003, Sect. 2.4). In this case, $\boldsymbol{et}$ can be further compressed by applying the following postprocessing procedure. For each pair (i, j) with $et_j \geq et_i$ and $r_{ik} \cdot r_{jk} < 0$ for some $k \in \mathcal{R}$ we put $\hat{d}_{ij} := \max\{\hat{d}_{ij}, 0\}$ and restore the transitivity of matrix $\hat{D}$ by applying the Floyd-Warshall algorithm. Since $\boldsymbol{et}$ is resource-feasible, any schedule $\boldsymbol{t}$ satisfying the temporal constraints $t_j \geq t_i + \hat{d}_{ij}$ is feasible as well. In particular, this holds true for the compressed earliest schedule $\boldsymbol{et}' = (\hat{d}_{0j})_{j \in V} \leq \boldsymbol{et}$.

4 Computational experiments

We tested the performance of different versions of the SGS using the solvable instances of the project duration problem with $n = 100$ events and 5 storage resources from Neumann and Schwindt (2002). The test set contains 90 randomly generated instances, 57 of which possess a feasible solution. We considered three types of precedence-feasible permutations π: the ***et***-list, which orders the events according to nondecreasing earliest occurrence times, the ***lt***-list, where events are arranged in order of nondecreasing latest occurrence times, and randomly generated lists. In total, we carried out the four experiments listed in Table 1.

Experiments A to D stepwise introduce extensions E1 to E3 into the basic SGS with postprocessing E4. For each instance, 100 executions of the SGS are distributed over the list types as shown in the second column of the table. The CPU time limit was set to 10 ms per execution, and the smallest project duration found in previous runs was defined as a deadline. While the randomization in experiment A comes from outside the SGS via the event lists, the randomization in B to D is implemented inside the SGS via acceptance probability $p < 1$ (we chose $p = 0.4$). The SGS was implemented in Java 11 under Eclipse and ran on a PC with one core and 4 GHz clock pulse operating under Windows 10.

In the right part of Table 1 we list the mean percentages p_{opt}, p_{feas}, and p_{no} of optimally solved, feasibly, but not optimally solved, and unsolved instances averaged over ten replications for each experiment. Δ_{opt} stands for the relative optimality gap of the instances solved to feasibility by the respective SGS version. All percentages refer to the best schedules found for each instance. Symbol t_{cpu} denotes the mean CPU time for 100 runs per instance in seconds.

Table 1. Experimental design and computational results

Experiment	`#et/#lt/#rand`	Expansions	p_{opt} [%]	p_{feas} [%]	p_{no} [%]	Δ_{opt} [%]	t_{cpu} [s]
A	1/1/98	E4	64.4	23.1	12.4	1.12	0.34
B	50/50/0	E1, E4	80.0	10.9	9.1	0.32	0.12
C	50/50/0	E1, E2, E4	91.4	5.8	2.8	0.29	0.16
D	50/50/0	E1 – E4	93.0	5.6	1.5	0.41	0.25

The successive additions of extensions consistently lead to higher percentages of optimally and of feasibly solved instances. In particular, the fully equipped SGS achieves good results with more than 90 % of instances solved to optimality and a mean optimality gap of 0.24 %, while the computational effort remains in reasonable order of magnitude.

In a further experiment we ran the SGS with expansions E1, E3 and the ***et*** and ***lt*** lists without time limit. As expected, the method delivered a feasible solution to each instance, which averaged over ten replications took 2.77 s for the ***et*** and 0.9 s for the ***lt*** list.

References

Carlier J., A. Moukrim, H. Xu, 2009, "The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm", *Discrete Appl Math*, Vol. 157, pp. 3631–3642.

Carlier J., A. Moukrim, A. Sahli, 2018, "Lower bounds for the event scheduling problem with consumption and production of resources", *Discrete Appl Math*, Vol. 234, pp. 178–194.

Fest A., R. H. Möhring, F. Stork, M. Uetz, 1998, "Resource-constrained project scheduling with time windows: A branching scheme with dynamic release dates", *Working Paper 596, Fachbereich Mathematik, TU Berlin*.

Laborie P., 2003, "Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results", *Artif Intell*, Vol. 143, pp. 151–188.

Neumann, K., C. Schwindt, 2002, "Project scheduling with inventory constraints", *Math Method Oper Res*, Vol. 56, pp. 513–533.

Neumann, K., C. Schwindt, J. Zimmermann, 2003, "Project Scheduling with Time Windows and Scarce Resources". Springer, Berlin

Stork, F., 2001, "Stochastic Resource-Constrained Project Scheduling". PhD Thesis, TU Berlin

Logic-based Benders Decomposition for preemptive Flexible Job-Shop Scheduling

Carla Juvin[1], Laurent Houssin[1,2], and Pierre Lopez[1]

[1] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
{carla.juvin,pierre.lopez}@laas.fr
[2] ISAE-SUPAERO, Université de Toulouse, France
laurent.houssin@isae-supaero.fr

Keywords: Flexible job-shop scheduling, preemption, Benders decomposition.

1 Introduction

In this paper, we address the preemptive flexible job-shop scheduling problem (pFJSSP). We propose three exact methods to solve the pFJSSP with makespan minimization objective: a Mixed Integer Linear Programming (MILP) formulation, a Constraint Programming (CP) formulation, and a Logic-Based Benders Decomposition (LBBD) algorithm.

The job-shop scheduling problem (JSSP) is one of the most studied NP-hard optimization problems. It consists of a set of jobs and a set of machines. Each job has a sequence of operations, each of which must be performed on a given machine. In order to satisfy present market, the production environment becomes more and more complex and flexible. The flexible job-shop problem(FJSSP) is an extension of the classical JSSP that allows an operation to be processed on any machine from a set of eligible machines.

The FJSSP has received considerable attention and both metaheuristics and exact methods have been developed to solve this problem (Brandimarte (1993), Shen *et al.* (2018)). Moreover, preemption is another important parameter when dealing with scheduling problems as it can have a positive impact on the objective function. The preemptive job-shop scheduling problem (pJSSP) has received limited attention and most of literature focus on approximation algorithms rather than exact methods. Among the exact methods, Le Pape and Baptiste (1998) develop a constraint programming approach, Ebadi and Moslehi (2013) model the pJSSP with a disjunctive graph and develop a branch-and-bound algorithm. To the best of our knowledge, only one study (Zhang and Yang (2016)) considers both preemption and flexibility for the JSSP. However, the problem addressed in this work has special characteristics (flexible workdays and overlapping in operations) that are not considered here.

Problem statement. An instance of the pFJSSP is defined as a set of n jobs $\mathcal{J} = \{1, \dots, n\}$ and a set of machines $\mathcal{M}$. Each job i consists of a sequence of n_i operations $\mathcal{O}_i = (i_1, \dots, i_{n_i})$. An operation $O_{i,j} \in \mathcal{O}_i$ of a job i must be performed by one of the machine from the set of eligible machines $\mathcal{M}_{i,j} \subseteq \mathcal{M}$. Let $p_{i,j,m}$ denote the processing time of operation $O_{i,j}$ that is processed on machine $m \in \mathcal{M}_{i,j}$. Each machine can process at most one operation at a time and preemption is allowed: the processing of operations can be interrupted and resumed later without penalty. Although an operation can be interrupted, it is assumed that it must be fully processed by one and the same machine.

2 Mathematical and Constraint programming models

Let us introduce a model that can be used as a mixed-integer program to solve the pFJSSP. It is based on a time-indexed formulation proposed by Bowman (1959) to solve the

preemptive job-shop problem. We have adapted it to add the notion of resource flexibility. Let $\mathcal{H} = \{1, 2, 3, \ldots, h\}$ be the time horizon. The (binary) decision variables are defined as follows:

- $x_{i,j,m}$ is equal to 1 if operation $O_{i,j}$ is processed on machine m;
- $y_{i,j,t}$ is equal to 1 if operation $O_{i,j}$ is in process at time t,

and the model:

$$\min C_{\max} \tag{1}$$

$$s.t. \quad \sum_{m \in \mathcal{M}_{i,j}} x_{i,j,m} = 1 \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i \tag{2}$$

$$\sum_{t=1}^{h} y_{i,j,t} \geq \sum_{m \in \mathcal{M}_{i,j}} x_{i,j,m} \times p_{i,j,m} \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i \tag{3}$$

$$\sum_{t'=t}^{h} y_{i,j,t'} \leq \max_{m \in \mathcal{M}_{i,j}} p_{i,j,m} \times (1 - y_{i,j+1,t}) \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i \setminus \{O_{i,n_i}\},\, t \in \mathcal{H} \tag{4}$$

$$\sum_{i \in \mathcal{J}} \sum_{j=1}^{n_i} x_{i,j,m} \times y_{i,j,t} \leq 1 \quad \forall m \in \mathcal{M},\, t \in \mathcal{H} \tag{5}$$

$$C_{\max} \geq (t+1) \times \sum_{i \in \mathcal{J}} y_{i,n_i,t} \quad \forall t \in \mathcal{H} \tag{6}$$

$$x_{i,j,m} \in \{0, 1\} \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i,\, m \in \mathcal{M}_{i,j} \tag{7}$$

$$y_{i,j,t} \in \{0, 1\} \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i,\, t \in \mathcal{H} \tag{8}$$

Note that Constraints (5) are nonlinear, but can be easily linearized since variables $x_{i,j,m}$ and $y_{i,j,t}$ are binary, the previous mathematical model thus becoming a MILP model.

Moreover, we propose another formulation of the problem using Constraint Programming. It is based on a formulation proposed by Polo-Mejía *et al.* (2020) to solve the Multi-Skill Project Scheduling Problem with partial preemption. We have adapted it to solve the pFJSSP. We introduce the following decision variables: $I_{i,j}$ represents the interval variable between the start and the end of the processing of operation $O_{i,j}$. Since activities can be performed in multiple machines, we introduce an optional interval variable $mode_{i,j,m}$ for each possible combination of an operation $O_{i,j}$ and a machine m. Each operation $O_{i,j}$ is divided into $p_{i,j,m}$ parts of unit duration, the optional variable $part_{i,j,k,m}$ representing the interval during which the k^{th} part of the operation $O_{i,j}$ is processed if it is executed on machine m.

$$\min C_{\max} \tag{9}$$

$$s.t. \quad C_{\max} \geq I_{i,n_i}.end \quad \forall i \in \mathcal{J} \tag{10}$$

$$endBeforeStart(I_{i,j}, I_{i,j+1}) \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i \setminus \{i_{n_i}\} \tag{11}$$

$$endBeforeStart(part_{i,j,k,m}, part_{i,j,k+1,m}) \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i,\, m \in \mathcal{M}_{i,j},\, k \in 1, \ldots, p_{i,j,m} - 1 \tag{12}$$

$$span(mode_{i,j,m}, part_{i,j,k,m} : \forall k \in 1, \ldots, p_{i,j,m}) \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i,\, m \in \mathcal{M}_{i,j} \tag{13}$$

$$alternative(I_{i,j}, mode_{i,j,m} : \forall m \in \mathcal{M}_{i,j}) \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i \tag{14}$$

$$presenceOf(mode_{i,j,m}) \Rightarrow presenceOf(part_{i,j,k,m}) \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i,\, m \in \mathcal{M}_{i,j},\, k \in 1, \ldots, p_{i,j,m} \tag{15}$$

$$noOverlap(part_{i,j,k,m} : \forall i \in \mathcal{J},\, j \in 1, \ldots, n_i,\, k \in 1, \ldots, p_{i,j,m}) \quad \forall m \in \mathcal{M} \tag{16}$$

3 Logic-based Benders decomposition

A logic-based Benders decomposition approach (Hooker (2007)) is proposed to solve the problem under consideration. It consists in dividing the problem into a machine assignment master problem and a preemptive job-shop scheduling subproblem.

The master problem is an assignment problem, each operation needs to be assigned to a machine. The decision variable $x_{i,j,m}$ is equal to 1 if operation $O_{i,j}$ is processed on machine m. We propose the following MILP model:

$$\min C_{\max} \tag{17}$$

$$\sum_{m \in M} x_{i,j,m} = 1 \quad \forall i \in \mathcal{J},\, O_{i,j} \in \mathcal{O}_i \tag{18}$$

$$\text{Benders' cuts} \tag{19}$$

$$x_{i,j,m} \in \{0,1\} \tag{20}$$

However, at first iteration as there is no Benders cut, there is no link between assignment variables $x_{i,j,m}$ and the objective value $C_{\max}$. That is why we propose to include a subproblem relaxation in the master problem. The subproblem relaxation is based on the following three ideas:

1. The makespan is at least equal to the sum of the processing times of the operations of the same job.
2. The makespan is at least equal to the sum of the processing times of the operations assigned to the same machine.
3. Let consider a subset of operations assigned to the same machine. The makespan is at least equal to the following quantity: the minimum release date of the subset plus the sum of the processing times of the operations of the subset plus the minimum delivery time of the job of the last processed operation of this subset.

At each iteration we obtain a feasible solution $P_h = \{(i,j,m) \,|\, x_{i,j,m} = 1\}$ of the master problem which contains all assignments. Then, the subproblem consists in a pJSSP, which can be solved by two different methods:

- a CP model similar to the one proposed for the pFJSSP, but for which the set of eligible machines $M_{i,j}$ for each operation $O_{i,j}$ is reduced to the machine m assigned by the master problem $(i,j,m) \in P^h$;
- a Branch-and-Bound algorithm proposed by Ebadi and Moslehi (2013).

When the subproblem is solved, we obtain the optimal makespan $C^h_{\max}$ for a given assignment P^h. We can deduce the following cut:

$$C_{\max} \geq C^h_{\max}\Big(1 - \sum_{(i,j,m) \in P^h} (1 - x_{i,j,m})\Big) \tag{21}$$

At each iteration, this cut is added to the master problem as a Benders cut, and the master problem is resolved with this new constraint.

4 Numerical results and Conclusions

For computational tests, we use CPLEX for solving the MILP model and CP Optimizer for the CP model. The computation time was limited to 10 minutes. We use classical instances for the FJSSP[3]. Table 1 shows the results. The first column shows the benchmarks

Table 1. Number of pFJSSP instances proved to be optimal within 10 min CPU

Benchmark	MILP	CP	LBBD (sp:CP)	LBBD (sp:B&B)
Brandimarte (15)	0	5	9	9
Hurink edata (66)	1	2	9	24
Hurink rdata (66)	1	1	1	20
Hurink vdata (66)	1	2	7	25
DPPaulli (18)	0	0	0	0
ChambersBarnes (21)	0	0	0	0
Kacem (4)	3	3	4	4
Fattahi (20)	8	12	13	17

under study (and the number of instances they contain), the following ones the number of instances solved to optimality using the methods described in the previous sections.

According to our computational experiments, CP method is superior to the MILP one and solves more instances to optimality. On the other hand, we can see that the pFJSSP benefits from the decomposition since LBBD methods are more efficient compared to the others. More specifically, solving the subproblem using Branch-and-Bound (fifth column) is faster than using CP (fourth column) and thus enables to solve a greater number of instances. We also notice that for the most difficult benchmarks (DPPaulli and Chambers-Barnes) no instance could be solved optimally by any of these methods within 10 minutes.

Conclusion. We have proposed three methods to solve the pFJSSP where the objective is the minimisation of the makespan: a mathematical model, a constraint programming model, and a logic-based Benders decomposition algorithm. Numerical results have shown that the MILP becomes inefficient for difficult instances. Also, our logic-based Benders decomposition outperforms mathematical programming and constraint programming to find optimal solutions. Our current work consists in improving the solving methods, in particular the decomposition ones to further increase the number of solved instances.

References

Brandimarte P., 1993, "Routing and scheduling in a flexible job shop by tabu search", *Annals of Operations Research*, Vol. 41, pp. 157-183.

Bowman E.H., 1959, "The schedule-sequencing problem", *Operations Research*, Vol. 07, pp. 621-624.

Ebadi A. and Moslehi G., 2013, "An optimal method for the preemptive job shop scheduling problem", *Computers & Operations Research*, Vol. 40, pp. 1314-1327.

Hooker J., 2007, "Planning and scheduling by logic-based Benders decomposition", *Operations Research*, Vol. 55, pp. 588-602.

Le Pape C. and Baptiste P., 1998, "Resource constraints for preemptive job-shop scheduling", *Constraints: An international journal*, Vol. 03, pp. 263-287.

Polo-Mejía O., Artigues C., Lopez P., and Basini V., 2020, "Mixed-integer/linear and constraint programming approaches for activity scheduling in a nuclear research facility", *International Journal of Production Research*, Vol. 58, pp. 7149-7166.

Shen L., Dauzère-Pérès S. and Neufeld J. S., 2018, "Solving the flexible job shop scheduling problem with sequence-dependent setup times", *European Journal of Operational Research*, Vol. 265, pp. 503-516.

Zhang J. and Yang, J. 2016, "Flexible job-shop scheduling with flexible workdays, preemption, overlapping in operations and satisfaction criteria: an industrial application", *International Journal of Production Research*, Vol. 54, pp. 4894-4918.

[3] `http://opus.ub.hsu-hh.de/volltexte/2012/2982/` – Last accessed October 2021

Large neighborhood search for a multi-mode resource constrained scheduling problem with resource leveling objective

Tom Portoleau[1,2], Christian Artigues[1], Tamara Borreguero Sanchidrián[3,4], Alvaro García Sánchez[4], Miguel Ortega Mier[4] and Pierre Lopez[1]

[1] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
{tom.portoleau,christian.artigues,pierre.lopez}@laas.fr
[2] IRIT, CNRS and University of Toulouse, France
[3] AIRBUS. Paseo John Lennon S/N. Getafe (28906) Spain
tamara.borreguero@airbus.com
[4] Industrial Engineering and Logistics Research Group, ETSII, Universidad Politécnica de Madrid. José Gutierrez Abascal 2 (28006) Madrid
{alvaro.garcia,miguel.ortega.mier}@upm.es

Keywords: multi-mode resource-constrained project scheduling, resource leveling, constraint programming, large neighborhood search.

1 Introduction

The aeronautical industry has experienced an in-depth transformation in the last years. The demand for aircrafts has increased but also the complexity and customization of the products. As a result, aircraft manufacturers have had to produce more units of more complex aircrafts while trying to reduce time to market, production lead times and costs. In line with these objectives, this paper proposes solution methods for the detailed scheduling of stations in an aircraft final assembly line. This problem, described in (Borreguero *et. al.* 2021), can be classified as a multi-mode resource-constrained project scheduling problem (MMRCPSP) with a resource leveling objective. In (Gerhards 2020) the author proposes a constraint programming (CP) model to solve the multi-mode resource investment problem (MMRIP). Formally, this problem is very similar to the assembly line scheduling problem at stake here. The notable differences are, firstly, the presence of non-renewable resources, and secondly, a slightly different objective function, as it considers a weighted sum of the maximums of the resources used, while the assembly line scheduling problem considers the maximum peak objective without weights. We propose a large neighborhood search (LNS) heuristic with neighborhoods tailored to the resource-leveling objective, with the aim to improve the performance of the CP exact methods and of the heuristic approach currently used by the company on large-scale industrial instances. As a result, the large-neighborhood search approach significantly improves the heuristic currently used by the aircraft manufacturer for assembly line scheduling. It also brings significant improvements to the method proposed by (Gerhards 2020) in the multimode resource investment problem when short CPU times are required.

2 Problem statement and constraint programming formulation

In the assembly line scheduling problem the maximum cycle time CT is fixed. We have a set of tasks W with a release date 0 and a due date CT. We have a set P of operator profiles and a set A of station areas. Each task $w \in W$ can be performed by a subset $P_w \subseteq P$ of operators profiles and by a number of operators that must lie in $M_w = \{MN_w, \ldots, MX_w\}$. The baseline duration of a task $w \in W$ is denoted by D_w. If task $w \in W$ is performed

by $o \in M_w$ operators of profile $p \in P_w$, we apply a reduction coefficient Γ_{opw} such that its duration is equal to $\Gamma_{opw} D_w$. Furthermore, each task is performed in an area indicated by $A_{aw} = 1$ if task is performed in area $a \in A$. Each area has a maximum capacity of C_a operators. There is also a set PR of standard precedence constraints such that $(w, w') \in PR$ means that w' cannot start before the end of w, a set NP of non-overlapping constraints between pairs of tasks such that $\{w, w'\} \in NP$ means that w and w' cannot be processed in parallel and a set MT of maximal time lag constraints such that $(w, w') \in MT$ means that the start time of w' must not exceed the end time of w plus a fixed time lag Δ.

We formally define the problem through the presentation of its constraint programming formulation. For modeling and solving, we use the CP Optimizer constraint-based scheduling library (Laborie *et. al.* 2018). Below we refer to the basic modeling elements we use. We refer to (Laborie *et. al.* 2018) for a more detailed definition of these elements. Each task $w \in W$ is modeled as an `interval` decision variable T_w. Each possible mode of a task (number of operators o and resource profile p) is modeled as an `interval optional` decision variable T_{wop} linked to the task by an `alternative` constraint. Precedence constraints and maximal time lag constraints are modeled by `endBeforeStart` and `startBeforeEnd` constraints, respectively. Task consumption on a resource is modeled as a `pulse` function. We use also `noOverlap` constraints for constraints related to set NP. The model comes as follows:

$\min \sum_{p \in P} n_p$
`dvar interval` T_w `in` $0..CT$, $\forall w \in W$
`dvar interval optional` T_{wop} `in` $0..CT$ `size` $\Gamma_{opw} D_w$, $\forall w \in W$, $p \in P_w$, $o \in M_w$
`alternative`$(T_w, (T_{wop})_{p \in P_w, o \in M_w})$, $\forall w \in W$
`endBeforeStart`$(T_w, T_{w'})$, $\forall (w, w') \in PR$
`startBeforeEnd`$(T_{w'}, T_w, -\Delta)$, $\forall (w, w') \in MT$
$\sum_{w \in W | A_{aw}=1} \sum_{p \in P_w} \sum_{o \in M_w}$ `pulse`$(T_{wop}, o) \leq C_a$, $\forall a \in A$
$\sum_{w \in W, o \in M_w}$ `pulse`$(T_{wop}, o) \leq n_p$, $\forall p \in P$
`noOverlap`$(T_w, T_{w'})$, $\forall \{w, w'\} \in NP$

3 A large neighborhood search approach

LNS is generally applied to scheduling problems where the objective is to minimize a time-related criterion or an outsourcing cost. A typical large neighborhood of a solution in this context consists in selecting a time interval and fixing all activities scheduled outside the interval and compacting as much as possible the activities scheduled over the interval, as it was done for the RCPSP in (Palpant *et. al.* 2004). Notably, the default search of the IBM CP Optimizer we used in the previous section also implements an LNS method based on this principle (Laborie *et. al.* 2018). This is not what we should do for the considered problem, as compacting a schedule as much as possible would inevitably increase the resource usage. In the problem considered here, we aim at minimizing the maximal use of a given resource. We aim at identifying the set of tasks that are involved in the maximal resource peaks. Consider a solution $\mathcal{S}$ where each task $w \in W$ has start time $\bar{S}_w \in [0, CT]$, a number of assigned operators $\bar{o}_w \in M_w$ for operator profile $\bar{p}_w \in P_w$ and a maximal number of operators $\bar{n}_p$ for each operator profile $p \in P$. The set of peak tasks is the set of all critical sets as defined below:

Definition 1. *A critical set $\tilde{W}$ is a set of overlapping tasks that reaches the maximal number of operators for at least one profile $p \in P$. More formally: $\exists t \in [0, CT]$, $\exists p \in P$, $\forall w \in \tilde{W}$, $\bar{p}_w = p$, $\bar{S}_w \leq t < \bar{S}_w + \Gamma_{\bar{o}_w \bar{p}_w w} D_w$ and $\sum_{w \in \mathcal{C}} \bar{o}_w = \bar{n}_p$.*

In fact, the resource usage only changes at the beginning or the end of a task. Let $\mathcal{T}$ denote the set of different start and end times of the tasks. The set of all critical sets can be enumerated by a sweep algorithm that tests the condition of definition 1 for each set built by the task that overlaps each time point in $\mathcal{T}$. Algorithm 1 describes the sweep algorithm that computes the set of all peak tasks $\mathcal{C}$ in $\mathcal{O}|W|^2|P|$ time.

Algorithm 1 The `sweep` algorithm for peak task computation

Require: A problem $\mathcal{P}$ and a solution $\mathcal{S} = \{(\bar{S}_w, \bar{p}_w, \bar{o}_w)_{w \in W}, (\bar{n}_p)_{p \in P}\}$
  $\mathcal{C} \leftarrow \emptyset$
  $\mathcal{T} \leftarrow \{\bar{S}_w | w \in W\} \cup \{\bar{S}_w + \Gamma_{\bar{o}_w \bar{p}_w w} D_w | w \in W\}$
  for $p \in P$ **do**
    for $t \in \mathcal{T}$ **do**
      $\tilde{W} \leftarrow \emptyset$; $cons \leftarrow 0$
      for $w \in W$ **do**
        if $\bar{p}_w = p$ **and** $\bar{S}_w \leq t < \bar{S}_w + \Gamma_{\bar{o}_w \bar{p}_w w} D_w$ **then**
          $\tilde{W} \leftarrow \tilde{W} \cup \{w\}$; $cons \leftarrow cons + \bar{o}_w$
        end if
      end for
      if $cons = \bar{n}_p$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \tilde{W}$
      end if
    end for
  end for
  return $\mathcal{C}$

In order to generate a high quality neighborhood, we let free all the tasks that contribute to the maximal use of the objective resource (the ones belonging to the peak set computed by the `sweep` algorithm) and the tasks that must precede them by a precedence constraint, and we fix the others. We then solve this new problem given the bound provided by the value of the initial solution and the constraints induced by the fixed tasks, within a limited time. If a solution has been found, it replaces the initial solution as the best solution and we start over. However, if no solution was found, we solve a new problem, fixing fewer tasks and setting a greater solving time, using a self-adaptive principle (Palpant *et. al.* 2004). To be more specific, each time the solver is unable to find a solution, we fix 10% less activities and add 10 seconds to the maximum solving time. These values were determined empirically using the instances from the benchmark considered in the previous sections. In our implementation, we use CP Optimizer as a black box to solve different generated subproblems using the constraint programming model described in Section 2. Algorithm 2 provides the pseudo-code of our implementation of the LNS method for the aircraft assembly line scheduling problem. Note that `presenceOf`(T_{wop}) in the CP Optimizer language is a constraint that enforces the presence of the optimal task T_{wop}, while `startAt`(T_w, t) is a constraint that fixes the start time of task T_w to value t. These two constraints are used to fix the modes and the start times of the tasks in $W \setminus W'$ while the tasks in W' are freed and form the LNS subproblem. Note that τ' is a value much lower than τ giving the amount of time devoted to the CP solver to get an initial solution.

4 Experimental results

We first compare LNS with CP optimizer and with a heuristic used by the aircraft manufacturer (Borreguero *et. al.* 2021) on a set of 7 industrial instances having from 90 up

Algorithm 2 LNS for the aircraft assembly line scheduling problem

Require: An aircraft assembly line scheduling problem $\mathcal{P}$ in the form of a constraint programming model (Section 2) and a time limit τ
Initialize solution $\mathcal{S}^* = \{(S^*_w, p^*_w, o^*_w)_{w\in W}, (n^*_p)_{p\in P}\}$ by solving $\mathcal{P}$ with CP Optimizer under time limit τ'
$\pi_{ratio} \leftarrow 100$; $\tau_{base} \leftarrow 10$; $\tau_{inc} \leftarrow 0$
while elapsed time $< \tau$ **do**
  $\tilde{W} \leftarrow \texttt{sweep}(\mathcal{P}, \mathcal{S}_{best})$ (get the peak tasks)
  $W^* \leftarrow \tilde{W} \cup \{W' \in W | (w', w) \in PR\}$ (add the tasks that precede them)
  $W' \leftarrow$ a subset of W^* where we randomly select $\pi_{ratio}\%$ tasks
  $\mathcal{P}' \leftarrow \mathcal{P}$
  for $w \in W \setminus W'$ **do**
    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \texttt{presenceOf}(T_{wo^*_w p^*_w})$
    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \texttt{startAt}(T_w, S^*_w)$
  end for
  $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{\sum_{p\in P} n_p < \sum_{p\in P} n^*_p\}$
  Get solution $\mathcal{S} = \{(\bar{S}_w, \bar{p}_w, \bar{o}_w)_{w\in W}, (\bar{n}_p)_{p\in P}\}$ by solving $\mathcal{P}'$ with CP Optimizer under time limit $\min(t_{base} + t_{inc}, T - elapsedtime)$
  if $\sum_{p\in P} \bar{n}_p < \sum_{p\in P} n^*_p$ **then**
    $\mathcal{S}^* \leftarrow \mathcal{S}$; $\tau_{inc} \leftarrow 0$; $\pi_{ratio} \leftarrow 100$
  else if $\mathcal{S}$ is empty **then**
    $\tau_{inc} \leftarrow \tau_{inc} + 10$; $\pi_{ratio} \leftarrow \pi_{ratio} - 10$
  end if
end while
return $\mathcal{S}^*$

to 721 tasks. For a 15 min CPU time limit, LNS improves the CP optimizer solutions for 5 out of 7 instances from 5.2% to 17.6%. Compared to the heuristic used by the manufacturer, the improvement on 4 out of 7 instances goes from 4.5% to 27.7% and for one instance LNS finds a solution while the heuristic is unable to find one. We also compare LNS to the CP optimizer model proposed in (Gerhards 2020) on the MMRIP. The results displayed in Table 1 for different CPU times and numbers of tasks (from 30 to 100) show significant improvements. Detailed results are given in (Borreguero *et. al.* 2021).

Table 1. LNS improvement over CP (Gerhards 2020) for the MMRIP

Instance set	1 min	15 min	30 min
$MMRIP_{30}$	5.27%	11.15%	0.82%
$MMRIP_{50}$	5.61%	15.47%	1.14%
$MMRIP_{100}$	10.58%	17.46%	3.40%

References

Borreguero T., Portoleau T., Artigues C., García A., Ortega M., 2021, "Exact and heuristic methods for an aeronautical assembly line time-constrained scheduling problem with multiple modes and a resource leveling objective", LAAS report 21247, LAAS-CNRS, Toulouse, `https://hal.laas.fr/hal-03344445`.

Gerhards P., 2020, "The multi-mode resource investment problem: a benchmark library and a computational study of lower and upper bounds", *OR Spectrum*, Vol. 42, Num. 4, pp. 901-933.

Palpant M., Artigues C., Michelon P., 2004, "LSSPER: Solving the resource-constrained project scheduling problem with large neighbourhood search", *Annals of Operations Research*, Vol. 131, Num. 1-4, pp. 237-257.

Laborie P., Rogerie J., Shaw P.,Vilím P., 2018, "IBM ILOG CP Optimizer for scheduling", *Constraints*, Vol. 23, Num. 2, pp. 210-250.

Robust scheduling within SNCF railway maintenance centers

Rahman TORBA[1,2], Stéphane DAUZÈRE-PÉRÈS[2], Claude YUGMA[2], Cédric GALLAIS[1] and François RAMOND[3]

[1] SNCF Direction du Matériel, 93210 Saint-Denis, France
rahman.torba, cedric.gallais@sncf.fr
[2] Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, Centre CMP, Departement SFL, F - 13541 Gardanne, France
yugma, dauzere-peres@emse.fr
[3] SNCF Innovation & Recherche, 93210 Saint-Denis, France
francois.ramond@sncf.fr

Keywords: RCPSP, Multi-project, Multi-skill, Railway maintenance, MILP.

1 Introduction

The French national railway company, SNCF, is responsible for the maintenance of its rolling stock. The most heavy maintenance operations, and thus longest ones (several weeks), are carried out in ten different dedicated workshops. The aim is to renovate and modernize railway vehicles every 10 to 20 years, depending on their type and deterioration state. Not only train coaches are maintained, but most of the components as well (electronic cards, bogies, axles). Economic savings and the reduction of the environmental impact of the railway industry are the main challenges while increasing passenger comfort and service quality. These decisions are made at a strategic level since the costs of rolling stock are very high. The rolling stock lifecycle is doubled thanks to heavy maintenance (Le Quéré *et. al.* 2003). At a tactical level, when and how much to produce should be decided to meet the customer requirements.

Knowing rolling stock arrival dates in the maintenance workshop, we propose a MILP model to schedule maintenance operations respecting industrial constraints. Each rolling stock unit is considered as a project and operations as activities requiring a certain number of resources to be processed. Thus, we deal with the *multi-skill resource-constrained multi-project scheduling problem* (MSRCMPSP), as resources have multiple skills and several rolling stock units are maintained simultaneously, which is an extension of the *resource-constrained project scheduling problem* (RCPSP).

The RCPSP was first proposed by Pritsker *et. al.* (1969) who introduced a binary variable $x_{i,t}$ equals to 1 if activity i start at time t, 0 otherwise. Since then, the problem has been widely studied and various extensions were proposed. It is a proven NP-hard problem and most papers focus on a single project; sometimes with multi-skill resources. However, very few papers deal with both multiple projects and multi-skill shared resources (Habibi *et. al.* 2018). The objective is usually to minimize the makespan, i.e. the total time to perform all activities on available resources.

In this paper, we focus on minimizing the sum of weighted tardiness of the projects $\sum_{e=1}^{N} w_e * T_e$, and the sum of their weighted duration $\sum_{e=1}^{N} w_e * (C_e - S_e)$; where w_e is the weight of the rolling stock unit e, T_e its tardiness, C_e its completion time and S_e its starting time. The first objective is obvious, respecting client due date is a priority in order to respect train timetable (Ramond *et. al.* 2006). The second objective reflects the performance of the workshop and it is attractive for the client since the immobilization of rolling stock units is minimized. Variable S_e is introduced because it is easier to agree

with the customer on an arrival date than on a due date. By decreasing the lead times, machines idle times are minimized and maintenance costs are reduced.

2 Problem description

When trains enter into the workshop some technical tests and observations are performed to re-estimate the workload. Then, trains are uncoupled in several coaches. Components of coaches are dismantled and repaired in parallel. To reassemble the train, the activities of all coaches and components must be finished. Once coaches are coupled, the final activity consists of testing the train in order to check that safety and quality standards are respected. Figure 1 illustrates the activity precedence graph of a very simplified maintenance procedure of a rolling stock unit composed of only 2 coaches.

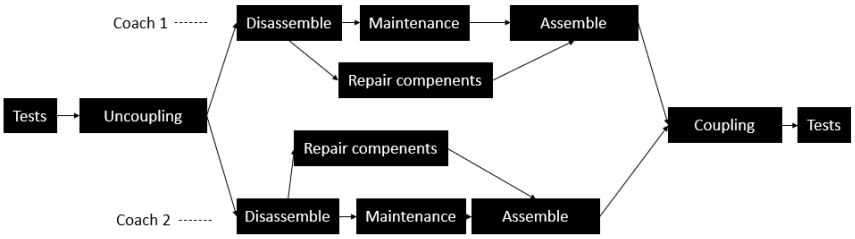


Fig. 1. Simplified activity precedence graph of rolling stock heavy maintenance

To give a few figures, a workshop processes around 6 to 10 trains simultaneously and thousands activities over a horizon of 1 to 2 years. Therefore, this is a very large planning problem.

So, we have a set of activities A to schedule, with processing time p_a and precedence relationship represented as a couple (a, a') meaning that activity a must be executed before a', during a time horizon T. Each activity needs a set of capacitated $r_{a,k}$ resources of type $k \in K$. Regarding resource types, machines and human operators must be differentiated. A machine in our case is a location (place), equipped with rails and with several facilities (for instance a garage pit and rolling stock roof access installation). Thus, by analogy with operators, machines are considered as multi-skill resources.

Contrary to machines, operators have working hours that must be taken into account. Due to long-horizon scheduling and uncertainty of the number of operators with a given skill, we do not assign operators but only assure to not exceed a capacity threshold for each team $r \in R$ and skill $k \in K$ at each period $t \in T$. The threshold should be configurable for each team. In fact, depending on rolling stock types and their entering date, certain teams or skills are not yet well sized. Furthermore, managers can create different scenarios with different capacity configurations and take mid-term decisions.

However, since the workshop configuration is very unlikely to be modified, we assign to each activity the location with the facility required to process this activity. Thus, this is a multi-skill resource assignment problem. The goal it to find a resource feasible solution that optimizes one or several criteria and respects maintenance procedure constraints such as precedence constraints and transport times between activities, modelled usually as time lags (Laurent *et. al.* 2017).

3 Mathematical modeling

Two main modeling approaches for the RCPSP can be found in the literature (Koné *et. al.* 2011):

- **Continuous time modeling**, where activities can start at any time of the scheduling horizon. The associated models are based on disjunctive constraints between activities sharing the same resources (Azem *et. al.* 2007).
- **Discrete time modeling**, where activities can only start at a given period $t \in 1, 2...T-1, T$. The number of variables, and the performances of the associated models are very sensitive to the discretization step and the scheduling horizon T since the number of decision variables increases.

As mentioned in Section 2, human resources have working hours. Furthermore, the number of operators with a given skill is uncertain on a long horizon and because it is time varying, we use a discrete time model for our problem. This modeling approach is widely used for the RCPSP since time dependent constraints are easy to write, and discrete time models have good linear relaxation bounds compared to continuous time models because of the "big-M" required in disjunctive constraints (Azem *et. al.* 2007).

Our MILP model is adapted from the model of Bellenguez and Néron (2005) and Pritsker *et. al.* (1969). The main binary decision variables are listed below:

- $X_{m,a,t} \in \{0,1\}$ which is equal to 1 if activity $a \in A$ start at period $t \in T$ at location $m \in M$, and 0 otherwise.
- $Y_{m,i,a} \in \{0,1\}$ which is equal to 1 if facility $i \in I_m$ of localisation m is used for processing a, and 0 otherwise. This variable allows us to handle the multi-skill aspect of the problem.
- $S_{a,t} \in \{0,1\}$ which is equal to 1 if activity a start at period t, and 0 otherwise. Let us note that this is an auxiliary variable used to keep a similar structure as the model proposed by Pritsker *et. al.* (1969). To be more explicit, $S_{a,t} = \sum_{m \in M} X_{m,a,t}$.

No preemption is allowed, each resource is assigned at most to one activity in a given period t and there are precedence constraints with lag time $l_{a,a'}$ between activities. For the sake of brevity, we do not present the full model but we introduce and study the performances of two different ways of writing precedence constraints:

$$\sum_{t=1}^{T} tS_{a,t} + p_a + l_{a,a'} \leq \sum_{t=1}^{T} tS_{a',t}; \qquad \forall (a,a') \in P_a \tag{1}$$

and,

$$\sum_{l=1}^{t+p_a-1} lS_{a,l} + p_a + l_{a,a'} \leq \sum_{l=t}^{T} lS_{a',l}; \qquad \forall t \in T, \forall (a,a') \in P_a \tag{2}$$

The first one is the "classical" way, and the second one is an original disaggregated approach, firstly considered by Christofides *et. al.* (1987) to reinforce the first one.

4 Preliminary numerical results

In our computational experiments, we analyze, on small instances designed using industrial data, the performances of the two modelling approaches. The results in Table 1 show that the discretization step "Step(h)" has a considerable impact on computational times for both modelling methods. The "classical" method is way faster than the disaggregated approach. However, for "difficult" instances, adding disaggregated constraints improves the computational times.

Table 1. Preliminary results on small instances based on industrial data (CPU<30min)

		Duration			**Tardiness**		
Instance	Step(h)	$CPU_1(s)$	$CPU_2(s)$	$CPU_{12}(s)$	$CPU_1(s)$	$CPU_2(s)$	$CPU_{12}(s)$
	6	**9**	18	13	**9**	37	14
E2Act45	4	**11**	324	16	**9**	134	22
	2	104	935	**97**	45	1357	**33**
	6	576	1306	**448**	1410	1751	**779**
E3Act73	4	960	1538	**853**	1251	-	**936**
	2	**1344**	-	1337	1695	-	**1392**
	6	1042	1788	**928**	1789	-	**1639**
E4Act90	4	-	-	**1709**	-	-	-
	2	-	-	-	-	-	-

5 Conclusions and perspectives

We proposed a Mixed Integer Linear Program to model the scheduling problem of heavy maintenance operations on rolling stock, which is a *multi-skill resource-constrained multi-project scheduling problem* (MSRCMPSP). The model was tested and validated using real industrial data. However, heavy maintenance operations are very uncertain (processing time, additional workload, etc.), and our main perspective is to study the stochastic version of the problem, using for instance the notion of service level proposed in Dauzère-Pérès *et. al.* (2008). A second perspective is to explore heuristics to solve large industrial instances and have the proposed schedules validated by the maintenance planners.

References

Azem, S., Aggoune, R., & Dauzere-Peres, S., 2007, "Disjunctive and time-indexed formulations for non-preemptive job shop scheduling with resource availability constraints", *IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 787-791.

Bellenguez, O., Néron, E., 2005, "Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills", *International Conference on the Practice and Theory of Automated Timetabling*, pp. 229-243.

Christofides, N., Alvarez-Valdés, R. & Tamarit J., 1987, "Project scheduling with resource constraints: A branch and bound approach", *European Journal of Operational Research*, Vol. 29, pp. 262-273.

Dauzère-Pérès, S., Castagliola, P., & Lahlou, C., 2008, "Service Level in Scheduling, In Flexibility and Robustness in Scheduling" *John Wiley & Sons, Ltd*, pp. 99-121.

Habibi, F., Barzinpour, F., & Sadjadi, S., 2018, "Resource-constrained project scheduling problem : Review of past and recent developments", *Journal of Project Management*, Vol. 3, pp. 55-88.

Koné, O., Artigues, C., Lopez, P., & Mongeau, M., 2011, "Event-based MILP models for resource-constrained project scheduling problems", *Computers and Operations Research*, Vol. 38, pp. 313.

Laurent, A., Deroussi, L., Grangeon, N., & Norre, S., 2017, "A new extension of the RCPSP in a multi-site context : Mathematical model and metaheuristics", *Computers & Industrial Engineering*, Vol. 112, pp. 634-644.

Le Quéré, Y., Sevaux, M., Tahon, C., & Trentesaux, D., 2003, "Reactive scheduling of complex system maintenance in a cooperative environment with communication times", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 33, pp. 225-234.

Pritsker, A. A. B., Waiters, L. J., & Wolfe, P. M., 1969, "Multiproject scheduling with limited resources : A zero-one programming approach", *Management Science*, Vol. 16, pp. 93-108.

Ramond, F., Dauzére-Pérès, S., & de Almeida, D., 2006, "Scheduling moves within railcar maintenance centers", *IFAC Proceedings Volumes*, Vol. 39, pp. 405-410.

Insights and results for the offline and online weighted capacitated parallel machine scheduling problem

Ilan Reuven Cohen* Izack Cohen† Iyar Zaks‡

We focus on capacitated parallel machine scheduling problems for modeling modern cloud computing environments as well as batch manufacturing processes and wafer fabrication processes [1]. The characteristic that separates between capacitaed machine scheduling problems and their widely researched non-capacitated counterparts is that a capacitated machine can process concurrently several jobs, up to its capacity.

Cloud computing systems such as those of Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP) include multiple components that impact their performance; one of those components is the scheduling module that allocates jobs to machines. Typically, decision-makers who seek to elevate the utilization of their data centers concentrate on improving their scheduling algorithms. Our focus in this research is on developing an efficient scheduling algorithm for the capacitated identical parallel machine scheduling problem.

Surprisingly, the classic version of the identical parallel machine scheduling problem describes well many real-world cloud computing environments. In fact, we were inspired into this research from a cloud computing environment of a large organization in which one of the authors worked. This environment can be modelled via the weighted capacitated identical parallel machines model. In the considered system, the weights manifest the possibility of jobs with different importance (for example, due to their reference to different customers or projects).

We seek to develop an algorithm with provable performance guarantees and, at the same time, one that exhibits good average performance. Such a desired algorithm A for a minimization problem P would have a ρ approximation ratio such that for any instance I of P, $A(I) \leq \rho \cdot OPT(I)$, where $OPT(I)$ is the value of an optimal solution for I.

*The Industrial & Information Systems Engineering (IISE) track in the Faculty of Engineering, Bar-Ilan University, Israel. ilan-reuven.cohen@biu.ac.il

†The Industrial & Information Systems Engineering (IISE) track in the Faculty of Engineering, Bar-Ilan University, Israel. izack.cohen@biu.ac.il

‡Faculty of Industrial Engineering and Management, The Technion, Israel. iyarzaks@gmail.com

Many studies developed approximation algorithms for the non capacitated version of the problem; for example, Eastman, Even, and Isaacs in [2] proved that scheduling according to the Weighted Shortest Processing Time (WSPT) priority rule provides a constant approximation algorithm. Kawaguchi and Kyan [3] improved the WSPT approximation ratio (ρ) to $(1+\sqrt{2})/2$ and proved that it is tight.

For the capacitated setting the research is scarce; Im, Naghshnejad, and Singhal [4] were the first to develop a constant approximation algorithm for minimizing the *non-weighted* sum of completion times. They combined the Smallest Volume First (SVF) priority rule and the Shortest Processing Time (SPT) priority rule. Our work extends theirs into the more general, weighted case, by combining the Weighted Smallest Volume First (WSVF) and WSPT rules, aiming to develop the first algorithm with a constant approximation ratio for the weighted completion time problem in the capacitated setting. Moreover, our analysis improves their approximation ratio for the non-weighted case and the suggested algorithm demonstrates favorable performance compared to several other alternatives.

Since the concerned problem is $\mathcal{NP}$-hard, we study heuristic approaches and develop provable approximation guarantees for their offline version. We focus on an algorithm that prioritizes the jobs with the smallest volume-by-weight ratio. We bound its approximation ratio using a decreasing function of the ratio between the highest resource demand of any job and the server's capacity. We use this algorithm for scheduling jobs with resource demands equal to or smaller than 0.5 of the server's capacity in conjunction with the classic weighted shortest processing time algorithm for jobs with resource demands higher than 0.5. By doing this, we create a hybrid, constant approximation algorithm for two or more machines.

We also develop a mixed integer linear program that can solve small problem instances to optimality and demonstrate that the suggested algorithm finds near-optimal solutions. Then, we conduct a comprehensive numerical experiment with synthetic and real-life problem instances with up to 30,000 jobs and 80 machines. We benchmark our WSVF algorithm with other alternatives such as the SPT, WSPT and SVF. The results indicate that the WSVF performs favorably with respect to the other alternatives.

Next, we develop an online WSVF-based algorithm that handles settings in which new jobs are continuously and randomly arriving to the system. The objective function is to minimize the total weighted flow time. As in the offline version of the problem, the algorithm uses the WSVF priority rule but in a way that is adapted to the online setting. Each time a job completes its processing or at preset intervals, the queue is reordered while taking into account the new jobs that arrived. We checked three different allocation policies for the online algorithm that fit different situations.

The first version, which we call Batch dispatch, sorts the batch of jobs that arrived to the scheduling system during the last time interval by WSVF and schedules them on the machines; that is, for each job the processing start time is set - even if the start time is sometime in the future. Via this policy, the customer immediately knows the machine and the start time at which its job will run, and accordingly when it is expected to complete its processing. Moreover, this

policy guarantees that no job will be unfairly stuck in the line.

The second policy emulates the parallel scheduling scheme without back-filling. That is, a job is allocated to a machine only if it can start its processing immediately. Under this policy, the queue of jobs is updated whenever a new job arrives. In such a case the customer does not receive information about when its job will be processed and there is also a possibility that a job will not be processed for a long time if jobs with a higher priority arrive to the system frequently.

The third policy, which is denoted as online continuous dispatch with back-filling, is similar to the second policy but with the option for jobs to be scheduled before higher priority jobs that cannot be scheduled immediately due to lack of machine capacity.

Our numerical experiments indicate that the second version of the algorithm outperforms the first one by almost 50% and the third outperforms the second policy by about 10%. That is, by the first policy the customers receive full information about the processing machine and completion time of their job but the price of providing this information is a significantly higher average completion time.

We list below the contributions of our research:

1. We investigate the offline and online versions of the capacitated parallel machine scheduling problems and provide theoretical and numerical results.
2. We provide a polynomial-time algorithm with a $\left(1+\frac{1}{1-\alpha}\right)$-approximation ratio, if the ratio between the jobs' demands and the servers' capacities is at most α. Our algorithm also improves the best known approximation guarantees for the non-weighted version—see, [4].
3. Moreover, we provide a polynomial-time constant $4+o(1/M)$-approximation ratio for job scheduling on two or more machines.
4. We formulated a mathematical program for solving small problem instances to optimality so we can benchmark them with the WSVF algorithm.
5. Via extensive numerical experiments with realistic size problem instances, we demonstrate the favorable performance of the offline and online versions of the suggested algorithms compared to other alternatives.

Finally, we mention three possible extensions to our work. The first significant theoretical extension may consider capacitated problems with multiple capacitated resources. In real-world cloud computing environments, CPU, memory, storage and bandwidth may be scarce resources. To realize such an extension, one would need to consider a multidimensional demand for each job. A second extension would be to use our methods to find performance guarantees in stochastic environments in which, for example, processing duration can be characterized using probabilistic knowledge. The third research direction would be to develop a model that accommodates jobs with release dates and deadlines. Such research may also be considered to be a natural extension of the current model.

References

[1] Purushothaman Damodaran, Omar Ghrayeb, and Mallika Chowdary Guttikonda. "GRASP to minimize makespan for a capacitated batch-processing machine". In: *The International Journal of Advanced Manufacturing Technology* 68.1-4 (2013), pp. 407–414.

[2] Willard L Eastman, Shimon Even, and I Martin Isaacs. "Bounds for the optimal scheduling of n jobs on m processors". In: *Management science* 11.2 (1964), pp. 268–279.

[3] Tsuyoshi Kawaguchi and Seiki Kyan. "Worst case bound of an LRF schedule for the mean weighted flow-time problem". In: *SIAM Journal on Computing* 15.4 (1986), pp. 1119–1129.

[4] Sungjin Im, Mina Naghshnejad, and Mukesh Singhal. "Scheduling jobs with non-uniform demands on multiple servers without interruption". In: *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE. 2016, pp. 1–9.

Disjunctive graph model for flexible job-shop scheduling problem with transportation and limited buffer space

Lucas Berterottière[1], Claude Yugma[1] and Stéphane Dauzère-Pérès[1]

Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, Centre CMP, Manufacturing Sciences and Logistics Department, F - 13541 Gardanne, France
{berterottiere,yugma,dauzere-peres}@emse.fr

Keywords: Job-shop scheduling, Disjunctive Graph, Transportation, Buffers

1 Introduction

The production of microelectronic devices is a highly complicated and cost-intensive process. In a semiconductor manufacturing facility (fab), a huge number of processing steps (until 800) in different work areas are required for the production of a single job (lots of wafers). In this context working on scheduling decisions is a challenging task. Furthermore, scheduling decisions have a strong impact on key performance indicators such as throughput and cycle time (Mönch et al. (2011)).

In this paper, we are concerned with solving an extension of flexible job-shop scheduling problems stemming from semiconductor manufacturing. The Flexible Job Shop Problem (FJSP) is an extension of the classical job shop scheduling problem which allows an operation to be processed by any machine from a given set. The problem is to assign each operation to a machine and to order the operations of the machines, such that the maximal completion time (Makespan) of all operations is minimized.
We extend this problem by integrating machine-to-machine transport jobs. The capacity of each vehicle is unitary i.e. each vehicle can move only one job at a time. Every job starts from an initial common deposit. Our goal is to solve the flexible job-shop scheduling problem while considering transportation and storage management of jobs to minimize Makespan. We formulate this problem through disjunctive graph modeling.

2 Related works

The problem under consideration is a generalization of the flexible job-shop scheduling problem which is already known to be NP-hard (Garey, M., Johnson, D. 1979). The literature is full of work on flexible job shop scheduling problems as Dauzère-Pérès, S., Paulli, J. (1997) and Kacem, I. *et. al.* (2002).

This problem adds to the flexible job-shop scheduling the transportation management and the limited capacity of the buffers. Three aspects have to be underlined for the problem, the flexibility of machine assignment, the transportation management, and the limited capacity of the buffers. Seen separately, these problems have been studied in the literature. We can find papers that introduce routing in job-shop scheduling decisions. Ulusoy, G., Bilge, U. (1993) and Bilge, U. and Ulusoy, G. (1995) proposed such a problem and decompose the master problem into two subproblems: production operations scheduling and transport operations scheduling. Hurink, J., Knust, S. (2005) introduced a disjunctive graph for the job shop with one single robot, then Lacomme, P., Larabi, M. (2007) extended to a job shop scheduling with multiple automated guided vehicles (AGV). More recently Zeng, C., Tang, J., Yan, C. (2014) proposed a neighborhood structure for the disjunctive graph model to solve the blocking job-shop with AGV (BJS-AGV). The closest article to our topic is

Zhang, Q., Manier, H., Manier, M-A. (2013) which uses a disjunctive graph modeling but differs on the method used as they adapt a Shifting Bottleneck heuristic.

To our knowledge, integrating these aspects (transportation, flexible scheduling and storage management) into the problem has rarely been addressed in the literature. Integrating transport in a job-shop is even more relevant if it is flexible, because in a classical job-shop, the transport requests are known while in a flexible job-shop, the choice of the machine has an impact on the travel time of the job. Dealing with these aspects together is also motivated by modern semiconductor manufacturing factories. Actually, we now find in fabs optimized algorithms (schedulers) used for production, one move from reactive to predictive transport management where transportation and storage decisions can be planned. Moreover, in these modern fabs, the cost of automated vehicle management for transportation is very low and can be neglected. Thus, given these considerations, it is relevant to integrate production, transport and storage decisions for better production planning.

3 Problem description

As already written, the problem we are studying here is an extension of the FJSP with transport, where an input and output loading port for each machine and limited capacity buffers for job storage are considered. An instance of the problem is composed of a set of job, which are a sequence of multiple consecutive operations, a set of machines with a defined number of input and output buffer space and a set of vehicles with unit capacity. As we are studying a flexible job-shop, each operation has a set of machines on which it can be processed. The processing times of production operations are machine dependents while the travel times of the vehicle doesn't depend on the vehicle affected.

The objective is to determine an assignment of jobs to machines, a sequence of production operations on the machines, and using buffers when necessary, the transportation sequences for vehicles for minimizing the Makespan C_{max}.

4 Disjunctive graph modeling

The disjunctive graph is given by $G = (V, E_c, E_d)$, where V denotes the set of nodes of G, E_c the set of conjunctive arcs and E_d the set of disjunctive arcs. A node $v \in V$ represents an operation of a job. The initial conjunctive arcs, defined as directed arcs, represent precedence constraints between operations in a job. The disjunctive arcs, defined as undirected arcs, represent the need to schedule on one machine one node at one end before the other. Once this graph has been constructed, a possible schedule can be determined by finding, for each machine, a chain of oriented arcs from the disjunctive arcs. The conjunctive graph is a solution to the scheduling problem if there is no cycle. Each precedence constraint is represented.

We add transport operations between the production operations. Those operations are all linked together by as many arcs as there are vehicles. Once all the arcs are oriented such as there is a planning, the transport arcs are weighted by the travel time between two machines. To represent the blocking constraints, we use the technique of McCormick, S.*et. al.* (1989). A disjunctive arc between two operations, one blocking the other, is replaced by two oriented arcs from each node to the predecessor of the other. The resulting solution is left with only one of the two arcs. This technique is used for vehicles, which can only process one operation at a time and will be used for the evaluation of a solution when adding buffers.

It is only possible to add buffers when a solution is found. The resulting conjunctive graph is extended by adding a node before and after every node representing a production

operation. They represent the input and the output buffers. Every job follows the rule first in, first out (FIFO).

5 Solution approach and results

In this section, we summarize by a pseudo-code the steps of the Tabu search we followed to obtain a solution:

Step 1. Find an initial solution using a topological sort of the vertices.
Step 2. While $Maxiter$ iterations without improvement on the Makespan, repeat:
- Search the neighborhood to find the best non-tabu move.
- Perform the best non-tabu move.
- Update the makespan and the heads and tails of each nodes.
- Update the Tabu list.

Step 3. Return the best solution.

We use a randomly generated initial solution by taking every production operation in a topological order then affect it to the least loaded machine on which this operation can be processed. Then we sort the transport operation according to their release date and similarly affect them to the least loaded vehicle.
We use a Tabu search method inspired by Dauzère-Pérès, S., Paulli, J. (1997) improved in **?**. To use a local search method, we need to define a neighborhood from a solution. The challenging part is that the neighbor must not contain a cycle. We adapt the neighborhood structure to guarantee that no cycle is created.
One of the main differences between our method and the previous method is that we are working on two connected graphs. The first is the graph with the production and transport operations and the second is the same with the buffers. This change occurs at the time of the update of the Makespan and of the heads and tails. We will present results comparing the evaluation of heads and tails on the first conjunctive and the extended one. The modification and evaluation of the extended connective graph is time-consuming, but it allows to accurately evaluate the heads and tails with the blocking constraints on the buffers.

Instance	C^*_{max} (MILP)	LAHC	Tabu search
fjspt1	134	138	134
fjspt2	114	114	114
fjspt3	120	120	120
fjspt4	114	114	116

Table 1. Comparison with Late Acceptance Hill-Climbing of Homayouni, S. M., Fontes, D., B., M., M. (2021) on 4 instances of Deroussi, L., Norre S. (2010)

The results on the table 1 shows that our results are matching those of Homayouni, S. M., Fontes, D., B., M., M. (2021) on the instances of Deroussi, L., Norre S. (2010) that are modified versions of instances of small-sized instances proposed by Bilge, U. and Ulusoy, G. (1995).

6 Conclusion

In this paper, we have addressed a complex problem is motivated by the semiconductor manufacturing industry. We have considered a Flexible Job-Shop scheduling problem with transportation and limited capacity buffer to be solved in an integrated approach. A disjunctive graph representation is used for modeling this complex problem. We have obtained promising results on a benchmark instances of Deroussi, L., Norre S. (2010) for FJSP with transport but without storage constraints.
Other properties obtained and result analysis will be reported at the conference.

References

Bilge, U. and Ulusoy, G., 1995, 'Time window approach to simultaneous scheduling of machines and material handling system in an FMS", *Operations Research*, Vol. 43, pp. 1058-1070.

Dauzère-Pérès, S., Paulli, J., 1997, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search", *Annals of Operations Research*, Vol. 70, pp. 281-306.

Deroussi, L., Norre S., 2010, "Simultaneous scheduling of machines and vehicles for the flexible job shop problem", *International Conference on Metaheuristics and Nature Inspired Computing*, pp. 1-2.

Garey, M., Johnson, D., 1979, "Computers, Complexity, and Intractability".

Homayouni, S. M., Fontes, D., B., M., M., 2021, "Production and transport scheduling in flexible job shop manufacturing systems", *Journal of Global Optimization*, Vol. 79, pp. 463–502.

Hurink, J., Knust, S., 2005, "Tabu search algorithms for job-shop problems with a single transport robot", *European Journal of Operational Research*, Vol. 162, pp. 99-111.

Kacem, I., Hammadi, S., Borne, P., 2002, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems", *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, Vol. 32, pp. 1-13.

Lacomme, P., Larabi, M., 2007, "A Disjunctive Graph for the Job-Shop with Several Robots", *MISTA Conference, Paris*, pp. 285-292.

McCormick, S., Pinedo, M., Shenker, S. *et. al.*, 1989, "Sequencing in an assembly line with blocking to minimize cycle time", *Operations Research*, Vol. 37, pp. 925-935.

Mönch, L., Fwoler, J.W., Dauzère-Pérès, S., Mason, S.J., Rose, O., "A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations", *Journal of Scheduling*, Vol. 14, pp. 583–599.

Ulusoy, G., Bilge, U., 1993, "Simultaneous scheduling of machines and automated guided vehicless", *International Journal of Production Research*, Vol. 31, pp. 2857-2873.

Zeng, C., Tang, J., Yan, C., 2014, "Scheduling of no buffer job shop cells with blocking constraints and automated guided vehicles", *Applied Soft Computing Journal*, Vol. 24, pp. 1033-1046.

Zhang, Q., Manier, H., Manier, M-A., 2014, "A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints", *International Journal of Production Research*, Vol. 52:4, pp. 985-1002.

Application of Quantum Approximate Optimization Algorithm to Job Shop Scheduling Problem

Tomasz Pecyna[1], Krzysztof Kurowski[1], Rafał Różycki[2], Grzegorz Waligóra[2] and Jan Węglarz[2]

[1] Poznań Supercomputing and Networking Center, IBCH PAS, Poland
tpecyna, krzysztof.kurowski@man.poznan.pl
[2] Poznań University of Technology, Institute of Computing Science, Poland
rafal.rozycki, gwaligora, jan.weglarz@cs.put.poznan.pl

Keywords: Job Shop Scheduling Problem, Quantum Approximate Optimization Algorithm, Scheduling, Quantum Computing, Heuristic, Approximation.

1 Introduction

The Job Shop Scheduling Problem (JSSP) has always been considered as one of the most complex scheduling problems. Optimizing the makespan of a given schedule generally involves using dedicated algorithms, local search strategies, or metaheuristics. These approaches, however, rely on classical computational power, which is bounded by the physical limits of microcontrollers and power issues. Inspired by the promising results achieved for Quantum Annealing (QA) approaches to solve JSSP (Venturelli et al. 2016), we propose an approach that uses gate-model quantum architecture with the Quantum Approximate Optimization Algorithm (QAOA) (Farhi et al. 2014) as an alternative to QA. Our approach, in contrast to Amaro et al. (2022), shows that we can take advantage of educated guess strategy to increase the chances of finding the optimal solution to a basic JSSP instance.

2 Formulation

In this paper we use the following formulation of the JSSP. There are J jobs $\mathcal{J} = \{j_1, \ldots, j_J\}$ and each of them has O_j operations $\mathcal{O}_j = \{o_{j1} \to \cdots \to o_{jO_j}\}$ to be processed in predefined order. Each operation must be processed on a specified and distinct machine from a set of M machines, $\mathcal{M} = \{m_1, \ldots, m_M\}$, and only one operation can be processed by a machine at a given time. The objective is to find the minimum makespan.

Inspired by Venturelli et al. (2016) we use the time-indexed JSSP representation. We define $x_{k,t} = 1$ if operation o_k starts at time t and $x_{k,t} = 0$ otherwise, where t is bounded by a deadline T common for all jobs, and k is a running index representing a position of an operation in a list concatenating operations over all jobs:

$$[\underbrace{o_{11}, \ldots, o_{1O_1}}_{j_1}, \underbrace{o_{21}, \ldots o_{2O_2}}_{j_2} \ldots, \underbrace{o_{J1}, \ldots o_{JO_J}}_{j_J}] = [\underbrace{o_1, \ldots, o_{k_1}}_{j_1}, \underbrace{o_{k_1+1}, \ldots o_{k_2}}_{j_2} \ldots, \underbrace{o_{k_{J-1}+1}, \ldots o_{k_J}}_{j_J}]. \tag{1}$$

We can now define a set of feasibility constraints. Firstly, the operation must start once and only once, which is expressed by the following formula:

$$h_1(x) = \sum_k \left(\sum_t x_{k,t} - 1 \right)^2 = 0. \tag{2}$$

Secondly, there can be only one operation running at a given machine at any time, i.e.:

$$h_2(x) = \sum_m \left(\sum_{k,t,k',t' \in A_m \cup B_m} x_{k,t} x_{k',t'} \right) = 0, \tag{3}$$

where A_m constrains an operation $o_{k'}$ from starting on a machine m if o_k is still running on the machine, and B_m constrains two operations from starting at the same time.

Let us denote l_k as the processing time of operation k. The last feasibility constraint is defined so that the original order of the operations is kept for all the operations:

$$h_3(x) = \sum_{n=1}^{J} \left(\sum_{\substack{k_{n-1}<k<k_n \\ t+l_k>t'}} x_{k,t} x_{k+1,t'} \right) = 0. \tag{4}$$

To promote low-makespan schedules we take advantage of the time-indexed representation by deriving an additional term that will put a penalty on any non-optimal schedule:

$$h_4(x) = \sum_{n=1}^{J} (J+1)^{t_{k_n}}, \tag{5}$$

With JSSP formulated, let us shortly remind the basics of QAOA to use it as the optimization algorithm. Having R qubits, the QAOA alternately applies the cost Hamiltonian H_C and a mixing Hamiltonian H_B to the $|+\rangle^{\otimes R}$ state p times, achieving a final state ψ:

$$|\psi_p(\overrightarrow{\gamma}, \overrightarrow{\beta})\rangle = e^{-i\beta_p H_B} e^{-i\gamma_p H_C} \dots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_C} |+\rangle^{\otimes R}, \tag{6}$$

where the variables γ and β play the role of variational parameters to be optimized by a classical algorithm. The parameter p is a key parameter of QAOA which defines circuit depth and influences solution quality. The cost Hamiltonian is the objective function constructed of a sum of binary clauses, in our case it is a sum of eqs. (2) to (5) taking Pauli-Z matrices as arguments, i.e. $H_C(\sigma^z) = h_1(\sigma^z) + h_2(\sigma^z) + h_3(\sigma^z) + h_4(\sigma^z)$. The mixing Hamiltonian H_B usually takes the form of a sum of Pauli-X matrices $H_B = \sum_{r=1}^{R} \sigma_r^x$.

The goal of QAOA is to find such parameters γ^* and β^* so that the expected value, which we will call energy in the next section, is minimized:

$$F_p(\overrightarrow{\gamma}, \overrightarrow{\beta}) = \langle \psi_p(\overrightarrow{\gamma}, \overrightarrow{\beta}) | H_C | \psi_p(\overrightarrow{\gamma}, \overrightarrow{\beta}) \rangle \tag{7}$$

3 Experiments and results

Due to computational power limitations the following analysis was conducted on a toy JSSP instance. The toy instance consists of 3 jobs and each of them contains between 1 and 2 operations with processing times ranging from 1 to 2 units. The number of machines is 3. The makespan for the optimal schedule for this instance is $\tau = 3$.

3.1 Educated guess strategy

The succeeding optimal variational QAOA parameter sequences $\beta^* = \beta_1^*, \dots, \beta_i^*, \dots, \beta_p^*$ and $\gamma^* = \gamma_1^*, \dots, \gamma_i^*, \dots, \gamma_p^*$ have been found to form patterns, i.e. their values should increase monotonically with increasing i and also, that they should interpolate the space with increasing p allowing to use the so-called educated guess strategy (Zhou et al. 2020). We have experimentally found that the patterns also exists when using our JSSP formulation. In Fig. 1 we present a visualisation of the behaviour of the variational parameters together

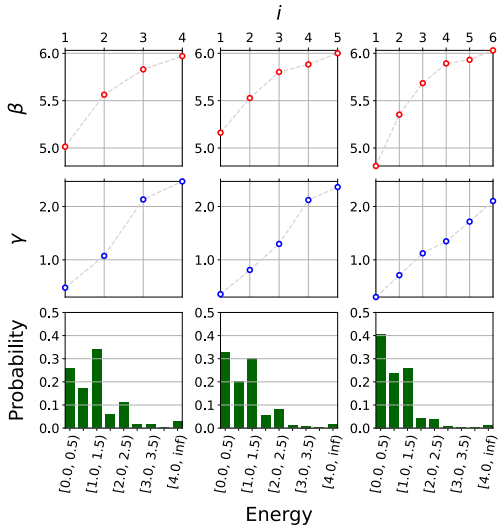


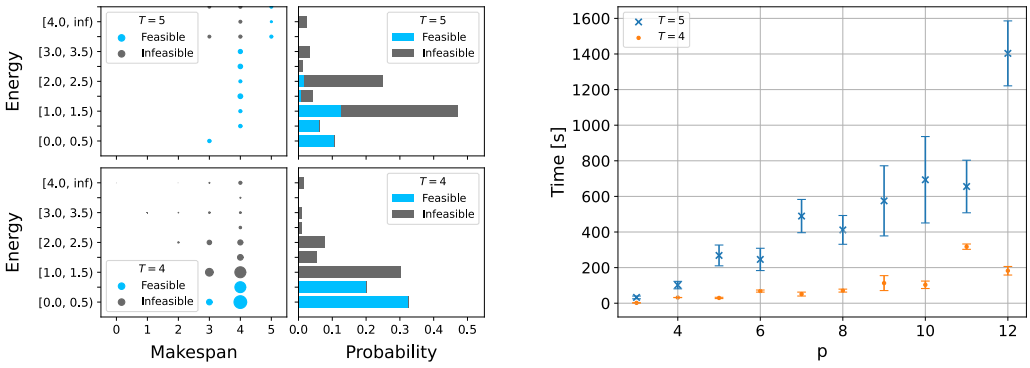
Fig. 1: Results for the basic JSSP instance obtained by using the educated guess strategy. The subplots show 3 pairs of optimal parameters β_i and γ_i $(i : 1 \dots p)$ as well as corresponding energy probabilities of the cost Hamiltonian.

with energy probabilities of the cost Hamiltonian. We can clearly see that, indeed, both γ and β tend to monotonically increase their values in the domain of increasing i in a single circuit, and that they tend to interpolate the space in the domain of increasing circuit depth p. We can also see that, the circuit depth affects the energy probabilities reaching over 40% chance of measuring low energy solutions when the depth is $p = 6$.

3.2 Performance analysis

We present the relation between energy and makespan for $p = 5$ in Fig. 2a. We can see a clear pattern that the lower the energy is, the higher is the probability of obtaining a feasible solution with a low makespan. Moreover, only the lowest range of energies correspond with the optimal solution. Interestingly, restricting the maximum feasible time to $T = 4$ resulted in rise in probability of measuring solutions with low energy, therefore lower makespan. This property might be advantageous when using any strategy based on repeated querying the circuit, e.g., starting with large T, measuring any feasible solution with makespan lower than T, and then setting this makespan value as T for the next iteration. Consequently, this would, again, accelerate the process of finding the optimal solution.

Additionally, in Fig. 2b we show the optimization time. We can observe that the time needed to optimize the variational parameters is higher when $T = 5$, than $T = 4$ and raises quickly. It is because the number of variables needed to solve the problem growths with T. In our case, the toy instance with $T = 4$ needs (after the pruning described in eg., Kurowski et al. (2020)) 11 variables, while setting $T = 5$ raises the needed number of variables to 16. Note, that in this paper all the experiments were made using a quantum simulator which means that all of the possible states had to be processed sequentially. This should not be discouraging, however, because there already exist several real quantum machines with multiple qubits capable of performing computations on all binary states simultaneously. In case of QAOA's time complexity, it is no longer dependent on the instance size but instead it is a function of the parameter p. Even though these are not directly comparable, it is shown in Farhi & Harrow (2019) that it is enough to obtain quantum advantage.



(a) (b)

Fig. 2: (a) Left hand-side: relationship between makespan and energy. Size of the markers are proportional to the probability of measuring a solution with given makespan and energy range. Right hand-side: marginal distribution of energy over the makespan. The top row presents the results for time $T = 5$, while the bottom row: $T = 4$. (b) Time needed for optimization of the variational parameters in the function of length of simulated quantum gate-model circuit. The plot compares two different maximum feasible times T.

4 Conclusions

This paper demonstrates how to efficiently apply QAOA to the JSSP to find the optimal solution and pave the way for solving more complex scheduling problems. We investigated the behaviour of the algorithm in series of experiments and demonstrated the relation between energy and makespan. We also discussed our experiences gained from real computational experiments. All the presented results can be quickly adopted and extended by gate-model application developers, especially for initial testing and experimental verification of new quantum-based approaches for CPU intensive quantum simulations.

Bibliography

Amaro, D., Rosenkranz, M., Fitzpatrick, N., Hirano, K. & Fiorentini, M. (2022), 'A case study of variational quantum algorithms for a job shop scheduling problem', *EPJ Quantum Technology* **9**(1).

Farhi, E., Goldstone, J. & Gutmann, S. (2014), 'A quantum approximate optimization algorithm', *arXiv:1411.4028: Quantum Physics* .

Farhi, E. & Harrow, A. W. (2019), 'Quantum supremacy through the quantum approximate optimization algorithm', *arXiv:1602.07674: Quantum Physics* .

Kurowski, K., Węglarz, J., Subocz, M., Różycki, R. & Waligóra, G. (2020), Hybrid quantum annealing heuristic method for solving job shop scheduling problem, *in* 'Computational Science – ICCS 2020', Springer International Publishing, Cham, pp. 502–515.

Venturelli, D., Marchand, D. J. J. & Rojo, G. (2016), 'Quantum annealing implementation of job-shop scheduling', *arXiv:1506.08479: Quantum Physics* .

Zhou, L., Wang, S.-T., Choi, S., Pichler, H. & Lukin, M. D. (2020), 'Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices', *Physical Review X* **10**(2), 021067.

Assembly Line Performance Analysis Based on Aircraft Preliminary Design: a Scheduling Approach

Anouck Chan[1], Stéphanie Roussel[1] and Thomas Polacsek[1]

ONERA/DTIS, Université de Toulouse, F-31055 Toulouse, France
{anouck.chan,stephanie.roussel,thomas.polacsek}@onera.fr

Keywords: aircraft design, assembly line, RCPSP.

1 Introduction

Conceiving and building a complex product, such as an aircraft, is not only about its specifications but also about how it is built. Because of its complexity, each aircraft has a dedicated industrial system. Thus, when a new aircraft is conceived, so is its associated industrial system. Both designs are interdependent, and design choice of one may impact the design, the performance and the construction of the other.

In the aeronautical domain, the current development cycle is mainly sequential: the aircraft is conceived first, then its means of production is. As described in Polacsek et al. (2017), such an approach raises several issues, and, following the industry 4.0 trend, many recent works follow a simultaneous engineering approach in which both systems (the aircraft and its industrial system) are designed simultaneously. Among others, one challenge raised by such an approach is to allow aircraft designers to estimate the consequences of their choices at different stages of the process with respect to the industrial system.

Recent works address this problem. In Sanchidrián (2019), the author lists several publications that focus on conceiving aircraft manufacturing systems at different stages of the development process. Among those references, in Pralet et al. (2018), the authors evaluate aircraft designs with respect to an existing assembly line.

In this work, we follow the latter approach in order to allow aircraft designers to compare aircraft designs with respect to their assembly line performances. However, we are interested in earlier steps of the development process. We propose a Constraint Programming model of the industrial system at early stages of the process, along with its performance features, and a model of the aircraft designs in terms of high level assembly tasks (Section 2). Then, we show the results associated with two real industrial aircraft designs (Section 3). Finally, in Section 4, we briefly conclude and describe future works.

2 Problem model

2.1 Assembly line presentation

We first describe the main features of an assembly line. We focus on pulsed lines, which are composed of several *workstations*. The aircraft moves from a station to the next at a regular time interval called *takt-time*. Therefore, the total time spent by the aircraft in the assembly line, or *makespan*, is equal to takt-time multiplied by the total number of stations. An assembly line is efficient if it has a short takt-time and a small number of stations (*i.e.* a short makespan). A set of actions (or *activities*) are performed on each station by technicians. These activities are located in specific areas of the aircraft (or *zones*) in which only a maximum number of people can work simultaneously.

In early design stages, the assembly line does not yet exist. More precisely, the number of stations, the takt-time and the makespan are not known. In this work, we focus on the best takt-time and makespan (the smallest ones) that can be reached for a given design.

We also have a limited knowledge about tools and machines that will be available on the assembly line. Indeed, we are only aware of the existence of *dimensioning resources*, *i.e.* resources in limited supply because of their size, their high price or their uniqueness. We also consider possible *incompatibilities* between resources, meaning that some resources cannot belong to the same station.

Note that in this work, as done in Pralet et al. (2018), we do not directly address aircraft architecture design, but consider instead a set of assembly operations that allow to produce the aircraft.

2.2 Constraint Programming approach

In this part, we describe the modelling of the problem, which could be seen as a kind of Resource-Constrained Project Scheduling Problem (RCPSP) extended with resource incompatibilities, neutralisation, hard and simple precedence relations and for which we consider specific criteria.

The assembly line is composed of a set of dimensioning *resources*, denoted $\mathcal{R}$. Each resource $r \in \mathcal{R}$ has a capacity cap_r that represents the number of tasks that can performed simultaneously by r. Then, the aircraft is divided in several *zones*, denoted $\mathcal{Z}$. A *zone* $z \in \mathcal{Z}$ is a physical area of the aircraft. It has a *capacity* cap_z that represents the maximum number of technicians who can work in the zone at the same time.

$\mathcal{A}$ is the set of assembly operations of the aircraft. Each activity $a \in \mathcal{A}$ has a duration d_a and occupy one or more zones of the aircraft. For each activity $a \in \mathcal{A}$ and each zone $z \in \mathcal{Z}$, occ_a^z is the number of technicians required for performing a in z. An activity may *neutralise* a set of zones $neutr_a$. For example, safety inspection operations must be made without any technician passing through the inspected zone, and thus neutralises it. An activity may also require specific machines and tools resources. For all $a \in \mathcal{A}$ and $r \in \mathcal{R}$, $consu_a^r$ represents the consumption of the resources r by the activity a.

We consider two types of *precedence* relation between activities. The first one is $\mathcal{P}_s \in \mathcal{A}^2$ and is called *simple precedence*. If a_i and a_j are two activities in $\mathcal{A}$, $(a_i, a_j) \in \mathcal{P}_s$ means that a_j must start after the end of the a_i. The second type, *hard precedence* is denoted $\mathcal{P}_h \in \mathcal{A}^2$. If (a_i, a_j) belongs to $\mathcal{P}_h$ then a_j has to start at the end of a_i. The precedence graph induced by $\mathcal{P}_s$ and $\mathcal{P}_h$ is assumed to be acyclic.

Finally, we consider an *incompatibility* symmetric relation $\chi \subset \mathcal{R}^2$ that models resources that cannot belong to the same station. Formally, (r_1, r_2) belongs to χ expresses that resources spans of r_1 and r_2 cannot overlap. Resource span of $r \in \mathcal{R}$ is the smallest time interval that contains all intervals of activities consuming r.

In order to formalise this problem with a Constraint Programming approach, for each activity $a \in \mathcal{A}$, we consider a variable $\texttt{start}_a$ whose domain is $[\![0, H]\!] \subset \mathbb{N}$ (where H is a given upper bound of the makespan) and that represents the start date of a. We do not detail here the formalisation of the constraints described along with the model description.

Example 1 (Toy example). Consider the following build process: activities, resources and zones along with their capacities are respectively $\mathcal{A} = \{a_1, \ldots, a_5\}$, $\mathcal{Z} = \{z_1, z_2\}$, $\mathcal{R} = \{r_1, r_2, r_3\}$, $cap_{z_2} = 2$ and $cap_{z_1} = cap_{r_1} = cap_{r_2} = 1$, $cap_{r_3} = 2$. Precedence relations are $P_s = \{(a_2, a_5), (a_3, a_4), (a_1, a_3)\}$, $P_h = \{(a_1, a_2)\}$ and $\chi = \{(r_1, r_3)\}$. Consumption of zones and resources by activities are described on Figure 1. Figure 2 represents a possible activities configuration satisfying all the constraints. Dashed rectangles represent zone neutralisation, for instance, a_2 and a_3 both neutralise z_1 at the same time. Activity a_5 has to wait until the end of a_4 because of incompatibility between resources r_3 and r_1.

	d	z_1	z_2	r_1	r_2	r_3
a_1	2	1	n	1		
a_2	1	n	1	1		
a_3	2	n	1		1	
a_4	1		1	1		
a_5	2	1				2

n stands for zone neutralisation.

Fig. 1: Resources consumption, zones occupation and neutralisation activities in Example 1.

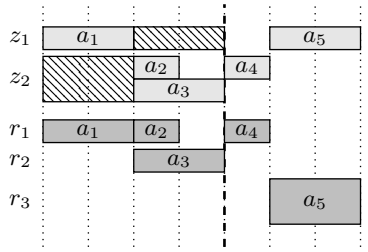


Fig. 2: A build process scheduling

The objective function should take into account the performance elements mentioned earlier, that is minimising makespan and minimising takt-time. We choose to consider the projection of scheduling onto stations by influencing, among other things, the takt-time, first, then the makespan (lexicographic objective function). In order to model the takt-time, we first consider the span interval of each resource. If a resource is attached to a unique station, resources spans will be a lower bound of takt-time. But only considering resource spans is not representative enough as all stations must have the same duration. This is why we fix an arbitrary number of stations and try to dispatch all resources span intervals in at most one station. This corresponds to minimising the number of cuts of resource spans by stations start/end dates. Because of the different relations between elements and the fact that all intervals have the same duration, the existence of a problem schedule without cut for a given scheduling horizon H is not guaranteed. For example, in Figure 2, consider $H = 8$ and a division in 2 stations, which means that takt-time is at most 4. A cut minimisation would put the end of the first station at time 4 (dashed line). In fact, the resource-span of r_1 is cut by this line as it starts at time 0 and ends at time 5. The other spans are not cut, thus the criterion has a value of 1. Once the number of cuts is minimised, we minimise the makespan. In the example, the makespan with one cut is 7.

All constraints and criteria in this model can be encoded using intervals and modelling structures of IBM ILOG Optimization Programming Language (OPL).

3 Experiments

The proposed model has been applied on two real industrial data sets depicting similar aircrafts, namely Design1 and Design2. In these two data sets, we have a set of 5 dimensioning resources composed of two types of machines: the first type is constituted by robots (R1 and R2) and the second by specific tools (T1, T2 and T3). They all have a capacity equal to 1 but note that it would be possible to increase it (done for some experiments not presented in this paper). Robots and specific tools have an incompatibility relation ($(R_i, T_j) \in \chi$ for all $i \in [1, 2]$ and $j \in [1, 3]$). We have about 180 tasks, 48 zones with capacity 1 by data set. Design1 has about 700 precedence relations and Design2 about 300.

For each experiment, we give the algorithm a maximum time limit of 60 seconds for all experiments except for the division into seven or more stations that were given a limit of 120 seconds. In addition, each result presented in the abstract has an optimal gap lower than 0.15 for cases with less than eight stations, lower than 0.25 for the eight stations case. The scheduling horizon H is equal to the sum of all activities durations. Such a value was large enough to allow the dispatch of resources to stations without any cut by resource span.

The experiments were run with CpOptimizer 20.1 on 20-core Intel(R) Xeon(R) CPU E5-2660 v3 2.60GHz, 64GB RAM, Ubuntu 18.04.5 LTS.

Table 1: Results in *takt-time* by number of stations

k	**4**	**5**	**6**	**7**	**8**
Design1	9h54'12	8h53	6h50'15	5h51	5h30
Design2	9h51	9h7'12	9h7'12	9h7'12	9h7'12

Table 1 represents the results obtained on our data sets by varying the number of stations from 4 to 8. We can notice that with the exception of the 4 stations configuration, Design1's takt-times are better than those associated with Design2. In a previous experiment, not presented in this paper, we have calculated for each Design, the overall takt-time lower bound from resource span. Their values are 5h30 for Design1 and 9h7'12 for Design2. Thus, we can notice that Design1 reaches its lower bound in the 8 stations configuration, while Design2 in the 5 stations configuration. So, unless we manage to reduce these bounds we can not anymore decrease either takt-time or makespan. Depending on the number of stations chosen, the two designs are Pareto optimal when considering takt-time and makespan.

4 Conclusion and future works

In this paper, we have proposed a scheduling-based analysis tool that allows us to evaluate the preliminary aircraft design against its performance on a pulsed assembly line. The experiments along with the industrial feedback show that the approach is promising.

In some experiments not presented in this paper, we have tested different assembly line configurations in which activities could choose among various resources of the same type (*e.g.* two robots of type $R1$). It strongly improves the results of Design2. Thus, a first future work could be to extend the model to add flexibility in the choice of resources *(RCPSP with modes)*. Another way to continue this work would be to consider costs associated with the number of stations and takt-time in order to differentiate solutions that are Pareto optimal in our current approach. Then, we have allowed activities to overlap on more than one station, but in real world instances, some may require to be performed in a single station. It might therefore be necessary to add this constraint in future models. Finally, the addition of an uncertainty on activities' duration could allow us to propose a more robust schedule with respect to the assembly line.

References

Polacsek, T., Roussel, S., Bouissière, F., Cuiller, C., Dereux, P. & Kersuzan, S. (2017), Towards thinking manufacturing and design together: An aeronautical case study, *in* 'Conceptual Modeling - 36th International Conference, ER 2017, Proceedings', Vol. 10650 of *Lecture Notes in Computer Science*, Springer, pp. 340–353.

Pralet, C., Roussel, S., Polacsek, T., Bouissière, F., Cuiller, C., Dereux, P., Kersuzan, S. & Lelay, M. (2018), A scheduling tool for bridging the gap between aircraft design and aircraft manufacturing, *in* 'Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS', AAAI Press, pp. 347–355.

Sanchidrián, T. B. (2019), Scheduling with limited resources along the aeronautical supply chain: from parts manufacturing plants to final assembly lines, PhD thesis, Industriales.

A fix-and-optimize heuristic for the resource renting problem

Max Reinke and Jürgen Zimmermann

Clausthal University of Technology, Germany
max.reinke@tu-clausthal.de, juergen.zimmermann@tu-clausthal.de

Keywords: project scheduling, resource renting, fix-and-optimize heuristic.

1 Introduction

The resource renting problem with general temporal constraints (RRP/max) was introduced by Nübel (2001), and is an extension of the well-known resource availability problem (RAP). The objective for the RRP/max is to minimize the total resource costs for a project with general temporal constraints. Typically, each activity in a project uses resources during its execution. In many cases, those resources have to be rented, for example, heavy machinery on construction sites. The renting of resources induces two types of costs, namely time-independent procurement cost and time-dependent renting cost. In this paper, we present a MIP based fix-and-optimize heuristic for the RRP/max. In Section 2 the RRP/max is described formally. Section 3 presents our fix-and-optimize heuristic. Finally, in Section 4 the results of a computational study for the heuristic are presented.

2 Problem description

We assume that the underlying project is given as an activity-on-node network $N = \langle V, E; \delta \rangle$, where the nodes V represent the activities and arcs $E \subseteq V \times V$ represent general temporal constraints between activities. For every arc $\langle i, j \rangle \in E$ a weight δ_{ij} is given. The prescribed project deadline $\bar{d}$ is observed by the arc weight $\delta_{n+1,0} = -\bar{d}$. A sequence of start times S_i for all activities $i \in V$ represents a schedule $S = (S_0, S_1, \ldots, S_{n+1})$, which is called time-feasible if $S_j - S_i \geq \delta_{ij}$ for all $\langle i, j \rangle \in E$. For activity i all time-feasible start times are given by $W_i = \{ES_i, ..., LS_i\}$ with the earliest start time ES_i and latest start time LS_i. Moreover, each activity $i \in V$ is assigned a duration $p_i \in \mathbb{Z}_{\geq 0}$ and has to be performed without preemption. We suppose that renewable resources $k \in \mathcal{R}$ are used to perform the project, where the execution of activity i needs an amount $r_{ik} \in \mathbb{Z}_{\geq 0}$ of resource $k \in \mathcal{R}$. Examples for this type of resources are machines or workers. They are not depleted when used for executing an activity but can be assigned for the duration p_i and then used again later for different activities. We assume all resources have to be rented, and resources can only be used while they are rented and thus available to the project. Associated with renting resources are two types of costs, procurement cost c_k^p arise every time additional resource units are obtained and represent for example setup costs or overhead. The second type of cost are the time-dependent renting cost c_k^r per unit and period, those cost represent the rate for which one unit of resource k can be rented for a time period. A resource is available at point in time t, if it was procured in the interval $[0, t]$ and availability of one unit of resource k for t periods incurs cost of $c_k^p + t \cdot c_k^r$. The resource availability is determined by a renting policy $\varphi_k(S, t)$, which specifies when resources are procured or released. The policy must be determined such that $\varphi_k(S, t) \geq \sum_{i \in V | S_i \leq t < S_i + p_i} r_{ik}$ for all $k \in \mathcal{R}$ and $t \in T$ holds. The objective of the RRP/max is to find a time-feasible schedule and a renting policy that minimize the total costs incurred by renting resources. For the case $c_k^p < c_k^r$ an optimal renting policy is to procure additional resources when the resource demand has a

positive step discontinuity at t and to release idle resources immediately, since procuring new resources is less expensive than keeping unused resources available. In general $c_k^p > c_k^r$ holds, here procurement costs of a resource k are higher than renting costs for one period. In this case, we can define $span = \lfloor c_k^p / c_k^r \rfloor$ as the maximum number of time periods for which it is beneficial to hold unused resources if they are used again later in the project.

For the RRP/max we use a time-indexed formulation with binary variables x_{it} for all $i \in V$ and $t \in T$, where $x_{it} = 1$ if activity i starts at point in time t. To model the resource demand, we use the variable z_{kt} representing the amount of resource $k \in \mathcal{R}$ needed at $t \in T$. The renting policy $\varphi_k(S,t)$ is modeled using positive variables a_{kt} and w_{kt}, which represent the number of units of resource k procured or released at point in time t. With the defined variables, the following MILP can be formulated.

$$\text{Min} \quad \sum_{k\in\mathcal{R}} c_k^p \sum_{t\in T} a_{kt} + \sum_{k\in\mathcal{R}} c_k^r \sum_{t\in T} t \cdot (w_{kt} - a_{kt}) \tag{1.1}$$

$$\text{s.t.} \quad \sum_{t\in W_i} x_{it} = 1 \qquad (i \in V) \tag{1.2}$$

$$\sum_{\tau=t}^{LS_i} x_{i\tau} + \sum_{\tau=ES_j}^{\min\{LS_j, t+\delta_{ij}-1\}} x_{j\tau} \leq 1 \qquad (\langle i,j\rangle \in E, t \in T) \tag{1.3}$$

$$\sum_{i\in V} r_{ik} \sum_{\tau=\max\{ES_i, t-p_i+1\}}^{\min\{t,LS_i\}} x_{i\tau} \leq z_{kt} \qquad (k \in \mathcal{R}, t \in T) \tag{1.4}$$

$$\sum_{\tau=0}^{t} (a_{k\tau} - w_{k\tau}) \geq z_{kt} \qquad (k \in \mathcal{R}, t \in T) \tag{1.5}$$

$$\sum_{t\in T} a_{kt} - \sum_{t\in T} w_{kt} = 0 \qquad (k \in \mathcal{R}) \tag{1.6}$$

$$a_{kt}, w_{kt}, z_{kt} \in \mathbb{Z}_{\geq 0} \qquad (k \in \mathcal{R}, t \in T) \tag{1.7}$$

$$x_{it} \in \{0,1\} \qquad (i \in V, t \in W_i) \tag{1.8}$$

Constraints (1.2) state that every activity has to be started exactly once. (1.3) ensure that all temporal restrictions between activities are satisfied. We use the formulation of Christofides *et al.* (1987), in the form used by Rieck *et al.* (2012) for general temporal constraints. Lower bounds on the value of variables z_{kt} are assigned by constraints (1.4). The available (rented) resources at time t are given by $\sum_{\tau=0}^{t}(a_{k\tau} - w_{k\tau})$, hence inequalities (1.5) ensure a feasible renting policy $\varphi_k(S,t) \geq z_{kt}$ for all $k \in \mathcal{R}$ and $t \in T$ is obtained. All added resources have to be withdrawn by the end of the project, see (1.6). Also, additional restrictions on the renting policy are used as proposed in Reinke and Zimmermann (2020).

3 A fix-and-optimize heuristic

To solve the RRP/max, we propose a fix-and-optimize heuristic, based on the basic principle presented in Sahling *et al.* (2009). The general procedure is to iteratively create and solve a predetermined number of subproblems $l \in \mathcal{L}$. For a given schedule S the values for x_{it} can be set, resulting in the problem to obtain an optimal renting policy, which can be solved with the algorithm provided by Nübel (2001) with a time complexity of $\mathcal{O}(n \log n)$. When solving the MILP for an instance of the RRP/max, determining the binary scheduling variables x_{it} requires most of the computational effort. In general, the fix-and-optimize heuristic, outlined in Algorithm 1, aims to define subproblems with less binary decision variables to consider, which are therefore easier to solve. For each subproblem a set of activities I_{opt} is chosen to be optimized. From this we derive the set of fixed activities $I_{fix} \coloneqq V \setminus I_{opt}$, for which the corresponding binary scheduling variables x_{it} are set to fixed values $\bar{x}_{it}$. The fixed values are taken from the optimal solution of the preceding subproblem. This reduces the number of activities to be scheduled in each iteration to $|I_{opt}| < |V|$. A subproblem is constructed by adding the following constraints to the MILP.

$$x_{it} = \bar{x}_{it} \quad (i \in I_{fix}, t \in T) \tag{1.9}$$

Note the real-valued variables z_{kt}, a_{kt} and w_{kt} are not fixed. A standard MIP solver is then used to solve the subproblem to optimality. The next subproblem is defined by choosing a new set I_{opt}.

Algorithm 1: General algorithm for the heuristic

```
begin
    Generate a starting solution S_start = (S, a, w);
    Set l := 1;
    while l ≤ L do
        Choose I_opt ;
        Set I_fix := V \ I_opt ;
        for i ∈ I_fix do
            Set x_it = x̄_it ∀(i, t);
        Solve RRP-SUB using CPLEX;
        Set I_opt = ∅ and I_fix = ∅;
        Set l := l + 1;
    Return locally optimal solution (S, a, w) ;
```

A starting solution (S_{start}, a, w) for the fix-and-optimize heuristic is obtained by a priority-rule based construction heuristic. First all activities for which $TF_i = LS_i - ES_i = 0$ are scheduled at their earliest start time. Then an optimal renting policy (a, w) for the corresponding partial schedule $S^{\mathcal{C}}$, where $\mathcal{C}$ is the set of already scheduled activities, is computed. While not all activities are scheduled, the next i from the set of unscheduled activities $\bar{\mathcal{C}}$ is determined by a priority-rule. Moreover, $S_i = t^*$ is set, where t^* is some point in time for which the total resource cost are minimized for the new partial schedule, including i, with the corresponding renting policy. The optimal solution for the RRP/max can be found among the set of quasistable schedules (Neumann *et al.* (2003)), thus only t^* which lead to a quasistable schedule have to be considered. We denote $ES_i(S^C)$ ($LS_i(S^C)$) as schedule dependent earliest (latest) start time and respectively $W_i(S^C) = \{ES_i(S^C), ..., LS_i(S^C)\}$ as set of possible start times. Also $ST(S^C)$ contains all points in time t where an activity starts in S^C and $CT(S^C)$ where an activity is completed. With this the set of decision points is given by $D_i(S^C) := \{ES_i(S^C), LS_i(S^C)\} \cup \{W_i(S^C) \cap CT(S^C)\} \cup \{t \in W_i(S^C) | t + p_i \in ST(S^C)\}$, to obtain a quasistable schedule. When constructing a starting solution we examined the priority-rules GRD, GRR and MST as found in Neumann *et al.* (2003). Preliminary test showed a good starting solution generally yields better results. GRD provided the best starting solutions for most instances.

To choose the m_{opt} activities incorporated in set I_{opt}, we devised three strategies named *Random*, *Add* and *Tree*. For the first strategy *Random* a number of m_{opt} activities are chosen at random from set V of all activities. Maximal time lags can result in cases where $|W_i(S)| = 1$, then the starting time of an activity is constraint to one point in time t. These activities can be identified by their schedule dependent total float time $TF_i(S) = 0$. In the second strategy *Add*, m_{opt} activities are taken at random from among eligible activities I_{elig}, where initially an activity i is eligible if $TF_i(S) = 0$. Every time an activity i is chosen, I_{elig} is updated to include all activities j for which $S_j - S_i = \delta_{ij} | (j \in Succ_i \cap j \in I_{opt})$ or $|S_i - S_j = \delta_{ij}| (j \in Pred_i \cap j \in I_{opt})$ holds in the current solution. The third strategy uses the property of quasistable schedules to be represented by a tree of binding temporal constraints where $S_j - S_i = \delta_{ij}$ and schedule induced precedence constraints. Here we only consider binding temporal constraints, which results in a number of disjointed subtrees in N. When an activity i is chosen randomly from I_{elig}, all activities belonging to the same subtree are also added to I_{opt}.

4 Performance analysis

To fathom the performance of our fix-and-optimize algorithm a computational study was conducted. The heuristic was implemented in C++ where the subproblems are solved by using *IBM CPLEX* v.12.8.0, on a computer with an Intel Core i7-7700 with 4.2 GHz and 64 GB RAM under Windows 10. As problem instances we used adaptations of the well-known benchmark test set UBO (Schwindt 1998) where we introduced a project deadline $\bar{d} = \alpha \cdot ES_{n+1}$ with $\alpha = \{1, 1.25, 1.5\}$ and procurement costs $c_k^r = CQ \cdot c_k^p$ with $CQ = 0.25$. For every combination of parameters α and CQ, 30 instances with $n = \{20, 50\}$ and an order strength of 0.5 were tested. Some results of our study are shown in Table 1. The given *gap* is the average deviation of the found solutions compared to the best solution found by *CPLEX* after 600s for $n = 20$ and 3600s for $n = 50$, for all 30 instances with the same parameter combination. For all runs, based on preliminary tests, $\mathcal{L} = 20$ was chosen. Generally we can observe a good performance of the heuristic across all tested instances. Especially for the instances with $n = 50$ and $\alpha = 1, 5$, the fix-and-optimize heuristic greatly outperformed CPLEX by obtaining significantly better solutions in a reasonable time. Of the tested selection strategies for I_{opt}, we can see the best performance regarding the *gap* from the strategy *Tree*. Concerning the number of activities in I_{opt} we tested $|I_{opt}| = 6$ and $|I_{opt}| = 10$ and found the larger number to yield better *gap* but also required greatly increased solution times.

Table 1. Results of the computational study

			Random		*Add*		*Tree*		*Random*		*Add*		*Tree*	
			$\|I_{opt}\| = 6$		$\|I_{opt}\| = 6$		$\|I_{opt}\| = 6$		$\|I_{opt}\| = 10$		$\|I_{opt}\| = 10$		$\|I_{opt}\| = 10$	
n	α	CQ	*gap*	*time*	*gap*	*time*	*gap*	*time*	*gap*	*time*	*gap*	*time*	*gap*	*time*
20	1	0,25	0,6	4,0	0,7	4,3	0,6	3,0	0,1	25,6	0,1	20,6	0,1	9,4
20	1,25	0,25	3,9	6,7	3,9	7,0	2,9	13,5	2,5	87,8	2,7	89,9	1,6	98,1
20	1,5	0,25	3,9	14,8	3,7	12,5	2,0	94,2	1,6	137,2	2,5	260,9	1,2	172,5
50	1	0,25	2,4	11,4	2,5	12,8	2,3	19,5	1,8	32,4	1,7	33,8	2,0	28,2
50	1,25	0,25	2,3	19,5	2,5	19,3	1,6	89,0	1,4	54,2	1,1	52,5	1,1	170,0
50	1,5	0,25	-6,6	33,5	-6,8	37,8	-7,2	278,0	-7,6	95,2	-7,6	150,6	-8,0	363,4

References

Christofides N., Álvarez-Valdés R., Tamarit J.M., 1987, "Project scheduling with resource constraints: A branch and bound approach", *European Journal of Operational Research*, Vol. 29, pp. 262-273.

Neumann K., Schwindt C., Zimmermann J., 2003, "Project Scheduling with Time Windows and Scarce Resources", Springer, Berlin Heidelberg, 2nd edition.

Nübel H., 2001, "The resource renting problem subject to temporal constraints", *OR Spectrum*, Vol. 23, pp. 359-381.

Rieck J., Zimmermann J., Gather T., 2012, "Mixed-integer linear programming for resource leveling problems", *European Journal of Operational Research*, Vol. 221, pp. 27-37.

Reinke M., Zimmermann J., 2020, "A Comparison of two MILP formulations for the resource renting problem", *Accepted Paper at the 17th International Conference on Project Management and Scheduling*. Available on: https://pms2020.sciencesconf.org/298538/document

Sahling F., Buschkühl L., Tempelmeier H., Helber, S., 2009, "Solving a multi-level capacitated lot sizing problem with multi-period set up carry-over via a fix-and-optimize heuristic", *Computers & Operations Research*, Vol. 36, pp. 2546-2553.

Schwindt C., 1998, "Generation of resource-constrained project scheduling problems subject to temporal constraints", *Technical Report WIOR-543*, University of Karlsruhe.

Extending Smith's Rule with Task Mandatory Parts and Release Dates

Bonnin Camille[1,2], Nattaf Margaux[1], Malapert Arnaud[2] and Espinouse Marie-Laure[1]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP**, G-SCOP, 38000 Grenoble, France
camille.bonnin@grenoble-inp.fr, marie-laure.espinouse@grenoble-inp.fr, margaux.nattaf@grenoble-inp.fr
[2] Université Côte d'Azur, CNRS, I3S, France
arnaud.malapert@univ-cotedazur.fr

Keywords: 1-Machine Scheduling, Flow Time, Mandatory Parts, Release Dates, Preemption.

1 Context and problem description

Smith's rule is well-known in scheduling for solving the one-machine problem in which the flow time is minimized, i.e. $1||\sum C_j$. A first extension of this rule including release date and allowing preemption has been developed in Brucker P. (2006). This rule is called the Modified Smith Rule (MSR). This paper aims to extend MSR to include the notion of task mandatory part, a key concept in Constraint Programming (CP). This extension aims at computing lower bounds for the more general problem $1|r_j;d_j|\sum C_j$ and to use it to improve the solving of this problem by CP.

CP is, currently, a well established method to solve scheduling problems as shown in Baptiste P. et al. (2001). However, constraint programming techniques are oriented toward feasibility and makespan minimization. Recently, some efforts have been made to integrate other objective functions such as the maximum lateness, or the weighted flowtime. Recently, a lot of efforts have been made to integrate several objective functions such as the maximum lateness, or the weighted flow time like Kovács A. and Beck J. C. (2011). Still, few of them consider the flow time as objective.

One crucial notion in CP is the task mandatory part. A Mandatory Part (MP) is the time interval in which the non-preemptive task is executed in each feasible solution of the problem. It is defined as the intersection between the time interval where the task starts at its release date and the time interval where the tasks ends at its deadline. In this intersection (if it is not empty), the task is sure to be executed in each feasible solution of the problem. Typically, in CP, task time-windows narrow during the solving. Thus, the MP size increases.

A key concept in CP is filtering. Filtering consists in removing value from the domain of a variable that will not lead to feasible solutions. In practice, this process is used very often during the solving procedure. Thus, the algorithm used in the filtering process must be as quick as possible. As shown in Nattaf M. and Malapert A. (2020), it is possible to use lower bounds to design filtering algorithms. Therefore, it is interesting to compute lower bounds for $1|r_j;d_j|\sum C_j$ in polynomial time. Since it is an $\mathcal{NP}$-hard problem, we cannot find a polynomial algorithm to solve it. Thus, to find efficient lower bounds for $1|r_j;d_j|\sum C_j$, we need to consider its polynomial relaxations. Among those relaxations, only two can be solved in polynomial time, $1|r_j;pmtm|\sum C_j$ and $1|sp-graph|\sum w_jC_j$. In this paper, the first relaxation is considered. Then, to improve the lower bounds for $1|r_j;d_j|\sum C_j$ a new constraint is added to the relaxed problem: the mandatory part constraint (MP). This constraint ensures that the MPs of the tasks are respected.

** Institute of Engineering Univ. Grenoble Alpes

2 Modified Smith's Rule with Mandatory Parts (MSRMP)

There already exist algorithms that compute a lower bound for $1|r_j;d_j|\sum C_j$ in polynomial time. One of those algorithms is the modified Smith rule given by Brucker P. (2006) that solves $1|r_j;pmtn|\sum C_j$. This rule is an adaptation of the Smith rule and states that at each release date and at each competition time (C_j) of a task, the available task with the shortest remaining processing time must be scheduled (a task can then be interrupted by the release of a quicker task).

In this section, we are describing a rule to find the optimal solution of a relaxation of $1|r_j;d_j|\sum C_j$, $1|r_j;pmtn;MP|\sum C_j$. This rule is based on the modified Smith rule in which we are adding the execution of the mandatory parts (MP) of the tasks. It is called the Modified Smith's Rule with Mandatory Parts (MSRMP).

Rule 1 (Modified Smith's Rule with Mandatory Parts (MSRMP)) *Schedule the mandatory parts (MPs) in the correct time slots. At each release date or completion time of a task, schedule an unfinished task that is available and has the shortest remaining processing time. In case of equality, schedule the unfinished task without MP or with the earliest MP.*

Example 1.
Table 1 and Figure 1 present an example of the application of the MSRMP with six tasks. The MPs are in gray. To execute this rule, we begin by computing and scheduling the MPs and then, we apply the modified Smith rule on the rest by taking care of the equality cases. We can notice that even if tasks 3 and 5 are finished on time, task 1 is too long and finishes late, a long time after its MP.

Table 1: Example the application of the MSRMP - instance-

j	1	2	3	4	5	6
$\mathbf{p_j}$	9	5	2	3	6	3
$\mathbf{r_j}$	0	0	2	9	9	14
$\mathbf{d_j}$	15	∞	4	∞	17	∞

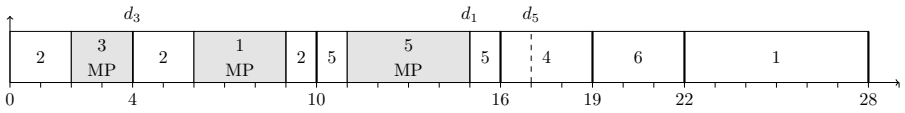


Fig. 1: Example of application of the MSRMP

Proposition 1 (Optimality of the Modified Smith Rule with Mandatory Parts). *The Modified Smith Rule with Mandatory Parts gives an optimal solution to the* $1|r_j;pmtn;MP|\sum C_j$ *problem.*

Proof. Clearly, the rule gives a feasible solution to the $1|r_j;pmtn;MP|\sum C_j$ problem.

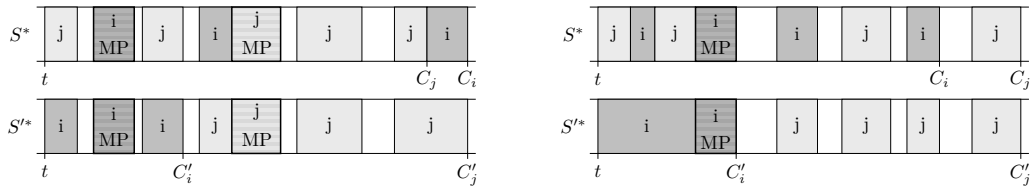
The proof relies on an exchange argument and is similar to the one of Theorem 4.9 in Brucker P. (2006). Therefore, we only present the exchange argument which is different here.

Let us assume that for the same feasible instance of $1|r_j;pmtn;MP|\sum C_j$, S is the schedule obtained with MSRMP and S^*, an optimal one. Those two schedules are identical until time t, where task i is scheduled in S, and task j in S^*. We then construct a new schedule S'^* based on S^* by re-scheduling, after t, the non-mandatory parts of i before those of j. As $r_j \leq t$ and $r_i \leq t$, S'^* is a feasible schedule. Lemma 1 shows that S'^* is still optimum. We then take S'^* as S^*, update t and redo the same reasoning until S'^* and S are the same. We then have proven the optimality of S.

Lemma 1 (Optimality of the exchange argument) *Let assume S^* and S'^* are constructed as in the proof of Proposition 1, the objective value of S'^* is no greater than those of S^*.*

We define $\widehat{p_k}$ as the remaining processing time of a task k, p_k its processing times, C_k and C'_k its completion time respectively in S^* and S'^*, and $[lst_k, eft_k[= [d_k - p_k, r_k + p_k[$ its MP.

Proof. We need to consider three cases based on the presence and the position of the MPs of i and j after t. We can notice that by the definition of an MP, there are executed at simultaneously in S^* and S'^* and so cannot be executed at time t.



(a) Case 2 (b) Case 3

Fig. 2: Examples of the cases 2 and 3 of the proof

Case 1 : neither i nor j has an MP after t. It is the modified Smith rule.

Case 2 : the mandatory part of i and/or j is in the middle of its execution. This case is represented by Figure 2a. By definition, the MPs of i and j are executed simultaneously in S^* and in S'^*. Also, as the MP of i (resp. j) is in the middle of the execution of i (resp. j) in S'^*, it has no influence on C'_i (resp. C'_j). So we can ignore the MP of i (resp. j) and return to case 1 or 3 depending on the situation.

Case 3 : i and/or j ends by its MP in S'^*. If j has an MP located at the end of its execution, as the MP of j is scheduled simultaneously in S^* and S'^*, we can deduce that $C'_j = C_j = eft_j$. By construction, we have that $C'_i \leq C_i$ so the objective value of S'^* is no greater than those of S^*. If, as in Figure 2b, it is i that have its MP at the end of its execution in S'^*, then by definition of the MP, if i finishes at eft_i, then it starts at r_i and is executed without preemption. So in S'^*, i is processed in $[t, eft_i[$. As S^* and S'^* differ after t, at least one part of i is scheduled after its MP in S^*, so $C_i > C'_i = eft_i$. For $C'_j = \max(C_i, C_j)$, if $C'_j = C_j$, then S'^* is strictly better than S^*, which contradict the optimality of S^*. If $C'j = C_i$, then $\widehat{p_i} < \widehat{p_j}$. Indeed, $\widehat{p_i}$ cannot be equal to $\widehat{p_j}$ as otherwise j would have been scheduled before i in S (if j has an MP after t, it is after those of i, else i could not be scheduled in all $[t, eft_i[$). Thus $\widehat{p_j} > \widehat{p_i}$ and then :

$$\begin{aligned} \triangle_{F_T} &= C_i - C'_i + C_j - C'_j \\ &= \cancel{C_i} - C'_i + C_j - \cancel{C_i} \\ &= C_j - C'_i \\ &= C_j - eft_i \end{aligned}$$

and $\triangle_{F_T} \geq 0$ as $C_j > eft_i$ ($\widehat{p_j} > \widehat{p_i}$). It is also impossible to have i and j finishing together by their MP as otherwise, they will be processed simultaneously on $[t, \min(eft_i, eft_j)[$ which is impossible as there is only one resource.

The MSRMP can be implemented in polynomial time. An example of such implementation, whose complexity is $O(n \log n)$, is a sweep line algorithm. Indeed, each task generates one event linked with its release date, and two events linked with the start and end of its mandatory part (if any). Then, the events are sorted in non-decreasing order where ties are broken by the earliest start of the mandatory part. Finally, events are processed sequentially so that a preemptive schedule is built under MSRMP. The worst-case complexity of

this algorithm is coming from the events sort and the selection of the task with the shortest remaining processing time.

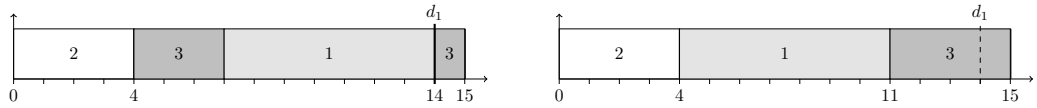
3 Non optimality of MSRMP with deadlines (d_j)

MSRMP computes in polynomial time a better lower bound of $1|r_j;d_j|\sum C_j$ than $1|r_j;pmtn|\sum C_j$ as $1|r_j;pmtn|\sum C_j$ is a relaxation of $1|r_j;pmtn;MP|\sum C_j$. However, its direct adaptation to include the deadlines by prioritazing tasks that just have the time to finish is not optimal, as shown by this example.

Example 2.
Table 2 and Figure 3a present an example of the application of the MSRMP with deadlines and three tasks. This adaptation of MSPMP is trivial: if a task has just the time to finish before being late, it has higher priority than all the other tasks. Figure 3a gives us the objective value of the MSRMP with deadlines on the instance of Table 2: it is 33 (0 + 4 + 14 + 15). However, Figure 3b gives a better solution for the same problem as its objective value is 30 (4 + 11 + 15). As Figure 3b is not following the MSRMP with deadlines, we can deduce that this rule is not optimal.

Table 2: Example the application of the MSRMP with d_j -instance-

j	1	2	3
$\mathbf{p_j}$	0	0	0
$\mathbf{r_j}$	7	4	4
$\mathbf{d_j}$	14	∞	∞



(a) MSRMP with d_j (b) Optimal solution

Fig. 3: Example of the non optimality of MSRMP with d_j

4 Conclusion and prospects

In this paper, we extended Smith rule to $1|r_j;pmtn;MP|\sum C_j$. It can solve this problem in polynomial time and so be used as a lower bound for $1|r_j;d_j|\sum C_j$ to improve its CP filtering algorithms. We also show that the adaptation of the rules to include the deadlines is not trivial. Even though, $1|r_j;d_j|\sum C_j$ and $1|r_j;d_j;pmtn|\sum C_j$ are $\mathcal{NP}$-hard, finding an algorithm which takes the deadline into account in a better way than ours is a challenging research direction.

References

Baptiste P., Le Pape C. and Nuijten W., 2001, "Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problem", *International Series in Operations Research & Management Science*, Vol. 39, Verlag Springer US, pp. xiii-198, https://doi.org/10.1007/978-1-4615-1479-4.

Brucker P., 2006,"Single Machine Scheduling Problem", *Scheduling Algorithms*, Springer, Berlin, Heidelberg, ch. 04, pp. 61–106, https://doi.org/10.1007/978-3-540-69516-5.

Kovács A. and Beck J. C., 2011, "A global constraint for total weighted completion time for unary resources ", *Constraints*, Vol. 16, No. 1, pp. 100–123, https://doi.org/10.1007/s10601-009-9088-x.

Nattaf M. and Malapert A., 2020, "Filtering Rules for Flow Time Minimization in a ParallelMachine Scheduling Proble", *CP 2020: Principles and Practice of Constraint Programming*, pp. 462-477, https://doi.org/10.1007/978-3-030-58475-7_27, https://hal.archives-ouvertes.fr/hal-3013857.

A realistic hybrid flow shop scheduling problem with availability restrictions, priorities, and machine qualifications

Christin Schumacher[1] and Dominik Mäckel[2]

[1] TU Dortmund University, Department of Computer Science - Modeling and Simulation, Germany christin.schumacher@tu-dortmund.de
[2] TU Dortmund University, Department of Computer Science - Modeling and Simulation, Germany dominik.maeckel@tu-dortmund.de

Keywords: heuristics, machine availability, priority, machine qualification, hybrid flow shop, makespan.

1 Introduction

While real-world problems in practice contain numerous restrictions, problems in the literature mostly consider a few specifications at a time [1]. Our problem of an automotive supplier includes two production stages, where both stages contain unrelated machines ($FH2,\ ((RM^{(k)})_{k=1}^{2})$). Machine qualifications have to be considered (M_j), jobs might skip a stage (*skip*), priority groups of jobs need to be respected (*prec*), machines might not be available at beginning the production (rm), and machines have times of unavailability (*unavial*) [2, 3]. The objective is to minimize makespan C_{max}. According to Graham et al. [4], our problem can be categorized as follows:

$$FH2,\ ((RM^{(k)})_{k=1}^{2}) \mid M_j, prec, skip, rm, unavail \mid C_{max} \tag{1}$$

In the following, heuristics and metaheuristics are compared to determine the best solution method, concerning short computation times. Furthermore, a mixed-integer problem (MIP) is used for comparison. All developed methods are tested on historical data of the use case so that the most promising algorithms for the company can be found.

2 Related Work

Even for the problem of two identical parallel machines on one stage, and one machine on the other stage, Gupta [5, pp. 359–360] has proved *NP-hardness*. So hybrid flow shop problems are considered as *NP-hard*. Due to high computation times for *NP-hard* problems, MIP models, even for small instances of jobs, are often not applicable in practice. However, they are useful for benchmarks.

To the best of our knowledge, no study has developed a heuristic algorithm or a MIP for our problem (1). In their literature overviews Ribas, Leisten, and Framiñan [6], Komaki, Sheikh, and Malakooti [7], and Ruiz and Vázquez-Rodríguez [2] mention several hybrid flow shop problems. Listed problems of papers conform at most with one or two restrictions to M_j, *prec*, *skip*, *rm* or *unavail*. The problem which is most familiar to our application case is provided by Ruiz, Şerifoğlub, and Urlings [1] and is at the same time one of few papers dealing with six realistic restrictions in one hybrid flow shop environment. But, since our application case differs in several restrictions from the mentioned papers, algorithms in literature cannot be applied for (1).

3 Heuristic and metaheuristic methods

For the presented scheduling problem, eight priority rules, one genetic algorithm (GA), twelve local search strategies, and four simulated annealing variants are developed. Applied priority rules are extensions of Shortest Processing Time (SPT) [3], Longest Processing Time (LPT) [3], Johnson's algorithm and NEH [1]. For second stage scheduling, priority rules are combined with Earliest Completion Time (ECT) or job-based relation (JBR). ECT selects the job with the shortest completion time from all jobs to be scheduled and JBR preserves the permutation of jobs from the first stage. All developed priority rules use ECT for machine assignment at the stages, where the algorithm successively schedules the next element of an ordered list of jobs to the machine that can complete the job first and is available at the dedicated time and eligible for the job. Each priority group gets scheduled separately beginning with the highest priority.
Local search strategies have been tested in six variants:

- The neighborhood can be explored by generating a defined amount of moves from the current schedule and selecting the solution with best objective value, in case of improved makespan (Steepest Descent) or by selecting a random neighbor and accepting the solution, in case of improved makespan (Random Descent).
- The neighborhood can be build by "shift" and "swap" moves. "Shift" moves a random job to a random position in the schedule. "Swap" selects two random jobs and exchanges their positions. Both operations preserve machine qualifications and priority groups.
- Steepest descent can be executed using a tabu list that contains a finite set of prohibited moves to solutions that have already been visited. Tabu search also accepts solutions with equal C_{max}.
- Again, for all variants, ECT or JBR can be chosen for scheduling the second stage (see priority rules).

Simulated Annealing is expanded from the algorithm of Kirkpatrick, Gelatt, and Vecchi [8]. It enables escaping from local minima by accepting solutions with larger C_{max} by a given probability defined by the current temperature that is decreasing over the optimization process. It is also combined with neighborhood strategies "shift" or "swap". ECT or JBR can be applied to schedule the second stage.

In addition, genetic algorithm (GA) by Wu, Liu, and Wu [9] is modified. If a crossover or mutation operation results in a non-feasible chromosome, the chromosome is reset after the particular operation and the further algorithm again uses the original instance. To ensure priority orders, priorities are ensured in each operation.

4 Results

Developed scheduling heuristics are applied to the historical production data of the company. The first step to evaluate the algorithms is to determine the benchmark using a developed MIP by the authors. For our MIP model, we have extended the model by Ruiz, Şerifoğlub, and Urlings [1] to the constraints *unavail* and *prec*. Even after 3 days of computation time, the optimality of some solutions for weeks with high production volume cannot be proven via the MIP. On average of evaluated weeks, the gap is about 13 %. As well, a few weeks are tested with a maximum computation time of 14 days. Despite the high additional computation time, no significant improvements are generated.

Also, results of heuristics and metaheuristics did not provide better C_{max}-values. Thus, it can be concluded that the benchmarks of MIP are almost optimal. In the following, heuristics and metaheuristics are compared to benchmarks in order to measure their performance as follows: $\Delta C_{max} = \frac{heuristic\ solution\ -\ MIP\ solution}{MIP\ solution}$.

In contrast to MIP, all heuristic algorithms need less than one minute to provide solutions. Using ΔC_{max}, priority rules are compared over 20 selected calendar weeks (Figure 1). Besides, Table 1 calculates averages for all weeks. The evaluation shows, that SPT has the worst performance with both second stage strategies and NEH is the best priority rule with both second stage strategies. In the midfield, LPT dominates Johnson's algorithm.

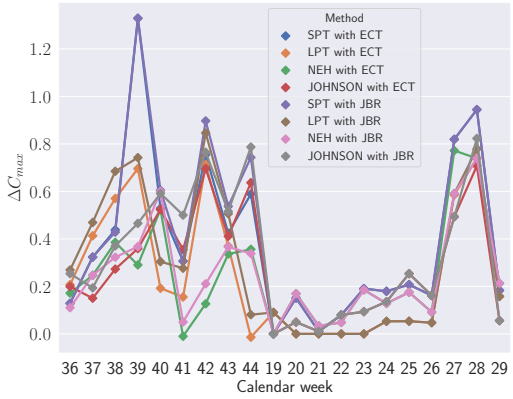


Fig. 1: ΔC_{max} of heuristics over the selected evaluation period.

Method	avg. ΔC_{max}
NEH with ECT	0.249141
NEH with JBR	0.249370
LPT with ECT	0.251440
JOHNSON with ECT	0.282263
LPT with JBR	0.297500
JOHNSON with JBR	0.329924
SPT with ECT	0.387423
SPT with JBR	0.411233

Table 1: Comparison of ΔC_{max} for selected heuristics using ECT as second stage strategy.

In the field of metaheuristics, GA with 90 000 objective function evaluations dominates the other metaheuristics (see figure 2 and table 2). Since the local search strategies evaluate 5 000 iterations, the parameters of the genetic algorithm have been reduced in "GA5000" to match with local search strategies regarding the number of function evaluations. Tabu search performs slightly better than local search strategies without tabu list in both cases. When "shift" is applied, it dominates "swap" in all cases. Also, simulated annealing is dominated by the other metaheuristics. Therefore, simulated annealing and "swap" are not considered in the tables and figures of this paper. All local search strategies perform significantly best by applying NEH as the initial solution. This shows a significant influence of the initial solution on the solution quality.

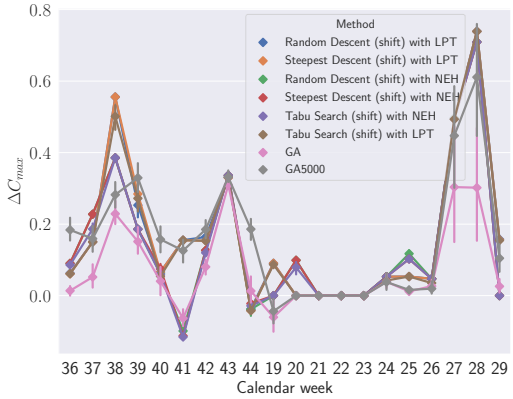


Fig. 2: ΔC_{max} of metaheuristics over the selected evaluation period.

Method	avg. ΔC_{max}
GA	0.073729
Tabu Search with NEH	0.134955
Steepest Descent with NEH	0.139813
Random Descent with NEH	0.139913
GA5000	0.156668
Tabu Search with LPT	0.162064
Random Descent with LPT	0.166863
Steepest Descent with LPT	0.167259

Table 2: Comparison of ΔC_{max} for selected metaheuristics using ECT as second stage strategy and shift move for local searches.

5 Conclusion

For our problem (1) a MIP, several heuristics, and metaheuristics have been developed and tested in manufacturing production. Regarding priority rules, NEH performs best. Comparing metaheuristics, a genetic algorithm with 90 000 function evaluations provides the best results followed by tabu search which only needs 5 000 iterations and lower computation time. For real-world applications, in future work, algorithms have to be modularized, so that it is possible to synthesize them automatically and reduce manual modeling effort.

References

[1] Rubén Ruiz, Funda Sivrikaya Şerifoğlub, and Thijs Urlings. "Modeling realistic hybrid flexible flowshop scheduling problems". In: *Computers and Operations Research* 35.4 (2008), pp. 1151–1175.

[2] Rubén Ruiz and José Antonio Vázquez-Rodríguez. "The hybrid flow shop scheduling problem". In: *European Journal of Operational Research* 205.1 (2010), pp. 1–18.

[3] Michael L. Pinedo. *Scheduling. Theory, Algorithms, and Systems.* 5th ed. Cham: Springer International Publishing, 2016.

[4] R.L. Graham et al. "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey". In: *Annals of Discrete Mathematics* 5 (1979), pp. 287–326.

[5] Jatinder N. D. Gupta. "Two-Stage, Hybrid Flowshop Scheduling Problem". In: *The Journal of the Operational Research Society* 39.4 (1988), p. 359.

[6] Imma Ribas, Rainer Leisten, and Jose M. Framiñan. "Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective". In: *Computers & Operations Research* 37.8 (2010), pp. 1439–1454.

[7] G. M. Komaki, Shaya Sheikh, and Behnam Malakooti. "Flow shop scheduling problems with assembly operations: a review and new trends". In: *International Journal of Production Research* 57.10 (2019), pp. 2926–2955.

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing". In: *Science* 220.4598 (1983), pp. 671–680.

[9] Yi Wu, Min Liu, and Cheng Wu. "A genetic algorithm for solving flow shop scheduling problems with parallel machine and special procedure constraints". In: *Proceedings of the 2003 Proceedings of the Second International Conference on Machine Learning and Cybernetics*. New York, N.Y and Piscataway, N.J: IEEE, 2003, pp. 1774–1779.

Two-stage stochastic/robust single machine scheduling based on groups of permutable jobs

Louis Riviere[1,2,3], Christian Artigues[1,2] and Hélène Fargier[1,3]

[1] Artificial and Natural Intelligence Toulouse Institute, Université de Toulouse, France
[2] LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France
[3] IRIT, Université de Toulouse, CNRS, UPS, Toulouse, France

Keywords: Scheduling, Uncertainty, Permutable operation groups.

1 Introduction

This paper considers a single machine scheduling problem with release dates, due dates, precedence constraints and aiming for the minimization of the maximum lateness or the sum of completion times over a sampled set of release dates scenarios, either in a stochastic or robust setting. The standard 2-stage approach for stochastic and robust scheduling on a single machine (Kouvelis and Yu 1997) considers first-stage decisions that output a full ordering (sequence) of the jobs (J-SEQ) on the machine and second-stage decisions that set the job start times in accordance with the information of a given scenario. In this paper, we study the performances of an alternative 2-stage solution method based on groups of permutable jobs. The method, initially e in (Artigues *et. al.* 2016), considers first-stage decisions that output only a partial ordering in the form of a sequence of groups of permutable jobs (G-SEQ). Given a scenario, the second stage policy orders the jobs inside each group according to the earliest realized release date first heuristic.

In this paper, we introduce a constraint programming approach to compute an optimal G-SEQ solution but show that in a limited time, it provides poor quality solutions on the largest instances compared to a standard J-SEQ solution approach. We then design several heuristics that use a portion of the time limit to obtain good quality J-SEQ starting solutions and then switch to G-SEQ solutions via greedy or local search. The best G-SEQ heuristics outperform all of the standard J-SEQ approaches under the same time limit.

2 Problem definition

The problem consists in scheduling a set of jobs N given uncertain release dates described by a set of discrete scenarios S. We define a problem with $|N|$ jobs and $|S|$ scenarios as a tuple $\mathcal{P} = (P, R, D, E, \sigma, \gamma)$ where $p_i \in P$ is the duration of job i, $r_i^s \in R$ is the release date of job i in scenario s, $d_i \in D$ is the due date of job i, E is the set of precedence constraints ($(i, j) \in E$ iff job i must be scheduled before job j). Parameter $\sigma \in \{max, avg\}$ is the objective aggregator amongst scenarios, and $\gamma \in \{L_{max}, \sum C_i\}$ is the objective type. Classically $\sigma = max$ corresponds to robust optimization while $\sigma = avg$ corresponds to stochastic optimization.

Solutions to scheduling problems are usually schedules, i.e a start time for each job. But with uncertain parameters, such solutions might either be invalid or too conservative. That is why we study two types of 2-stage solution methods, that take advantage of the information available when a scenario occurs to adjust the schedule.

We consider two different settings for the first stage decision : respectively job sequences (J-SEQ) and sequences of groups of permutable jobs (G-SEQ):

Job sequences A valid job sequence (J-SEQ) is a total ordering of the jobs such that for any pair $(i, j) \in E$, $i \prec j$ in the sequence. Given a J-SEQ and a scenario, the best associated start times are the left-shifted ones as semi-active schedules are dominant for our objectives (Kouvelis and Yu 1997).

Group sequences A sequence of groups of permutable jobs (G-SEQ) is an ordered partition of the set of jobs: $\pi = G_1|G_2|\dots|G_k$ where $G_i \subseteq N$; $G_i \cap G_j = \emptyset$; $\bigcup_{i=1..k} G_i = N$. It is a valid solution if for any pair $(i, j) \in E$, the group of job i is ordered strictly before the group of job j in the G-SEQ. With this condition, a G-SEQ represents a potentially large set of $\Pi_{i=1..k}|G_i|!$ valid jobs sequences. Given a G-SEQ and a scenario, we use the "Earliest release date first" heuristic to compute a full job sequence, and then the corresponding left-shifted schedule. The "Earliest release date first" heuristic allows for non anticipation about the scenario : jobs are scheduled when they are ready.

We notice that J-SEQs are a special case of G-SEQs, where each group contains a single job only. As such, G-SEQs are in theory better than J-SEQs, but due to the much larger search space, in a limited time, it is not easy to predict which approach will give the best results.

3 Constraint programming models

In this Section we present a constraint programming (CP) model for each approach. For both models, we let jobs have a different start time in each scenario, but all start times must be consistent with a unique J- or G-SEQ across all scenarios. Algorithm 1 describes the key constraints in the CP model used for G-SEQ computation using IBM CP Optimizer (CPO) modeling.

Algorithm 1 CP model for G-SEQ computation

for $s \in S$ **do**
  $StartOf(Job[i, s]) \geq r_i^s \quad \forall i$
  $NoOverlap(Sequence(Job[:, s]))$
  for $(i, j) \in N^2$ **do**
    $(i, j) \in E \rightarrow g_i < g_j$ {g_i is the group number of job i}
    $g_i < g_j \rightarrow Job[i, s]$ before $Job[j, s]$
    if $Release(i, s) \leq Release(j, s)$ **then**
      $g_i == g_j \rightarrow EndOf(Job[i, s]) \leq StartOf(Job[j, s])$
    else
      $g_i == g_j \rightarrow EndOf(Job[j, s]) \leq StartOf(Job[i, s])$
    end if
  end for
end for

The CP model for J-SEQ computation is derived from IBM's *"stochastic jobshop"* example, which uses the **IloSameSequence()** constraint to enforce sequence unity across all scenarios. Both models encapsulate the online scheduling policy associated with selected solutions.
However, we will see in section 5 that even though G-SEQs are better in theory, for larger instances within 5 minutes, CPO gets much better results using the J-SEQ model than the G-SEQ model. In order to take advantage of the good performance of the J-SEQ solver, but still retain in some of the possible gain and flexibility of G-SEQs, we then introduce heuristic G-SEQ solvers based on a good J-SEQ starting solution.

4 Heuristics

In addition to the two CP models described in Section 3, several solvers were implemented and compared. The idea for most of them is to use as a starting point a good J-SEQ provided by a short run of the J-SEQ CP model with CPO. For each heuristics, parameters were selected empirically, the goal being not provide a state of the art G-SEQ solver, but to show that even with limited time, it can be more efficient to use G-SEQs than J-SEQs.
On the G-SEQ side, we introduced the following heuristics:

- Warm-CP* : G-SEQ CP model guided with a starting J-SEQ solution.
- Greedy : A simple greedy heuristic, after shuffling the jobs, at each step it inserts the jobs in an existing group or between existing groups to maximize the objective.
- Taboo* : A taboo list algorithm based on the neighborhood defined by moving one job from a group to another group or between existing groups.
- GA* : A genetic algorithm in which the crossover operator is an adaptation of the one-point crossover to G-SEQs : if π_A and π_B are the selected individuals, with x being the crossover point, the offspring is the concatenation of the first x groups of π_A ($G_1^{\pi_A}|\dots|G_x^{\pi_A}$) with the groups of π_B (in which the jobs already in $G_1^{\pi_A}|\dots|G_x^{\pi_A}$ are removed). Mutation operators include moving jobs (as in Taboo), switching group orders, merging groups and splitting groups.

On the J-SEQ side we also propose GA, Greedy and Taboo heuristics producing standard J-SEQ solutions based on similar neighborhoods and other components as their G-SEQ counterparts for fair comparisons.

5 Experimental results and comparisons

THe solvers have been tested on a custom benchmark of instances generated using 4 parameters: N: the number of jobs; S : the number of scenarios; Δ : the variability of the release date around a base value for each scenario; and Π : the density of the precedence graph. The solvers were run on a single worker (Xeon E5-2695 v4 @ 2.10GHz processor) with a 5 minutes timeout. The CP models were solved by IBM CP Optimizer 20.1. For warm start solvers (indicated by * in section 4) the output of the J-SEQ CP model after 1 minute was used as starting point for the solver for the remaining 4 minutes.

We compare the solvers based on :

- performance (best/score) : the ratio of the best score reached by any solver to the score reached by the solver on the same instance. Hence the best solution for a given instance has a performance of 1.
- majoring ($\#best/\#instance$) : the percentage of instances where the solver had the best solution among compared solvers.

We first compare the exact solution approaches in terms of the above-defined performance criterion (columns CP and Warm-CP in Table 1). For small problem, the G-SEQ CP model manages to solve most instances optimally and has a better performance than the J-SEQ CP model, which is conform to the theory. As the number of job increases, the J-SEQ CP model performs in average better than the G-SEQ CP model, especially for L_{max} objective (although there is a high standard deviation in the data due to the nature of L_{max} objective). The performance of Warm-CP shows that providing a starting solution to the G-SEQ CP model significantly improves its performances, but it still performs worse than the J-SEQ CP model on the largest instances. Also, contrary to what

could be expected, increasing the number of scenarios relatively decreases the G-SEQ CP model performances for L_{max} objectives, but increasing scenario variability does increase its relative performance.

We now turn to the comparisons of heuristics. Table 1 shows that over tested instances, the G-SEQ GA consistently performs better than other solvers on larger instances, while almost as good as exact G-SEQ solvers on small instances. The G-SEQ GA also obtains the the largest percentage of best solutions (0.82).

The G-SEQ Greedy performs remarkably well on the $N = 10$ and $N = 20$ instances and then its performance quickly decreases but remain higher than the G-SEQ CP model, which puts emphasis on the weakness of the latter. J-SEQ-restricted counterparts of heuristics perform worse than their unrestricted counterparts and slightly worse than the J-SEQ CP model on average.

In general, the relative performance and majoring rate of all J-SEQ solvers is lower on instances with higher variation Δ, with the notable exception of J-SEQ Greedy, for which an improvement is observed . On the other hand, the relative performance of J-SEQ based solvers improves on instances with higher precedence density Π, which can be expected since the possibility of grouping jobs is decreased.

Overall, several G-SEQ solvers are able to improve the average performance of the input J-SEQ more than the J-SEQ CP model within the same time, showcasing that it can be beneficial to compute G-SEQ solutions, even within limited time. GA seems to be a good starting point to implement a truly efficient G-SEQ solver in the future. Efficient constraint propagation on the G-SEQ CP model is also a future research direction.

Table 1. Performance comparison for all solvers (best score/solver score)

		G-SEQ					J-SEQ			
obj	N	CP	Warm-CP	Taboo	GA	Greedy	CP	Taboo	GA	Greedy
avg	10	**1.0000**	**1.0000**	0.9961	0.9997	0.9998	0.9537	0.9537	0.9537	0.9537
L_{max}	20	0.9397	0.9993	0.9910	**0.9967**	0.9899	0.9479	0.9473	0.9476	0.9450
	50	0.5717	0.9783	0.9934	**0.9981**	0.7561	0.9648	0.9620	0.9627	0.7439
	100	0.3983	0.9639	0.9876	**0.9931**	0.5732	0.9853	0.9726	0.9720	0.5680
avg	10	0.9980	0.9997	0.9983	0.9997	**1.0000**	0.9793	0.9792	0.9793	0.9793
$\sum C_i$	20	0.9611	0.9923	0.9949	**0.9984**	0.9968	0.9748	0.9743	0.9749	0.9749
	50	0.8690	0.9813	0.9946	**0.9988**	0.9596	0.9786	0.9784	0.9798	0.9590
	100	0.8394	0.9575	0.9712	**0.9988**	0.8876	0.9726	0.9631	0.9725	0.9060
max	10	**1.0000**	**1.0000**	0.9934	0.9996	**1.0000**	0.9614	0.9614	0.9614	0.9614
L_{max}	20	0.9913	**1.0000**	0.9885	0.9971	0.9921	0.9624	0.9624	0.9624	0.9591
	50	0.6950	0.9958	0.9923	**0.9968**	0.7709	0.9786	0.9785	0.9785	0.7664
	100	0.4728	0.9896	0.9945	**0.9954**	0.6142	0.9934	0.9898	0.9893	0.6069
max	10	**1.0000**	**1.0000**	0.9968	0.9993	0.9999	0.9771	0.9771	0.9771	0.9771
$\sum C_i$	20	0.9600	0.9935	0.9911	**0.9969**	0.9955	0.9749	0.9741	0.9745	0.9748
	50	0.8496	0.9856	0.9921	**0.9970**	0.9543	0.9836	0.9814	0.9822	0.9604
	100	0.8154	0.9653	0.9711	**0.9943**	0.8796	0.9902	0.9660	0.9735	0.8971

References

Artigues C., Billaut J.C., Cheref A., Mebarki N., Yahouni Z., 2016, "Robust Machine Scheduling Based on Group of Permutable Jobs", in Doumpos M., Zopounidis C., Grigoroudis E. (Eds) *Robustness analysis in decision aiding Optimization, and Analytics*, pp.191-220, Springer.

Kouvelis P., Yu G/, 1997, "Robust Discrete Optimization and Its Applications", Springer.

A relaxation-based generation scheme for the RCPSP/max,π

Mareike Karnebogen and Jürgen Zimmermann

Clausthal University of Technology, Germany
mareike.karnebogen@tu-clausthal.de, juergen.zimmermann@tu-clausthal.de

Keywords: project scheduling, partially renewable resources, generation scheme

1 Introduction

In the context of project scheduling, alternative resource types like partially renewable resources, which were first mentioned by Böttcher *et al.* (1999), gain more and more relevance. Partially renewable resources are characterized by the fact that they are only required by an activity, if its execution takes place in a defined set of periods, whereas for the complementary set of periods no resource consumption applies. Hence, partially renewable resources constitute a generalized form of renewable and non-renewable resources and, e.g. allow to model special working time restrictions like weekend working arrangements. In this paper, the resource-constrained project scheduling problem with partially renewable resources and time windows (RCPSP/max,π) is considered. As a generalization of the RCPSP/π it is hard to solve ($\mathcal{NP}$-hard in the strong sense). Thus we present a relaxation-based generation scheme, which finds good feasible solutions within a short amount of time. In Section 2 a problem description and a corresponding mixed-integer linear formulation (MILP) is given. Section 3 addresses the developed generation scheme. Finally, in Section 4 a preliminary performance analysis is presented.

2 Problem description

Within the RCPSP/max,π, the underlying project is given by an activity-on-node network $G = (V, E, \delta_{ij})$ with minimim and maximum start-start-time lags. Each activity $i \in V = \{0, 1, \ldots, n+1\}$ has a deterministic processing time $p_i \in \mathbb{Z}_{\geq 0}$ and can not be interrupted during its execution. Furthermore, a maximal project duration $\bar{d}$ is given, which is observed by a time lag $\delta_{n+1,0} = -\bar{d}$ between the end and the start of the project. Let d_{ij} be the longest distance from node $i \in V$ to node $j \in V$ in the project network and $ES_i = d_{0i}(LS_i = -d_{i0})$ the earliest (latest) start time of activity $i \in V$, respectively. Then, the set of all integer time-feasible start time points of activity i results in $\mathcal{T}_i = \{ES_i, ES_i + 1, ..., LS_i\}$.

In addition to time restrictions, a set of partially renewable resources $\mathcal{R} = \{1, \ldots, m\}$ is given. To each of those resources $k \in \mathcal{R}$ a set $\Pi_k \subseteq \{1, 2, \ldots, \bar{d}\}$ of not necessarily connected periods of the planing horizon is assigned. For all time periods contained in Π_k it is assumed that the capacity of the resource is in total restricted to R_k, whereas in periods not contained in Π_k the resource k is not available. In addition, it is supposed, that activity $i \in V$ only consumes r_{ik} units of resource $k \in \mathcal{R}$ per time period contained in Π_k and thus the execution of activity i is not linked to the resource availability as shown in the following example: Assuming there are two employees ($k = 1, 2$) who can take on an all-day task i ($r_{i1} = r_{i2} = 1$) on any day of the week. Employee 1 is available Monday and Tuesday ($\Pi_1 = \{1, 2\}$), employee 2 the remaining three days of the working week ($\Pi_2 = \{3, 4, 5\}$). Task i can then be carried out on all 5 days, but - depending on how it is scheduled - it is done either by employee 1 (start time $S_i \in \{1, 2\}$) or by employee 2 ($S_i \in \{3, 4, 5\}$).

Consequently, the composite resource consumption of activity i of resource k depends on the number of periods $t \in \Pi_k$ during those the activity is executed and can be calculated by $r_{ik}^c(S_i) = |\{S_i + 1, S_i + 2, \ldots, S_i + p_i\} \cap \Pi_k| \cdot r_{ik}$ (Watermeyer and Zimmermann 2020).

The objective of the RCPSP/max,π is to find a schedule $S = (S_0, S_1, \ldots, S_{n+1})$ minimizing the project duration S_{n+1} and simultaneously complies with all time and resource restrictions. The time restrictions are met if $S_j - S_i \geq \delta_{ij} \, \forall \, \langle i, j \rangle \in E$, whereas the resource restrictions are observed if the accumulated resource demand of all capacitated periods $t \in \Pi_k$ not exceeds R_k for any partially renewable resource $k \in \mathcal{R}$. So formally, the RCPSP/max,π can be stated as follows:

$$\left.\begin{array}{lll} \text{Minimize} & f(S) = S_{n+1} & \\ \text{subject to} & S_j - S_i \geq \delta_{ij} & (\langle i, j \rangle \in E) \\ & S_0 = 0 & \\ & \sum_{i \in V} r_{ik}^c(S_i) \leq R_k & (k \in \mathcal{R}) \\ & S_i \in \mathbb{Z}_{\geq 0} & (i \in V) \end{array}\right\} \text{(RCPSP/max,}\pi\text{)}$$

3 Generation scheme

The basic principle of our generation scheme is starting with the ES-Schedule gradually resolving resource conflicts until we obtain a time- and resource-feasible schedule. Thereby, the resource relaxation of the RCPSP/max,π is used, which, according to Watermeyer and Zimmermann (2020), can be formulated as follows:

$$\left.\begin{array}{lll} \text{Minimize} & f(S) = S_{n+1} & \\ \text{subject to} & S_j - S_i \geq \delta_{ij} & (\langle i, j \rangle \in E) \\ & S_i \in W_i & (i \in V) \end{array}\right\} \text{(RCPSP/max,}\pi^{relax}\text{)}$$

Vector $W = (W_i)_{i \in V}$ describes start time restrictions for the project activities $i \in V$ with $W_i \subseteq \{0, 1, ..., \bar{d}\}$ for all $i \in V \setminus 0$ and $W_0 = \{0\}$. For the resource relaxation, $W_i := \mathcal{T}_i$ for all $i \in V$ applies. In the following, $S_T(W)$ describes the feasible area of problem RCPSP/max,π^{relax}, and $ES(W)$ is the unique minimal point of this space, which exists unless $S_T(W) = \emptyset$ and can be determined as shown in Watermeyer and Zimmermann (2020).

Algorithm 1 shows our generation scheme. In the initialization process for each activity $i \in V$ the composite resource demand $r_{ik}^c(t)$ according to $t \in \mathcal{T}_i$ for each partially renewable resource $k \in \mathcal{R}$ and the minimal composite demand r_{ik}^{min} are calculated. Afterwards, for each activity $i \in V$ a set of all potential start times W_i is established containing all points in time $t \in \mathcal{T}_i$, omitting those who are certainly known to be resource-unfeasible. Based on W_i, the maximal composite demand r_{ik}^{max} is computed. Furthermore, a counter u is set to zero and a tabu list Θ_k for each resource $k \in \mathcal{R}$ are initialized.

In the main step, which is just performed if a time-feasible schedule exists, i.e. $S_T \neq \emptyset$, and if the minimum possible resource requirement r_{ik}^{min} of all activities $i \in V$ in total does not exceed the capacity R_k for any resource $k \in \mathcal{R}$, first Schedule S is set to the unique minimalpoint $ES(W)$. Then, the set of resources is determined for which a resource conflict exists, i.e. the total resource consumption exceeds the given capacity. If this set is empty, schedule S is both time- and resource-feasible and the algorithm terminates. Otherwise, one of the conflict resources $k^* \in \mathcal{R}^{conflict}$ is selected based on a certain priority rule. In the next step, all activities $i \in V$, whose resource consumption of resource k^* can be reduced, are determined to evaluate potential candidates to resolve the chosen resource conflict. If there is no such activity, i.e. if the set V^{pot} is empty, a so-called reverse step is carried out. Otherwise, an activity $i^* \in V^{pot}$ is selected priority rule-based and its maximal composite

Algorithm 1 Generation Scheme

1: $r^c_{ik^*}(t) := |\{t+1, t+2, \ldots, t+p_i\} \cap \Pi_k| \cdot r_{ik}$ for all $i \in V$, $k \in \mathcal{R}$ and $t \in \mathcal{T}_i$
2: $r^{min}_{ik} := \min_{t \in \mathcal{T}_i} r^c_{ik^*}(t)$ for all $i \in V$ and $k \in \mathcal{R}$
3: $W_i := \{t \in \mathcal{T}_i | r^c_{ik}(t) \leq R_k - \sum_{j \in V \setminus i} r^{min}_{ik}$ for all $k \in \mathcal{K}\}$ for all $i \in \mathcal{V}$
4: $r^{max}_{ik} := \max_{t \in \mathcal{W}_i} r^c_{ik^*}(t)$ for all $i \in V$ and $k \in \mathcal{R}$
5: $r^{LB}_{ik} := r^{min}_{ik}, r^{UB}_{ik} := r^{max}_{ik}$
6: $u := 0$, $\Theta_k = \emptyset$ for all $k \in R$
7: **if** $\mathcal{S}_T(W) = \emptyset$ **then** terminate
8: **while** true **do**
9: $S := ES(W)$
10: $\mathcal{R}^{conflict} := \{k \in \mathcal{R} | \sum_{i \in V} r^c_{ik^*}(S_i) > R_k\}$
11: **if** $\mathcal{R}^{conflict} = \emptyset$ **then** break
12: **else**
13: priority based choice of resource $k^* \in \mathcal{R}^{conflict}$
14: $V^{pot} := \{i \in V | r^c_{ik^*}(S_i) > 0 \wedge r^c_{ik^*}(S_i) - r^{LB}_{ik^*} > 0 \wedge i \notin \Theta_{k^*}\}$
15: **if** $V^{pot} = \emptyset$ **then** $u := u + 1$ and **Reverse Step**
16: **else**
17: priority based choice of activity $i^* \in \mathcal{I}^{pot}$
18: $r^{UB}_{i^*k^*} := r^c_{i^*k^*}(S_{i^*}) - r_{i^*k^*}$
19: **for** $t \in W_i$ **do**
20: **if** $r^c_{i^*k^*}(t) > r^{UB}_{i^*k^*}$ **then** $W_i := W_i \setminus \{t\}$
21: **if** $\mathcal{S}_T(W) = \emptyset$ **then**
22: $r^{UB}_{i^*k^*} := r^c_{i^*k^*}(S_{i^*})$
23: $\Theta_{k^*} := \Theta_{k^*} \cup \{i^*\}$
24: **else**
25: update W_i, r^{min}_{ik} and r^{max}_{ik} for all $i \in V$ and $k \in \mathcal{R}$
26: **return** S

resource consumption is set to $r^c_{i^*k^*}(S_{i^*}) - r_{i^*k^*}$. This additional restriction on the resource consumption of activity i^* is now converted into a start time restriction, i.e. all point in times $t \in W_{i^*}$ for which $r_{i^*k^*}(t) > r^{UB}_{i^*k^*}$ applies are removed from W_{i^*}. Then, it is verified whether a feasible solution for RCPSP/max,π^{relax} still exists due to this additional start time restriction. If not, the restriction is undone and activity i^* is added to the tabu list of resource k^*, i.e. i^* cannot be selected for the next time to resolve the resource conflict of k^*. Otherwise, W_i, r^{min}_{ik} and r^{max}_{ik} are updated for all $i \in V$ and $k \in \mathcal{R}$. The main step is repeated until a both time- and resource-feasible schedule has been determined or the algorithm terminates in the course of the reverse step.

The reverse step is executed if there is no possible operation to resolve the selected resource conflict, which means $S_T(W)$ no longer contains a resource-feasible schedule. In case u is higher than a given number of reverse steps $\hat{u}$ the generation scheme terminates, i.e. no feasible schedule could be found. Otherwise, start time restrictions or decisions made in previous iterations, which ensure that currently existing resource conflicts arise or cannot be resolved, are undone. In a first step, the tabu list Θ_{k^*} is cleared. Thereafter, due to the fact that the wrong start time restrictions can not clearly be determined, four different strategies were examined:

1. a random number of the last decisions made is reversed ($\sigma = 1$).
2. return to the initial state ($W_i = \mathcal{T}_i$ for all $i \in V$) ($\sigma = 2$).
3. W_i is reset for activities $i \in V$, which are potentially candidates to solve the resource conflict, i.e. $r_{ik^*}(S_i) - r^{LB}_{ik^*} > 0$ ($\sigma = 3$).
4. for potentially candidates it examined whether binding time or resource constraints are the reason why their resource consumption of k^* cannot be reduced and the start time restrictions of the corresponding activity i and/or other activities are reset ($\sigma = 4$).

Finally, W_i, r^{min}_{ik} and r^{max}_{ik} are updated for all activities $i \in V$ and resources $k \in \mathcal{R}$.

4 Performance analysis

Our generation scheme was coded in C++. For the choice of resource k^* the priority rules RDd (maximal total "Resource Demand dynamic" first) and ROd (maximal "resource capacity overrun dynamic" first) were tested, whereas for the activity selection the rule TFd was used. For the reverse step the four strategies mentioned above were examined. To evaluate the generation scheme, non-trivial instances with $n \in \{20, 50, 100\}$ and $m = 30$ from the instance set established by Watermeyer and Zimmermann (2020) were solved. Table 1 shows our results compared to results of the construction-based generation scheme of Karnebogen and Zimmermann (2021) as well as to results obtained by the mixed-integer linear programming solver *IBM® ILOG® CPLEX®* 12.8.0 applied on the MIP model of Watermeyer and Zimmermann (2020) with a runtimelimit of one hour. All runs were done on an Intel Core i7-7700K CPU with 4.2 GHz and 64 GB RAM under Windows 10. Displayed are the percentage of instances (%feas), for which our generation scheme was able to find feasible solutions, the average percentage gap (%Gap) referred to the best lower bound of the MIP found in at most 3.600 seconds and the average computing time (∅CPU) in seconds required for 100 runs with $\hat{u} = 100$.

Table 1. Preliminary results of the computational study

		RDd				*ROd*				GS^{constr}	*CPLEX*
		$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$\sigma = 4$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$\sigma = 4$		
	%feas	93.98	93.63	95.35	95.87	93.29	93.46	95.52	96.04	98.45	100.00
$UBO20^{\pi}$	%Gap	8.62	7.27	6.49	6.63	8.59	7.10	6.51	6.86	5.42	0.46
	∅CPU	1.71	4.59	2.95	3.17	2.34	5.10	3.22	3.43	3.19	143.55
	%feas	91.04	92.23	93.23	92.63	91.63	92.63	93.43	92.83	96.81	96.81
$UBO50^{\pi}$	%Gap	36.35	33.19	33.79	33.19	36.01	32.72	33.52	32.95	33.75	12.88
	∅CPU	16.90	53.42	21.70	23.42	19.15	67.58	23.87	25.53	30.85	2175.01
	%feas	95.82	96.24	96.45	96.66	96.03	96.66	97.08	97.08	97.08	63.26
$UBO100^{\pi}$	%Gap	87.17	83.41	85.63	85.08	87.14	81.88	85.10	84.73	85.83	86.06
	∅CPU	165.96	257.95	201.90	214.08	177.09	274.04	220.35	230.37	112.50	2985.09

The results show that our generation scheme is able to generate feasible solutions for most tested instances. For small instances ($UBO20^{\pi}$ and $UBO50^{\pi}$) the MIP-Solver outperforms both generation schemes, even though the run-time increases drastically. For the bigger instances ($UBO100^{\pi}$), which are more relevant in practice, the generation schemes are able to find way more feasible and slightly better solutions in under four minutes than the MIP in one hour. For these instances, the combination of the resource selection rule *ROd* and reverse strategy $\sigma = 2$ performs best.

References

Böttcher J., A. Drexl, R. Kolisch and F. Salewski, 1999, "Project scheduling under partially renewable resource constraints", *Management Science*, Vol. 45, pp. 543-559.

Karnebogen M. and J. Zimmermann, 2021, "A Generation Scheme for the Resource-Constrained Project Scheduling Problem with Partially Renewable Resources and Time Windows", *Book of Extended Abstracts of 17th International Conference on Project Management and Scheduling*, Toulouse, pp. 195-198.

Neumann K., C. Schwindt, and J. Zimmermann, 2003, "Project scheduling with Time Windows and Scarce Resources", 2nd ed., Springer, Berlin.

Watermeyer K. and J. Zimmermann, 2020, "A branch-and-bound procedure for the resource-constrained project scheduling problem with partially renewable resources and general temporal constraints", *OR Spectrum*, Vol. 42, pp. 427-460.

Total Core Idle Time minimization for the permutation flowshop scheduling problem

Paula Sanchez-de los Reyes[1], Paz Perez-Gonzalez[1]

Industrial Management, School of Engineering, University of Seville
psdelosreyes, pazperez@us.es

Keywords: flowshop, metaheuristics, machine idle times.

1 Introduction

The permutation flowshop scheduling problem (PFSP) has been widely studied considering different objective functions as makespan or total completion time, see e.g. (Fernandez-Viagas, Ruiz & Framinan 2017, Fernandez-Viagas, Molina-Pariente & Framinan 2020). In this paper, the total core idle time minimization is considered due to its practical interest (Maassen, Perez-Gonzalez & Günther 2020). The total core idle time is defined as the machines idle times between jobs, avoiding the front/back idle times generated by the first/last job in a given schedule. The problem PFSP with total core idle time (TCIT) is NP-hard in the strong sense (Maassen et al. 2020), so approximate methods should be developed to solve realistic sizes instances in reasonable computational times. Related literature can be consulted in the recent work by (Maassen et al. 2020).

In this paper, the main idea is to test methods proposed in the literature for related problems, in order to check their efficiency for this objective. Related problems can be PFSP with makespan and total completion time, due to the relation between these objectives and machine idle times (Maassen et al. 2020). We have adapted approximate methods in the literature developed for makespan to TCIT: the constructive heuristic NEH (Nawaz, Enscore & Ham 1982), and the metaheuristics IG (Ruiz & Stützle 2007), IGALL (Dubois-Lacoste, Pagnozzi & Stützle 2017) and VBIH (Tasgetiren, Pan, Kizilay & Gao 2016). It has not been possible to include in this work the state-of-the-art method by (Fernandez-Viagas & Framinan 2019), due to the high computational effort needed by this method in the adaptation, since the accelerations and critical path methods applied to the makespan are not adaptable to the TCIT, with similar complexity to the total completion time, see e.g. (Fernandez-Viagas et al. 2020). Regarding to the total completion time, according to (Pan & Ruiz 2013), we have adapted the best constructive heuristic with respect to the quality of the solution: the LR-NEH. Finally, taking into account that the idle time is part of the TEC (total energy consumption) objective, those methods proposed by (Öztop, Tasgetiren, Eliiyi, Pan & Kandiller 2020) for the PFSP with total completion time and TEC as objectives, and (Öztop, Tasgetiren, Kandiller, Eliiyi & Gao 2020) for the hybrid flowshop scheduling problem with makespan and TEC has been adapted: PF-NEH and NEH-M as constructive heuristics, and IG, IGALL, VBIH as metaheuristics.

Therefore, seven methods have been adapted: four heuristics (Section 2) and three metaheuristics (Section 3). Results are presented in Section 4. Among the adapted methods, the best results are provided by the VBIH. Finally a variant of this method is proposed, giving the best results between the tested methods.

2 Constructive Heuristics

In this section we describe the constructive heuristics NEH, NEH-M, PFH-NEH and LR-NEH previously mentioned in Section 1 and adapted for the TCIT objective.

In the NEH, the jobs are initially sorted in descending order of their total processing times. The first job is selected to obtain a partial solution with size one. Then, each job of the initial sequence is inserted into all possible positions of the new partial sequence to find the best position, until all jobs are scheduled.

The NEH-M(X) heuristic takes into account the strong impact of the first job of the sequence in the objective function. Initially, the jobs are sorted in the same way than in the NEH. Then, X solutions are generated by choosing different jobs as the first job. In the h^{th} iteration, the job number h of the initial sequence is chosen as the first job of the partial solution. Once the first job of this h^{th} solution is selected, the NEH procedure is applied. Finally, the best among the X solutions is selected as the final solution.

In the LR-NEH(X) heuristic, d jobs are selected to generate a partial solution by the LR method (Liu & Reeves 2001). The LR(X) constructs the solution by appending each job based on an index function (the sum of the weighted total machine idle time and the artificial total flow time). The job with the minimum value of this index function is selected. Ties are broken by selecting the one with the minimum weighted total machine idle time. Then, the NEH is applied to the partial solution of size d by inserting the $n-d$ jobs. Due to the impact of the first job into the objective function, this procedure is repeated X times, choosing in the h^{th} iteration, the job number h of the initial sequence by LR as the first job of the partial solution.

The PFH-NEH(X) sorts the jobs in ascending order of a function computing the sum of the front delays and total processing times of each job. Then, X solutions are generated choosing different jobs as the first job. In the h^{th} iteration, the job number h of the initial sequence is chosen as the first job of the partial solution. Once the first job of this h^{th} solution is selected, the method PF is applied, constructing the solution by appending each job based in an index function, computed as the sum of the blocking and idle times on the machines. Once the complete sequence is obtained, the NEH and LS are applied.

3 Metaheuristics

In this section, IG, IGALL and VBIH are presented as the metaheuristics adapted for our problem. Additionally, the proposal, denoted as VBIH-P is described too. Note that, in the adaptation, all the constructive heuristics previously described has been tested as initial solutions for the metaheursitics.

The original Iterated Greedy (IG) algorithm starts with an initial solution obtained by the NEH heuristic. In the destruction procedure, d jobs are removed from the current sequence and stored in a partial sequence. In the construction phase, the d jobs are reinserted one by one in the best position. Then a local search based on first improvement with general swap, named LS and used in the rest of the methods in this section, is applied to the complete solution. After LS, it should be decided whether or not the new sequence is accepted as the initial solution for the next iteration. A simple simulated annealing-like acceptance criterion with a constant temperature (parameter denoted TP) is employed.

The IGALL algorithm is similar to the IG, with the difference that the IGALL applies LS to each partial solutions after the destruction. It uses the same parameters d and TP.

The VBIH algorithm originally developed by (Tasgetiren et al. 2016) for the PFSP with total completion time and blocking constraint, is similar to the IG and IGALL. (Öztop, Tasgetiren, Eliiyi, Pan & Kandiller 2020, Öztop, Tasgetiren, Kandiller, Eliiyi & Gao 2020) consider as initial sequence the PFH-NEH. The difference is that the destruction procedure removes from the current solution a block of jobs, with an increasing size $bs \in [bmin, bmax]$ in each iteration, applying LS. Afterwards, the block is inserted in all possible positions,

choosing the best one. Again, LS is applied to the complete solution. The same simple simulated annealing-like acceptance criterion with parameter TP is employed.

The VBIH-P is the method proposed in this study. It is based in the strong impact of the first job in the objective function. The VBIH-P follow the same procedure of the VBIH previously described with some differences: the local search applied to the partial sequence is not included, revealing a good performance of the destruction and construction for the TCIT objective. Additionally, the simulated annealing-like acceptance criterion is removed. In the proposal, if the new solution is better than the current one, the new solution is always accepted as the incumbent solution. Otherwise, the incumbent solution will become the last solution explored in the construction process. In our procedure, the last solution explored is the result of inserting the block in the first position of the partial sequence. Note that this metaheuristic only has as parameters $bmin$ and $bmax$.

4 Computational Evaluation

In this section, the performance of the methods is analysed. For all the methods, the well known Taillard testbed (Taillard 1990) has been used (see the sizes in Tables 1 and 2 with n the number of jobs, and m the number of machines). Each size has 10 instances, being in total 120 instances in this testbed. For each method, the average relative deviation index (ARDI) has been computed by the following equation:

$$RDI_i = \frac{TCIT_{im} - TCIT_{imin}}{TCIT_{imax} - TCIT_{imin}} \tag{1}$$

with $TCIT_{im}$ the objective value for the instance i by the method m, $TCIT_{imax}$ the maximum value obtained for the instance i by all the methods, and $TCIT_{imin}$ the minimum.

First, the constructive heuristics (NEH, NEH-M, LR-NEH and PFH-NEH) have been compared. The parameter X was calibrated and the best results were obtained for $X = n$, with n the number of jobs of the instance, when $n \leq 200$, or $X = 1$ otherwise. Table 1 show the results of ARDI for each value of n and m, and the total in the last row (in bold the best results per size). It can be observed that the best results are provided by the NEH-M for almost all the sizes, being the best method in average.

Regarding the metaheuristics, IG, IGALL, VBIH and VBIH-P are compared. All the constructive heuristics have been tested as initial sequences. Due to space reasons, Table 2 only presents the results for the initial sequence providing the best results in the complete computational experiment, the NEH-M. In this study, the parameters used are the following: IG and IGALL $d = 4$ and $TP = 0.4$, VBIH $bs \in [bmin = 2, bmax = 5]$ and $TP = 0.4$, and, finally, VBIH-P $bs \in [bmin = 2, bmax = 5]$. For all the methods, the stopping criteria has been $n \cdot m \cdot 30$ milliseconds. The results clearly indicate that the best method in terms of ARDI is the VBIH-P. In a statistical analysis carried out shows that VBIH-P is statistically better than IG, IGALL and VBIH (details are not included due to space reasons). However, it should be noticed that the VBIH-P gets worse as n increases, being outperformed by IG when $n = 500$. The total result shows that our proposal outperforms the other efficient metaheuristics for the problem.

Acknowledgements

This work is supported by the projects Ibsos (AT17_5920_US), Quid Pro Quo (AT17_5967_US), EFECTOS (US-1264511) and DEMAND (P18-FR-1149) funded by Junta de Andalucia, and the project Assort (PID2019-108756RB-I00) funded by the Spanish Government.

Table 1. Constructive Heuristics: ARDI

n	m	LR-NEH	NEH	NEH-M	PFH-NEH
20	**5**	0.1722	0.6945	0.3338	**0.0831**
	10	0.2054	0.4669	**0.0652**	0.2631
	20	0.2579	0.2491	**0.0000**	0.1429
50	**5**	**0.0657**	0.4770	0.1908	0.1640
	10	0.1343	0.3173	**0.0405**	0.1806
	20	0.2722	0.2379	**0.0000**	0.1838
100	**5**	0.1282	0.4876	**0.0732**	0.4174
	10	0.0744	0.3699	**0.0643**	0.1684
	20	0.2045	0.1104	**0.0000**	0.1640
200	**10**	0.1881	0.0766	**0.0000**	0.3019
	20	0.1700	0.0657	**0.0000**	0.1595
500	**20**	0.0356	**0.0401**	**0.0401**	0.1318
Total		0.1590	0.2994	**0.0673**	0.1967

Table 2. Metaheuristics: ARDI

n	m	IG	IGALL	VBIH	VBIH-P
20	**5**	0.4521	0.5974	0.3387	**0.0091**
	10	0.5740	0.6408	0.5203	**0.0333**
	20	0.4712	0.5838	0.4548	**0.0243**
50	**5**	0.3955	0.4933	0.3000	**0.0000**
	10	0.3774	0.5120	0.2789	**0.1059**
	20	0.4740	0.5209	0.3873	**0.0529**
100	**5**	0.1835	0.2407	0.1835	**0.0333**
	10	0.3823	0.4805	0.2830	**0.0257**
	20	0.3713	0.5370	0.2667	**0.0798**
200	**10**	0.3323	0.3309	0.2436	**0.2012**
	20	0.3956	0.4493	0.2896	**0.1844**
500	**20**	**0.2930**	0.6155	0.5902	0.4275
Total		0.3918	0.5002	0.3447	**0.0981**

References

Dubois-Lacoste, J., Pagnozzi, F. & Stützle, T. (2017), 'An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem', *Computers and Operations Research* **81**, 160–166.

Fernandez-Viagas, V. & Framinan, J. M. (2019), 'A best-of-breed iterated greedy for the permutation flowshop scheduling problem with makespan objective', *Computers and Operations Research* **112.**

Fernandez-Viagas, V., Molina-Pariente, J. M. & Framinan, J. M. (2020), 'Generalised accelerations for insertion-based heuristics in permutation flowshop scheduling', *European Journal of Operational Research* **282**(3), 858–872.

Fernandez-Viagas, V., Ruiz, R. & Framinan, J. M. (2017), 'A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation', *European Journal of Operational Research* **257**(3), 707–721.

Liu, J. & Reeves, C. R. (2001), 'Constructive and composite heuristic solutions to the P//â^'Cischeduling problem', *European Journal of Operational Research* **132**(2), 439–452.

Maassen, K., Perez-Gonzalez, P. & Günther, L. C. (2020), 'Relationship between common objective functions, idle time and waiting time in permutation flow shop scheduling', *Computers & Operations Research* p. 104965.

Nawaz, M., Enscore, E. & Ham, I. (1982), 'A Heuristic Algorithm for the m-Machine , n-Job Flow-shop Sequencing Problem', *Omega* **11**(1), 91–95.

Öztop, H., Tasgetiren, M. F., Eliiyi, D. T., Pan, Q.-K. & Kandiller, L. (2020), 'An energy-efficient permutation flowshop scheduling problem', *Expert Systems with Applications* **150**, 113279.

Öztop, H., Tasgetiren, M. F., Kandiller, L., Eliiyi, D. T. & Gao, L. (2020), 'Ensemble of metaheuristics for energy-efficient hybrid flowshops: Makespan versus total energy consumption', *Swarm and Evolutionary Computation* **54**(September 2019).

Pan, Q. K. & Ruiz, R. (2013), 'A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime', *Computers & Operations Research* **40**(1), 117–128.

Ruiz, R. & Stützle, T. (2007), 'A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem', *European Journal of Operational Research* **177**(3), 2033–2049.

Taillard, E. (1990), 'Some efficient heuristic methods for the flow shop sequencing problem', *European Journal of Operational Research* **47**(1), 65–74.

Tasgetiren, M. F., Pan, Q.-K., Kizilay, D. & Gao, K. (2016), 'A Variable Block Insertion Heuristic for the Blocking Flowshop Scheduling Problem with Total Flowtime Criterion', *Algorithms* **9**(4), 71.

Just-In-Time Flexible Job Shop with Stochastic Processing Times

Camilo Rodríguez-Espinosa[1], Eliana González-Neira[1]

Pontificia Universidad Javeriana, Bogotá, Colombia
camiloa.rodriguez@javeriana.edu.co, eliana.gonzalez@javeriana.edu.co

Keywords: Stochastic flexible job shop, Simheuristic, Earliness, Tardiness, Robustness.

1 Introduction

In a Flexible Job Shop (FJS) environment, each job has a particular process path that must be completed and, for each process, different machines can perform the task(Pinedo; 2016). This problem exists in many important industries such as textile manufacturing, chemical material processing, aeronautical, semiconductor manufacturing, steelmaking, among others. Due to its combinatorial nature, the FJS is classified as NP-Hard (Demir and İşleyen; 2013; Yang et al.; 2020).

One important aspect to improve customer service level, is the consideration of Just-In-Time (JIT) philosophy (Kong et al.; 2017). This philosophy involves, traditionally, not finishing jobs too early, generating inventory handling costs, and not delivering jobs too late, incurring in penalties with the clients. That is why earliness and tardiness objectives are included as objective functions.

Another key element to consider, is the presence of uncertainties in manufacturing operations. That is why in this project the processing times of the jobs are modeled as random variables. This work uses the predictive approach in which both stochastic and robust modeling are implemented simultaneously. On the one hand, the processing times are modeled through lognormal probability distributions, and a simheuristic method is implemented to solve the problem (Juan et al.; 2015). On the other hand, a robust measure is included as one of the objective functions, ensuring that the predictive schedule does not differ excessively from the realized schedule.

Based on Wu et al. (2020), the objectives to be analyzed are: i) the predictive Earliness+Tardiness (ET), considering deterministic processing times; and ii) the Earliness+Tardiness Risk which is the expected delay of the realized Earliness+Tardiness (ET^R), considering stochastic processing times. Mathematically it can be defined as in equation (1):

$$\textit{Earliness+Tardiness Risk (ET Risk)} = E[max(ET^R - ET, 0)] \tag{1}$$

Taking into account the above elements, this paper solves a stochastic FJS problem, with lognormal processing times, to obtain the Pareto frontier of predictive Earliness+Tardiness and Earliness+Tardiness Risk. To solve the problem a NSGA-II simheuristic is proposed. The NSGA-II is characterized by its low computational requirements, elitist approach, and parameter-less sharing approach to improve solutions at different applications (Deb et al.; 2002).

The rest of the abstract is organized as follows. The next section presents the proposed simheuristic. The computational experiments and results of the simheuristic performance will be reported in Section 3.

2 Proposed NSGA-II simheuristic

This section describes the proposed simheuristic. Bearing in mind the main elements of the NSGA-II, the principles of a simheuristic (Juan et al.; 2015) and the characteristics of the problem, the following aspects are going to be explained: chromosome definition, initial population, order and selection, crossover operator, mutation operator and Monte Carlo simulation.

Chromosome definition: Based on Sun et al. (2019), the coding used for this problem is a real number between $[0, M_{ij})$ for each operation of each job, being M_{ij} the total number of machines that can process the j-th operation of job i. The integer part of the number represents the selected machine to process the job. For example if machines 3, 4 and 7 can process the job and the integer part is 1 (considering 0 as the first position) the machine 4 is selected. The fractional part of the number provides the priority of the job in this operation.

Initial population: For the initial population, some solutions were generated with the application of six dispatching rules, taking the expected values of processing times and adapting them for the FSJ problem. The rules selected were Earliest Due Date (EDD), Shortest Processing Time (SPT), Longest Processing Time (LPT), Critical Ratio (CR), Average Processing time per Operation (AVPRO), and Slack per Remaining Operations (S/OR). The remaining solutions of the initial population were generated at random.

Order and selection: After the initial population is generated, there is a group of N chromosomes. This group will be classified in Pareto frontiers, according to the fast non-dominated sorting procedure and crowding distance. Once the chromosomes are ordered, a pair of chromosomes are selected randomly to apply the crossover operator, until Q offspring are obtained. As a result, a population with size N+Q is obtained. Then the chromosomes of the population are rearranged with the same criteria, and the first N chromosomes are saved. The previous process will be repeated until the maximum number of generations is reached or the running time of the algorithm is ended.

Crossover operator: A random number between 0 and 1 is generated for each position of the chromosome, which defines how the crossover will be performed to create an offspring chromosome. When the random number is less than the crossover probability, the offspring takes the value of the parent's gene in this position. Otherwise, it takes the value of the mother's gene in this position.

Mutation operator: For each offspring, a random number between 0 and 1 is generated to determine if the mutation mutation operator is applied or not to that chromosome. The mutation consists of choosing one job of the chromosome and reversing the order of the fractional values of the operations.

Monte Carlo simulation: Each of the chromosomes generated through the NSGA-II, is simulated through a Monte Carlo simulation to calculate the Earliness+Tardiness Risk, by using a lognormal distribution.

3 Computational experiments

This section presents the computational experiments carried out to evaluate the performance of the proposed simheuristic in comparison with the simulation of solutions given by six dispatching rules. Besides, the behavior of the objective functions is analyzed under different coefficients of variation of the log-normal distribution.

A total of 66 instances from Brandimarte (1993) and Hurink et al. (1994) were evaluated. Considering that these instances do not have a due time window for each job, they were created randomly.

Each instance was run two times with the simheuristic under four different coefficients of variation of processing times (0.25, 0.50, 0.75, 1.00), for a total of 528 Pareto Frontiers. The best value of each objective function obtained in each Pareto Frontier was compared with the simulation of the solutions given by the six mentioned dispatching rules under the same coefficients of variation. Thence, the percentage improvement given by the simheuristic was measured (see equation 2).

$$Improvement = \frac{ObjectiveFunction_{DispatchinghRule} - ObjectiveFunction_{Best}}{ObjectiveFunction_{DispatchingRule}} \quad (2)$$

Table 1 presents the average percentage improvement given by the simheuristic in comparison with the six mentioned dispatching rules.

Table 1. Average percentage improvement of each objective function in comparison with dispatching rules.

Dispatching rule	**ET (%)**	**ET Risk (%)**
EDD	74.01	47.28
SPT	80.13	50.76
LPT	86.83	58.82
CR	71.52	60.88
AVPRO	80.95	48.99
S/RO	55.67	55.82

A non-parametric MANOVA was executed to evaluate the effect of different coefficients of variation in both objective functions. In Table 2 the p-values of MANOVA are presented.

Table 2. Non-parametric MANOVA for the objective functions with different coefficients of variation.

Factor	**ET**	**ET Risk**
Instance	0.001	0.001
Coefficient of variation	0.039	0.001
Instance*Coefficient of variation	0.029	0.001

According to the results obtained, the coefficient of variation has a significant effect on the two objective functions. The averages and standard deviations of both objectives

are shown in Table 3. It can be observed that, as the coefficient of variation increases the Earliness+Tardiness Risk also augment whereas the Earliness+Tardiness is stable. These tests allow us to emphasize the importance of studying randomness when uncertainties are present. Also, it is essential to make an accurate probability distribution fitting to obtain adequate results.

Table 3. Behavior of the objective functions according to the coefficient of variation.

C.V	ET AVG	ET SD	ET Risk AVG	ET Risk SD
0.25	739.58	1313.95	91.34	161.18
0.50	748.05	1299.17	355.41	503.72
0.75	766.67	1313.77	926.51	1039.34
1.00	763.46	1295.01	1757.87	1676.89

For future research, it is recommended to analyze the behavior of other probability distributions and the addition of other stochastic parameters in the model. Finally, we recommend to evaluate this problem with other simheuristic modeling to observe any variation in the results of the objective functions.

Bibliography

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search, *Annals of Operations Research* **41**: 157–183.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation* **6**(2): 182–197.

Demir, Y. and İşleyen, S. K. (2013). Evaluation of mathematical models for flexible job-shop scheduling problems, *Applied Mathematical Modelling* **37**(3): 977–988.

Hurink, J., Jurisch, B. and Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines, *OR Spektrum* **15**: 205–215.

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M. and Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems, *Operations Research Perspectives* **2**: 62–72.

Kong, L., Li, H., Luo, H., Ding, L., Luo, X. and Skitmore, M. (2017). Optimal single-machine batch scheduling for the manufacture, transportation and JIT assembly of precast construction with changeover costs within due dates, *Automation in Construction* **81**: 34–43.

Pinedo, M. L. (2016). *Scheduling*, 5th edition edn, Springer International Publishing.

Sun, L., Lin, L., Li, H. and Gen, M. (2019). Cooperative co-evolution algorithm with an MRF-based decomposition strategy for stochastic flexible job shop scheduling, *Mathematics* **7**.

Wu, Z., Sun, S. and Yu, S. (2020). Optimizing makespan and stability risks in job shop scheduling, *Computers and Operations Research* **122**: 104963.

Yang, Y., Huang, M., Wang, Z. Y. and Zhu, Q. B. (2020). Robust scheduling based on extreme learning machine for bi-objective flexible job-shop problems with machine breakdowns, *Expert Systems with Applications* **158**: 113545.

A Novel Continuous-Time Mixed-Integer Linear Programming Model for the Multi-Mode Resource-Constrained Project Scheduling Problem

Nicklas Klein

Department of Business Administration, University of Bern, Switzerland
nicklas.klein@unibe.ch

Keywords: Resource-Constrained Project Scheduling, Multiple Modes, Mixed-Integer Linear Programming.

1 Introduction

The resource-constrained project scheduling problem (RCPSP) has drawn the attention of many researchers, not only due to its computational challenges but also because of its wide range of practical applications. In many real-world projects, the durations and resource requirements of activities are not fixed, but there may be tradeoffs between higher resource requirements and smaller processing times. These tradeoffs can be represented through multiple execution modes of the activities, which is considered an extension of the RCPSP. This extension is called multi-mode RCPSP (MRCPSP) and has been widely studied in the literature (cf., e.g., Mika *et. al.* (2015) for an overview).

In the MRCPSP a set of activities is given that can be executed in different modes and a set of completion-start precedence relations between pairs of activities. The activities require renewable resource capacities, e.g., personnel, and also nonrenewable resources, e.g., raw materials. While renewable resources are constrained during each time period, nonrenewable resources are limited throughout the whole project. The optimization problem determines the start time and execution mode of each activity such that the renewable and nonrenewable resource capacities are not exceeded, all precedence relations are met and project completion time (makespan) is minimized.

Multiple mixed-integer linear programming (MILP) models for the MRCPSP have been introduced in the literature. These models can be divided into two categories: discrete-time (DT) and continuous-time (CT) models. In DT models, the time horizon is split into equidistant time intervals, and each activity can only start at the beginning of those intervals. To the best of our knowledge, the current state-of-the-art DT model was introduced by Talbot (1982). For each activity, the model includes a binary variable for each combination of a start time and a mode to indicate whether the activity starts at the respective time in the respective mode. It is, therefore, of pseudopolynomial size. In contrast, CT models allow continuous activity start times. To the best of our knowledge, the current state-of-the-art CT model was proposed by Gnägi *et. al.* (2019). Their model relies on explicitly assigning resource units to activities, which also leads to a pseudopolynomial model. Therefore, both state-of-the-art models include a large number of variables and constraints if the time horizon or the resource capacities, respectively, are large.

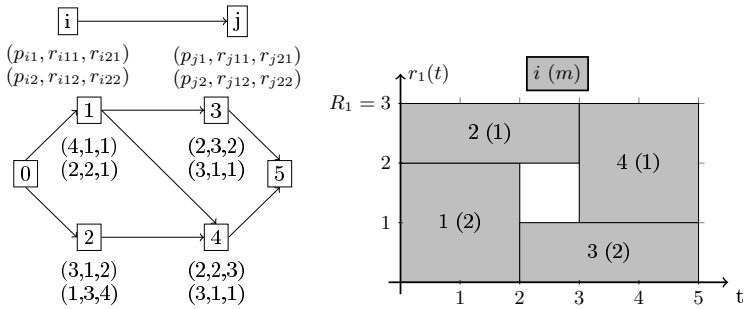
In this paper, we analyze a novel continuous-time MILP model for the MRCPSP, which is of polynomial size (compact). We introduce two kinds of sequencing variables that, in combination, allow us to identify which activities are processed in parallel and whether renewable resource constraints are met. We tested the novel model against the current state-of-the-art DT and CT models on two sets of benchmark instances containing 20 activities per instance. The results indicate that the novel model outperforms both considered

models from the literature on instances with relatively long activity durations and performs comparably well on instances with rather short durations.

The remainder of this paper is structured as follows. In Section 2, we provide an illustrative example of the MRCPSP. In Section 3, we present the variables used in the novel model. The results of our computational studies are given in Section 4. Finally, in Section 5, we conclude our findings and provide an outlook for further research.

2 Illustrative example

In this section, we illustrate the problem setting of the MRCPSP using the example of a project consisting of $n = 4$ real activities. We denote the set of activities as $V = \{1, \ldots, 4\}$. The activities require units of one renewable resource ($k = 1$) and one nonrenewable resource ($k = 2$). To model the project start and completion, we add two fictitious activities 0 and 5 to the problem. These activities have a length of zero and no resource requirements. Each activity $i \in V$ can be executed in two different modes, i.e., $M_i = \{1, 2\}$. The precedence relations of the considered example are displayed in the activity-on-node network in Fig. 1(a); an arc between two activities (i, j) denotes that activity i has to be completed before the start of activity j. Furthermore, the graph displays the activity durations p_{im}, the renewable resource requirements r_{i1m} and the nonrenewable resource requirements r_{i2m} for each real activity $i \in V$ and both modes $m \in \{1, 2\}$. The capacity of the renewable resource is $R_1 = 3$ and that of the nonrenewable resource is $R_2 = 8$. An optimal solution to the described instance is displayed in Fig. 1(b).



(a) Precedence relations, durations and resource requirements

(b) Optimal solution

Fig. 1. Illustrative example for the MRCPSP.

3 Variable types in the novel model

In this section, we explain the variable types used in the novel CT model through our illustrative example. First, we use continuous variables S_i to represent the start time of each activity $i \in V \cup \{n+1\}$. To model the mode choices, we introduce a binary variable x_{im} for each real activity $i \in V$ and each execution mode $m \in M_i$ of that activity. Furthermore, for each pair of activities $i, j \in V \cup \{n + 1\},\ i \neq j$, we introduce two kinds of sequencing variables. The first type is denoted as y_{ij} and is equal to one if activity i is completed before the start of activity j and zero otherwise. These variables are also used to model the

precedence relations. The second type is denoted as z_{ij} and is equal to one if activity i starts anytime before or at the same time as activity j and zero otherwise. The combination of these two sequencing variables allows us to check at the start of each activity which other activities have already been started but not yet completed. Together with a continuous auxiliary variable, this allows us to model the renewable resource constraints, as we will demonstrate using our illustrative example from Section 2.

We display the values of the start time variables, mode-choice variables and, for a selection of interesting activity pairs, the sequencing variables in the optimal solution in Fig. 2. First, we consider the pair of activities one and two, which start at the same time. Therefore, both variables z_{12} and z_{21} are equal to one, and due to the overlap, the y-variables are equal to zero. This means that the renewable resource requirement of activity two has to be considered at the start of activity one and vice versa. Second, we look at the pair of activities three and four, where activity four starts later than the start but still before the completion of activity three. Consequently, we do not have to consider activity four at the start of activity three because it has not yet started ($z_{43} = 0$). From the perspective of activity four, however, we have to consider activity three because it started processing ($z_{34} = 1$) but is not yet completed ($y_{34} = 0$). Last, we consider activities one and three, where activity three starts after the completion of activity one. As a result, variables z_{13} and y_{13} are equal to one, and z_{31} and y_{31} are zero. When starting activity three, we have to consider the requirements of activity one because it starts earlier ($z_{13} = 1$), but since it is already completed ($y_{13} = 1$), we can ignore it.

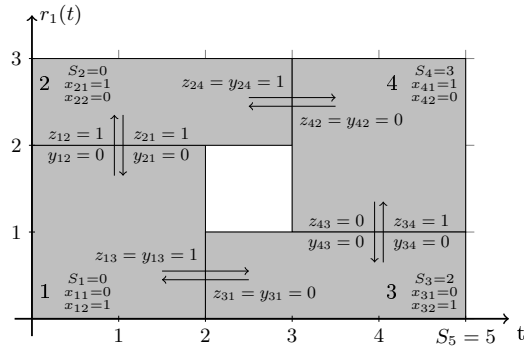


Fig. 2. Illustration of the variables used in the novel model.

4 Computational results

In this section, we present the results of our computational studies. We analyzed the performance of the novel model and compared it to the DT model of Talbot (1982) and the CT model of Gnägi *et. al.* (2019). As test sets, we used the J20 benchmark instances from the PSPLIB (Kolisch and Sprecher 1996) and a modification of these instances. The J20 set contains 554 feasible instances with 20 activities, each of which can be executed in three modes and requires capacities of two renewable and two nonrenewable resources. Since the activity durations in the J20 set are rather short, we considered an additional set of instances (D20). Gnägi *et. al.* (2019) created this set by enlarging the durations of half of the activities in each of the J20 instances. We implemented all models in Python 3.8

and used Gurobi 9.1.2 as the solver. For each instance, we set a time limit of 500 seconds and limited the maximum number of threads to two.

For both instance sets and each model, we report the following key figures: the number of instances for which a feasible and proven optimal solution was found and the average MIP gap and CPU time over all instances of the respective set (Table 1); the respective *best* values are marked in bold. We observe that for set J20, the novel model solves more instances to optimality than the CT model and, on average, has a smaller MIP gap and solution time. Compared to the DT model, it can solve as many instances to optimality. For set D20, the novel model performs best, solving all instances in (on average) under one second of CPU time.

Table 1. Computational results for the J20 and D20 instance sets.

J20	# Feas	# Opt	Avg. MIP-gap (%)	Avg. time [sec]
Tal82	**554**	**541**	**0.14%**	**22.42**
Gna19	**554**	490	1.47%	73.08
Novel	**554**	**541**	0.44%	24.07

D20	# Feas	# Opt	Avg. MIP-gap (%)	Avg. time [sec]
Tal82	520	457	4.44%	161.03
Gna19	**554**	548	0.04%	12.55
Novel	**554**	**554**	**0.00%**	**0.82**

5 Conclusion and outlook

In this paper, we introduced a novel, compact continuous-time MILP model for the multi-mode RCPSP. The model relies on two types of sequencing variables that allow us to identify overlaps between pairs of activities. Our computational results indicate that the novel model can outperform the state-of-the-art CT model on all instances and the DT model for instances with long durations.

For future research, we propose to analyze whether adding valid constraints to the novel model can further improve its performance. Furthermore, we plan to modify existing CT models for the single-mode RCPSP to the MRCPSP and compare the performance to our novel model. Moreover, we intend to adapt the novel model to other RCPSP variants.

References

Gnägi M., T. Rihm and A. Zimmermann and N. Trautmann, 2019, "Two continuous-time assignment-based models for the multi-mode resource-constrained project scheduling problem", *Computers & Industrial Engineering*, Vol. 129, pp. 346-353.

Kolisch, R., A. Sprecher, 1996, "PSPLIB - A project scheduling problem library", *European Journal of Operational Research*, Vol. 96, pp. 205-216.

Mika M., G. Waligóra and J. Węglarz, 2015, "Overview and state of the art", In: Schwindt C., Zimmermann J. (eds) *Handbook on Project Management and Scheduling Vol. 1*, pp. 445-490. Springer, Cham.

Talbot F.B., 1982, "Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case", *Management Science*, Vol. 28(10), pp. 1197-1210.

Aggregation techniques for scheduling on parallel machines in semiconductor manufacturing

Jérémy Berthier[1,2], Stéphane Dauzère-Pérès[1], Claude Yugma[1], Alexandre Lima[2]

[1] École des Mines Saint-Étienne, UMR 6158 LIMOS, CMP, Department of Manufacturing Sciences and Logistics, Gardanne, France
j.berthier@emse.fr, dauzere-peres@emse.fr, yugma@emse.fr
[2] STMicroelectronics Crolles, Department of Decisional Solutions, Team of Full Automation and Simulation, Crolles, France
jeremy.berthier@st.com, alexandre.lima@st.com

Keywords: Parallel machines, aggregation techniques, semiconductor manufacturing.

1 Introduction

Semiconductor manufacturing includes the most complex manufacturing processes. Mönch et al. (2011) and Mönch et al. (2012) showed that the performances of the industry highly rely the effective scheduling of semiconductor wafer fabrication facilities. Scheduling problems to be addressed at the operational level involve a rich set of constraints and criteria. As a result, optimization algorithms are increasingly preferred over dispatching rules, in particular in complex production areas such as the photolithography area considered in this paper.

2 Problem description

The scheduling problem in the photolithography area consists in scheduling a set of jobs on a set of parallel photolithography machines. Each job requires an additional resource, called reticle or mask, that can be transported from one machine to another. Various papers in the literature have studied different version of this problem, and have presented mathematical models and heuristics. For example, Mönch et al. (2002) propose a genetic algorithm to solve the photolithography scheduling problem while considering send-ahead wafers. A recent short overview of the literature can be found in Bitar et al. (2021).

The characteristics of the problem are summarized below.

Jobs Each job needs to be processed with a given priority. Few jobs have precedence constraints (at most two operations, and the problem is not formally considered to be a flexible job-shop scheduling problem), and must not exceed a maximum time lag. Engaged maximum time lag constraints when starting the schedule are modelled by assigning a deadline to each job.

Photolithography machines Each photolithography machine is qualified to process a limited subset of the jobs. Machines can only process one job at time, and may not be available before a certain date. The processing of a job on a machine cannot be interrupted (no preemption), and processing times depend on both the job and the machine. Finally, a machine-dependent and sequence-dependent setup time is required to start the processing of a job.

Reticles Each job requires one reticle, which is available in a single copy. Jobs are assumed to have a competitive access to reticles (otherwise, considering them would be trivial). Besides, transporting a reticle between two machines takes a transportation time that depends on the location of the machines.

3 Multi-objective scheduling

The scheduling problem is addressed through an Integer Linear Programming (ILP) model which relies on Bitar et al. (2016) and Bitar (2021). The scheduling horizon is divided into periods of equal duration. A feasible solution is represented by binary decision variables associated with each triplet (job, machine, period). Several linear constraints are defined in order to model operations within the photolithography area. Moreover, all jobs must be scheduled before the end of the time horizon.

Several objective functions are defined based on the factory requirements. All the criteria must be considered in the optimization problem, which makes it multi-objective. Overall, three categories of criteria are studied:

- Criteria implementing relaxed operational constraints, such as the minimization of a total risk function associated with the maximum time lag constraints and deadlines;
- Criteria to meet production targets of the manufacturing area:
  - Minimize the sum of the completion times of jobs,
  - Maximize the number of jobs scheduled before a certain time threshold;
- Criteria to improve the operational efficiency:
  - Minimize the total setup time of machines,
  - Minimize the number of reticle moves.

In the following, only the minimization of the sum of of the completion times of jobs is considered to compare the aggregation methods presented in the next section.

4 Aggregation methods

The ILP problem discussed in Section 3 is intractable for large instances (see Section 5.1) when the scheduling horizon is discretized in minutes. To improve its efficiency, we propose to reduce the problem size using two aggregation methods. Section 4.1 presents a method that batches (or combines) multiple jobs using the same reticle to create a single job. In Section 4.2, the use of larger periods (time steps) than one minute when discretizing the scheduling horizon is proposed.

4.1 Batching similar jobs

The first method consists in batching similar jobs, i.e. combining multiple jobs using the same reticle as one single job. Indeed, jobs with the same reticle belong to the same setup group, which means that no setup time is required between these jobs. Moreover, batching jobs helps to generate more acceptable solutions from a practical point of view, in particular by avoiding unnecessary mask changes on the same machine and reticle transportation between machines.

Jobs in the same batch are selected such that:

- They are not involved in a precedence relationship;
- They are not in-process;
- No setup times are required between any sequence of the jobs;
- They have at least one common qualification, which means that $\cap_{j \in b} \mathcal{M}_j \neq \emptyset$.

In a batch, jobs are ordered using Smith's rule applied to the average total processing time of the batch over qualified machines, i.e. by ascending order of the criterion $\frac{\sum_{\cap_{j\in b}\mathcal{M}_j} p_{j,m}}{\omega_j \cdot |\cap_{j\in b}\mathcal{M}_j|}$. In the optimization model, each batch b becomes a new job whose priority is $\omega_b \leftarrow \max_{j\in b} \omega_b$.

4.2 Increasing the time step

The second method consists in increasing the discretization period, or time step. This method is motivated by the small dispersion of the processing times in the photolithography area, contrary to academic instances where ranges are generally much larger. Besides, the realized processing times are in practice often different than the scheduled ones (since average processing times are used) because of the uncertainty in the production processes.

If Δ denotes the time step, any time-related parameter α_t is modified as follows: $\alpha_t \leftarrow \frac{\alpha_t}{\Delta}$. Since decision variables are time-indexed, this results in dividing the number of decision variables by Δ. The resulting schedule is adjusted in a post-processing algorithm that uses the initial value of α_t.

5 Numerical results

These experiments have been conducted with 5 industrial instances derived from one of the photolithography areas of STMicroelectronics, that comprises about 23 machines and less than 80 reticles. All the experiments are conducted using IBM ILOG CPLEX 12.9.

5.1 Computational results

As shown in Table 1, solution time of the ILP model grows very fast with the number of jobs to be scheduled. For example, up to 1,000,000 binary decision variables can be generated. Such computational times are not tractable since jobs must be scheduled in real time. This motivated the development of the two aggregation methods tested hereinafter.

Table 1. Solution time as a function of the number of jobs

Number of jobs	40	45	60	70	90	120
Min solution time [in s]	*0.2*	*3*	*9*	*27*	*102*	*7,365*
Average solution time [in s]	0.7	14	18	60	1,376	7,415
Max solution time [in s]	*1.4*	*41*	*31*	*172*	*5,743*	*7,488*

5.2 Aggregation methods

Both methods show promising results for small and medium size instances, as illustrated in Tables 2 and 3. Regarding the method in Section 4.1, few jobs (around 16) are batched because more than half of the reticles are required by exactly one job. Nevertheless, the serial batching of jobs generates a significant time saving (56.8% on average) while keeping a reasonable gap to the optimal objective function (1.5% on average). Although the gap to an optimal solution is sparsely dispersed, time savings have a larger range (from 99% to -2%) that depends on the number of batches and their sizes.

Table 2. Average solution time saving and optimality gap with batched jobs

Instance	**Nb of batches**	**Average batch size**	**Time saving**	**Gap**
A (84 jobs)	7	2.5 jobs	35%	0.5%
B (82 jobs)	6	3 jobs	99%	2%
C (92 jobs)	6	3.5 jobs	76%	2%
D (87 jobs)	3	3 jobs	-2%	1%
E (87 jobs)	5	3.4 jobs	78%	2%
Average (86.4 jobs)	**5.4**	**3.08 jobs**	**56.8%**	**1.5%**

For the sake of the experiments, the time discretization is adjusted with two values: the minimum and maximum processing times on machines (5 min and 15 min, respectively). As expected, when the time step is increased, the gap to the optimal slightly deteriorates (from 0.6% to 2.3%) but the time saving is improved and (from 99.6% to 98.1%). Contrary to to the first method, the time saving is better controlled but not the optimality gap, which is a little more scattered.

Table 3. Average solution time saving and optimality gap for time step increase

Instance	$\boldsymbol{\Delta = \max_{j,m}\{p_{j,m}\}} \approx 15\ min$		$\boldsymbol{\Delta = \min_{j,m}\{p_{j,m}\}} \approx 5\ min$	
	Time saving	**Gap**	**Time saving**	**Gap**
A (84 jobs)	99.8%	4.3%	99.1%	0.4%
B (82 jobs)	99.9%	4.1%	99.8%	0.7%
C (92 jobs)	99.3%	0.9%	97.3%	0.3%
D (87 jobs)	99.1%	1.5%	95.6%	1.3%
E (87 jobs)	99.7%	0.6%	98.6%	0.3%
Average (86.4 jobs)	**99.6%**	**2.3%**	**98.1%**	**0.6%**

6 Conclusions

Two aggregation methods were presented to solve a scheduling problem on parallel machines in semiconductor manufacturing. Numerical results on industrial instances were conducted using an advanced ILP model (that will be presented in detail at the conference). The aggregation methods are motivated by the industrial requirements and specific features of the problem, and show promising results that will be used to improved the multi-objective optimization of the problem.

References

Bitar A., Dauzère-Pérès S., and Yugma C. (2021). Unrelated parallel machine scheduling with new criteria : Complexity and models. Computers Operations Research, page 105291, 2021.

Bitar A., Dauzère-Pérès S., Yugma C., and Roussel R. A Memetic Algorithm to Solve an Unrelated Parallel Machine Scheduling Problem with Auxiliary Resources in Semiconductor Manufacturing. Journal of Scheduling, 19(4): 367-376, 2016.

Mönch, L. (2002). A genetic algorithm heuristic applied to stepper scheduling. In Proceedings of the International Conference on Modelling and Analysis of Semiconductor Manufacturing, Tempe, USA, (pp. 276-281).

Mönch L., Fowler J. W., Dauzère-Pérès S., Mason, S. J., and Rose O.(2011). A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. Journal of Scheduling, 14(6), 583-599.

Mönch L., Fowler J. W., and Mason S. J. (2012). Production planning and control for semiconductor wafer fabrication facilities: Modeling, analysis, and systems. New York: Springer.

New empirical and artificial data instances for the multi-skilled resource-constrained project scheduling problem: data generation and implementation

Jakob Snauwaert[1], Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
jakob.snauwaert@ugent.be mario.vanhoucke@ugent.be
[2] Operations and Technology Management Centre, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, United Kingdom

Keywords: Project Management, Resource-Constrained Project Scheduling, Multi-skilled Resource Assignment.

1 Introduction

In recent years, organizations have been driven by a combined effort of trying to execute cost-efficient processes while allowing for customer specific request to individualize the final product design (Gutjahr, *et al.* 2010). This operational trade-off induced a requirement for highly efficient as well as flexible workers in current organizations (Riley, *et al.* 2017). As such, today's workers need not only be productive, they also need to be able to handle a substantial amount of different jobs and to agree with having very dynamic job descriptions. A direct consequence from this is that current workforces consist of a set of complementary multi-skilled resources that are capable of executing as many different projects as possible (Nembhard and Uzumeri 2000). This fixated vision on skills and competences is especially important because it is less likely or improbable that these skills and competences can be duplicated or imitated, which yields an important advantage over other competitive organizations. Consequently, investments in development and training of human resources, with improved results in mind, are a critical part of people management and of the project's success (Riley, *et al.* 2017).

The research that will be presented in this abstract revolves around project scheduling with resource constraints, more specifically, it focuses on resources as human beings and how they can be employed optimally in a project. It deals with the efficiency and involvement of skills in projects and on the integration of multi-skilled teams. The main focus lies on the *multi-skilled resource-constrained project scheduling problem*, further abbreviated as MSRCPSP, which is an activity scheduling and resource assignment problem with resources who master several different skills. This problem is an extension of the resource-constrained project scheduling problem (RCPSP), which is proven to be NP-hard (Blazewicz, *et al.* 1983). Overviews of the research on the RCPSP and its extensions can be found in Hartmann, *et al.* (2010) and Hartmann, *et al.* (2021). There are two main differences between the RCPSP and the MSRCPSP. Firstly, the resource requirements are now skill requirements, which means that activities do not only require a certain number of resources to be executed, but these multi-skilled resources also need to have an appropriate set of skills to be able to perform the jobs specified by the activity. Secondly, each resource is considered to be an individual human being that masters a certain set of skills instead of a uniform resource. Consequently, this means that the multi-skilled resources can not work on every activity in the project and the resource assignment subproblem becomes more difficult to solve. Finally, the goal of the MSRCPSP is to construct feasible activity schedules and feasible multi-skilled resource assignments that minimises, in most cases, the project makespan.

In the last two decades, the number of research articles on the MSRCPSP has increased substantially. Néron (2002) was the first to introduce resources that were characterized by skillsets into the RCPSP. In the years after this publication, multiple heuristic and exact solutions methods were presented for the MSRCPSP by, among others, Bellenguez and Néron (2007) and Correia, *et al.* (2012). More recently, a considerable growth in the number of research papers studying the MSRCPSP has occurred, most of which proposed a metaheuristic approach (Almeida, *et al.* (2018), Zabihi, *et al.* (2019) and Snauwaert and Vanhoucke (2021)). An extensive overview of scheduling problems with multi-skilled resources is addressed by Afshar-Nadjafi, B. (2020).

2 Problem Definition

In the MSRCPSP, a project can be represented by an acyclic activity-on-the-node network $G = (N, A)$, in which the minimum precedence relationships with a time-lag of zero are characterised by the arc set A and the activities by the node set N. The project consists of $|N| - 2$ real activities and two additional dummy activities that represent the start and the end of the project. We make the assumption that both these dummy activities have no resource or skill usage and a duration of zero. The activities are topologically ordered, that is, an activity always has a higher label than all of its predecessors. The project is said to be scheduled without pre-emption within the precedence relations and with a set of renewable multi-skilled resources R, with $R = 1, ..., |R|$, resulting in a baseline schedule with an activity starting time s_i and finishing time f_i for each activity $i \in N$ that minimises a certain objective function, e.g makespan or cost.

Each activity $i \in N$ has a certain predefined duration p_i and a demand for renewable resources, that is defined differently than for the RCPSP due to the presence of skills in the problem formulation. While the resource constraints for the RCPSP are defined as a set of renewable resource types (with index $k = 1, ..., |K|$), and each activity i is linked to a resource k by its resource demand r_{ik} for each resource type k, the resource requirements are adjusted for skills in the MSRCPSP. Each activity $i \in N$ requires r_{ij} resources that master skill j of the set of available skills J for project G. More specifically, an activity i has a certain resource demand that is defined as a skill requirement for the skill j of a resource k. Note that a resource k can only be assigned to no more than one activity every time unit $t \in T$.

Every resource k from the set R masters at least one skill of the set of skills J, with $J = 1, ..., |J|$, defined by the *categorical skills* b_{jk}, which is equal to 1 if resource k masters skill j and 0 otherwise. The number of resources that masters a certain skill j is equal to the total skill availability a_j of that skill j. Furthermore, in some extensions to the MSRCPSP the multi-skilled resources are also characterized by *hierarchical skills*. These hierarchical skills are represented by the skill level distribution b^l_{jk}, which defines the hierarchical level $l \in L$ at which a resource $k \in R$ masters the skill $j \in J$. Similarly to b_{jk}, b^l_{jk} is equal to 1 if resource k masters skill j at level l and 0 otherwise. As such we assume that for the MSRCPSP without hierarchical skills for multi-skilled resources, the number of levels $|L|$ is equal to 1. The order of the hierarchical skills can indicate the inherent differences between the resources that can be outed in a substantial variety of ways dependent on the problem at hand and its specifics. For instance, higher hierarchical skill level values can reduce the processing time of an activity, while lower levels can increase the processing time. Along the same lines, resources with higher skill levels can be more expensive, qualitative, flexible or efficient, all of which will influence the final project schedule in their own manner.

3 Data generation procedure

The main structure of the generation procedure is displayed in Figure 1. For each instance, we start off by retrieving a network from the Rangen2 generator (Vanhoucke, *et al.* 2008) based on the number of activities $|N|$ and the serial-parallel network indicator SP. Next, the skill requirements r_{ij} are calculated from the project skill factor $SF \in [0, 1]$ parameter, which determines the number of skill types that are required by each activity in the project, and a desired variation in the skill requirements $SF_\alpha \in [0, 1]$, which is equal to 0 if all activities require the same amount of skill types. The variation in skill requirement increases with an increasing value of SF_α.

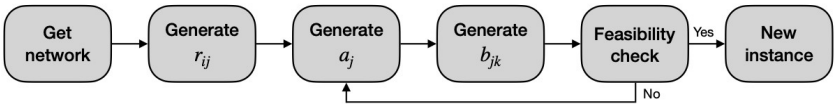


Fig. 1. Flow of data generation procedure

Based on the skill requirements r_{ij} and a project skill strength parameter $SS \in [0, 1]$ that determines the total skill scarceness in the project, we calculate the skill availability a_j for all skills $j \in J$. The average skill availability in the project is directly related to the value of the project skill strength SS. Additionally, we incorporate the skill strength variability $SS_\alpha \in [0, 1]$ that introduces variability into the skill availabilities of all skills $j \in J$. As such, we can generate instances where all skills are equally scarce and instances where some skill types are very scarce while others are widely available. Afterwards, we generate the skill distribution b_{jk} of the multi-skilled workforce using the acquired skill availability a_j, the resource availability parameter $RA \in [0, 1]$ and the resource availability variability $RA_\alpha \in [0, 1]$. The resource availability RA determines the number of resources over which the available skills will be distributed, while the resource availability variability allows us to generate workforces in which all resources master the same amount of skills or workforces that consist of a combination of single-skilled and completely-skilled resources. Finally, we implement the rejection method for this data generator by performing a feasibility check on the new instance. When generating the skill distribution b_{jk} we always assure that the required skills for all activities are available in the workforce, but due to the fact that multiple skills can be mastered by the same resources, it could be that some activities turn out to be infeasible for the generated workforce. Therefore, we solve an assignment problem for each activity to check whether the complete instance is feasible. For the SP, we generated instances with values from the following set $\{0.1, ..., 0.9\}$. Since we generated instances with 4 skill types, $|S| = 4$, the instances included SF values ranging from 0.25 to 1. For all other parameters, instances can be found for the full range of the parameter $[0, 1]$.

In addition to these artificial instances, we have also gathered a total of 7 empirical instances for the MSRCPSP that are collected from a software company and companies that operate in the railway construction industry. Of these instances, 4 have a workforce with only single-skilled resources, while the other 3 projects were executed by a multi-skilled workforce. These empirical instances are used to validate and substantiate the artificially generated instances of this procedure.

4 Conclusion

In this abstract, we present a new data generation procedure for the multi-skilled resource-constrained project scheduling problem. The generator makes use of multiple skill-specific parameters that determine the skill requirements and the skill distribution among the resources. Additionally, we have incorporated variability parameters that are based on the $\frac{\alpha}{\alpha_{max}}$ principle (Labro and Vanhoucke 2008) in order to introduce variation of the requirements over the activities and variation over the resources in terms of skill availability. All artificial instances of the MSLIB set from this procedure as well as the empirical instances for the MSRCPSP are available on our website `https://www.projectmanagement.ugent.be/research/project_scheduling/MSRCPSP`.

Acknowledgements

The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, FWO and the Flemish Government - department EWI.

References

Afshar-Nadjafi, B. (2020). Multi-skilling in scheduling problems: A review on models, methods and applications, *Computers & Industrial Engineering*, page 107004.

Almeida, B. F., Correia, I., and Saldanha-da Gama, F. (2018). A biased random-key genetic algorithm for the project scheduling problem with flexible resources, *TOP*, pages 1-26.

Bellenguez-Morineau, O., Néron, E., 2007. A branch-and-bound method for solving multi-skill project scheduling problem, *RAIRO-Operations Research*, 41 (2), pp. 155-170.

Blazewicz, J., Lenstra, J. K. and Kan, A. R., 1983. Scheduling subject to resource constraints: classification and complexity, *Discrete applied mathematics*, 5 (1), pp. 11-24.

Correia, I., Lampreia-Lourenc Ì§o, L., Saldanha-da Gama, F., 2012. Project scheduling with flexible resources: formulation and inequalities, *OR spectrum*, 34 (3), pp. 635-663.

Gutjahr W.J., Katzensteiner, S., Reiter, P., Stummer, C., and Denk, M., 2010, Multi-objective decision analysis for competence-oriented project portfolio selection, *European Journal of Operational Research*, 205 (3), pp. 670-679.

Hartmann, S. and Briskorn, D., 2010. A survey of variants and extensions of the resource-constrained project scheduling problem, *European Journal of operational research*, 207(1):1-14.

Hartmann, S. and Briskorn, D., 2021. An updated survey of variants and extensions of the resource-constrained project scheduling problem, *European Journal of operational research*,

Labro, E. and Vanhoucke, M. (2008). Diversity in resource consumption patterns and robustness of costing systems to errors. *Management Science*, 54(10):1715-1730.

Nembhard, D. A., and Uzumeri, M. V., 2000. Experiential learning and forgetting for manual and cognitive tasks, *International journal of industrial ergonomics*, 25 (4), pp. 315-326.

Néron, E., 2002. Lower bounds for the multi-skill project scheduling problem, *Proceeding of the Eighth International Workshop on Project Management and Scheduling*, pp. 274-277.

Riley, S. M., Michael, S. C. and Mahoney, J. T., 2017. Human capital matters: Market valuation of firm investments in training and the role of complementary assets, *Strategic Management Journal*, 38 (9), pp. 1895-1914.

Snauwaert, J. and Vanhoucke, M. 2021. A new algorithm for resource-constrained project scheduling with breadth and depth of skills, *European Journal of Operational Research*, 292(1):43-59.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., and Tavares, L. V. 2008. An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 187(2):511-524.

Zabihi, S., Kahag, M. R., Maghsoudlou, H., and Afshar-Nadjafi, B. 2019. Multi-objective teaching-learning-based meta-heuristic algorithms to solve multi-skilled project scheduling problem, *Computers & Industrial Engineering*, 136:195-211.

Maximal slacks between lower bounds of the makespan on parallel processors

Jacques Carlier[1] and Claire Hanen[23]

[1] Heudiasyc UMR CNRS 7253, Sorbonne Universités, Université de Technologie de Compiègne, Compiègne, France
[2] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France‘
[3] UPL, Université Paris Nanterre, France
jacques.carlier@hds.utc.fr, claire.hanen@lip6.fr

Keywords: scheduling, makespan, parallel processors, worst case bound, lower bound

1 Introduction

Several lower bounds are known for a long time for the m- machines scheduling problem with release dates and tails usually denoted by $P|r_i, q_i|C_{\max}$. The three main bounds are the preemptive bound, the Jackson Pseudo Preemptive makespan (Carlier & Pinson 1998) and the Energetic Reasoning bound (Erschler & Lopez 1990). There has been a lot of efforts in the literature to improve the computation time of these bounds, among them we can cite Baptiste et al. , 1999, Tercinet et al. , 2004, Hidri et al. , 2008, Ouellet & Quimper , 2018, Carlier et al. , 2021. Such bounds are the basis of most branch and bound algorithms or approximation algorithms for more complex problems (Néron 2008, Laborie & Nuijten 2008, Haouari et al. 2012, Haouari et al. 2014, Carlier & Néron 2003, Hanen et al. 2021).

In a recent paper, Carlier et al. , 2020 characterized mathematically these three classical lower bounds for the m-machine scheduling problem. The paper give some insights into the ways in which the bounds differ and their experiments show how close to each other are these bounds.

The aim of our work is to exhibit tight values of the maximal slack between the three lower bounds, to which we add the upper bound on the optimal preemptive schedule defined by the Jackson preemptive schedule (Jackson 1955). We prove that the slack is always less than the maximum processing time of a job, and we characterize more accurately the worst case slack depending on the structure of the schedules or pseudo-schedules. Proofs are omitted in this extended abstract. Section 2 defines the notations for the rest of the paper. In Section 3 we characterize the structure of a worst case and we get a tight bound on the slack between pseudo-preemptive and preemptive bound. Section 4 measures the worst case slack between energetic bound and pseudo-preemptive bound, and also provide structure of worst cases. And finally Section 5 discuss the slack between the jackson preemptive schedule and the jackson pseudo-preemptive schedule.

2 Problem definition and notations

We consider in this paper the two problems usually denoted by $P|r_i, q_i, pmtn|C_{\max}$ and $P|r_i, q_i|C_{\max}$ defined by:

- A set of n jobs $\mathcal{T}$
- For each job $J_i \in \mathcal{T}$, a release date r_i,a processing time p_i, a tail q_i
- m parallel processors

A non preemptive schedule σ assigns to each job J_i a completion time $C_i(\sigma) \geq r_i + p_i$ such that at most m jobs are performed at each time t:

$$\forall t, |\{J_i, C_i(\sigma) - p_i \leq t < C_i(\sigma)\}| \leq m$$

In a preemptive schedule the jobs might be interrupted several times before their completion, and is usually defined by an instantaneous rate function $\rho_i(t)$ for each job J_i and each time t. The optimization criteria is in both cases the makespan $C_{\max}(\sigma)$ defined by :

$$C_{\max}(\sigma) = \max_{J_i \in \mathcal{T}} C_i(\sigma) + q_i$$

In Carlier & Pinson , 1998 is defined the notion of pseudo-preemptive schedule, in which the instantaneous rate might at some condition be greater than 1, and which defines a lower bound on the makespan of the optimal preemptive schedule. Energetic reasoning introduced in Erschler & Lopez , 1990 gives a lower bound on the optimal makespan of a non preemptive schedule. In the rest of the paper we denote by $C^{PR}_{\max}$ the optimal makespan of a preemptive schedule, by $C^{JPPS}_{\max}$ the optimal makespan of a pseudo-preemptive schedule, and by $C^{ER}_{\max}$ the lower bound based on energetic reasoning.

3 Slack between pseudo-preemptive and preemptive bound

During this work, we defined an algorithm to build a preemptive schedule from the optimal pseudo-preemptive schedule, the makespan of which is at most $C^{JPPS}_{\max} + p_{\max}$. We then used the Gale conditions (Gale 1957) on the optimality of network flows to characterize the structure of the optimal preemptive schedule with subsets of tasks and their time intervals. Then assuming a critical subset, we defined eight elementary transformations of an instance that do not decrease $C^{PR}_{\max}$ and do not increase $C^{JPPS}_{\max}$ nor $p_{\max}$.

This allows us to characterize the structure of the worst case instance as illustrated by figure 1 and call it a Bandoneon instance which is composed with K subsets of bridge jobs and $K+1$ pillars of sand jobs (i.e. very small jobs). From this structure we derive a mathematical program whose variables are the slack between $C^{PR}_{\max}$ and $C^{JPPS}_{\max}$ and for each bridge k the volume of its part x_k performed in its left pillar (resp z_k in its right pillar), its thickness e_k and the volume of its part performed between the pillars y_k in the optimal preemptive schedule.

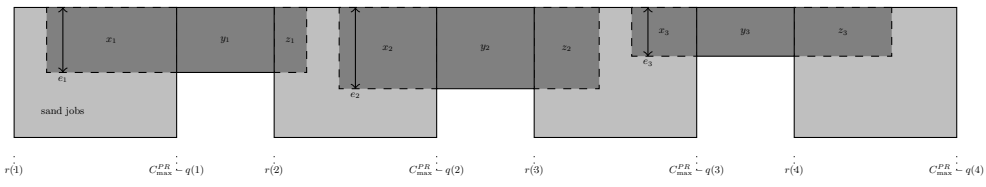


Fig. 1. Optimal preemptive schedule of a bandoneon instance with 3 bridges

Based on the study of dominant solutions of this program, we then establish the following bound on the slack between $C^{PR}_{\max}$ and $C^{JPPS}_{\max}$ and we prove it is tight. In the worst case instance all bridges have the same thickness.

Theorem 1. *for a worst case instance with K bridges, we get*

$$C^{PR}_{\max} - C^{JPPS}_{\max} \leq \min\left(\frac{m-1}{m}, \frac{(\sqrt{K+1}-1)^2}{K}\right) \times p_{\max}$$

Fig. 2. Pseudo-preemptive schedule of the bandoneon instance of Figure 1

4 Slack between energetic bound and preemptive bounds

Carlier et al. , 2020 gives a characterization of critical intervals for the energetic bound $C^{ER}_{\max}$ with respect to the intermediate bound called crossing jobs bound and denoted by $C^{CJ}_{\max}$. In this secton we use this result to analyse the slack between $C^{ER}_{\max}$, $C^{PR}_{\max}$ and $C^{JPPS}_{\max}$. For a given makespan C, a job J_i is said to be a crossing job at time t if $t \in [C - q_i - p_i, r_i + p_i)$. The crossing job bound corresponds to the minimal value of C the ending point such that there are no more than m crossing jobs at each time t.

Case 1 We first consider instances such that $C^{PR}_{\max} > C^{ER}_{\max}$. Then we can establish that for some dominant bandoneon instances, we have $C^{ER}_{\max} = C^{JPPS}_{\max}$. So for this case the maximum slack is given by theorem 1.

Case 2 We now consider instances for which $C^{PR}_{\max} < C^{ER}_{\max}$. Among these instances we can build what we call the rake instance $\mathcal{R}$, build from $m+1$ jobs with unit processing times. for this instance $C^{PR}_{\max}(\mathcal{R}) = \frac{m+1}{m}$ whereas $C^{ER}_{\max}(\mathcal{R}) = 2$ so that the difference is $\frac{m-1}{m}$. And finally in both cases by analyzing the relation to $C^{CJ}_{\max}$ and the presence of crossing jobs, we get the maximal slack :

Theorem 2.

$$\left|C^{PR}_{\max} - C^{ER}_{\max}\right| \leq \frac{m-1}{m} p_{\max} \quad and \quad C^{ER}_{\max} - C^{JPPS}_{\max} \leq \frac{m-1}{m} p_{\max} \tag{1}$$

5 Slack between pseudo-preemptive lower bound and Jackson preemptive upper bound

Sanlaville , 1995 proved that the Jackson preemptive schedule (which gives a suboptimal preemptive schedule) achieves a slack with the optimal schedule not larger than $\frac{m-1}{m} p_{max}$. In this section we give a new proof of this result by comparing the makespan of the Jackson preemptive schedule to the pseudo-preemptive bound.

Theorem 3.

$$C^{JPS}_{\max} - C^{JPPS}_{\max} \leq \frac{m-1}{m} p_{max}$$

We prove that this slack is asymptotically tight using a special purpose Bandoneon instance.

6 Conclusion

Using modeling, algebra, algorithms and scheduling theory, we established that the main lower bounds for our problem are very close to each other even in the worst case. This theoretically confirms previous experiments. Moreover, as in Carlier et al. , 2020, the bounds on slacks can be extended to the cumulative scheduling problem (CuSP). So, when using such bounds in branch and bound methods, it seems reasonable to use the one with the least algorithmic complexity.

References

Baptiste, P., Le Pape, C. & Nuijten, W. , 1999, "Satisfiability tests and time - bound adjustments for cumulative scheduling problems", *Annals of Operations Research* **92**(0),pp. 305–333.

Carlier, J. , 1987, "Scheduling jobs with release dates and tails on identical machines to minimize the makespan", *European Journal of Operational Research* **29**,pp. 298–306.

Carlier, J. & Pinson, E. , 1998, "Jackson's Pseudo Preemptive Schedule for the $Pm|r_i, q_i|C_{\max}$ scheduling problem", *Ann. Oper. Res.* **83**,pp. 41–58.

Carlier, J. & Néron, E. , 2003, "On linear lower bounds for the resource constrained project scheduling problem", *Eur. J. Oper. Res.* **149**(2),pp. 314–324.

Carlier, J., Pinson, E., Sahli, A. & Jouglet, A. , 2020, "Comparison of three classical lower bounds for the Cumulative Scheduling Problem", *(submitted)* .

Carlier, J., Sahli, A., Jouglet, A. & Pinson, E. , 2021, "A faster checker of the energetic reasoning for the cumulative scheduling problem", *International Journal of Production Research* **0**(0),pp. 1–16.

Erschler, J. & Lopez, P. , 1990, Energy-based approach for task scheduling under time and resources constraints, *in* '2nd International Workshop on Project Management and Scheduling, Compiègne (France)', pp. 115–121.

Gale, D. , 1957, "A theorem on flows in networks.", *Pacific Journal of Mathematics* **7**(2),pp. 1073 – 1082.

Hanen, C., Kordon, A. M. & Pedersen, T. , 2021, "Two deadline reduction algorithms for scheduling dependent tasks on parallel processors", *in* P. J. Stuckey, ed., 'CPAIOR 2021,', Vol. 12735 of *Lecture Notes in Computer Science*, Springer, pp. 214–230.

Haouari, M., Kooli, A. & Néron, E. , 2012, "Enhanced energetic reasoning-based lower bounds for the resource constrained project scheduling problem", *Comput. Oper. Res.* **39**(5),pp. 1187–1194.

Haouari, M., Kooli, A., Néron, E. & Carlier, J. , 2014, "A preemptive bound for the resource constrained project scheduling problem", *Journal of Scheduling* **17**(3),pp. 237–248.

Hidri, L., Gharbi, A. & Haouari, M. , 2008, "Energetic reasoning revisited: application to parallel machine scheduling", *Journal of scheduling* **11**(4),pp. 239–252. Publisher: Springer.

Jackson, J. R. , 1955, "Scheduling a production line to minimize maximum tardiness", *management science research project* . Publisher: University of California.

Laborie, P. & Nuijten, W. , 2008, "Constraint programming formulations and propagation algorithms", *in* 'Resource Constrained Project Scheduling', John Wiley & Sons, Ltd, pp. 63–72.

Néron, E. , 2008, "Resource and precedence constraint relaxation", *in* 'Resource Constrained Project Scheduling', John Wiley & Sons, Ltd, pp. 37–48.

Ouellet, Y. & Quimper, C.-G. , 2018, " A $\mathcal{O}(n \log^2 n)$ checker and $\mathcal{O}(n^2 \log n)$ filtering algorithm for the energetic reasoning", *in* W. J. v. Hoeve, ed., "CPAIOR 2018', Vol. 10848 of *Lecture Notes in Computer Science*, Springer, pp. 477–494.

Sanlaville, E. , 1995, 'Nearly on line scheduling of preemptive independent tasks', *Discret. Appl. Math.* **57**(2-3), 229–241.

Tercinet, F., Lenté, C. & Néron, E. , 2004, "Mixed satisfiability tests for multiprocessor scheduling with release dates and deadlines", *Oper. Res. Lett.* **32**(4),pp. 326–330.

Comparative Study of Two Machine Learning Tasks in Project Scheduling

Weikang Guo[1], Mario Vanhoucke[1,2,3] and José Coelho[1,4]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
weikang.guo@ugent.be, mario.vanhoucke@ugent.be, jose.coelho@uab.pt
[2] Technology and Operations Management, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, United Kingdom
[4] INESC - Technology and Science, Porto (Portugal) and Universidade Aberta, Portugal

Keywords: project scheduling, machine learning, performance prediction.

1 Introduction

Machine learning (ML) plays a major role in many scientific applications and has become one of the most promising growing fields of research. The major ML approaches can be classified into two main categories: *supervised learning* (Parvin *et. al.* 2013) and *unsupervised learning* (Minaei-Bidgoli *et. al.* 2014). Multi-label classification and label ranking are two important tasks in supervised learning. This research aims to provide a comparative study of these two tasks for resource-constrained project scheduling.

The *resource-constrained project scheduling problem* (RCPSP) has received widespread attention from researchers and has become one of the most popular problems to be solved in the project management and scheduling literature. The basic version of the RCPSP assumes that a single project consisting of a number of activities should be scheduled in order to finish the project as soon as possible. The activities are interrelated by finish-to-start precedence relations with a minimal time-lag of zero, and cannot be preempted. Moreover, each activity makes use of a set of renewable resources, for which their availabilities are limited. A schedule is said to be feasible if it respects the precedence constraints and the resource constraints, and is optimal if no other feasible schedule exists with a lower project duration (known as the project makespan).

As a generalization of the job-shop scheduling problem, the RCPSP is known to be NP-hard in the strong sense (Blazewicz *et. al.* 1983). Therefore, the branch-and-bound (B&B) technique is the most common way to deal with this problem to get optimal solutions (Brucker *et. al.* 1998). Some branch-and-bound algorithms can be found in Stinson *et. al.* (1978). Coelho and Vanhoucke (2018) developed a branch-and-bound procedure that takes into account all best performing components from literature. This so-called *composite branch-and-bound procedure* (CBP) is used as the basis of the current study. It makes use of 48 different configurations which are combinations of various branching schemes, search strategies, lower bounds, and branching orders.

In the current study, 24 configurations are selected and each project instance is solved by each of these configurations truncated after 1 minute. Most configurations could not find the optimal solutions within such a short time and therefore the goal of two machine learning tasks is to obtain the best performing configuration for a project based on the characteristics of the project instance (i.e. project indicators). Since each project instance may have more than one configuration that generates the best possible makespan, selecting the best configuration can be treated as a *multi-label learning* problem. In this work, multi-label classification and label ranking tasks will be introduced to automatically classify or rank the performance of 24 configurations, and then select the best performing configuration to solve the project instance to optimality (a near optimality if truncated early). The purpose of *multi-label classification* is to find the mapping between the project indicators and the performance of the 24 configurations of the CBP in order to predict

which configurations can produce the best solution for the project instance. The goal of *label ranking* is to learn a mapping between project indicators and a full ranking list of the 24 configurations of the CBP to provide preference information on the quality of these 24 configurations for the project instance.

This is, to the best of our knowledge, the first research study to investigate the relation between project indicators and the full ranking list of the performance of various configurations of different branch-and-bound procedures using a prediction model. We will compare the performance of the two different learning tasks (multi-label classification and label ranking) on a test set of benchmark instances, and their performance will be compared with the performance of the individual configurations of the CBP.

2 Problem description

The resource-constrained project scheduling problem (RCPSP) can be formulated as follows: A set of activities N, numbered from a dummy start node 0 to a dummy end node $n + 1$, is to be scheduled without pre-emption on a set R of renewable resources types. Each renewable resource $k \in R$ has a constant availability a_k per period. Each nondummy activity $i \in N$ has a deterministic duration d_i and requires $r_{i,k}$ units of resource type $k \in R$. The start and end dummy activities 0 and $n + 1$ represent the start and completion of the project, which their duration and renewable resource requirement equal to zero. A project network is represented by a topological ordered activity-on-the-node (AoN) format where A is the set of pairs of activities between which a finish-start precedence relationship with time lag 0 exists. We assume graph $G(N, A)$ to be acyclic. A schedule S is defined by a vector of activity start times and is said to be feasible if all precedence and renewable resource constraints are satisfied. The objective of the problem type is to find a feasible schedule within the lowest possible project makespan, and hence, the problem type can be represented as $m, 1T|cpm|C_{max}$ using the classification scheme of Herroelen *et. al.* (1999) or as $PS|prec|C_{max}$ following the classification scheme of Brucker *et. al.* (1999).

Project indicators: As mentioned in the introduction, our multi-label classification and ranking tasks aim to find the mapping between the project indicators and the performance of 24 configurations of the CBP. To that purpose, the most well-known project indicators are used in our study that describe the activity, the network, and the resource characteristics of a project instance. In total, 12 project indicators are used, classified as *activity indicators* (Number of activities), *network indicators* (Coefficient of Network Complexity (CNC), Order Strength (OS), Serial/Parallel (SP), Activity Distribution (AD), Length of Arcs (LA), Topological Float (TF), and Length of long arcs (I_5)) and *resource indicators* (Resource Factor (RF), Resource Use (RU), Resource Strength (RS), Resource Constrainedness (RC)).

Various configurations: Coelho and Vanhoucke (2018) have shown that their *composite lower bound strategy* not only improves the current solutions of individual configurations, but also generates new best-known solutions for existing datasets. This branch-and-bound procedure is, to the best of our knowledge, the most complete procedure that integrates the best performing components from the literature, which is why we have used it in the current study. The main components of the 24 configurations we selected are given below.

- Search strategy: Either the minimum lower bound strategy (LBS) or the upper bound strategy (UBS) is used.
- Branching scheme: The branching scheme consists of the activity start branching (AST), the parallel branching (PAR), and the serial branching (SER).
- Branching order: The selection of a node at each level of the tree can be done using the activity ID (AID) or the best lower bound (BLB).
- Composite lower bound (CLB): Four lower bounds are used in the algorithm, and they are integrated in a composite way as described in the original paper as CLB0,

CLB4, CLB8, and CLB12. CLB0 only calculates the critical path based lower bound (LB_{cp}), CLB4 is defined as the CLB0 plus the critical capacity lower bound (LB_{cc}), the critical sequence lower bound (LB_{cs}), the incompatible pairs lower bound (LB_{ip_0}) and the basic version of the node packing lower bound (LB_{np_0}). CLB8 is defined as the CLB4 plus two extensions of the node packing lower bound (LB_{np_2} and LB_{np_1}), one parallel machine based lower bound, and the incompatible triplets lower bound. CLB12 is defined as the CLB8 plus an extended version of the machine lower bound, the lower bound found by *reduction by precedence*, the lower bound found by *reduction by core times*, and the lower bound found by *reduction by time periods*.

3 Machine learning framework

The machine learning framework consists of three different steps, which are briefly outlined along the following lines.

Step 1. Data Split. In the first step, the dataset is split into a training set, a validation set, and a testing set. The training set is used in the learning phase to build a relation between the project indicators (inputs, features) and the performance of different configurations (outputs, labels). For the multi-label classification task, the classification model will predict whether a configuration can generate the best possible solution among the set of 24 configurations. Specifically, in case a specific configuration generates the best possible makespan, the configuration is set to 1 (positive), and 0 (negative) otherwise. For the multi-label ranking task, the ranking model will predict a full ranking list of 24 configurations based on the makespan values of the instance in the dataset. Both the classification model and the ranking model are built to learn the relationship between the 12 project indicators and these labelled configurations. In the current study, decision tree multi-label classification models proposed by Guo *et. al.* (2021) are used and compared with one ranking model.

Step 2. Hyper-parameter optimization. In this step, the validation set is used to find the optimal parameter settings for the classification and ranking models. More specifically, the classification and ranking models are fine-tuned by setting the parameters to different values to obtain their best possible values.

Step 3. Performance evaluation. Once the training and validation steps are finished, the parameters that yield the best possible values for classification and ranking models are selected, and both types of models are now retrained on the whole set (including the training set and the validation set). After that, the learned relationship is applied to the testing set to predict the final performance of the classification and ranking models. For the testing set, the project indicators for each project instance are calculated and entered in the classification and ranking models to select one or more configurations to solve the instances in this set. For the classification task, it is possible that more than one configuration can be predicted to be positive, in which case all these positive labels are used to solve the instance, after which the best makespan is reported as the best possible makespan and the algorithm stops. For the ranking task, the predictions generated by the classification model are the full ranking list of 24 configurations, and so only the top x configurations ($x \leq 10$) in the prediction are used to solve the instances of the test set.

4 Preliminary results

The aim of the computational experiments is twofold. First, the performance of the two learning tasks for predicting the configurations of the branch-and-bound procedure will be compared within a time limit of 60 seconds. Second, the quality of the obtained solutions generated by the classification and ranking models will be compared with the solution of the individual configurations, both for the best found lower bounds and upper bounds. These results consist of the best performing single configuration, as well as the best solution found over all 24 configurations (indicated in Table 1 as OPT). It should be

noted that the classification and ranking methods can never find a solution better than OPT, but should at least perform better than any individual configuration.

Table 1 shows the results for the upper bounds (UB) and lower bounds (LB) expressed as the sum of the makespan over all instances in the test set. The table shows that the classification models and the ranking model outperform the single best configuration, and the solutions of the ranking model lie closer to the optimal case (OPT) than the solutions for the classification models. These (limited) results illustrate the power of machine learning for selecting best-performing components of an algorithm, and in the workshop presentation, more detailed comparisons and results will be given.

Table 1. Performance comparison of various methods.

Various Methods	UB	Various Methods	LB
OPT	37,643	OPT	25,818
Label ranking	**37,681**	Label ranking	**25,816**
Multi-label classification	40,173	Multi-label classification	25,812
The single best configuration (C22[1])	41,118	The single best configuration (C23[2])	25,773

[1] C22 indicates the configuration which is a combination of UBS, PAR, BLB and CLB12.
[2] C23 represents the configuration which is a combination of LBS, PAR, BLB and CLB12.

5 Conclusions

This work compares the performance of two types of machine learning tasks for solving the RCPSP, which aim at mapping known project indicators with the performance of various configurations of a composite B&B procedure from the academic literature. The computational experiments show that both machine learning tasks outperform any single-best configuration, and often come close to the optimal case. The future of this research should focus on extending these machine learning tasks to solve other scheduling problems, such as the *resource-constrained multi-project scheduling problem* (RCMPSP) and *multi-mode resource-constrained project scheduling problem* (MRCPSP).

References

Blazewicz, J., J. Lenstra and A. Kan, 1983, "Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics", *Discrete Applied Mathematics*, Vol. 5(1), pp. 11-24.

Brucker, P., S. Knust, A. Schoo and O. Thiele, 1998, "A branch and bound algorithm for the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 107(2), pp. 272-288.

Brucker, P., A. Drexl, R. Möhring and K. Neumann, 1999, "Resource-constrained project scheduling: Notation, classification, models, and methods", *European journal of operational research*, Vol. 112(1), pp. 3-41.

Coelho, J., M. Vanhoucke, 2018, "An exact composite lower bound strategy for the resource-constrained project scheduling problem", *Computers & Operations Research*, Vol. 93, pp. 135-150.

Guo, W., M. Vanhoucke and J. Coelho, 2021, "Automatic detection of the best performing priority rule for the resource-constrained project scheduling problem", *Expert Systems with Applications*, Vol. 167, pp. 114116.

Herroelen, W., E. Demeulemeester and B. De Reyck, 1999, "A classification scheme for project scheduling", *In Project scheduling*, pp. 1-26.

Minaei-Bidgoli, B., H. Parvin, H. Alinejad-Rokny, H. Alizadeh, and W. Punch, 2014, "Effects of resampling method and adaptation on clustering ensemble efficacy", *Artificial Intelligence Review*, Vol. 41(1), pp. 27-48.

Parvin, H., H. Alinejad-Rokny, B. Minaei-Bidgoli and S. Parvin, 2013, "A new classifier ensemble methodology based on subspace learning", *Journal of Experimental & Theoretical Artificial Intelligence*, Vol. 25(2), pp. 227-250.

Stinson, J., E. Davis and B. Khumawala, 1978, "Multiple Resource Constrained Scheduling Using Branch and Bound", *A I I E Transactions*, Vol. 10(3), pp. 252-259.

A Method to Find Criticalities in Project Networks with Feeding Precedence Relations

Lucio Bianco[1], Massimiliano Caramia[1], Stefano Giordani[1], and Alessio Salvatore[2]

[1]University of Rome "Tor Vergata" and [2]National Research Council - IAC, Italy
bianco,caramia,giordani@dii.uniroma2.it

Keywords: Feeding Precedence Relations, Critical Activities, Makespan.

1 Background

This paper deals with project networks with feeding precedence relations. Feeding precedences were firstly introduced by Kis *et al.* (2004) and extend the classical precedence relations of a Resource-Constrained Project Scheduling Problem (RCPSP). These kinds of precedences occur in all of those applications in which it is not possible to calculate the exact durations of the activities like in the case of production planning environment, e.g., make-to-order manufacturing, which commonly requires the so-called project-oriented approach. Under this viewpoint, a project consists of activities each one representing a manufacturing process and the effort associated with their execution varies over time. When it comes to this kind of problems, the literature uses variable activity execution intensity models (see, e.g., Kis (2005)). As the durations of the activities cannot be taken into account, the traditional finish-to-start precedence relations, as well as the Generalized Precedence Relations (GPRs, see, e.g., Bartusch *et al.* (1988), Elmaghraby and Kamburowski (1992)), are not in charge of giving a direct realist mapping of the problem into constraints as feeding precedence relations are able to do (see, Kis *et al.* (2004), Kis (2006)). Feeding precedence relations dealt with in this paper are inspired by previous work of Alfieri *et al.* (2011), Bianco and Caramia (2011), and Bianco and Caramia (2012). They are of four types:

- Start-to-%Completed ($S\%C(g_{ij})$) constraint between two activities (i, j). This constraint imposes that the processed fraction of successor activity j of i can be greater than $0 \leq g_{ij} < 1$ only if the execution of i has already started.
- %Completed-to-Start ($\%C(q_{ij})S$) constraint between two activities (i, j). This constraint impose that successor activity j of i can be started only if i has been processed for at least a fractional amount $0 < q_{ij} \leq 1$.
- Finish-to-%Completed ($F\%C(g_{ij})$) constraint between two activities (i, j). This constraint imposes that the processed fraction of successor activity j of i can be greater than $0 \leq g_{ij} < 1$ only if the execution of i has already completed.
- %Completed-to-Finish ($\%C(q_{ij})F$) constraint between two activities (i, j). This constraint imposes that successor activity j of i can be completed only if i has been processed for at least a fractional amount $0 < q_{ij} \leq 1$.

2 Problem definition

In our problem, we are given a (directed acyclic) graph $G = (V, A)$ representing a project network where $V = \{1, \ldots, n\}$ is the set of n activities to be carried out without preemption and A is the set of arcs modeling the precedence relations between (ordered) pairs of activities. An example is shown in Figure 1: for example, the feeding precedence relation related to arc $(1, 2)$ states that the processed percentage of activity 2 can be greater than 60% only if activity 1 has already started. More in general, numbers in parentheses are the fractions of execution associated with each constraint.

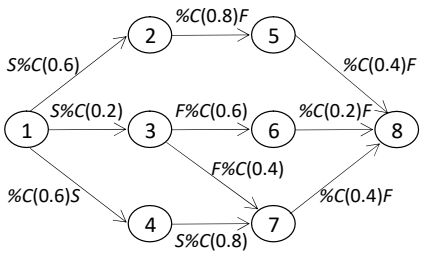
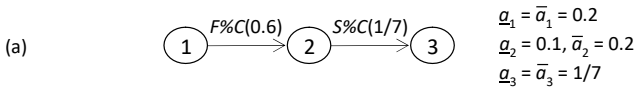
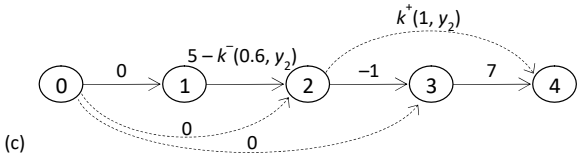


Fig. 1. Example of a project network with feeding precedence relations

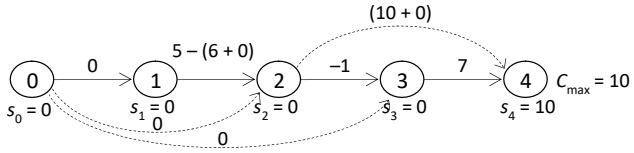




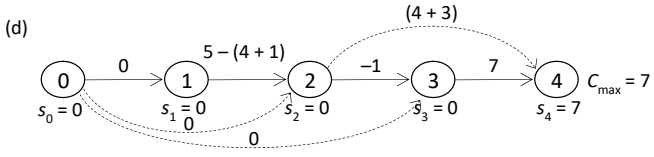
$k^-(0.6, y_2) = \lfloor \min\{0.6, y_2\}/\underline{a}_2 \rfloor + \lfloor (0.6 - \min\{0.6, y_2\})/\overline{a}_2 \rfloor$

$k^+(1, y_2) = \lceil \min\{1, y_2\}/\underline{a}_2 \rceil + \lceil (1 - \min\{1, y_2\})/\overline{a}_2 \rceil = \lceil y_2/\underline{a}_2 \rceil + \lceil (1 - y_2)/\overline{a}_2 \rceil$

1) $y_2 = 1$ (activity 2 is executed at its minimum speed $\underline{a}_2 = 0.1$)



2) $y^*_2 = 0.4$ (activity 2 is firstly executed at its min. speed $\underline{a}_2 = 0.1$ for 40% of its work and then at its max. speed $\overline{a}_2 = 0.2$ for the remaining 60% of its work)



3) $y_2 = 0$ (activity 2 is executed at its maximum speed $\overline{a}_2 = 0.2$)

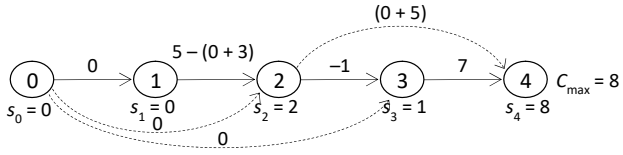


Fig. 2. The effect of executing an activity with different min-max speed execution profile

Our focus is on finding the start time s_i and the execution intensity profiles π_i (i.e., the fractions of work done in the time slots of the planning horizon) of each activity $i \in V$ that warrant the minimum completion time (or duration) of the project, i.e., the minimum project makespan $C^*_{\max}$. In our problem definition, we have two assigned parameters, say $\underline{a}_i$ and $\bar{a}_i$, with $0 \leq \underline{a}_i \leq \bar{a}_i \leq 1$, associated with each activity $i \in V$, which represent its minimum and maximum execution intensities (speeds), respectively, in each time slot in which the activity is executed. That is, activity i can be executed with variable execution intensity (speed) ranging in $[\underline{a}_i, \bar{a}_i]$, during its execution period (with the exception of its last execution time slot if the remaining work to be done is less than $\underline{a}_i$). The number of execution time slots of activity i represents its duration.

We note that finding the optimal solution of our problem is not straightforward, since adopting the solution where all the activities are executed at their maximum speed does not lead in general to a minimum project duration. An example of this occurrence is shown in Figure 2d, where three distinct solutions, corresponding to different execution profiles for activity 2, are represented for the project network of Figure 2a. In particular, Figure 2d shows that executing activity 2 at its maximum speed does not assure the minimization of the project makespan. Solution number 2) is the optimal solution (minimizing the activity start time and the project makespan) as it will be clear next. Notwithstanding the aim of minimizing the project makespan, in this paper, we pose the primary goal of finding a specific network representation of the problem showing which activities may be defined as *critical*. To the best of our knowledge this task has not been carried out on project networks with feeding precedence relations. In fact, the literature has plenty of results on the criticalities in project networks with finish-to-start constraints with zero time lags and with the more general case of generalized precedence relationships (see, e.g., Bianco *et al.* (2021)), but no attempt has been made to define a method to find criticalities in the problem setting considered in this paper.

3 The proposed method

Given the problem definition reported in the previous section, and the consideration made therein, the proposed method is based on the following theorems for which, for space reasons, we omit their proofs.

Let us denote with $\pi_i^{(\underline{a}_i, \bar{a}_i)}(y_i)$ the *min-max speed execution profile* for activity i where the initial fraction y_i (with $0 \leq y_i \leq 1$) of its work is done at its minimum speed $\underline{a}_i$ and the remaining fraction $(1 - y_i)$ at its maximum speed $\bar{a}_i$.

Theorem 1. *Given any feasible execution profile π_h for each one of the activities h preceding activity i, there exists a min-max speed execution profile $\pi_i^{(\underline{a}_i, \bar{a}_i)}(y_i)$ for activity i that allows i to start at its earliest possible time, with respect to the given execution profiles of its predecessors.*

In particular, let y_i^* be the minimum value among the initial fractions y_i that fulfills Theorem 1.

Theorem 2. *Executing each activity i with min-max speed execution profile $\pi_i^{(\underline{a}_i, \bar{a}_i)}(y_i^*)$ allows all the activities to start at their earliest start time and to minimize the project makespan.*

Given this result, we are able to transform the original project network $G = (V, A)$ into a standardized project network $N = (V', A')$, with $V' = V \cup \{0, n+1\}$, where 0

and $n+1$ are the initial and final dummy nodes (activities), and $A \subset A'$, and with start-to-start $SS_{ij}^{\min}(\ell_{ij}(y_i, y_j))$ precedence relations with minimum time lag $\ell_{ij}(y_i, y_j)$. So doing, we generalize the Bartusch *et al.*'s transformation for standardizing GPRs project network. Figures 2(c) shows the standardization of project network of Figure 2(a) (or of the equivalent project network of Figure 2(b) that includes the initial and final dummy activities). The standardized network of Solution 2) shown in Figure 2d, with the optimal values y_i^* according to Theorem 2, is the optimal standardized network minimizing the activity earliest start times. Notwithstanding, we are able to prove the following theorem that allows us to determine the optimal values y_i^* in linear time with $|A|$.

Theorem 3. *The earliest start time of each activity $i \in V$ can be determined on the standardized network N by means of a forward recursion, where the optimal value of variable y_i^* is determined as the minimum value of y_i that minimizes the length of the longest path from dummy initial node 0 to node i. All the time lags encompassing y_i can be determined just by substituting y_i^* found so far without affecting the optimality of the start times and of the project makespan.*

4 Ongoing and future work

The algorithm to find the critical path and the critical activities at the moment is able to determine the earliest start time schedule. We are working on the construction of the latest start schedule to analyze whether activities on a critical path are characterized by having the same earliest and latest start time. Moreover, we are doing the same for the earliest and latest finish schedules to analyze if a difference between such values may occur or not for the activities on the critical paths. Our conjecture, at the moment, is that the criticality of an activity does not necessary imply that its total floats, computed on its earliest and latest start times, and on its earliest and latest finish times, respectively, are both equal to zero, but that this occurs at least for one of them.

References

Alfieri A., Tolio T., Urgo M., 2011, "A project scheduling approach to production planning with feeding precedence relations", *International Journal of Production Research*, Vol. 49(4), pp. 995–1020.

Bartusch M., Möhring R.H., Radermacher F.J., 1988, "Scheduling project networks with resource constraints and time windows", *Annals of Operations Research*, Vol. 16, pp. 201–240.

Bianco L., Caramia M., 2011, "Minimizing the completion time of a project under resource constraints and feeding precedence relations: a Lagrangian relaxation based lower bound", *4OR*, Vol. 9(4), pp. 371–389.

Bianco L., Caramia M., 2012, "Minimizing the completion time of a project under resource constraints and feeding precedence relations: an Exact Algorithm", *4OR*, Vol. 9(4), pp. 361–377.

Bianco L., Caramia M., Giordani S., 2021, "Project scheduling with generalized precedence relations: A new method to analyze criticalities and flexibilities", *European Journal of Operational Research*, available online, doi: 10.1016/j.ejor.2021.07.022.

Elmaghraby S.E., Kamburowski J., 1992, "The analysis of activity networks under generalized precedence relations (GPRs)", *Management Science*, Vol. 38(9), pp. 1245–1263.

Kis T., Erdös G., Márkus A., Váncza J., 2004, "A project-oriented decision support system for production planning in make-to-order manufacturing", *ERCIM News*, Vol. 58, pp. 66–67.

Kis T., 2005, "A branch-and-cut algorithm for scheduling of projects with variable-intensity activities", *Mathematical Programming*, Vol. 103(3), pp. 515–539.

Kis T., 2006, "RCPS with variable intensity activities and feeding precedence constraints". In: Józefowska J., Węglarz J. (eds.) *Perspectives in Modern Project Scheduling*, Int. Series in Oper. Res. & Manag. Sci., vol 92. Springer, Boston, MA, pp. 105–129.

A branch and bound approach for stochastic 2-machine flow shop scheduling with rework

Lei Liu[1] and Marcello Urgo[1]

Politecnico di Milano, Mechanical Dept., Italy
lei.liu, marcello.urgo@polimi.it

Keywords: Flow shop, Rework, Markovian chain, Branch and bound

1 Introduction and problem statement

Turbine blades are one of the most expensive components in gas turbines for power generation, due to materials used and the complex manufacturing process. For this reason, their re-manufacturing is an economically viable approach to obtain refurbished parts for the maintenance of gas turbines. Re-manufacturing processes, differently from production of new parts, are characterized by a considerable degree of uncertainty. With respect to turbine blades, the repair process entails the removal of the hard coating and the damaged parts, the addition of the missing material through an additive manufacturing processes and their grinding. Hence, an additional material removal phase is required by means of electrical discharge technologies, to obtain the final desired shape. Within the described process, two of the most relevant re-manufacturing activities are the addition of materials through a welding process and the following grinding process. Moreover, blades quite always need to be reworked by the repetition of the same sequence of operations, thus competing for the same resources. Blades are processed in batches, consisting of a set of blades of the same stage of the turbine. The number of blades in each batch is not known in advance. In fact, some of the blades in the batch could be too damaged to be repaired and must be substituted with new ones. The processing times for each batch of blades in the different phases, included the rework ones, also entails a certain degree of uncertainty. Blades with a higher degree of damages requires longer processing times respect to less severe damages. The uncertainty associated to these factors is embedded in the processing times associated to batches of blades, described through a probability distribution.

In this paper, we focus on the scheduling of the two re-manufacturing phases described above, i.e., welding and grinding, modeling the process through a stochastic 2-machine permutation flow shop scheduling problem with rework. A set of jobs N, representing batches of blades, must be processed on two machines, M_1 and M_2 in sequence. The sequence of the jobs on the two machines is the same. After their processing on the second machine, jobs will need a rework cycle on both M_1 and M_2 (Fig 1). Rework jobs are grouped in an additional set N'. The processing time of a job $j \in N \cup N'$ on machine M_i, denoted as f_{ij}, is modeled as an independent random variable following a general distribution.

After the first processing, blades undergo an inspection to determine the parameters of the rework process. The inspection is operated offline respect to the flow shop. For this reason, in order to consider the time needed for this phase, we state that rework jobs can be processed not earlier than 2 jobs after the corresponding original job, unless the jobs to be processed are less than 2. To provide an example, let us consider a schedule referring to 4 jobs $[a, b, c, d]$ and their corresponding rework jobs $[a', b', c', d']$. Thus, a full repair schedule can only be one of the following: $[a, b, c, a', d, b', c', d']$, $[a, b, c, d, a', b', c', d']$ and $[a, b, c, a', b', d, c', d']$.

The objective function considered is the minimization of the Value-at-Risk (VaR) (Urgo, M. and Vancza, J. 2019) of the makespan, with the aim to provide a robust solution. To

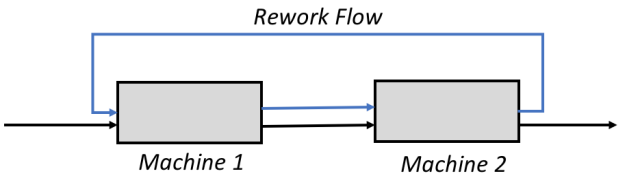


Fig. 1. Manufacturing environment

address this scheduling problem, we propose a branch and bound approach. The processing of the jobs is modeled through a Markov chain (Kulkarni, V.G. and Adlakha, V.G. 1986), whose time to absorption correspond to the makespan of the schedule, enabling the calculation of the VaR. To extend the approach beyond exponential processing times, phase-type distributions are used, due to their capability to approximate general distributions (Bladt, M. 2005).

2 Branch and bound algorithm

The branching scheme is aimed at the definition of a full schedule, containing both original and rework jobs. A forward branching scheme is used, sequencing the jobs starting from the beginning of the schedule. Due to the need to respect the constraints affecting the sequencing of rework jobs, nodes in the branching tree which are in conflict with these constraints are pruned before being evaluated.

A heuristic rule (Baker, K.R. and Trietsch, D. 2011) is exploited to obtain an initial upper bound for the search. This schedule is obtained by arranging all the jobs according to the decreasing order of $(1/E(j_1) - 1/E(j_2))$ with $E(j_1)$ and $E(j_2)$ being the expected value of the processing times of job j on machines 1 and 2 respectively. If the resulting schedule is in conflict with the constraints affecting the sequencing of rework jobs, they are shifted towards the right until the conflicts are eliminated.

To illustrate the approach for the calculation of the VaR of a schedule, let us consider an Activity on Arc (AoA) network of activities modeled as an acyclic directed graph $G = (V, A)$. Each arc in G represents an activity while the nodes in V represents states. At a given time t, an activity can only be active, dormant or idle (Kulkarni, V.G. and Adlakha, V.G. 1986). If we consider the schedule $[a, b, c, a', d, b', c', d']$, corresponding AoA network is reported in Fig.2. Hence, the set of states modeling the execution of the network, constituting the support of the Continuous Time Markov Chain (CTMC), can be obtained (Fig. 3).

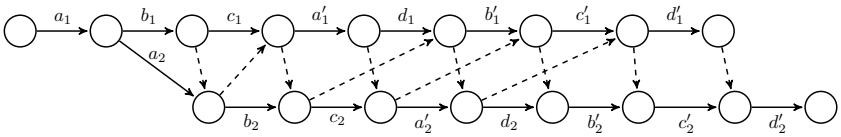


Fig. 2. AoA activity network for a two-machine flow shop with rework

Starting from this, the set of states is further enriched to consider phase-type distributions. In fact, these distribution can be in turn defined through a CTMC. Thus, each of the states in Fig. 3 represents a set of states defined by the phase type modeling the processing times of the different activities. The infinitesimal generator of the extended CTMC can be obtained, starting from the one associated to the states in Fig. 3, using a Kronecker alge-

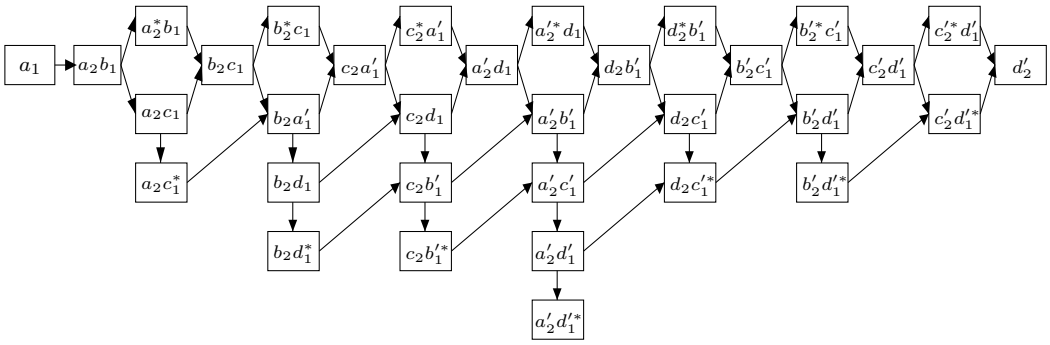


Fig. 3. States generation scheme

bra approach (Angius, A. *et. al.* 2021). The distribution of the time to absorption of the described CTMC and the quantile corresponding to the VaR_α can be obtained according to Eq. 1,

$$F(t) = 1 - \beta e^{Tt}\mathbf{1}, \quad \alpha = 1 - \beta e^{T*VaR}\mathbf{1} \tag{1}$$

The described approach applies to full schedules and can support the analysis of leaf nodes in the search tree. For the evaluation of the other nodes, let us consider a partial schedule with s jobs already sequenced. For these jobs, an approach similar to the one described for leaf nodes can be used, constituting the leftmost part of the network of activities in Fig. 2. On the contrary, for the remaining $n - s$ jobs, their processing times on the two machines can be modeled through two activities (X_1 and X_2) whose processing times are the sum of the ones of the original jobs. The AoA activity network for a partial schedule $[a, b, c, a', \ldots]$ is represented in Fig. 4. Grounding on this network, precedence constraints among operations on M_1 and M_2 for unsequenced jobs are relaxed. Since the VaR of the makespan is a regular objective function, relaxing constraints will provide a lower bound for the VaR of the complete schedules derived from the partial one considered. Thus, the approach described above can be used to generate the associated CTMC and estimate the lower bound of the VaR (Urgo, M. and Vancza, J. 2019).

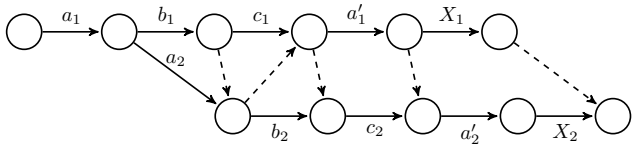


Fig. 4. AoA activity network for a partial schedule

3 Numerical experiments

A set of test instances has been generated considering $n = 6$ jobs, thus a total of 12 jobs including the rework ones. Processing times are modeled through phase-type distributions randomly generated by the BuTools library (Horvath, G. and Telek, M. 2016) by providing values for the mean and number of phases. The value of the mean is randomly sampled from three different uniform distributions with support [0,20],[30,50] and [60,80]. The number of phases is randomly sampled between 1 and 4. Different risk levels α (10 and 20%) are

considered for the optimization. The results of the experiments are reported in Table 1 and Fig.5, showing the performance of the branch-and-bound algorithm in terms of solution time, number of evaluated nodes and average evaluation time per node.

Table 1. Results

Job No.	Risk level (%)	Solution time(s)				Evaluated nodes			
		Mean	Min	Max	SD	Mean	Min	Max	SD
6	10	877.6	110.8	1383.8	404.1	12262	1627	22433	6569
	20	900.9	143.6	2009.3	563.2	11442	2665	36028	9824
ALL		889.8	110.8	2009.1	480.9	11830	1627	36028	8223

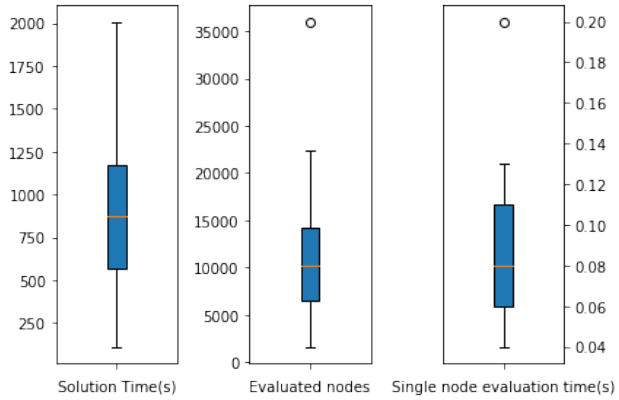


Fig. 5. Performance of algorithm

Grounding on the experiments, the proposed algorithm is able to find the optimal schedule in about 20 minutes, with 11830 nodes evaluated on average, and the average single node evaluation time is about 0.08 seconds. According to these results, the time to solve larger instance is likely to be rather large, future works will address the tighter lower bounds and effective job insertion dominance rules. Nevertheless, in the considered industrial environment, the number of jobs to schedule is in line with the one considered in the computational experiments, thus the proposed approach is valuable for the company.

References

Angius, A., Horvath, A. and Urgo, M., 2021, " A Kronecker Algebra Formulation for Markov Activity Networks with Phase-Type Distributions", *Mathematics*, 9(12), p.1404.

Baker, K.R. and Trietsch, D., 2011. Three heuristic procedures for the stochastic, two-machine flow shop problem. Journal of Scheduling, 14(5), pp.445-454.

Bladt, M., 2005. A review on phase-type distributions and their use in risk theory. ASTIN Bulletin: The Journal of the IAA, 35(1), pp.145-161.

Horvath, G. and Telek, M., 2016. BuTools 2: a Rich Toolbox for Markovian Performance Evaluation. In VALUETOOLS.

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

Urgo, M. and Vancza, J., 2019. A branch-and-bound approach for the single machine maximum lateness stochastic scheduling problem to minimize the value-at-risk. Flexible Services and Manufacturing Journal, 31(2), pp.472-496.

Problem-specific Priority Rules for the Resource-Constrained Project Scheduling Problem with Alternative Subgraphs

Rojin Nekoueian[1], Tom Servranckx[1] and Mario Vanhoucke[1,2,3]

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

Keywords: Resource-Constrained Project Scheduling, Alternative Subgraphs, Priority Rules.

1 Introduction

The scheduling of activities subject to resource and precedence constraints in order to minimise the project makespan is known as the resource-constrained project scheduling problem (RCPSP) (Kolisch and Hartmann 1999). This problem is known to be NP-hard (Blazewicz *et. al.* 1983). Most research studies assume that the project structure is deterministic and completely known in advance. Based on discussions with practitioners, researchers observed that large and complex project structures might consist of alternative work packages rather than a deterministic project network. In case that we relax the assumption of a deterministic project structure, we consider a more general project scheduling problem that consists of alternative, substitutable ways to execute the project (Servranckx and Vanhoucke 2019). In this research, the RCPSP is therefore extended with alternatives in order to allow a flexible network topology. The RCPSP with alternative subgraphs (RCPSP-AS) is the problem of selecting one alternative for a work package, i.e. a subset of activities in the project, among a set of existing alternatives. This problem includes two specific subproblems. First, an alternative work package should be selected (i.e. selection subproblem) and then it should be scheduled considering constrained resources (i.e. scheduling subproblem) (Servranckx and Vanhoucke 2019). There are various problems considering alternatives with different therminology. Besides the RCPSP-AS, there exist other, similar scheduling problems that consider alternatives such as the RCPSP with alternative process plan for production scheduling (Capek *et. al.* 2012). However, these scheduling problems differ from the RCPSP-AS in terms of problem features and terminology.

Schedule generation schemes (SGS) are the core of most heuristic solution procedures for the RCPSP as they start from scratch and build a feasible schedule by the stepwise extension of a partial schedule (Kolisch and Hartmann 1999). To generate a single or multiple schedules, researchers develop heuristics such as priority rule (PR)-based scheduling. A PR defines the order of project activities to be employed in a SGS. PRs can be static or dynamic. A static PR creates an activity list before employing SGS and once the activity list is made it is saved and the activities' priority in the list will not change. A dynamic PR updates the priority of activities in the list each time an activity is scheduled. Therefore, the priority of the remaining activities will change each time an activity is scheduled, but without rescheduling the activities in the partial schedule. If a (static or dynamic) PR generates a prioritized activity list that results in a high-quality schedule (i.e. low project makespan), less effort will be needed for the improvement of this schedule. Therefore, academics constantly search for better PRs for various scheduling problems. There are plenty

of studies in the literature which develop new and more efficient PRs for specific problems. Kolisch and Hartman (1999) and Kolisch and Hartman (2006) investigate on PRs for the RCPSP. Cooper (1976) investigates two heuristic methods using a number of static and dynamic PRs for the RCPSP. Capacho et. al. (2009) assign tasks of the assebly line to workstations and examine two different PRs for selection and scheduling of the alternative subgraphs in assembly line balancing problem (ASALBP). In this paper, we are going to examine previously developed PRs and investigate PRs which are specifically practical for RCPSP-AS.

2 Problem Statement

We consider a directed acyclic graph that represents the activity-on-the-node (AoN) project network. The activities should be scheduled on a set of renewable resource types for which the per-period available amount of each resource type is a previously known and assumed constant over time. However, in RCPSP-AS, certain activities can be excluded from the schedule, while the project still delivers on the required outcomes. In this problem, the set of all activities consists of two mutually exclusive subsets of activities: the set of *fixed activities* that should always be executed in order to complete the project, and the set of *alternative activities* that are optional. Hence, the precedence and resource constraints of an alternative activity should only be satisfied when the activity is scheduled. The objective of RCPSP-AS is to select for each alternative subgraph exactly one alternative branch such that the makespan of the resulting project is minimised. There are several special types of activities that need to be identified in order to define alternative subgraphs. A *principal activity* causes the decision amongst different choices in the project structure. Therefore, this fixed or alternative activity should have at least two mutually exclusive direct, alternative successors of which at most one should be selected. An alternative activity that is the direct successor of a principal activity is a *branching activity*. A *terminal activity* terminates the decision amongst different choices in the project structure. An *alternative subgraph* is an induced subgraph consisting solely of alternative activities that originate from the same principal activity with the inclusion of the corresponding terminal activity. A subset of activities in the alternative subgraph that consists of the branching activity as well as all its transitive successors is called an *alternative branch*. An *alternative path* is a subset that consists of the set of fixed activities and the logical feasible set of selected alternative activities.

Servranckx and Vanhoucke (2019) have identified two relations between alternative subgraphs in the project network. A *nested* alternative subgraph is an alternative subgraph that exists in another alternative subgraph. It might be that an alternative for one work package is connected to an alternative of the same or a different work package. This implies that decisions made in different work packages with alternatives are interconnected. Projects that consist of this type of project structure are called *linked* projects.

3 Example

An example of a linked nested project is shown in Fig. 1. Activity 1 is a principal activity and activity 9 is a terminal activity for the first alternative subgraph. One alternative path to be mentioned is {1, 6, 7, 8, 9}. Considering the static PR named most immediate successors (Slowinski and Weglarz 1989) and the tie-breaker rule of minimum activity number, the list of prioritised activities for this example project is: {1, 2, 4, 3, 5, 6, 7, 8, 9}. According to this list of prioritized activities, branching activity 2 is prioritised. Therefore this branching activity will be selected and all activities not related to this activity will be

removed. Activity 2 triggers a nested subgraph. Note that activity 4 is linked to activity 7 of another alternative and thus activity 7 and its successors in the other alternative will be selected if branching activity 4 is selected. After deleting act. 3 and 6, the new activity list is: {1, 2, 4, 5, 7, 8, 9}. This updated activity list will be used for scheduling.

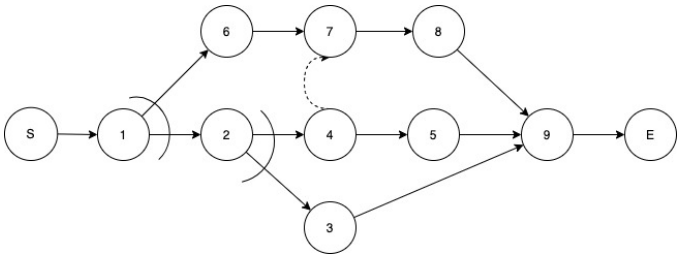


Fig. 1. An example of a linked nested project

4 Methodology

Since the RCPSP-AS consists of two subproblems (i.e. selection and scheduling subproblem), a feasible schedule can be obtained in two ways. First, a single PR is used to make a selection between the branching activities and, subsequently, schedule the corresponding selected activities. In this study, we will test various PRs that were traditionally developed for the RCPSP (Slowinski and Weglarz 1989). Second, two different priority rules can be used for the selection and scheduling subproblem, respectively. To select an alternative branch, we only consider the priority of immediate successors. For the scheduling subproblem, we use a PR similar to the PR that is used in case that both subproblems are scheduled using a single PR. We employ a serial schedule generation scheme (SSGS) to provide a feasible as soon as possible schedule for a given activity list.

5 Results

We find the average makespan for 36,000 project instances with different characteristics provided by Servranckx and Vanhoucke (2019) employing PRs developed for RCPSP and PRs by Capacho et. al. (2009). In Capacho et. al. (2009) selection PRs are defined as NP selects for each subassembly the subgraph with the minimum number of precedence relations; TT selects the subgraphs that involve the minimum total processing time of tasks; and NT selects the subgraphs that involve the minimum number of tasks. In Table 1, we show the average project makespan in case that two different PRs are used for the selection and the scheduling subproblem (upper part of Table 1) or a single PR is used for both subproblems (lower part of Table 1). The results show that employing a specific selection PR that is different from existing scheduling PRs in the literature generates schedules with shortened project duration for the RCPSP-AS. The selection of a subgraph with minimum processing time of activities dramatically decreases average project makespan in comparison to other selection and scheduling PRs for the RCPSP. Moreover, latest work station and minimum slack are scheduling PRs that produce shortened average project makespan for the tested project instances.

We call the PRs employed in the work of Capacho et. al. (2009) problem-specific since they distinguish between selection and scheduling PRs and their proposed PRs minimize the average makespan for the tested RCPSP-AS project instances.

Table 1. Average makespan of 36,000 instances for PRs in the literature

	(Capacho *et. al.* 2009)				
	Maximum task time	Minimum earliest work-station	Minimum latest work-station	Minimum slack	Maximum number of immediate successors
NP	212.47	212.26	207.97	208.62	209.92
TT	**203.74**	**202.15**	**196.71**	**197.39**	**199.62**
NT	212.76	212.50	208.37	209.05	210.37
	(Slowinski and Weglarz 1989)				
	Longest Processing Time	Early Start Time	Late Start Time	Minimum slack	Most immediate successors
	234.72	231.13	249.49	221.81	226.11

6 Conclusion

In complex and uncertain projects, the assumption of deterministic project structure is not always valid. To deal with this gap between research and practice, the RCPSP-AS has been developed. In this paper, we find single schedules for the RCPSP-AS employing PRs from the literature. We conclude that employing a specific PR for the selection and scheduling subproblem separately reduces the project makespan dramatically in comparison to using a single PR for both subproblems.

References

Blazewicz J., J. K. Lenstra and A.R. Kan, 1983, "Scheduling subject to resource constraints: classification and complexity", *Discrete applied mathematics*, 5(1), pp. 11-24.

Capacho L., R. Pastor, A. Dolgui and O. Guschinskaya, 2009, "An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem", *Journal of Heuristics*, 15(2), pp.109-132.

Capek R., P. Šucha and Z. Hanzálek, 2012, "Production scheduling with alternative process plans", *European Journal of Operational Research*, 217(2), pp.300-311.

Cooper D.F., 1976, "Heuristics for scheduling resource-constrained projects: An experimental investigation", *Management Science*, 22(11), pp. 1186-1194.

Kolisch R., S. Hartmann, 1999, "Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis", In *Project scheduling* pp. 147-178. Springer, Boston, MA.

Kolisch R., S. Hartmann, 2006, "Experimental investigation of heuristics for resource-constrained project scheduling: An update.", *European journal of operational research*, 174(1), pp. 23-37.

Servranckx T., M. Vanhoucke, 2019, "A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs", *European Journal of Operational Research*, 273(3), pp. 841-860.

Slowinski R., J. Weglarz, 1989, "Heuristic algorithms for resource constrained project scheduling: A review and an empirical analysis", In *Advances in Project Scheduling* pp. 113-134. Elsevier.

A comparison of two project forecasting methods using risk models: Structural Equation Modeling and Bayesian Networks

Izel Unsal Altuncan[1], Mario Vanhoucke[1,2,3] and Annelies Martens[1]

[1] Ghent University, Belgium
izel.unsalaltuncan@ugent.be
annelies.martens@ugent.be
[2] Vlerick Business School, Belgium
[3] University College London,UK
mario.vanhoucke@ugent.be

Keywords: Risk modelling, Project forecasting, Project simulation

1 Introduction

Project risks are controlled in literature as if they arise independently during project execution. However in practice, project risks often arise as dependent events. Therefore, in order to assure effective control of project risks, the hidden structure consisting of causal paths between interrelated risks should be identified. In addition to the qualitative knowledge on the causal interrelations, quantitative information on the level of causality is also required to completely exploit the benefits of the causal structure between project risks. Recently, directed networks consisting of quantitative parameters to represent the level of causal interaction between risks, the so-called *risk models*, have been receiving attention within the field of project management.

Risk models have been frequently used for risk analysis and risk response planning in project management for the last two decades, however, only a limited number of research papers have focused on using risk models for project forecasting. The way risk models predict project duration and cost is fundamentally different from the existing Earned Value Management (EVM) method. While EVM is based on periodic performance measurement during project progress, risk models only make use of static project data which can be obtained prior to the start of the project (i.e. activity network, sensitivity measures). Since no research has been explicitly conducted on project forecasting through risk models using a variety of project data, the applicability and effectiveness for forecasting is still vague. To the best of our knowledge, two causal modelling methods have been used in literature for forecasting project duration and cost, namely Structural Equation Modeling (SEM) and Bayesian Networks (BN). The goal of this paper lies in comparing the forecasting accuracy of the two methods using a large variety of project data. The outline of this manuscript is as follows. In Section 2 we briefly discuss the general approach consisting of three phases and provide details for the methodology framework of each phase. In Section 3 we discuss the general accuracy of the models for various test data settings. Finally in Section 4 we briefly discuss the planned future work.

2 General approach

SEM is an approach for linear modelling consisting of measurement models and a structural model, while BN is an approach for probabilistic inference through a single model. Since the methods are fundamentally different from each other we propose a specific ap-

proach which is explicitly designed for comparing the two methods using the same dataset, through identical topological structures.

The computational experiment is applied in four phases as displayed in Figure 1. In Phase 1, we propose a theoretical risk model based on background knowledge in project management. In Phase 2, we generate a variety of artificial projects and collect available empirical project data. In Phase 3, we train and test the theoretical risk model of Phase 1, using the artificial and empirical project data of Phase 2. Model training in this research refers to validating the theoretical model and determining the optimal parameter set using k-fold cross validation, while model testing refers to making use of the optimal model for forecasting project duration. Finally in Phase 4, we compare the forecasting accuracy of the SEM and the BN for different training data. In the remainder of this section we briefly discuss the highlights of each phase.

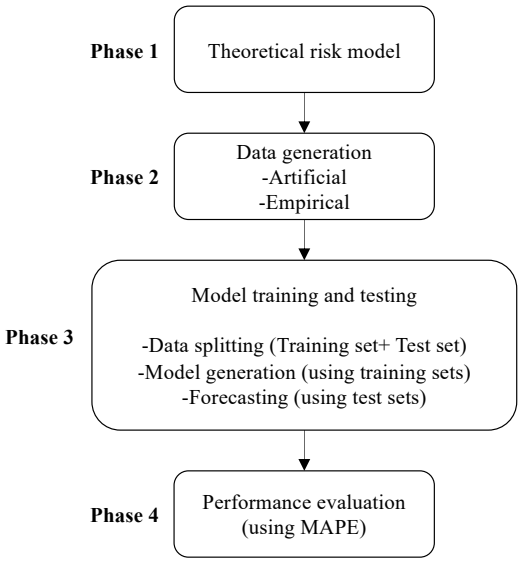


Fig. 1. General framework

Phase 1 Theoretical model: Since risk variables within the SEM approach are considered to be not directly measurable, in the theoretical model of this research given in Figure 2, we propose observable indicators (represented by rectangles) for measuring risk variables (represented by ellipses). The network topology (NT), time sensitivity (TS) and cost sensitivity (CS) measures of a project are proposed to affect the time performance (TP) and the cost performance (CP). In order to cover all aspects, time and cost sensitivity are modelled in terms of the standard deviation (TSstd and CSstd) and the average (TSav and CSav).

Phase 2 Data generation: For empirical data, 29 projects are collected from an open source database from Batselier and Vanhoucke (2015). For artificial data, 900 project networks presented in Vanhoucke (2010a) are simulated under 9 different scenarios, by making use of different distributions for activity durations. Projects are simulated in two parts: static simulation and dynamic simulation. The static simulation aims at estimating the project performance prior to the start using sensitivity metrics, while the dynamic simulation aims at imitating the actual project progress and obtains the real duration, the

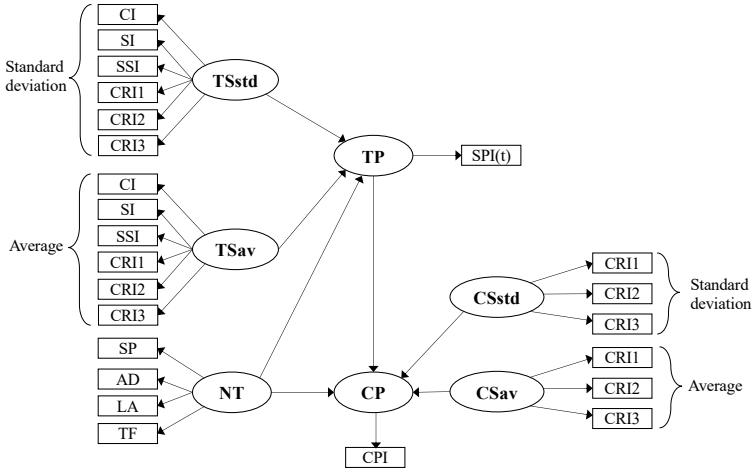


Fig. 2. Theoretical model

real cost, and thus the actual time performance (SPI(t)) and the actual cost performance (CPI).

Phase 3 Model training and testing: First, the artificial data (900 x 9 = 8100 simulated projects in total) is split into training set and test set. Then, artificial and empirical risk models are generated for the SEM and the BN, making use of the the observable indicator values obtained through both static simulations (sensitivity metrics) and dynamic simulations (SPI(t) and CPI) for the projects in the artificial training set and obtained through real-life project data for the projects in the empirical set. Once the risk models with optimal parameter sets are determined, we make use of the models for forecasting the time performance of projects using the network topology and sensitivity metrics in the artificial test set as input data.

Phase 4 Evaluation: In this phase, the resulting forecasts of the SEM and the BN are compared to the actual SPI(t) values for the projects in the test sets. Forecasting accuracy is measured using the Mean Absolute Percentage Error (MAPE), where the lower MAPE indicates a better forecasting performance.

3 Preliminary results

Overall, the relatively low MAPEs indicate that both the SEM and the BN are notable alternatives for forecasting project duration. The general results are as follows:

- The Length of Arcs (LA) and the Topological Float (TF) are excluded from the risk models since they are considered to be not strongly correlated with the network topology (NT) for forecasting both the time performance (TP) and the cost performance (CP). This is in line with the previous discussions which claim that only SP and AD have a significant influence on the forecasting accuracy (Vanhoucke (2010a)).
- Only the Schedule Sensitivity Index (SSI) and the Cruciality Index (CRI) are considered as explicit observable indicators for the time sensitivity. This is in line with the discussions in Vanhoucke (2010b) which confirms the further effectiveness of these metrics over the Criticality index (CI) and the Significance Index (SI).
- As given in Figure 3, the forecasting accuracy of the BN is better than SEM, regardless of the type of the training data. More specifically, when the artificial data is used for

model training, MAPE values for the BN are 7.19% lower on average, and when the empirical data is used for model training, MAPE values of the BN are 4.08% lower on average.

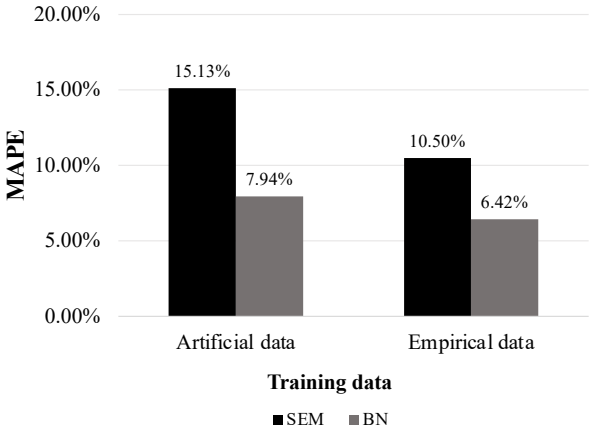


Fig. 3. Average MAPE for different training data

4 Future research

There are two future research intentions. First, we will investigate the impact of test project characteristics on the accuracy of the SEM and the BN. Second, we will compare the forecasting accuracy of the SEM and BN to the EVM methods.

References

Batselier, J. and Vanhoucke, M., 2015, Construction and evaluation framework for a real-life project database, *International Journal of Project Management*, 33:697-710.

Gupta, S. and Kim, H. W. , 2008, Linking structural equation modelling to Bayesian networks: Decision support for customer retention in virtual communities, *European Journal of Operational Research*, 190:818-833.

Vanhoucke M., 2010a, *Measuring time: Improving project performance using earned value management*, volume 136 of International Series in Operations Research and Management Science, Springer.

Vanhoucke, M., 2010b, Using activity sensitivity and network topology information to monitor project time performance, *Omega*,38, 359-370.

Vanhoucke M., 2012, *Project Management with Dynamic Scheduling*, Springer.

Using schedule risk analysis with resource constraints for project control

Jie Song[1], Annelies Martens[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
jieson.song@ugent.be, annelies.martens@ugent.be, mario.vanhoucke@ugent.be
[2] Technology and Operations Management, Vlerick Business School, Belgium
[3] UCL School of Management, University College London, United Kingdom

Keywords: Project Management, Project Control, Resource-constrained, Schedule Risk Analysis.

1 Introduction

Schedule Risk Analysis (SRA) has shown to provide reliable activity sensitivity information for taking corrective actions during project control. More precisely, by selecting a small subset of activities with high sensitivity values for taking corrective actions, the project outcome can be improved. In resource-constrained projects, disrupted activities can affect both their successors as well as other activities when resource conflicts are induced. Since SRA focuses solely on the project network to determine the sensitivity of activities, the traditional SRA metrics do not accurately reflect the activity sensitivity for resource constrained projects. In this paper, we extend the traditional SRA metrics to be adequate for resource constrained projects and a novel resource-based sensitivity metric is proposed. These metrics are referred to as resource constrained SRA metrics (RC-SRA). An extensive computational experiment is established to analyse the impact of constrained resources on the efficiency of the risk analysis and project control phase. In particular, two activity selection strategies to determine which activities should be taken corrective actions on are compared. Finally, the efficiency of two types of corrective actions will be reviewed.

2 Problem formulation

Schedule risk analysis (SRA) is a project management technique to analyze the risk of the baseline schedule. More precisely, uncertainty is added to the project by using activity duration distribution and Monte Carlo simulations. This activity uncertainty might affect the probability of activities being on the critical path (activity criticality) and might affect the project outcome (impact of activity uncertainty). Although both aspects are often discussed separately, the sensitivity of an activity is defined by the combination of them. Therefore, these three aspects are discussed independently along the following lines.

Probability of activity criticality. The criticality index (CI), introduced by Martin (1965), gives an indication of the probability that an activity lies on the critical path. It is a simple and straightforward metric, which has been widely studied in literature. Although the CI has been studied widely, it often fails in accurately identifying the weaknesses of the project, since it does not consider the impact of activity uncertainty on the project outcome.

Impact of activity uncertainty. To consider the impact of activity uncertainty on the project outcome, Williams (1992) proposes the significance index (SI) and the cruciality index (CRI). The SI is an indicator for the impact of activity uncertainty on the project duration and can be used to reflect the relative importance between project activities. The CRI measures the correlation between the activity durations and the final project duration. While the CRI only measures the linear relationship between the activity duration

and project duration, Cho and Yum (1997) propose that the relation between these two variables often follows a non-linear relation.

Combination of probability and impact. The risk of an activity is accurately measured by considering both the probability of activity criticality and impact of activity uncertainty. The schedule sensitivity index (SSI) is proposed to combine the standard deviation of the activity duration and the project duration (S_{di} and S_{Cmax}) with the CI, which is exactly a merge of the impact of uncertainty and the probability of activity criticality (PMBOK 2004). Ballesteros-Perez *et. al.* (2019) put forward the Criticality-Slack-Sensitivity (CSS) index, which is a refinement of the SSI metric and show that the CSS performs better than the existing activity sensitivity metrics in the literature.

Current study. The SRA metrics provide an indication of the sensitivity of project activities based on a network analysis. Hence, it is assumed that delays in an activity only affects its successors. In resource constrained projects, delays in activities do not only affect succeeding activities, but can also delay other activities when resource conflicts are induced. In this case, the traditional SRA metrics might not accurately indicate the actual sensitivity of the activities. Therefore, in this paper, SRA metrics for resource constrained projects (RC-SRA metrics) are proposed and evaluated. As Fig. 1 shows, the traditional SRA metrics for measuring the probability of criticality, the impact of activity uncertainty and a combination of the probability and impact are extended to the resource constrained project context. For a detailed discussion on RC-SRA metrics, we refer to the work of Song *et. al.* (2021). Second, to validate whether the RC-SRA metrics indicate the sensitivity of activities more accurately in a resource constrained project context, an extensive computational experiment is conducted. In the first experiment, two activity selection strategies to determine which activities should be taken corrective actions on are evaluated. Finally, the impact of two different types of corrective actions for resource constrained projects are compared.

	CPM		RCP
Probability of activity criticality	CI (Martin, 1965) Based on critical path	extension →	RCI Based on critical chain
Impact of activity uncertainty	SI (Wiliams, 1992) Based on earliest start schedule (ESS)	extension →	RSI Based on resource feasible schedule
	CRI (Wiliams, 1992) Connection activity duration and project duration		—
Combination of probability and impact	SSI (PMBOK, 2004) Using CI	extension →	RSSI Using RCI
Impact of constrained resources	—		Novel metric: RRUI

Fig. 1. Comparison of SRA and RC-SRA metrics

3 Methodology

3.1 Data generation

In order to test the impact of RC-SRA metrics on the project control process, a large set of fictitious projects with the well-considered topological network structure and resource constraindness is generated by a project network generator RanGen2 (Vanhoucke *et. al.* 2008). First, the topological structure of these fictitious project networks are indicated with the serial/parallel (SP) indicator, which takes a value in $[0, 1]$. Second, the resource constraindness (RC) measures the average resource amount of a resource type consumed by all the activities divided by the limited availability of this resource type, and the RC indicator takes a value in $[0, 1]$ as well. In the experiment, each combined SP-value (0.2, 0.4, 0.6, 0.8) and RC-value (0.2, 0.4, 0.6, 0.8) contains 100 projects. As a consequence, 1,600 ($= 4 \times 4 \times 100$) projects are generated and tested in the experiment.

3.2 Resource-constrained schedule risk analysis

RC-SRA consists of three steps, namely constructing a resource-feasible baseline schedule, running simulation for the project network, and measuring the sensitivity values of each activity, which is in line with traditional SRA. Subsequently, the result of the simulation runs is that each activity has a value for the sensitivity metrics. In order to simulate uncertain activity durations, Monte Carlo simulations are deployed to generate the real activity duration with the presence of uncertainty. A lognormal distribution that is skewed to the right is employed to model the actual activity duration using two shape parameters, with $\mu = 1.1$ and $\delta = 0.3$ (Martens and Vanhoucke 2019, Hu *et. al.* 2016).

3.3 Simulation run with corrective actions

During project execution, the project manager should select the activities for corrective actions and decide which type of corrective actions should be taken on the selected activities,

Activity selection strategies. To select activites for actions, two distinct strategies can be applied. First, a *normal strategy* (NS) adopts the traditional view on determining a small subset of activities for taking corrective actions (Vanhoucke 2010), i.e. a subset of most sensitive activities is selected at each tracking period. Second, a *sequential strategy* (SS) focuses on the highly sensitive activities, but only the activities for which actions lead to a makespan reduction are selected, while the other highly sensitive activities are not taken into further consideration.

Types of corrective actions. In a resource-feasible RCPSP schedule, two types of activity crashing are distinguished, with the aim of reducing the duration or the resource demand of the selected activities. The first type deploys the traditional viewpoint on activity crashing. In order to reduce the activity duration, the project manager invests additional resources to crash the selected activity. The second type can be viewed as a slightly adapted version of the traditional activity crashing. Instead of reducing the activity duration, the resource demand is decreased which results in increased activity duration.

4 Results

The crashing effectiveness (CE) is measured as a ratio between the reduction in project delays due to corrective actions ($Delay^{no} - Delay^{yes}$) and total delay before corrective actions ($Delay^{no}$) along the project progress (Eq. (1)). This metric measures the extent of

project delays that has been reduced by taking corrective actions.

$$Crashing effectiveness = \frac{1}{nrs_l} \sum_{I=1}^{nrs_l} (\frac{Delay^{no} - Delay^{yes}}{Delay^{no}}) \quad (1)$$

With $Delay^{no} = RD^{no} - PD$, and $Delay^{yes} = RD^{yes} - PD$, nrs_l, number of delayed projects in the simulation. The PD is the planned duration of the project. RD^{no} and RD^{yes} are the project duration obtained from the simulated project execution without and with effort.

First, the results show that the crashing effectiveness of the RSSI measure (10.47%, on average) is relatively higher than the RCI (9.32%), RSI (8.80%) and CRI (9.62%). Further, the SSI performs as effective as the RSSI measure when the RC is low. Since the lower RC values indicate that the resources are less restricted, the SSI can still provide useful activity sensitivity information for taking corrective actions during the project tracking process. Consequently, the SSI does not lose its initial value when the availability of resources is high. However, when the RC values increase, the crashing effectiveness of the RSSI approach is relatively higher (i.e. 8.01% vs. 5.40%) than the SSI measure. Thus, incorporating constrained resources in the RC-SRA metrics helps improve the accuracy of corrective actions in projects with more restricted resources during project tracking.

Second, the results show that, on average, the use of the sequential strategy results in a better crashing effectiveness. More precisely, the crashing effectiveness of the sequential strategy is particularly better than the normal strategy for parallel projects, while the difference is marginal for serial projects. This is due to the fact that the resources or precedence conflicts are more likely to occur in parallel projects during project tracking. The results also reveal that, on average, the crashing effectiveness is higher when the first type of corrective actions are used.

References

Ballesteros-Pérez P., K. M. Elamrousy and M. C. González-Cruz, 2019 "Non-linear time-cost trade-off models of activity crashing: Application to construction scheduling and project compression with fast-tracking", *Automation in Construction*, Vol. 97, pp. 229-240.

Cho J.G., B.J. Yum, 1997, "An uncertainty importance measure of activities in PERT networks", *International Journal of Production Research*, Vol. 35, pp. 2737-2758.

Hu X., N. Cui, E. Demeulemeester and L. Bie, 2016, "Incorporation of activity sensitivity measures into buffer management to manage project schedule risk", *European Journal of Operational Research*, Vol. 249, pp. 717-727.

Martens A., M. Vanhoucke, 2019, "The impact of applying effort to reduce activity uncertainty on the project time and cost performance", *European Journal Of Operational Research*, Vol. 277, pp. 442-453.

Martin J.J., 1965, "Distribution of the time through a directed, acyclic network", *Operations Research*, Vol. 13, pp. 46-66.

PMBOK., 2004, "A Guide to the Project Management Body of Knowledge, Third Edition", *Newtown Square, Pa.: Project Management Institute, Inc.*

Song J., A. Martens and M. Vanhoucke, 2021, "Using schedule risk analysis with resource constraints for project control", *European Journal of Operational Research*, Vol. 288, pp. 736-752.

Vanhoucke M., J. Coelho, D. Debels, B. Maenhout and L. Tavares, 2008, "An evaluation of the adequacy of project network generators with systematically sampled networks", *European Journal of Operational Research*, Vol. 187, pp. 511-524.

Vanhoucke M., 2010, "Using activity sensitivity and network topology information to monitor project time performance", *Omega The International Journal of Management Science*, Vol. 01, pp. 1-5.

Williams T.M., 1992, "Criticality in stochastic networks", *Journal of the Operational Research Society*, Vol. 43, pp. 353-357.

Project Planning for Engineering Automotive Production Systems

Maximilian Kolter, Martin Grunow, Rainer Kolisch and Thomas Stäblein

TUM School of Management, Technichal University Munich, Germany
max.kolter@tum.de, martin.grunow@tum.de, rainer.kolisch@tum.de,
thomas.staeblein@tum.de

Keywords: resource leveling, multi-project scheduling, outsourcing, external resources.

1 Introduction

In the automotive industry, there is a trend of growing product portfolios and decreasing product life cycles (Dusan *et al.* 2019). This trend results in a large number of new product introductions, for which not only new cars have to be engineered but also new production systems. The engineering of production systems are projects which serve the development of new production systems and consist of various engineering activities such as production equipment planning or layout planning. Car manufacturers need to cope with multiple overlapping production system engineering projects with short development times. However, they only have a limited workforce of engineers and thus rely on outsourcing to handle the increasing workload. Furthermore, a major European car manufacturer brought to our attention that there are strategic efforts to reduce the number of internal engineers despite this growing demand. Consequently, car manufacturers face the problem of scheduling their production system engineering projects such they make the best use of scarce internal engineers and use outsourcing in a targeted and sensible manner. This study aims to address this problem by developing a framework to schedule production system engineering projects such that internal engineers are utilized efficiently and outsourced activities are combined into attractive work packages for contractors.

The remainder of this paper is organized as follows: In Section 2, we formally define the problem. Next, in Section 3, we provide a brief literature review. Then in Section 4, we propose a mixed-integer programming formulation for the problem. Finally, in Section 5, we discuss some preliminary results.

2 Problem Definition

Each project $p \in \mathcal{P}$ depicts the engineering of a production system. All projects have to be scheduled within the discrete planning horizon $\mathcal{T}$ consisting of periods 1 to T. In order to start engineering the production system for a new car, a certain product development status of the car to be manufactured is required. We assume that the time this status is achieved is known and represents the release date for a project. The deadline of each project is dictated by the planned start-of-production (SOP), which aligns with the market introduction of the product. Each project p is associated with a set of engineering activities (e.g., layout planning, assembly planning) represented by $\mathcal{V}_p$.

The activities must be processed according to finish-to-start precedence relations that are represented by the set of arcs $(i, j) \in \mathcal{E}$. For processing activities, engineers with different skills (e.g., layout planner or assembly planner) are available. The set $\mathcal{R}$ denotes the different types of resources according to their skill. Each activity j requires exactly one specific skill $k \in \mathcal{R}$ for being processed and r_{jk} denotes the number of resources with skill k required during each period activity j is processed. Furthermore, activities can be processed

in two different modes $m \in \mathcal{M}$. Mode $m = 0$ to process an activity with internal resources and mode $m = 1$ to process activities with external resources (outsourcing). Each activity j must be processed in exactly one mode $m \in \mathcal{M}$ for a (mode-independent) duration of p_j. However, for reasons such as intellectual property protection, some activities must be processed with internal resources. We denote the modes allowed for activity j by $\mathcal{M}_j$.

The available number of internal engineers limits the number of activities processed with internal resources. We assume that a maximum of b_{kt} internal engineers with skill k is available in period t. Additionally, to account for the strategic goal of decreasing the size of the internal workforce, we assume the management prescribes a maximum $b_t < \sum_{k \in \mathcal{R}} b_{kt}$ for the size of the internal workforce (number of all engineers) for every period t.

The objective is to find a feasible schedule for the projects $p \in \mathcal{P}$ that first seeks to maximize the use of internal resources and second creates schedules for the outsourced work that are attractive for contractors. We define the attractiveness of work packages for contractors as follows. Work packages are more attractive if they (*i*) consist of consecutive activities with few interruptions, (*ii*) have small peaks in the resource requirements, and (*iii*) have little variance in the resource requirement over time.

3 Literature Review

Three streams of literature are relevant to the problem presented in Section 2. The first stream concerns the engineering of production systems and provides many qualitative insights and quantitative models for single engineering activities (see Heragu (2018)). However, this stream does not provide quantitative methods for the scheduling of the required engineering activities. The second stream concerns the resource leveling problem (RLP); for an extensive overview see Rieck and Zimmermann (2015). The problem subject to this study is close to two classical RLP variants. The objective of minimizing outsourcing is similar to the overload RLP (Kreter *et al.* 2014) and the objective of creating attractive outsourcing packages is similar to the total adjustment cost RLP (Rieck *et al.* 2012). The third stream of literature deals with project scheduling considering outsourcing. Only a few papers consider outsourcing in project scheduling, for instance Heimerl and Kolisch (2010). However, to the best of our knowledge, no model exists that addresses the creation of attractive outsourcing packages. However, we argue that attractive outsourcing packages are highly relevant for practical applications since they lead to smaller outsourcing costs.

4 Model Formulation

In this section, we present a MIP for the problem. The decision variables are summarized in Table 1. The proposed MIP is based on the multi-mode resource constrained project scheduling problem (MRCPSP) (Weglarz *et al.* 2011). For the scheduling decisions, we use step variables x_{jmt}, which are binary variables that take the value 1 if and only if activity j is started in or before period t in mode m (Artigues *et al.* 2015). We assume that prior to optimization for each activity j the earliest start times ES_j are computed using the duration and release dates and latest start times LS_j are computed using the duration and SOPs (deadlines). To reduce the number of variables, we only define step variables for the feasible starting times as specified by the interval $[ES_j, LS_j]$.

The MIP has three lexicographic objectives. The first objective (1) seeks to maximize the utilization of the internal workforce by minimizing the outsourced work. Objectives (2) and (3) seek to generate attractive outsourcing packages by smoothing resource demand profiles for the outsourced work. For this purpose, objective (2) minimizes the peak volume

Table 1: Summary of decision variables

Discrete Variables	
u_{kt}	Number of internal resources k utilized in period t
x_{jmt}	1, if activity j is started in mode m before or in the beginning of period t, 0 otherwise
Continuous Variables	
z_{kt}	Amount of outsourced work requiring resource k in period t
z_{kt}^+, z_{kt}^-	Positive and negative change of outsourcing volume of work requiring resource k from period $t-1$ to t
z_k^{max}	Peak volume of outsourced work requiring resource k

of the outsourced work and objective (3) minimizes changes in the outsourcing volume over time.

$$\min\ v_1 = \sum_{k\in\mathcal{R}}\sum_{t\in\mathcal{T}} z_{kt} \tag{1}$$

$$\min\ v_2 = \sum_{k\in\mathcal{R}} z_k^{max} \tag{2}$$

$$\min\ v_3 = \sum_{k\in\mathcal{R}}\sum_{t\in\mathcal{T}} (z_{kt}^+ + z_{kt}^-) \tag{3}$$

While optimizing (1) - (3), the following constraints have to be considered. The first block of constraints (4) - (6) defines the scheduling process. Constraints (4) ensure that each activity is scheduled exactly once in exactly one mode. Thereby, the start and finish of activities must adhere to finish-start precedence relations between activities as defined in constraint (5). Furthermore, constraints (6) ensure that once started, activities are not interrupted (no preemption).

$$s.t. \sum_{m\in\mathcal{M}_j}\sum_{t\in\mathcal{T}\setminus\{1\}} (x_{jmt} - x_{jm,t-1}) = 1 \qquad \forall j \in \mathcal{V} \tag{4}$$

$$\sum_{m\in\mathcal{M}_j} x_{jmt} \leq \sum_{m\in\mathcal{M}_i} x_{im,t-p_i} \qquad \forall (i,j)\in\mathcal{E},\ t\in\mathcal{T} : t > p_i \tag{5}$$

$$x_{jm,t-1} \leq x_{jmt} \qquad \forall j\in\mathcal{V},\ m\in\mathcal{M}_j,\ t\in\mathcal{T}\setminus\{1\} \tag{6}$$

The second block of constraints (7) - (12) defines the resource allocation. First, the internal processing of activities is limited by the number of internal engineers with skill k in constraints (7). For each internal resource k the number of required units has to be within the available capacity as defined in constraints (8) and (9). Work that exceeds this internal capacity must be outsourced. The volume of outsourced work is computed in (10). Subsequently, the peak of outsourced work and the changes in outsourcing volume over time are computed in constraints (11) and (12), respectively. Note, variable definition constraints are omitted for the sake of space.

$$\sum_{j\in\mathcal{V}_k} r_{jk}(x_{j0t} - x_{j0,t-p_j}) \leq u_{kt} \qquad \forall k\in\mathcal{R},\ t\in\mathcal{T} : t > p_j \tag{7}$$

$$u_{kt} \leq b_{kt} \qquad \forall k\in\mathcal{R},\ t\in\mathcal{T} \tag{8}$$

$$\sum_{k \in \mathcal{R}} u_{kt} \leq b_t \qquad \forall t \in \mathcal{T} \tag{9}$$

$$\sum_{j \in \mathcal{V}_k} r_{jk}(x_{j1t} - x_{j1,t-p_j}) - z_{kt} = 0 \qquad \forall k \in \mathcal{R},\ t \in \mathcal{T} : t > p_j \tag{10}$$

$$z_{kt} \leq z_k^{max} \qquad \forall k \in \mathcal{R},\ t \in \mathcal{T} \tag{11}$$

$$z_{kt} - z_{kt-1} - z_{kt}^{+} + z_{kt}^{-} = 0 \qquad \forall k \in \mathcal{R},\ t \in \mathcal{T} \setminus \{1\} \tag{12}$$

5 Numerical Results

We conducted preliminary tests to evaluate the capability of off-the-shelf solvers to solve MIP (1) - (12). For this purpose, we randomly generated projects, activities, and precedence relationships. We considered between 1 and 3 projects, 10 and 20 non-dummy activities per project, and 1 and 3 resources (18 instances total). We implemented the model in Python and solved it with Gurobi version 9.1. We solve the objectives (1) - (3) lexicographically in the order given. The experiments were performed on an Intel Core i5-7200U CPU 2.50GHz with 16 gigabytes of RAM under Windows 10 64-bit operation system. The CPU time limit was set to 3,600 seconds for each objective (1), (2), and (3).

Table 2: Results

Objective	Average runtime [sec]	Instances solved to optimality [%]	Average gap [%]	Max gap [%]
(1)	162	100.00	0.00	0.00
(2)	1,057	72.22	3.61	25.00
(3)	1,656	61.11	10.78	47.99

Table 2 reports the computational results. Considering that not all of these small instances could be solved optimal, finding optimal solutions for realistic instance sizes requires a more tailored solution approach. We are currently working on a decomposition approach that will be presented at the conference. Additionally, we will present an extensive computational study as well as a case study using real-world data.

References

Artigues C., O. Kone, P. Lopez, M. Mongeau, 2015, "Mixed-Integer Linear Programming Formulations", in Schwindt C., J. Zimmermann *Handbook on Project Management and Scheduling*, Switzerland, Springer International Publishing, pp. 17-41.

Dusan S., M. Molnar and G. Fedorko, 2019, "Shortening of Life Cycle and Complexity Impact on the Automotive Industry", *TEM Journal*, Vol. 8(4), pp. 1295-1301.

Heimerl C., R. Kolisch, 2010, "Scheduling and staffing multiple projects with a multi-skilled workforce", *OR Spectrum*, Vol 32(2), pp. 343-368.

Heragu S.S., 2018, *Facilities Design*, 4th edn., CRC Press, Boca Raton.

Kreter S., J. Rieck, J. Zimmermann, 2014, "The total adjustment cost problem: Applications, models, and solution algorithms", *Journal of Scheduling*, Vol 17(2), pp. 145-160.

Rieck J., J. Zimmermann, T. Gather, 2012, "Mixed-integer linear programming for resource leveling problems", *European Journal of Operational Research*, Vol 221(1), pp. 27-37.

Rieck J., J. Zimmermann, 2015, "Exact Methods for Resource Leveling Problems", in Schwindt C., J. Zimmermann *Handbook on Project Management and Scheduling*, Switzerland, Springer International Publishing, pp. 361-387.

Weglarz J., J. Jozefowska, M. Mika, G. Waligora, 2011, "Project scheduling with finite or infinite number of activity processing modes - A survey", *European Journal of Operational Research*, Vol 208(3), pp. 177-205.

Learning based heuristics for scheduling jobs with release dates on a single machine to minimize the sum of completion times

Axel Parmentier[1], Vincent T'kindt[2]

[1] CERMICS, Ecole des Ponts, Marne-la-Vallée, France
axel.parmentier@enpc.fr
[2] University of Tours,
LIFAT (EA 6300), ERL CNRS ROOT 7002, Tours, France
tkindt@univ-tours.fr

Keywords: Single machine, heuristics, machine learning.

1 Introduction

Consider the problem where n jobs have to be scheduled on a single machine. Each job j is defined by a *processing time* p_j and a release date r_j so that, in a given schedule, no job j can start before its release date. The machine can only process one job at a time and preemption is not allowed. The goal is to find a schedule s (permutation) that minimizes the total completion time $\sum_j C_j(s)$ with $C_j(s)$ the completion time of job j in schedule s. If $s = (j_1, \ldots, j_n)$, then

$$C_{j_1}(s) = r_{j_1} + p_{j_1} \quad \text{and} \quad C_{j_k}(s) = \max(C_{j_{k-1}}(s), r_{j_k}) + p_{j_k} \text{ for } k > 1.$$

When there is no ambiguity, we omit the reference to schedule s when referring to completion times. Following the standard three-field notation in scheduling theory, this problem is referred to as $1|r_j|\sum_j C_j$ and is strongly $\mathcal{NP}$-hard (Rinnooy Kan 1976). When there is no release dates, the corresponding $1||\sum_j C_j$ problem can be solved in $O(n\log(n))$ time using the SPT rule (shortest processing times first).

The $1|r_j|\sum_j C_j$ problem is a challenging problem which has been studied for a long time. In this work, we focus on heuristic algorithms which can be used to compute good solutions in a reasonable amount of time. Along the years, numerous heuristic algorithms have been proposed. We cite the `RDI` (Release Date Improvement procedure) local search based on the APRTF (Advanced Priority Rule for Total Flowtime) greedy rule proposed by Chand et al. (1996) and which requires $O(n^4\log(n))$ time. The `RBS` heuristic (Recovering Beam Search) developed by Della Croce and T'kindt (2002) is a truncated search tree approach that has been the heuristic with the best performances for a decade. The RBS heuristic requires $O(wn^3\log(n))$ time with w the beam width parametrizing the heuristic: notice that Della Croce and T'kindt (2002) considered the case $w = 1$. To the best of our knowledge, the state-of-the-art heuristic is a matheuristic `MATH` proposed by Della Croce et al. (2014) which provides solutions very close to the optimal ones but at the price of a large CPU time requirement.

In this work we propose learning based heuristics for the $1|r_j|\sum_j C_j$ problem which are compared to these milestones heuristics.

2 Using machine learning to transform hard into easy instances

The use of machine learning (ML) techniques within operations research (OR) algorithms is a recent but active and promising research area (Bengio et al. 2021). To the best

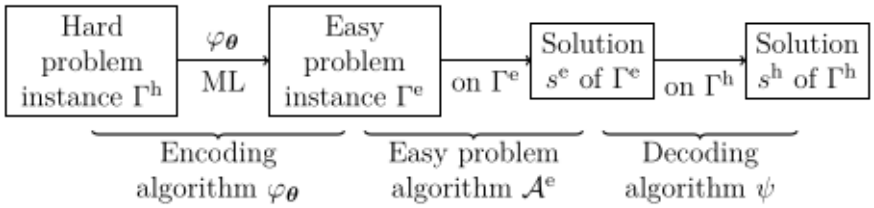


Fig. 1. ML to approximate hard problems by well-solved ones

of our knowledge, very few contributions of this kind have considered scheduling problems. In these works, ML is used to guide the solution process, *i.e.*, the proposed OR heuristic. In this paper, we elaborate on an original approach recently introduced by Parmentier (2021) and illustrated in Figure 1. A ML predictor $\varphi_{\boldsymbol{\theta}}$, which we call the encoding algorithm, is used to convert an instance $\mathit{\Gamma}^h$ of the *hard* $1|r_j|\sum_j C_j$ problem into an instance $\mathit{\Gamma}^e$ of the $1||\sum_j C_j$ problem. The latter is called the *easy* problem as it exists a practically efficient algorithm $\mathcal{A}^e$ to solve it (SPT rule). From an optimal solution s^e of $\mathit{\Gamma}^e$, a decoding algorithm is used to rebuild a solution s^h to $\mathit{\Gamma}^h$. Notice that, the processing time $\hat{p}_j$ of job j in $\mathit{\Gamma}^e$ is not equal to its processing time p_j in $\mathit{\Gamma}^h$, but to a linear combination of features computed from $\mathit{\Gamma}^h$. Several different decoding algorithms are proposed to obtain a schedule for the $1|r_j|\sum_j C_j$ problem, thus leading to different heuristics.

The main challenge to make such an approach working is to build an encoding algorithm $\varphi_{\boldsymbol{\theta}}$ such that the optimal solution of the instance $\mathit{\Gamma}^e$ leads to a good solution $\mathit{\Gamma}^h$ after decoding. As usual in ML, we first define an appropriate family of predictors $(\varphi_{\boldsymbol{\theta}})_{\boldsymbol{\theta}}$, and then seek (learn) the best parameter $\boldsymbol{\theta}$. We formulate the choice of $\boldsymbol{\theta}$ as a structured learning problem. Structured learning algorithms on the permutation group have been thoroughly studied in the literature on rankings, but with applications such as document retrieval or question answering that are quite far from scheduling. If the traditional learning approaches of this literature can in theory be applied in our context, we do not use them because the loss functions they use to evaluate if a ranking is a good approximation of another are not good evaluations of the quality of a $1|r_j|\sum_j C_j$ schedule. We instead propose a novel generic approach based on Fenchel-Young loss functions which can be applied to a large set of hard problem (Parmentier and T'kindt 2021).

In this abstract we don't detail how the predictor $\varphi_{\boldsymbol{\theta}}$ is built but we rather highlight how work the three proposed learning based heuristics.

3 Learning based heuristics

The first heuristic we propose, denoted by `PMLH` (*Pure Machine Learning Heuristic*) uses $\varphi_{\boldsymbol{\theta}}$ to transform the instance $\mathit{\Gamma}^h$ into an instance $\mathit{\Gamma}^e$ of the $1||\sum_j C_j$ problem. Then, it applies the SPT rule to obtain schedule s^e and evaluate this schedule on instance $\mathit{\Gamma}^h$. Heuristic `PMLH` requires $O(n\log(n))$ time.

It may happen that for instance $\mathit{\Gamma}^h$, in the schedule built from s^e, two consecutive jobs j and k, with $j \to k$, are such that $p_j > p_k$. Let t be the starting time of j in that schedule. Then, if $r_j, r_k \leq t$ the schedule is suboptimal and swapping j and k leads to a better schedule. In the second heuristic we propose, denoted by `IMLH` (*Improved Machine Learning Heuristic*), we apply a fast local search `LS` to repair these problematic cases, if

any. This local search has a $O(n^2)$ worst-case time complexity. To improve the schedule s^i obtained from LS, we finally apply the RDI. More precisely, we apply the version of RDI where, each time a rescheduling of jobs is necessary, we use the SPT rule on the $\hat{p}_j$'s, *i.e.* the processing time value of job j in instance Γ^{e}. The RDI heuristic requires $O(n^4 \log(n))$ time in the worst case. It follows that heuristic IMLH requires $O(n^4 \log(n))$ time in the worst case.

The third proposed heuristic, denoted by itMLH (iterative MLH), is an extension of heuristic IMLH, which is obtained by applying IMLH with m different perturbed versions $\boldsymbol{\theta}_k \in \mathbb{R}^d$ of the vector $\boldsymbol{\theta}$. In this approach, we choose to approximate Γ^{h} by m instances of the easy problem $\Gamma^{\mathrm{e}}_1, \ldots, \Gamma^{\mathrm{e}}_m$. We suggest to use the instances:

$$\Gamma^{\mathrm{e}}_i = \varphi_{\boldsymbol{\theta}_k}(\Gamma^{\mathrm{h}}) \tag{1}$$

with $\boldsymbol{\theta}_k = \boldsymbol{\theta} + z_k$ and $\boldsymbol{z}_k$ is a sample of a random variable $\boldsymbol{Z}$ on $\mathbb{R}^d$, that can be chosen arbitrarily. The heuristic itMLH starts by running IMLH with $\boldsymbol{\theta}$. It then iteratively launches m runs of IMLH, using $\boldsymbol{\theta}_k$ instead of $\boldsymbol{\theta}$ at iteration k, i.e., Γ^{e}_k instead of Γ^{e}. It finally returns the best solution found over the different runs of IMLH. To speed up the heuristic, at each iteration k, if the computed schedule s^{e_k} has already been obtained at a previous iteration $k' < k$, then neither LS nor RDI are applied and the heuristic starts iteration $(k+1)$. Similarly, RDI is not applied whenever the solution s^i_k obtained after LS has already been computed at a previous iteration. It follows that heuristic itMLH requires $O(mn^4 \log(n))$ time in the worst case.

4 Computational experiments

We first introduce how instances Γ^{h} of the $1|r_j|\sum_j C_j$ problem are randomly generated (Della Croce and T'kindt 2002). For a given instance of $n \in \{120, 140, 160, 180, 200, 300, 500, 1000, 1500, 2000, 2500\}$ jobs, processing times p_j are drawn at random following the uniform distribution on $[1; 100]$ and release dates r_j are drawn at random following the uniform distribution on $[1; 50.5 \times n \times \rho]$. Parameter ρ enables to generate instances of different difficulties: we consider $\rho \in \{0.2, 0.4, 0.6, 0.8, 1.0, 1.25, 1.5, 1.75, 2.0, 3.0\}$. For each combination of n and ρ, 30 instances are randomly generated.

In this abstract we provide a comparison of the heuristics on large instances. For a given instance Γ^{h}, the best known solution is the best solution found by the heuristics. Notice that a time limit of 120s is imposed in the experiments: as soon as for a given size n, the average running time of a heuristic exceeds 120s, this heuristic is no longer run for higher values of n.

Table 1 presents the obtained results: for each heuristic, columns T_{avg} and T_{max} indicate the average and maximum computation time. Columns δ_{avg} and δ_{max} indicate the average and maximum deviation to the best known solution with the deviation δ computed as follows:

$$\delta = 100.00 \times \frac{\sum_j C_j(H) - \sum_j C_j^{BNS}}{\sum_j C_j^{BNS}},$$

with $\sum_j C_j(H)$ the value of the solution returned by H, $\sum_j C_j^{BNS}$ being the value $\min_{H \in X}(\sum_j C_j(H))$ of the best solution known, and X the set of heuristics tested on instance Γ^{h}.

Table 1 shows that heuristics MATH and RBS are unable to solve the instances with more than $n = 300$ jobs due to the time limit of 120s. Heuristic RDI is able to solve instances with up to 2000 jobs but is slower than IMLH and itMLH. Regarding the deviations to the best known solutions, heuristic MATH remains the most efficient one, with an average deviation

equal to 0. Heuristic RBS is the second most efficient heuristic but, again, a slow one not able to solve large instances. Heuristics IMLH and itMLH provide very good results in terms of deviations with reduced average running times.
We can conclude from these experiments that:

1. Learning based heuristics offer a very good trade-off between the quality of the computed solution and the running time required. They also show very low deviations to the best solution known.
2. Heuristic RDI is outperformed by both IMLH and itMLH.
3. Heuristics MATH and RBS are unable to solve instances with more than 300 jobs due the time limit of 120s.

	PMLH				IMLH				itMLH			
n	$\delta_{avg}(\%)$	$\delta_{max}(\%)$	T_{avg} (s)	T_{max} (s)	$\delta_{avg}(\%)$	$\delta_{max}(\%)$	T_{avg} (s)	T_{max} (s)	$\delta_{avg}(\%)$	$\delta_{max}(\%)$	T_{avg} (s)	T_{max} (s)
120	1.015	6.872	0.00	0.00	0.076	0.873	0.00	0.00	0.035	0.353	0.38	1.00
140	0.985	9.146	0.00	0.00	0.061	0.611	0.00	1.00	0.030	0.159	0.48	1.00
160	0.830	7.860	0.00	1.00	0.059	0.885	0.02	1.00	0.029	0.237	0.47	1.00
180	0.880	6.211	0.01	1.00	0.057	0.665	0.01	1.00	0.026	0.237	0.58	1.00
200	0.843	8.245	0.01	1.00	0.053	0.709	0.02	1.00	0.023	0.325	0.65	1.00
300	0.800	3.096	0.00	1.00	0.042	0.515	0.05	1.00	0.018	0.197	0.98	1.00
500	0.874	3.535	0.02	1.00	0.032	0.423	0.12	1.00	0.003	0.061	1.67	3.00
1000	1.000	3.483	0.02	1.00	0.031	0.316	1.88	7.00	0.002	0.033	5.37	19.00
1500	1.088	3.668	0.09	1.00	0.029	0.234	12.29	53.00	0.003	0.025	19.08	87.00
2000	1.138	3.626	0.07	1.00	0.026	0.153	46.21	188.00	0.003	0.042	62.31	296.00
2500	1.172	3.624	0.11	1.00	0.021	0.156	128.88	539.00	0.000	0.000	166.49	865.00

	RDIA				RBS				MATH			
n	$\delta_{avg}(\%)$	$\delta_{max}(\%)$	T_{avg} (s)	T_{max} (s)	$\delta_{avg}(\%)$	$\delta_{max}(\%)$	T_{avg} (s)	T_{max} (s)	$\delta_{avg}(\%)$	$\delta_{max}(\%)$	T_{avg} (s)	T_{max} (s)
120	0.170	1.586	0.00	0.00	0.012	0.098	5.77	8.00	0.000	0.096	13.79	67.00
140	0.187	1.636	0.00	1.00	0.010	0.071	9.74	13.00	0.000	0.000	18.86	88.00
160	0.180	2.108	0.02	1.00	0.009	0.061	15.35	19.00	0.000	0.001	25.01	76.00
180	0.138	1.405	0.04	1.00	0.009	0.074	23.27	29.00	0.000	0.001	34.68	169.00
200	0.136	1.280	0.04	1.00	0.008	0.049	33.56	42.00	0.000	0.000	45.43	177.00
300	0.106	1.601	0.15	1.00	0.005	0.027	140.88	172.00	0.000	0.001	157.54	301.00
500	0.059	0.749	1.08	5.00	—	—	—	—	—	—	—	—
1000	0.042	0.808	15.33	76.00	—	—	—	—	—	—	—	—
1500	0.034	0.551	72.56	362.00	—	—	—	—	—	—	—	—
2000	0.025	0.754	220.46	1167.00	—	—	—	—	—	—	—	—
2500	—	—	—	—	—	—	—	—	—	—	—	—

Table 1. Comparison of the heuristics with best known solutions

References

Y. Bengio, A. Lodi and A. Prouvost. Machine Learning for Combinatorial Optimization: A Methodological Tour d'horizon. European Journal of Operational Research, 290(2):405-421, 2021.

S. Chand, R. Traub and R. Uzsoy. An iterative heuristic for the single machine dynamic total completion time scheduling problem. Annals of Operations Research, 23(7):641-651, 1996.

F. Della Croce, F. Salassa and V. T'kindt A hybrid heuristic approach for single machine scheduling with release times. Computers & Operations Research, 45:7-11, 2014.

F. Della Croce and V. T'kindt A recovering beam search algorithm for the one-machine dynamic total completion time scheduling problem. Journal of the Operational Research Society, 53:1275-1280, 2002.

A.H.C. Rinnooy Kan. Machine sequencing problem: classification, complexity and computation. Springer-Verlag, 1976.

A. Parmentier. Learning to Approximate Industrial Problems by Operations Research Classic Problems. Operations Research, 2021.

A. Parmentier and V. T'kindt Learning to solve the single machine scheduling problem with release times and sum of completion times. arXiv:2101.01082, 2021.

Automated design of priority rules for the RCPSP via efficient genetic programming approach

Jingyu Luo[1], Mario Vanhoucke[1,2,3] and José Coelho[1,4]

[1] Ghent University, Belgium
jingyu.luo@ugent.be, mario.vanhoucke@ugent.be
[2] Vlerick Business School, Belgium
[3] University College London, United Kingdom
[4] Universidade Aberta, Portugal
jose.coelho@uab.pt

Keywords: project scheduling, priority rules, genetic programming.

1 Introduction

The resource-constrained project scheduling problem (RCPSP) is one of the most challenging scheduling problems in practice, and effective but simple and intuitive rules-of-thumb are often used to obtain sufficiently good and practicable solutions for large real-life instances. This method is called the priority rule-based approach, which relies on a priority function that uses project information to calculate the priority for the activities in the project and ranks them in a certain order to be scheduled. However, the design of good priority rules is a non-trivial task. Even with significant knowledge and experience, experts and researchers are limited in the possibilities they can consider.

Since the priority rule-based method is a heuristic approach, the hyper-heuristic algorithm, which is motivated by automating the design of heuristic methods to solve complex problems, can be used to design priority rules. Recently, genetic programming (GP), one of the most popular hyper-heuristic algorithms, has emerged as a powerful approach to design efficient priority rules for the job-shop scheduling problem (Branke *et. al.* (2015)). However, the potential of applying a *genetic programming hyper-heuristic* (GPHH) to design priority rules has been scarcely explored in the domain of project scheduling, and one of its main drawbacks is that it often requires a long runtime before a well-acceptable priority rule is found. Therefore, in this work, we aim to deploy some improvements on the GPHH to reduce its runtime while still maintaining its performance.

The contribution of this work is threefold. First, we propose a "duplicate removal" technique to reduce the number of evaluations that need to be performed. Second, we design two different surrogate models for the RCPSP problem to avoid evaluating newly generated rules on the complicated original model. In addition, we investigate the impact of the training data on the algorithm's performance and the selection of the fitness function. Computational experiments indicate that our proposed methods can significantly improve the efficiency of the GPHH.

2 Efficient GPHH for RCPSP

2.1 The general procedure of GPHH

Genetic programming is widely used as a heuristic generation methodology (Burke *et. al.* (2013)), and in recent years, it has attracted the attention of many researchers in the field of operations research. In this work, a genetic programming hyper-heuristic is deployed to design priority rules for the RCPSP, and its general procedure is shown in Fig. 1.

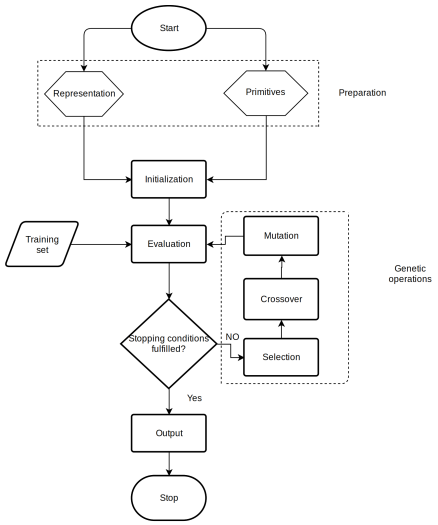


Fig. 1. The procedure of designing priority rules with GPHH

As can be seen from Fig. 1, first, the preparation process determines the representation of the priority rules and the primitives used to construct them. Since priority functions are expressions used to calculate the priority values, the primitives in this work consist of variables and mathematical operators that can be used to construct the expression of priority functions. In this work, we use the expression tree structure (Koza (1994)) to represent the expression of priority functions. Second, the GPHH starts with the initialization process, in which the initial population of priority rules is constructed using the primitives. Each rule can now be treated as an individual in the population. Subsequently, in the evaluation process, the fitness of individuals in the population will be evaluated by their performance on the training set. Based on the obtained fitness, genetic operations, including selection, crossover, and mutation, are applied to the current population to produce new rules for the next generation. Finally, once the stopping conditions are met, the global best individual will be the output of the GP and can be used to solve unseen projects.

2.2 Duplicate removal technique

In order to improve the efficiency of GPHH, a duplicate removal technique is deployed to reduce the number of evaluations performed by the algorithm. Since priority values are the expression value of the priority functions, we consider two types of duplicates: (i) The *expression value* duplicate, which is defined as individuals that always produce identical expression values despite the input data. (ii) The *expression structure* duplicate, which is defined as individuals with a similar or even identical structure with others, and therefore although they are sensitive to the input data, their expression value is equivalent to each other with the same input.

To detect the duplicate individuals, we record the priority value calculated by the individuals. We first sampled 100 activities from the training data, and each individual in the population is used to calculate the priority value for the selected activities. Subsequently, we remove the expression value duplicate by assigning a predefined fitness value to individuals that produce the same value for all the 100 activities. Finally, for the expression structure duplicate removal, we rely on comparing the expression value of the individual (not evaluated yet) with the evaluated ones. Two individuals are assumed to be equiva-

lent and have the same fitness value if they produce the same expression value on all the recorded activities. Thus, we can immediately assign the fitness value for an individual once it has an equivalent rule that is already evaluated.

2.3 Surrogate models

The surrogate models are used as approximation models that mimic the behavior of the original problem as closely as possible while still being fast and simple surrogates for the original problem. Thus, we can evaluate the individuals on the surrogate models, and the runtime can be reduced. In this work, two different types of surrogate models for the RCPSP are considered: the *phenotypic characterization* and the *simplified model*. The phenotypic characterization captures the behavior of the phenotype encoded in the individual and describes its characteristics, while the simplified model method approximates the fitness of the individuals by reducing the complexity of the original problem.

Phenotypic characterization. We use a vector consisting of a priority rule's decisions in a group of eligible sets as its phenotype. For that purpose, we first need to create the eligible sets, and each set consists of several activities to be selected. Subsequently, a decision vector for a priority rule can be constructed in two steps: ranking and comparing. In the ranking process, the rule to be characterized is used to rank the activities in each set. While in the comparing process, the rankings given by the current rule are compared with the reference rule. The decision of the rule in each set is denoted as the rank of the reference rule assigned to the activity receiving the highest rank by the rule. Once the decision vector of a rule is constructed, its Euclidean distances to the vector of all rules in a pre-configured database are computed. The fitness of the closest rule in the database is used as the estimated fitness of the new rule.

Simplified models. The simplified models approximate the fitness of the individuals by reducing the complexity of the original problem. This research will examine three strategies for designing simplified models for the RCPSP: (i) The complexity when calculating the makespan is reduced. (ii) The size of the instances used for training is reduced. (iii) The combination of (i) and (ii). The fitness of a priority rule can be calculated directly by the obtained results on the simplified models. Therefore, it does not need any extra data.

2.4 The impact of the training data and fitness functions

In this work, we investigate the performance and efficiency of GPHH with different training data and fitness evaluation functions, and they will be assessed by the performance of their output priority rules.

Training data. The full set of the J30 (Kolisch and Sprecher (1997)) and RG30 (Vanhoucke *et. al.* (2008)) dataset are used for training. We also generate two subsets of these datasets called subsets "J30_select" and "RG30_select". Moreover, a third subset is created by merging these two subsets (J30+RG30_select). These three subsets have the same values of the network topology and resource indicators as the original sets, but the size (i.e., number of instances) is much smaller.

Fitness functions. In this work, two types of fitness evaluation functions are considered:

1. The average objective function value

$$Fitness_{ind} = \frac{1}{T} \sum_{ins=1}^{T} Obj_{ins} \tag{1}$$

Where $Fitness_{ind}$ denotes the fitness value of an individual (priority rule) in the population, T is the number of project instances in the training set and Obj_{ins} is the objective value of project instance ins achieved by the individual.

2. The average deviation from the reference value

$$Fitness_{ind} = \frac{1}{T} \sum_{ins=1}^{T} \frac{Obj_{ins} - Ref_{ins}}{Ref_{ins}} \quad (2)$$

Where Ref_{ins} denotes the reference value of ins, obtained by any other method.

3 Experiment results

Duplicate removal. The experiment results of the duplicate removal indicate that it has no adverse effects on the performance and can significantly reduce the runtime (up to 50%). Therefore, we can rely on the duplicate removal technique to significantly reduce the runtime of the GP algorithm without losing any searching ability or performance.

Surrogate models. The results show that the simplified models, which focus on reducing the resource types and using small-sized projects as the training data, perform the best. Therefore, we used them to construct the combined models to reduce the complicity further. The combined simplified models can obtain similar results as the original model. However, only one-third of the runtime is required.

Training data and fitness functions. For the training data selection, the results show that using an extensive training set is inefficient, and a relatively small set can also design good rules. However, the diversity of the training data is still relevant. Therefore, the J30+RG30_select performs the best since it balances between diversity and efficiency in the best possible way. For the fitness function, on average, the fitness evaluation function of Eq. (2) with the best known-solution as the reference value performs the best. However, Eq. (1) also displays similar results. Given that the best known-solution is not always readily available, in such a case, using Eq. (2) with the CPM as the reference or using Eq. (1) are good alternatives since they show similar, albeit slightly worse results.

4 Conclusions

In this work, we aim to improve the efficiency of the GPHH for designing priority rules for the RCPSP. We designed a duplicate removal technique and two different types of surrogate models, and investigate the impact of the training data and fitness function selection. The experiments show that these two methods can significantly reduce the runtime without weakening the performance, and a well-sampled training set is sufficient for designing priority rules with a good performance. Moreover, although using a good reference value often results in better output, other evaluation criteria can also provide acceptable results.

References

Branke, J., S. Nguyen, C. Pickardt and M. Zhang, 2015, "Automated design of production scheduling heuristics: A review", *IEEE Transactions on Evolutionary Computation*, Vol. 20, pp. 110-124.

Burke, E., M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan and R. Qu, 2013, "Hyper-heuristics: A survey of the state of the art", *Journal of the Operational Research Society*, Vol. 64, pp. 1695-1724.

Kolisch R., S. Arno, 1997, "PSPLIB-a project scheduling problem library", *European journal of operational research*, Vol. 96, pp. 205-216.

Vanhoucke M., J. Coelho, D. Debels, B. Maenhout and L. Tavares, 2008, "An evaluation of the adequacy of project network generators with systematically sampled networks", *European Journal of Operational Research*, Vol. 187, pp. 511-524.

Koza, J. R., 1994, "Genetic programming as a means for programming computers by natural selection", *Statistics and computing*, Vol. 4(2), pp. 87-112.

A comparative analysis for bounding the project completion time distribution in stochastic project networks

Forough Vaseghi[1], Annelies Martens[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Gent 9000, Belgium
Forough.Vaseghi, Annelies.Martens @ugent.be
[2] Technology and Operations Management, Vlerick Business School, Gent 9000, Belgium
[3] UCL School of Management, University College London, London, United Kingdom
Mario.Vanhoucke@ugent.be

Keywords: Project Risk Analysis, Continuous Probability Distributions, Stochastic Activity Network.

1 Introduction

A project network consists of activities between which precedence relations exist. The duration of these activities is not deterministic, but varies due to the presence of risk and uncertainty. When the stochastic nature of activity durations is neglected, the expected project duration is underestimated. To solve this problem, stochastic activity networks (SAN) with variable activity durations are used to define the project completion time as a distribution rather than a deterministic value. A SAN can be defined as a directed acyclic graph-DAG, in which each of the activities' duration follows a pre-defined distribution. A DAG is a series-parallel graph (SP-graph) when its complexity index (CI) is equal to zero. This means that, to convert the graph to one node, we only need a sequence of convolution and product operators. Many project networks, however, have a CI higher than zero, meaning that they cannot be reduced to a single node using only series and parallel reductions (Sahner and Trivedi 1987). For these project networks, computing the exact makespan distribution is a #p-complete problem. As a results, many research efforts have focused on developing approximation methods to obtain lower and upper bounds for the makespan distribution, with a lower (upper) bound for the makespan distribution having a cdf which is higher than (lower than) or equal to the actual makespan cdf for each completion time t. The common idea in these studies is to transform the given network into a SP-graph for which the makespan distribution can be computed more easily. In SP graphs, if activity durations are stochastically independent, the project distribution can be obtained by creating the decomposition tree of the graph and taking the convolution and product of the distributions of the sons of a series nodes and parallel nodes, respectively. The goal of this study is to combine the tightest lower and upper bound methods such that the area between both bounds is minimized, meaning that the possible area in which the actual project completion distribution is situated is as low as possible. To this extent, the impact of the network complexity and structure on the quality of the considered bounding methods will be investigated for different types of activity duration distributions.

2 Literature review

Many methods have been used over the last decades to compute and evaluate the project completion time and makespan distribution of project, which can be classified into exact methods, alternative analytical approaches, simulation [Van Slyke (1963)], and PERT [Malcolm *et. al.* (1959)]. Some researchers have used exact methods to compute the project makespan distribution [Hagstrom (1990) & (Schmidt and Grossmann 2000)], but using these methods is not efficient considering the cost and time of the computation. This motivated many studies on developing analytical approximation methods. Since this study focuses on the comparison of lower and upper bounds, the remainder of this section will discuss these bounding methods in greater detail.

In approximation methods, the common and main idea is converting the network (DAG-directed acyclic graph) to a SP graph. Kleindorfer (1971) and Shogan (1977) provided lower

and upper bounds for the completion time distribution using recursive equations. Dodin (1985) used a duplication process (DP) to transform the original DAG into a SP graph, which provides an upper bound for project completion time. Spelde (1976) suggested considering all chains in the network and using pairwise disjoint chains for computing the upper and lower bounds respectively. Kleinoder (1982) proposed adding arcs and removing arcs to transform it into a SP graph to obtain upper and lower bounds respectively. Colajanni *et. al.* (2000) used a tree-like representation to decompose the original DAG into series or parallel subgraphs for an upper bound.

Ludwig *et. al.* (2001) have evaluated the methods by Kleindorfer, Dodin, and Spelde to bound or approximate the distribution function of the makespan of stochastic project networks for activity on arc (AOA) networks that are based on network transformations and compared each of these distributions to the project distribution obtained by simulation to find the best bound based on the average relative error and CPU time. In their experiments, they have considered factors like the number of supporting points, number of activities, and type of distribution (normal/uniform/triangle/gamma). They concluded that Spelde's heuristic procedure based on the Central Limit Theorem provides excellent estimates at virtually no computational expense. This approach is therefore a remarkable alternative to (computationally costly) simulation techniques which are mostly used in practice. Further, Colajanni *et. al.* (2000) have compared their approach named Tree-bound approach with the bounding methods by Shogan, Dodin, and Kleinoder for activity on node (AON) networks and compared each of these distributions to the exact project distribution based on the mean execution time. They claimed that none of the heuristics methods can guarantee the best result because the accuracy is deeply related to the graph topology and task time distribution.

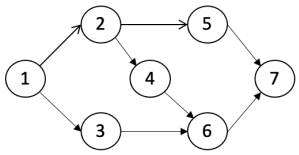
As mentioned, Ludwig *et. al.* (2001) and Colajanni *et. al.* (2000) used the simulated and exact makespan distributions to calculate the average relative error for each distribution obtained from different bounding methods to find the distribution which is the closest one to the actual distribution. However, they did not provide any information on the combination of best lower and upper bounds. In our analytical experiment, we will determine the best combination of LB and UB such that the area between LB and UB is as low as possible and the position of actual makespan distribution is known. Since we will compare the different lower bounds and upper bounds to each other, no simulation or exact determination of the makespan is required.

3 Experiments

In our experiments, we compare the bounding methods by Shogan, Dodin, Spelde, and the PERT technique to find the best and tightest upper and lower bounds for determining the project duration distribution of project networks with a CI higher than zero. For this comparison, the areas under the makespan cumulative distribution function (cdf) will be compared. For lower bound methods, the method with the lowest area under the distribution curve provides the tightest lower bound for the project makespan. Similarly, for upper bound methods, the method with the highest area under the distribution curve provides the tightest upper bound for the project duration. The advantage of this experiment is that simulation is not required to compare different bounding approaches. Another advantage is that the approximate position of the actual project makespan distribution can be situated between the lower and upper bound distributions. In the computational experiments, the bounding methods will be evaluated for different activity duration distributions (Beta/Weibull/Gamma), and they will be tested on a large dataset with varying values for the project network structure (SP, OS) and network complexity (CI).

4 Illustrative example and preliminary results

Let us consider the activity on node network in Fig 1 with a complexity index equal to 1 to show how the results of bounding methods can be different. In this example, the activity durations are assumed to be distributed lognormally and stochastically independent.



Activity	Mean	Variance
1	0	0
2	24	16
3	16	9
4	18	12.25
5	32	36
6	20	20.25
7	0	0

Fig. 1. An AON network with CI = 1 and list of activities with their means and variances.

As mentioned before, we compare the area under different LB and UB cdf curves. As shown in Table 1, for upper bound methods, the Dodin technique (Duplication Process) has the highest area, which corresponds to the lowest project completion ime on average (lowest stochastically). For lower bound methods, the Spelde (pairwise disjoint) technique has the lowest area, which corresponds to the highest project completion time on average (highest stochastically). So, these methods provide the tightest upper and lower bounds for this stochastic project network.

Table 1. Mean, variance and area under the CDF curve of makespan distribution for upper and lower bound methods.

Bounding methods	Mean	Variance	Area
UB			
Kleindorfer	63.718	42.097	136.275
Dodin (duplicate activity 2)	**63.145**	46.875	**136.854**
Spelde	63.695	36.906	136.304
Parallel composition of all chains	63.701	41.566	136.293
LB			
Kleindorfer	62.000	49.999	138.000
PERT	62.000	48.500	138.000
Spelde (pairwise disjoint)	**62.030**	48.607	**137.968**

In Fig 2, the worst and best UB and LB for makespan distribution and the areas between them in which the actual CDF is situated are shown. The size of this area (=area LB − area UB) is equal to 1.7242 and 1.1139 for the left and right graphs respectively. Therefore, selecting the best upper and lower bounds reduces the area where the actual makespan distribution can be situated. The goal of our further experiments is to investigate what the impact of the network structure and complexity is on the performance of the bounding methods. More precisely, we will examine which LB and UB generally result in the lowest area, and how project network characteristics such as the SP, OS and CI affect this performance.

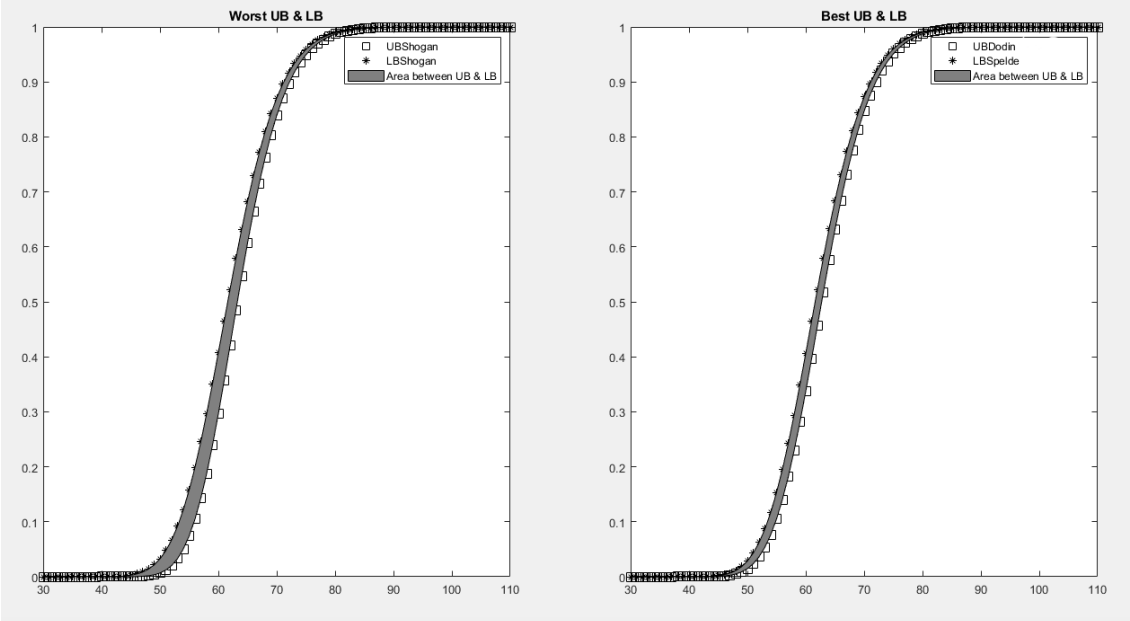


Fig. 2. Worst and best UB and LB for makespan distribution and the areas between them.

References

Colajanni M., F. Lo Presti and S. Tucci, 2000, "A hierarchical approach for bounding the completion time distribution of stochastic task graphs", *Performance Evaluation*, Vol. 41, pp. 1-22.

Dodin B., 1985, "Bounding the project completion time distribution in pert networks", *Operations research*, Vol. 33(4), pp. 862-881.

Hagstrom J., 1990, "Computational complexity of pert problems", *Networks*, Vol. 20, pp. 231-244.

Kleindorfer G., 1971, "Bounding distributions for a stochastic acyclic network", *Operations research*, Vol. 19, pp. 1586-1601.

Kleinoder W., 1982, "Stochastic analysis of parallel programs for hierarchical multiprocessor systems", PhD thesis, University of Erlangen, Nurnberg.

Ludwig R., R. Mohring and F. Stork, 2001, "A computational study on bounding the makespan distribution in stochastic project networksounding the completion time distribution of stochastic task graphs", *Annals of Operations Research*, Vol. 102, pp. 49-64.

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

Sahner R.A., K.S. Trivedi, 1987, "Performance and reliability analysis using directed acyclic graphs", *IEEE*, Vol. SE-13, pp. 1105-1114.

Schmidt C., I. Grossmann, 2000, "The exact overall time distribution of a project with uncertain task durations", *European Journal of Operational Research*, Vol. 126, pp. 614-636.

Shogan A., 1977, "Bounding distributions for a stochastic pert network", *Networks*, Vol. 7, pp. 359-381.

Spelde H., 1976, "Stochastische Netzplane und ihre Anwendung im Baubetrieb", PhD thesis, Rheinisch-Westfalische Technische Hochschule Aachen.

Van Slyke R.M., 1963, "Monte Carlo methods and the PERT problem", *Operations Research*, Vol. 11, pp. 839-860.

Human-centered interactions for project scheduling decision-aid in space industry

Hugo Chevroton[1], Cyril Briand[1], Philippe Truillet[2], Mélody Mailliez[3], Céline Lemercier[4]

[1] Université Toulouse 3 - Paul Sabatier, LAAS-CNRS
hchevroton,cbriand@laas.fr
[2] Université Toulouse 3 - Paul Sabatier, IRIT
philippe.truillet@irit.fr
[3] Université de Paris, LPS
melody.mailliez@parisdescartes.fr
[4] Université Toulouse 2 - CLLE-LTC
celine.lemercier@univ-tlse2.fr

Keywords: scheduling, decision-aid, human machine interaction, constraint programming.

Introduction

As modern production environments tend to be increasingly complex and stressful for production process supervisors (Khademi K. 2016), providing interactive decision support tools (DSTs) is seen as a relevant way to help humans better organize and monitor operations, in the face of production uncertainties. This organization is mainly related to task planning and scheduling, which are very complex functions involving many decision variables, a lot of non-trivial time and resource constraints, and a highly combinatorial search space (Lopez P. and Roubellat F. 2013). Beyond the management of this complexity, DSTs must help supervisors to cope with the occurrence of the (numerous) hazards that appear during the execution of the production plan (quality defects, supply disruptions, delays, breakdowns, etc.). Indeed, the real-time management of hazards is a major source of cognitive and emotional load for a supervisor (Robin Morris GW. 2004) as it implies adapting the schedule by frequently scheduling and re-scheduling production tasks so that the production plan remains consistent with the new constraints (Mailliez M. *et. al.* 2021). This is usually done in a hurry, under pressure from the hierarchy, also taking into account the stress of the operators who undergo the changes on the production floor.

The development of efficient DSTs is a major issue in the production field, the "Human-Machine" performances having to be taken into account in a global way. In (Peissner U. and Parasuraman R. 2013), the authors highlight the potential of DSTs and describe the requirements they should met. In the field of air traffic control, it is shown in (Metzger U. and Parasuraman R. 2005) that DSTs can allow a reduction of the mental load and an increase in the performance of the resolution of air conflicts, provided that they are reliable and easy to use. Several authors, see e.g. (Trentesaux D., P. Millot 2016), have alarmed that most DSTs for production supervision suffer from technocentrism: when a hazard occurs, the algorithms propose (and sometimes even impose) a solution to the problems faced by the decision makers, assuming that supervisors will be able to implement perfectly the solution, whatever the situation, respecting the expected response times. Consideration of the actual needs and capabilities of supervisors in the DST development process thus seems to be a prerequisite for the development of usable, accepted and effective tools.

The results presented in this paper synthesize the findings of an ongoing multidisciplinary project taking interest in human-centered design of DSTs related to production supervision. This work was conducted in partnership with a major French company specialized in space technologies, which provided our case study. The activity of this company

consists in assembling high-tech systems, each system being produced in a single unit. Interviews and observation campaigns with the supervisors of this company helped to understand the different decision-making processes and to extract the requirements that a DST should meet. Section 1 details the major requirements and specifies the key features that a DST should incorporate. Section 2 focuses on the decision aspects and the benefits that a constraint-programming (CP) approach could bring in the context of DSTs-supervisors interaction. Conclusions and future directions are drawn in the last Section.

1 Supervisor requirements and main DST features

The observed company builds systems composed of hundreds of high-tech components that fit together and connect to each other. Each system is specifically designed for one customer. Its assembly generates thousands of activities, the quality of each of which must be carefully controlled (because the system cannot be maintained once in service). The assembly activities are all performed manually and we distinguish between several operator skills depending on the nature of the activity. The accessibility of a specific part of the system depends on both its physical orientation (pan/tilt) and the number of operators already working on it The length of the project horizon commonly varies from 6 to 18 months. Expensive penalties have to be paid in case of tardiness. The schedule results from a cooperation between several supervisors, who play the roles of both project manager and chief-operator. Each one is in charge of the detailed schedule of a category of operators (e.g., mechanics, controllers, electricians) for each 8-hour shift, this schedule being dynamically changed to react to the numerous unpredictable events that occur during the process. There are also classical project managers having a long-term vision of the project portfolio, from both the delay and cost viewpoints. They allocate operators to projects along the time periods, according to the requirements of the supervisors, set-up milestones to be met, and manage the orders passed to the suppliers. Each category of project manager works with their own heterogeneous DSTs and (surprisingly) they are not numerically integrated together. Hence, the need for an integrated DST is strong so that the global consistency of the decisions can be checked easily all along the project. Project managers are more interested in finding one feasible solution than an optimal one. They clearly formulate the desire to keep the hand on the design of the schedule so that they can enforced decisions at any moment (e.g., to take non-modeled knowledge into account). The decision processes must therefore include humans in the loop and be consistent with their way of working. Hence, a DST should be kept focus on helping decision-makers to manage the problem complexity, quickly helping her/him to evaluate the consistency, the quality, and the consequences of her/his decisions. Autonomous decision making is only desired if the decision maker explicitly requests it (e.g., to automatically complete the construction of a schedule or to react quickly to unexpected disturbances).

Another major consequence of the human-in-the-loop requirement is to give support to help decision-makers in negotiating the constraints and finding a fair/efficient trade-off between their possibly conflicting objectives (Briand C. *et. al.* 2017). It should also support the whole project life cycle such that the various decision levels and horizons are integrated, which means that activity/resource/time decision variables can be disaggregated/aggregated into smaller/higher abstraction elements.

Another requirement is the management of disruptions that impact short-term scheduling decisions. DSTs obviously need to offer features that help supervisors to both prevent and react to disruptions. The trace and causes of the various decision changes should be saved so that decision-makers can explain to their hierarchy the path followed by the project and capitalize experience.

2 A CP-based approach

The wide variety of performance indicators, situations and decision-makers, as well as the multi-faceted nature of the production process, make the search for an optimal solution unnecessary. As mentioned before, the main objective is to provide decision makers with relevant information, presented in an understandable way, in order to facilitate the coordination and negotiation of decisions. The satisfiability of constraints is obviously a relevant property to be checked in real time. The ability to quickly compute good lower/upper bounds on well-targeted performance indicators is also of major interest to supervisors. In case of inconsistent constraints, providing explanations to decision makers and helping them to recover the desired satisfiability can also be of great help. Finally, the ability to quickly generate detailed feasible solutions is also useful.

The above features can be met in the constraint programming paradigm in which many researchers precisely focused for decades on designing algorithms able to efficiently prove constraint satisfiability, propagate time/resource constraints to refine variable domains, or provide minimal inconsistent constraint sets (see e.g. (Ceberio M. and Kreinovich V. 2014) for a survey) . Constraint propagation solvers now even advantageously rival against best top-ranking MILP solvers to quickly find good-quality schedules. Eventually, distributed constraint satisfaction techniques (Fioretto, F. *et. al.* 2018) can be used to negotiate constraints among a set of decision makers. The remainder of this section discusses a CP model for our company's project planning environment, specifically addressing how work, and resources can be disaggregated, i.e., how constraints can be settled to link the disaggregated/aggregated decision variables all together. It shows how the specific *bin_packing* global constraint can be advantageously used. Such constraint links the placement of sized/weighted items into bins and the capacity of the bins (Régin, J.C. and Rezgui, M. 2011).

The time horizon is assumed to be modeled as a set of period T of identical length, each period being indexed from 1 to $|T|$ (a period corresponds to a shift in our case study). Cumulative resources, each of them representing a set of disjunctive resources, are distinguished. K^* refers to as the set of all the disjunctive resources (i.e., the set of all operators or system states in our case study). $\mathcal{K}$ is the set of all the possible subsets of resources (e.g., a subset represents a category of operators or a specific state of the system). A subset K in $\mathcal{K}$ can be modeled as a cumulative resource, Q_K being its capacity. The project is defined by a set of tasks J. We refer to $\mathcal{A}$ as the set of precedence constraints, i.e., $(j \prec j') \Leftrightarrow (j, j') \in \mathcal{A}$. Each task $j \in J$ can be decomposed into p_j subtasks of duration equals to one period. A task has to be allocated to a set of periods (Dom_j is the index of periods where subtasks of j can be carried out) and a *set* of cumulative resources $\mathcal{K}_j \in \mathcal{K}$. Furthermore, each subtask of j must be assigned to a disjunctive resource belonging to K, for each $K \in \mathcal{K}_j$ and to a specific period of Dom_j.

The decision variable $x_{j,i}$ models the index of the period assigned to subtask i of task j ($j \in J$, $i \in [1, \ldots, p_j]$). The domain $Dom(x_{j,i})$ of $x_{j,i}$ is initialized to Dom_j. The value of variable $y_{k,t}$ is t ($t \in T$) if resource k ($k \in K^*$) is made available at period t, else 0 ($Dom(y_{k,t}) = \{0, t\}$). $w_{K,t}$ is the intensity of set of resources $K \in \mathcal{K}$) required at period t ($Dom(w_{K,t}) = [0, \ldots, Q_K]$). $w'_{K,t}$ is the capacity of resource K made available at period t ($Dom(w'_{K,t}) = [0, \ldots, Q_K]$). A dummy variable $w'_{K,0}$ is defined for unused resource units in order to keep the available capacity and the assigned capacity balanced for each K.

For all $K \in \mathcal{K}$, X^K is the array of all variables $x_{j,i}$ with $j \in J,\ K \in \mathcal{K}_j,\ i \in [1, \ldots, p_j]$. For all $K \in \mathcal{K}$, Y^K is the array of all variables $y_{k,t}$ with $k \in K, t \in T$. Finally, for all $K \in \mathcal{K}$, W_K (resp. W'_K) is the array of all variables $w_{K,t}$ (resp. $w'_{K,t}$), such t is in $\{0\} \cup T$. The CP model is presented below. Constraints (1) guarantee that two subtasks belonging to the same task are not executed in the same period. Constraints (2) model the precedence

constraints. Bin-packing constraints (3) models the link between x and w variables: x are the items to be assigned to bins w, where each w is associated with a set of resources and a specific period. Similarly, the link between y and w' variables is modeled by bin-packing constraints (4) : y are the items to be assigned to bins w', where each item $y_{k,t}$ with $k \in K$ has to be assigned either to bin $w'_{K,0}$ or $w'_{K,t}$. Constraints (5) ensure that, for each period, the number of assigned resources is higher than the resource consumption. All the above constraints are available in modern CP solvers with their specific propagators, which can be used together with the other solver features to check consistency, explain inconsistency, and find efficient bounds.

$$x_{j,i} < x_{j,i+1} \qquad \forall j \in J,\ \forall i \in [1, \ldots, p_j - 1] \tag{1}$$

$$x_{j,p_j} < x_{j',1} \qquad \forall (j, j') \in \mathcal{A} \tag{2}$$

$$bin\_packing(W_K, X_K) \qquad \forall K \in \mathcal{K} \tag{3}$$

$$bin\_packing(W'_K, Y_K) \qquad \forall K \in \mathcal{K} \tag{4}$$

$$w_{K,t} \leq w'_{K,t} \qquad \forall K \in \mathcal{K},\ \forall t \in T \tag{5}$$

Conclusion and perspectives

The major features that a DST should include in the context of project scheduling in space industry have been presented. A preliminary CP model has been provided in order to support the various human-in-the-loop decision processes. It integrates resource and task aggregation and will be updated soon to also deal with time abstraction. Concerning the interaction scenarios, the various way of using the algorithms that check consistency on this model, provide lower/upper bounds on performance indicators or recover consistency are currently under study. They will be implemented to assess the real usability, acceptability and efficiency of the proposed approach. Future research works will address the multi-agent nature of the decision problems, as well as the negotiation mechanisms they involved.

References

Briand C, Ngueveu SU, Šůcha P. Finding an optimal Nash equilibrium to the multi-agent project scheduling problem. J Sched. 2017 Oct;20(5):475-91.

Ceberio M. and Kreinovich V. Constraint Programming and Decision Making. Studies in Computational Intelligence 539, Springer 2014.

Fioretto, F., Pontelli, E. and Yeoh, W. Distributed constraint optimization problems and applications: A survey. Journal of Artificial Intelligence Research, 61, 623-698. 2018

Khademi K. Les processus cognitifs dans les activités d'ordonnancement en environnement incertain. *French Report*. Psychology. Université Toulouse le Mirail - Toulouse II,2016.

Lopez P, Roubellat F. Production Scheduling. John Wiley & Sons; 2013. 384p.

Mailliez M, Battaïa O, Roy RN. Scheduling and rescheduling operations using decision support systems: Insights from emotional influences on decision-making. Frontiers in Neuroergonomics.Vol. 02, 2021.

Metzger U, Parasuraman R. Automation in future air traffic management: effects of decision aid reliability on controller performance and mental workload. Hum Factors. 2005. 47(1):35-49.

Peissner M, Hipp C. Potenziale der Mensch-Technik-Interaktion für die effiziente und vernetzte Produktion von morgen. Fraunhofer-Verlag; 2013. 78 p.

Régin, J.C. and Rezgui, M. Discussion about constraint programming bin packing models. Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence. 2011.

Robin Morris GW. The Cognitive Psychology of Planning. Taylor & Francis; 2004. 256 p.

Trentesaux D, Millot P. A human-centred design to break the myth of the "magic human" in intelligent manufacturing systems. In: Service Orientation in Holonic and Multi-Agent Manufacturing. Cham: Springer International Publishing; 2016. p. 103-13.

On the relevance of the makespan service level for the flexible job shop scheduling problem under uncertainty

Mario Flores-Gómez[1], Valeria Borodin[1], Stéphane Dauzère-Pérès[1,2]

[1] Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, F-42023 Saint-Etienne, France
{mario.flores, valeria.borodin, dauzere-peres}@emse.fr
[2] Department of Accounting and Operations Management, BI Norwegian Business School, Oslo, Norway

Keywords: Stochastic scheduling, Service level, Random processing times, Monte Carlo sampling, Tabu search.

1 Introduction

The Flexible Job-shop Scheduling Problem (FJSP) is a generalization of the classical Job-shop Scheduling Problem (JSP), that includes a set of operations partitioned into a set of jobs. The operations in a job have to be executed in a specific order (routing) on a set of available machines. A machine can only perform one operation at a time, and each operation cannot be interrupted. An operation can be executed by any machine in a given subset specific to that operation (eligible machines). The processing times can be machine-dependent. Finding a solution to this problem is to determine both an assignment of operations to machines, and the sequence the operations must follow on each machine, while respecting the routing of every job. In this paper, the makespan (C_{max}) is considered as a required specification, which corresponds to the maximal completion time of all operations.

The Stochastic FJSP (SFJSP), where the processing times of operations in the machines are no longer supposed to be known, has been considerably less studied than its deterministic version. Daniels and Carrillo (1997) define, for a single machine scheduling problem, a β-robust schedule to model the likeliness of the total flow time across all jobs to be no worse than a given target level. This notion is extended by Beck and Wilson (2007) to deal with the stochastic JSP. They propose a number of techniques combining Monte Carlo simulation with solution approaches dedicated to the deterministic JSP (e.g. constraint programming or tabu search). In continuation of our work in (Gómez *et. al.* 2021), we extend the study on the relevance of the notion of *makespan service level* as defined in (Dauzère-Pérès *et. al.* 2008). In Section 2, the notion of *makespan service level* is formalized and our solution approach is presented, numerical results are provided and discussed in Section 3. Conclusions and perspectives are given in Section 4.

2 Problem statement and solution approach

In this paper, processing times are considered as random variables. As a consequence, the makespan of a sequence is also a random variable. Our goal is to maximize the probability that the makespan is lower than or equal to a given threshold T. This probability, denoted by $\alpha(S,T)$, is known as the *makespan service level* associated to sequence S and threshold T:

$$\alpha(S,T) = \mathbb{P}(C_{max}(S) \leq T)$$

To determine the service level of a sequence, a set of scenarios Ω is generated and an algorithm based on Monte Carlo simulation is implemented, as described in (Gómez

et. al. 2021). The service level is denoted $\alpha(S, T, \Omega)$. The proposed solution method is based on a competitive tabu search approach (Dauzère-Pérès and Paulli 1997), including a Monte Carlo sampling procedure to represent and deal with uncertainties. The results of computational experiments as well as the outline of the proposed method can be found in (Gómez *et. al.* 2021). Note that minimizing the makespan does not mean maximizing the service level.

3 Relevance of *makespan service level*

In this paper, we want to show the relevance of the *makespan service level* with other numerical results than the ones in (Gómez *et. al.* 2021). The set of scenarios Ω is randomly generated according to the beta probability distribution (Marshall and Olkin 2007), by extending the FJSP benchmark instances from Hurink *et. al.* (1994). In this paper, $|\Omega|$ is set to 1,000. The processing times in the benchmark instances are set as the mean parameter μ for every random variable used to generate the scenarios in Ω. The standard deviation σ is expressed as a fraction of μ. The support $[c, d] = [\mu - 0.2\mu, \mu + 0.8\mu]$ for every random variable is also defined using μ. Let S_μ be the sequence found by the tabu search in Dauzère-Pérès and Paulli (1997) for mean values of random processing times. Only the processing times of the operations in one job of the instance are considered as random variables to generate Ω.

Table 1. Characteristics of instances mt06, mt10 and mt20 with three lines per instance type (edata, rdata, vdata) for each level of flexibility.

Instance	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{J}\rvert$	Operations per job	Average $\lvert\mathcal{M}_i\rvert$	$C_{max}(S_\mu)$
mt06	6	6	6	1.15	55
				2	47
				3	47
mt10	10	10	10	1.15	881
				2	686
				5	655
mt20	5	20	5	1.15	1,091
				2	1,023
				2.5	1,023

3.1 Solving the FJSP for the worst-case scenario

A common practice in stochastic optimization is to use the worst-case scenario as reference to bound the problem. However, this scenario may not correspond to any of the scenarios in Ω, although it is a possible scenario because each scenario $\omega \in \Omega$ is generated according to independent random variables (Birge and Louveaux 2011).

The theoretical and the empirical worst-case scenarios are considered. The theoretical worst-case scenario ω^{th}_{worst} (resp. ω^{th}_{best}) is based on the maximum interval where the random processing times take the value of parameter d (resp. c). In the empirical worst-case scenario ω^{emp}_{worst} (resp. ω^{emp}_{best}), the value for every random processing time is the largest (resp. smallest) realisation in Ω. For each scenario $\omega \in \Omega$ and for every operation i and every machine j (ω_{ij}), we have $\omega_{ij} \leq \omega^{emp}_{worst,i,j} \leq \omega^{th}_{worst,i,j}$ and $\omega^{th}_{best,i,j} \leq \omega^{emp}_{best,i,j} \leq \omega_{ij}$. Let:

- S^{emp}_{worst} (resp. S^{emp}_{best}) be the sequence determined using the tabu search in (Dauzère-Pérès and Paulli 1997), where the makespan is minimized for the values of ω^{emp}_{worst} (resp. ω^{emp}_{best}),
- S^* be the sequence determined using the tabu search in (Gómez *et. al.* 2021) and Ω,
- $C_{max}(S, \omega)$ be the makespan of sequence S with the processing times in scenario ω.

Table 2. Makespan and *makespan service level* for every job from instance mt10-vdata: $T = 655$, $\alpha = \alpha(S, T, \Omega)$

Job	$S_\mu, \omega^{emp}_{worst}$		$(S^{emp}_{best}, \omega^{emp}_{best}), (S^{emp}_{worst}, \omega^{emp}_{best})$		S^*, μ	
	$C_{max}(S, \omega)$	α	$C_{max}(S, \omega)$	α	$C_{max}(S, \omega)$	α
1	744.20	0.932	655.00, 655.93	0.050, 1	655	1
2	743.72	0.935	655.00, 655.20	0.001, 1	655	1
3	768.69	0.979	655.00, 723.74	0.805, 0	655	1
4	880.86	0.905	597.00, 834.93	0.958, 0	655	0.978
5	702.62	0.856	655.00, 655.00	0.473, 1	655	1
6	724.30	0.985	655.00, 656.00	0.002, 1	655	0.998
7	690.87	0.953	655.00, 655.56	0, 1	655	1
8	778.63	0.864	655.00, 685.93	0, 0	655	0.898
9	841.38	0.863	655.00, 748.84	0, 0	655	0.956
10	751.98	0.919	655.00, 691.99	0.761, 0	655	0.989

For the optimization problem described in Section 2, $\alpha(S^{emp}_{worst}, T, \Omega) = 1$ means that the approach in Gómez *et. al.* (2021) is not necessary. However, as shown in Table 2, this is not very common since $\alpha(S^{emp}_{worst}, T, \Omega) \leq \alpha(S^*, T, \Omega)$ for all jobs. Therefore and as expected, solving ω^{emp}_{worst} does not guarantee that a sequence with a good *makespan service level* is determined. However, when the processing times of jobs 1, 2, 5, 6 or 7 are stochastic, a sequence is found that maximizes the *makespan service level* and with a very competitive makespan. This could possibly mean that, even for the values in scenario ω^{emp}_{worst}, the operations in those jobs are not critical for the computation of the makespan subject to $C_{max}(S^{emp}_{worst}, \omega) \leq T, \forall \omega \in \Omega$. Further experimentation is required.

3.2 Statistical evaluation of FJSP with best-case-scenario

Let S^{th}_{best} be the sequence determined by the tabu search in (Dauzère-Pérès and Paulli 1997), where the makespan for ω^{th}_{best} is minimized. The makespan for each sequence is computed using the processing times in its respective reference scenario.

As shown in Table 3, since $\forall i$, $\forall j$,

$$\omega^{th}_{best,i,j} \leq \omega^{emp}_{best,i,j} \leq \omega_\mu$$

the average values of $C_{max}(S^{th}_{best})$ and $C_{max}(S^{emp}_{best})$ are, in general, smaller than $C_{max}(S_\mu)$. However, the dispersion of random variables and the fact that processing times that are critical to the determination of the makespan are undervalued, lead to very under-performing solutions as shown by the average values of $\alpha(S^{emp}_{best}, T, \Omega)$ and $\alpha(S^{th}_{best}, T, \Omega)$. Further experimentation is required.

4 Conclusions

We are currently improving the modeling of uncertain processing times by characterizing industrial data. Our goal is to propose new benchmark instances based on real-life

Table 3. Average values of makespan and *makespan service level* for all instances. Three lines per instance type (edata, rdata, vdata) for each level of flexibility: $T = C_{max}(S_\mu)$ for each instance, $\alpha = \alpha(S, T, \Omega)$

Instance	$S^{th}_{best}, \omega^{th}_{best}$		$S^{emp}_{best}, \omega^{emp}_{best}$		S_μ		S^*
	$C_{max}(S, \omega)$	α	$C_{max}(S, \omega)$	α	$C_{max}(S)$	α	α
mt06	53.17	0.84	53.17	0.69	55	0.88	0.91
	46.50	0.61	46.51	0.21	47	0.88	0.92
	45.67	0.23	45.67	0.36	47	0.93	0.99
mt10	886.43	0.42	881.78	0.21	881	0.85	0.87
	684.00	0.05	684.94	0.03	686	0.80	0.89
	649.20	0.26	649.20	0.30	655	0.96	0.98
mt20	1,086.75	0.73	1,086.01	0.68	1,091	0.76	0.76
	1,020.00	0.09	1,019.00	0.36	1,023	0.71	0.71
	1,016.00	0.20	1,016.00	0.30	1,023	0.64	0.72

observations. Moreover, a set of experiments has been designed to analyze the impact of the cardinality of Ω with respect to the number of random variables in each instance, and to exploit the information provided by $S^{emp}_{best}, S^{th}_{best}, S^{emp}_{worst}$ and S^{th}_{worst}.

We are also investigating more efficient solution approaches that will not only rely on Monte Carlo simulation. In particular, considering dominance relationships between scenarios seems a promising research avenue.

Acknowledgements

This work was partly funded by the French Public Authorities through the Nano 2022 program, which is part of IPCEI (Important Project of Common European Interest).

References

Beck J.C. and N. Wilson, 2007, "Proactive Algorithms for Job Shop Scheduling with Probabilistic Durations", *Journal of Artificial Intelligence Research*, Vol. 28, pp. 183-232.

Birge, J.R. and F. Louveaux, 2011, "The value of Information and the Stochastic Solution", In *Introduction to stochastic programming*, pp. 163-180.

Daniels R.L. and J.E. Carrillo, 1997, "β-Robust scheduling for single-machines systems with uncertain processing times", *IIE Transactions*, Vol. 29, pp. 977-985.

Dauzère-Pérès S. and J. Paulli, 1997, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search", *Annals of Operations Research*, Vol. 70, pp. 281-306.

Dauzère-Pérès S., P. Castagliola and C. Lahlou, 2008, "Service Level in Scheduling", John Wiley & Sons, pp. 99-121.

Flores Gómez M., V. Borodin and S. Dauzère-Pérès, 2021, "A Monte Carlo based method to maximize the service level on the makespan in the stochastic flexible job-shop scheduling problem", *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)* pp. 2072–2077.

Hurink J., B. Jurisch and M. Thole, 1994, "Tabu search for the job-shop scheduling problem with multi-purpose machines", *OR Spektrum*, Vol.15, pp. 205-215.

Marshall A.W. and I. Olkin, 2007, "Gamma and Beta Functions", In *Life Distributions*, pp. 717-727.

Generation and Characterization of Real-World Instances for the Flexible Resource-Constrained Multi-Project Scheduling Problem

Hendrik Weber, Robert Brachmann and Rainer Kolisch

Technical University of Munich, Germany
hendrik.weber@tum.de, r.brachmann@tum.de, rainer.kolisch@tum.de

Keywords: resource-constrained scheduling, multi-project scheduling, instance generation

1 Introduction

The Resource-Constrained Project Scheduling Problem (RCPSP) is a well-known and extensively studied scheduling problem. A recent extension is the Flexible Resource-Constrained Multi-Project Scheduling Problem (FRCMPSP) where a portfolio of projects has to be scheduled subject to precedence constraints, scarce resources and resource profile constraints, such that the total weighted lateness is minimized. In this paper, we present a set of instances for the FRCMPSP based on data from a mid-sized ETO company. We systematically varied these instances in terms of instance size, resource availability, due date tightness, and resource flexibility leading to a set of 360 instances with up to 114 projects and 1,841 activities. The generated instances are characterized according to instance indicators from the literature. We develop a mixed-integer model and investigate its performance using a commercial solver.

2 Literature

The relevant literature for our study stems from two main streams of research. Firstly (resource-constrained) project scheduling and its variants and secondly available benchmark instance sets in the field of project scheduling.

The RCPSP, first studied by Pritsker *et al.* (1969), belongs as a generalization of the job shop scheduling problem to the class of NP-hard optimization problems (Blazewicz *et al.* 1983). Surveys on the RCPSP and its variants are provided by Hartmann and Briskorn (2010) and Hartmann and Briskorn (2021). In contrast to the RCPSP, where a single project is scheduled at a time, the RCMPSP deals with multiple projects simultaneously. The FRCPSP was first proposed by Kolisch *et al.* (2003) in the context of a real-world application in pharmaceutical research. It introduces flexible resource demand profiles to the RCPSP, which allow varying resource demand over the processing periods of an activity. Kis (2005) presents a MIP-formulation for the FRCPSP. Fündeling (2006) and Fündeling and Trautmann (2010) study variants of the FRCPSP with discrete and exact allocation of resource demand. Naber and Kolisch (2014) investigate the performance of four different mixed-integer formulations using solution quality and computation time as measures. The FRCMPSP, introduced by Brachmann and Kolisch (2021), is a more recent extension in the context of the RCPSP. It incorporates key features from two generalizations of the RCPSP, namley the adaptation of flexible demand profiles of the FRCPSP as well as addressing the multi-project environment of the RCMPSP.

With the multitude of proposed RCPSP-extensions also comes the requirement for suitable problem instances that serve as standardized benchmarks for researchers to test and evaluate their solution approaches. However, real-world problem data is scarce. The

first set of standardized test instances that gained popularity in the research community was proposed by Patterson (1984). In subsequent decades, reasearchers strived for problem sets that cover as much of the feature space as possible, leading to a variety of proposed instance generators and libraries, see Van Eynde and Vanhoucke (2020). However, these proposed instances all share the characteristic that they are artificially generated. In this research we provide an instance set which builds up on real-world instances, therefore allowing researchers to develop and test their approaches on realistic problems.

3 Problem Description

The FRCMPSP consists of two decisions that are to be made simultaneously in a multi-project and multi-resource environment. First, for each project of the portfolio, a schedule, including the start and finish times of the activities, must be determined subject to precedence relations of finish-start type, project arrival and due date, and feasible processing periods. Second, for each activity, an allocation of a given resource requirement between its start and finish, a so-called resource profile, has to be decided, taking into account resource capacity as well as upper and lower allocation bounds. The two decisions of the FRCMPSP are interdependent, as the schedule and the resource profile determine the start, finish, and duration of an activity. The objective is to minimize the total weighted lateness of projects, which permits positive lateness (tardiness) and negative lateness (earliness). Due date deviation is penalized with a linear function using individual penalty values for earliness and tardiness.

An application area for the FRCMPSP is aggregate production planning in Engineer-to-Order (ETO) environments (Brachmann and Kolisch 2021). In our example, we consider a mid-sized ETO company that designs, engineers, and manufactures customer-specific packaging machines for the pharmaceutical industry. Every order is planned and processed as a unique project consisting of various activities and requiring different resources. Each activity depicts a dedicated part of the project that requires one single resource. In the provided problem data, there are a total of 11 resources and 12 activity types, where each activity type requires exactly one resource.

4 Instance Generation and Characterization

The data for generating instances was provided by the ETO company, stemming from three different sources and being comprised of three different data sets: project network data, order data, and resource data. 360 instances were generated using the parameters presented in Table 1: instance size (number of projects, number of activities), resource availability, due date tightness, and resource flexibility. The small instances contain 38 projects with 624 activities and comprise the first third of all projects. The medium-sized instances comprise the first two-thirds of all projects, including 76 projects and 1,221 activities. Consequently, they also include the projects of the small instances. The large set of instances cover all 114 projects and 1,841 activities. Instances of the same size share the same super-project network and project sub-networks. However, they differ concerning all other instance parameters.

The resource availability ranges between 0.2 and 0.7. The parameter represents a percentage increase from the average resource capacity of the renewable resources. Five different due date tightness levels are used to systematically vary the due dates based on the arrival dates of projects and their respective critical path durations. E.g., for a due date tightness of 0.2, the project's due date d_p is to set to $d_p = \lceil a_p + (1 + 0.2) \cdot CP_p \rceil$, where a_p and CP_p denote the arrival date and the critical path duration of project p. Resource flex-

Table 1. Parameter levels

Parameter	Values
Instance size (projects (activities))	38 (624), 76 (1,221), 114 (1,841)
Resource availability	0.7, 0.6, 0.5, 0.4, 0.3, 0.2
Due date tightness	0.6, 0.5, 0.4, 0.3, 0.2
Resource flexibility	[0.10, 0.70], [0.15, 0.65], [0.20, 0.60], [0.25, 0.55]

ibility determines for each activity j and each resource r the minimum and the maximum of the resource allocation $(\underline{u}_{rj}, \overline{u}_{rj})$ as a percentage of the resource requirement wc_{rj} of the activity. That is, for a resource flexibility of [0.1, 0.7], for each activity and each resource $\underline{u}_{rj}/wc_{rj} = 0.1$ and $\overline{u}_{rj}/wc_{rj} = 0.7$ holds.

In order to position the instances within the existing literature on multi-project scheduling, we are characterizing them using well-known measures from the literature. These are the coefficient of network complexity, the order strength, the serial-parallel-indicator, the activity distribution, the resource sharing ratio, the normalized average resource loading factor, the resource strength, the utilization factor, and the variance of the utilization factor.

Figure 1 displays all 360 generated instances in a two-dimensional space parameter plot for normalized average resource loading factor (NARLF), utilization factor (UF) and variance of utilization factor (σ^2_{UF}).

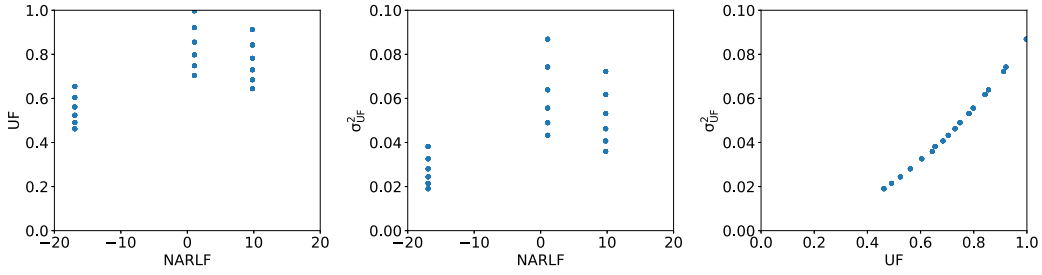


Fig. 1. Two-dimensional parameter space plots for generated instances in line with Van Eynde and Vanhoucke (2020)

5 Computational Study and Results

The problem is modelled as a mixed-integer program and solved with GUROBI 9.1 with a time limit of 7,200 seconds per problem instance. The computational results reported in Table 2 use the following five performance measures: The percentage of instances that are solved to optimality (Optimality), the percentage of instances for which a feasible solution could be obtained (Solved), the total weighted lateness (Objective value), the percentage gap between the best solution and the best bound (Gap), and the CPU time (CPU time). The performance measures are calculated as mean values over all instances of a subset of instances. The first column ("Average") aggregates the columns (4-7) that provide the measures for different levels of resource flexibility, where level 1 is the most flexible and level 4 is the least flexible setting.

Table 2. Computational results grouped by instance size

Size	Measure	Resource flexibility				
		Average	Level 1 [0.1, 0.7]	Level 2 [0.15, 0.65]	Level 3 [0.2, 0.6]	Level 4 [0.25, 0.55]
624	Optimality (%)	92.50	100.00	100.00	83.33	86.67
	Solved (%)	100.00	100.00	100.00	100.00	100.00
	Objective value	4.35	1.57	2.57	4.73	8.53
	Gap (%)	1.47	0.00	0.00	2.29	3.57
	CPU time	716.39	21.36	230.04	1361.37	1252.78
1221	Optimality (%)	68.33	86.67	76.67	60.00	50.00
	Solved (%)	100.00	100.00	100.00	100.00	100.00
	Objective value	27.93	20.17	24.00	29.90	37.63
	Gap (%)	6.61	3.25	4.34	8.24	10.58
	CPU time	2635.11	1469.43	2079.91	3213.30	3777.82
1841	Optimality (%)	31.67	40.00	40.00	33.33	13.33
	Solved (%)	87.50	93.33	90.00	90.00	76.67
	Objective value	69.60	64.25	65.37	81.00	67.70
	Gap (%)	14.77	15.05	12.04	16.47	15.64
	CPU time	5492.80	4850.44	5281.24	5393.97	6445.54

References

Blazewicz J., J. K. Lenstra, and A. H. G. Rinnooy Kan, 1983, "Scheduling subject to resource constraints: classification and complexity", *Discrete Applied Mathematics*, Vol. 5(1), pp. 11-24.

Brachmann R., R. Kolisch, 2021, "The impact of flexibility on engineer-to-order production planning", *International Journal of Production Economics*, Vol. 239.

Fündeling C.-U., 2006, "Ressourcenbeschränkte Projektplanung bei vorgegebenen Arbeitsvolumina", *Produktion und Logistik. DUV Deutscher Universitäts-Verlag.*

Fündeling C.-U., N. Trautmann, 2010, "A priority-rule method for project scheduling with work-content constraints", *European Journal of Operational Research*, Vol. 203(3), pp. 568-574.

Hartmann S., D. Briskorn, 2010, "A survey of variants and extensions of the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 207(1), pp. 1-14.

Hartmann S., D. Briskorn, 2021, "An updated survey of variants and extensions of the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 297(1), pp. 1-14.

Kis T., 2005, "A branch-and-cut algorithm for scheduling of projects with variable-intensity activities", *Mathematical Programming*, Vol. 103(3), pp. 515-539.

Kolisch R., K. Meyer, R. Mohr, and C. Schwindt, 2003, "Ablaufplanung für die Leitstrukturoptimierung in der Pharmaforschung", *Zeitschrift für Betriebswirtschaft*, Vol. 73, pp. 825-848.

Naber A., R. Kolisch, 2014, "MIP models for resource-constrained project scheduling with flexible resource profiles", *European Journal of Operational Research*, Vol. 239(2), pp. 335-348.

Patterson J., 1984, "A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem", *Management Science*, Vol. 30(7), pp. 854-867.

Pritsker A., L. Waiters, and P. Wolfe, 1969, "Multiproject scheduling with limited resources: A zero-one programming approach", *Management Science*, Vol. 16(1), pp. 93-108.

Van Eynde R., M. Vanhoucke, 2020, "Resource-constrained multi-project scheduling: benchmark datasets and decoupled scheduling", *Journal of Scheduling*, Vol. 23(3), pp. 301-325.

Budget allocation in risk prevention and risk protection considering risk interdependency

Xin Guan[1], Tom Servranckx[1], Mario Vanhoucke[1,2,3]

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

Keywords: budget allocation, risk prevention, risk protection, risk interdependency, project risk management

1 Introduction

Various risks may be present in each stage of the project life cycle, and the occurrence of each risk will lead to an adverse effect on project objectives, such as project delay, budget overrun, and even the failure of projects. To reduce this undesired effect and to ensure the successful project completion of project, there is a high demand for effective risk response strategies to mitigate the underlying project risks. In our study, any event that typified with an occurrence probability and adverse impact is regarded as a risk, such as the material shortage, blockage of equipment, etc.

So far, some models of risk response have been developed, such as the zonal-based models (Miller and Lessard 2001, Elkjaer and Felding 1999), decision tree-based models (Dey 2012), optimization-based models (Ben-David and Raz 2001, Sato and Hirao 2013) and the integrated models (Sherali *et. al.* 2011, Sherali *et. al.* 2008, Zhang and Guan 2018), which have made significant contributions for developing effective response plans. However, the main focus of the existing studies is on how to select appropriate strategies from a pool of alternative strategies within a limited budget, where the cost and effect of each strategy are fixed and known. In reality, the effects and costs of strategies are unknown, and the effect of each response strategy usually depends on the invested effort. Thus, determining the budget amount invested for each strategy and allocating budget among strategies are of great importance for the effective response to risks and the reasonable use of project budget. Despite the importance of budget allocation for risk response, few attempts have been made to solve this problem. Our study aims at proposing a method for reasonably allocating budget among risk response strategies to mitigate the project risks.

Given that risk is measured by its expected loss, which is multiplying *risk probability* and *risk loss*, the response strategies can be divided into two types (Fan *et. al.* 2008, Kuo *et. al.* 2019). *Risk prevention* is used to reduce the occurrence probability of the risk and implemented before risk occurs, while *risk protection* is used to reduce the impact on the project objectives when the risk does happen. Since the implementation of these two strategies may yield different risk reductions as well as incur different costs, trade-offs should be made between these two strategies. In addition, project risks are interdependent. The occurrence of one risk may increase the chance of an other risk, and the total impact will be aggravated if multiple risks occur simultaneously. Due to this interdependency, strategies used for dealing with one risk may also have an impact on the dependent risks. Thus, the interdependencies should be taken into account when making risk response decisions.

This study includes the interdependencies into the risk response model and provides analytical solutions, based on which the optimal budget allocation are obtained and the effects of risk and risk response characteristics, and the interdependencies are investigated.

2 Problem formulation

2.1 Problem statement

Suppose one project exposes two risks (i, $i = \{1, 2\}$) which interact with each other. Each risk has an initial probability P_i^0 and loss L_i^0, leading to a risk level R_i^0. Thus, the initial total risk level of the project is R^0 ($= R_1^0 + R_2^0$). To ensure the successful completion of the project, a target/acceptable risk level R ($R = \mu R^0$, $0 < \mu < 1$) is predefined by the project manager, and a smaller R implies a stricter response requirement. Thus, the goal of risk response is to reduce the total risk level from R^0 to R.

To this end, a prevention strategy to decrease the risk probability and a protection strategy to mitigate the risk loss are developed for each risk. The ex-post risk probability and loss are denoted as P_i and L_i for each risk, such that $R = \sum_i P_i L_i$. Given that the risk cannot be removed completely unless the project activity or scope is changed, minimal levels of risk probability and loss are set, denoted as $\underline{P_i}$ and $\underline{L_i}$, respectively, $\underline{P_i} \leq P_i \leq P_i^0$ and $\underline{L_i} \leq L_i \leq L_i^0$.

Several possible solutions exist to achieve the response goal: focus on a single risk or both risks as well as use risk prevention, protection or a combination for each risk. However, these possible solutions may incur different costs due to the risk interrelation and the different effects of response strategies. Thus, the purpose of risk response is to satisfy the response requirement at a minimal cost.

2.2 Model development

Based on the above analysis, the budget allocation problem (BAP) introduced by Guan *et. al.* (2021) can be formulated as a bi-level optimization problem. The lower-level problem is to allocate budget among risk prevention and risk protection for each risk given the response requirement for each risk (denoted as μ_i). The upper-level problem determines μ_i by allocating the response requirement μ between the two risks. The bi-level optimization model is developed in the following.

$$\textbf{Upper-level} \quad \min_{(\mu_i)} \quad Z = Z_1 + Z_2 = q_1 + r_1 + q_2 + r_2 \tag{1}$$

$$s.t. \quad P_1 L_1 + P_2 L_2 = \mu(R_1^0 + R_2^0) \tag{2}$$

$$\textbf{Lower-level} \quad \min_{(q_i, r_i)} \quad Z_i = q_i + r_i \quad \forall i \tag{3}$$

$$s.t. \quad P_i' L_i' = \mu_i R_i^0 \quad \forall i \tag{4}$$

Lower-level. Objective function (3) is to minimize the total response cost for each risk. q_i and r_i are the budget amounts allocated to prevention and protection strategies, which depends on the cost-effect relations of risk response strategies. Constraint (2) ensures the response requirement for each risk without considering risk interrelation to be satisfied, where P_i' and L_i' are the ex-post probability and loss of risk i after implementing prevention and protection strategies of risk i. Since the different ranges of probability and loss make it difficult to compare between risk prevention and protection, auxiliary variables $m_i (= \frac{P_i'}{P_0^i})$ and $n_i (= \frac{L_i'}{L_0^i})$ are introduced. Accordingly, constraint (4) transforms into Eq.(5):

$$Eq.(4) \quad \Rightarrow \quad m_i n_i = \mu_i, \quad \forall i \tag{5}$$

Following the decision procedure proposed in Guan *et. al.* (2021), analytical solutions ($m_i^\star$, $n_i^\star$) of the lower-level problem can be obtained.

Upper-level. Objective function (1) is to minimize the total response costs and constraint (2) forces the response requirement for the total risk level to be satisfied. The ex-post probability and loss of each risk (P_i, L_i) should be calculated based on the risk interdependencies. In this study, we classify the risk interdependencies into probability dependency and impact dependency, representing that the changes in one risk in terms of probability or impact will affect the probability or impact of the dependent risk, respectively. The following will take the probability dependency as an example to illustrate how risk interdependency affects the risk reductions, as shown in Figure 1.

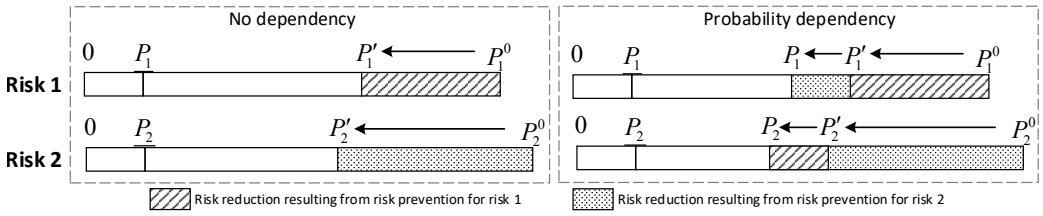


Fig. 1. Example of probability interdependency

Suppose risk prevention strategies are used for reducing both risk 1 and risk 2. If no interdependency exists, $P_i = P_i' = m_i P_i^0, i = 1, 2$ and $L_i = L_i^0$ where the reduction $P_i^0 \to P_i'$ is due to the prevention strategy for risk i. Thus, the ex-post total risk level is $R = \sum_i m_i P_i^0 L_i^0$. If probability dependency exists between the two risks, the prevention strategy for one risk also leads an reduction in the other risk. Thus, besides the reduction $P_i^0 \to P_i'$, a further reduction $P_i' \to P_i$ has been led by the prevention strategy for the other risk. Such that, a lower ex-post total risk level will be obtained when probability dependency exists. We calculate the ex-post probability of each risk as Eq. (6), where θ denotes the interdependency degree.

$$P_i = (m_i - \theta + m_j\theta)P_i^0, \quad j = 1, 2 \quad \& \quad j \neq i \tag{6}$$

When impact dependency exists, the ex-post loss of each risk can be computed in a similar way. Thus, constraint (2) can be reformulated as following:

$$Eq.(2) \quad \Rightarrow \quad \frac{R_1^0}{R_2^0} = \frac{\mu - B}{A - \mu} \tag{7}$$

where $A = (m_1 - \theta + m_2\theta)n_1$ and $B = (m_2 - \theta + m_1\theta)n_2$ with probability dependency, $A = m_1(n_1 - \theta + n_2\theta)$ and $B = m_2(n_2 - \theta + n_1\theta)$ when impact dependency exists, and $A = (m_1 - \theta + m_2\theta)(n_1 - \theta + n_2\theta)$ and $B = (m_2 - \theta + m_1\theta)(n_2 - \theta + n_1\theta)$ when risks are dependent in both probability and impact.

3 Results

In order to obtain the results, we conduct some numerical experiments that can be extended with computational experiments in the future. We consider a linear relation and a non-linear relation between the costs and effects of the risk response strategies. Concerning the results in the **linear** setting, we observe that the total response cost will reach its minimal value at the bound point of the response requirement for each risk. This implies that if mitigating one risk can satisfy the total response requirement, the

best decision is to deal with only one risk. Otherwise, both risks should be mitigated, where one risk should be reduced to its minimal level, and the remaining risk reduction should be controlled by handling the other risk. After deciding which risk to be reduced, the best response strategy for each risk depends on whether the risk can be reduced by a single strategy. The results with **non-linear** relations suggest that both risks should be reduced in some situations even if the response goal can be achieved by reducing only one risk. The reduction in each risk is affected by the risk characteristics, risk response characteristics, risk response requirement, and the interdependency degree. Regarding the **budget** amount allocated to each risk, the result shows that the budget allocated for each risk depends on the initial risk value, the risk response requirement and the unit response cost of the selected strategy, and the interdependency degree. Specifically, more budget is required at a strict risk response requirement, a high initial risk value, a high unit response cost, or a low interdependency degree.

4 Conclusion and future research

For the strategy selection, in the linear setting, the optimal decision is to implement one strategy if this can meet the response requirement. Otherwise, both risk prevention and protection are required. For mitigating each risk in the non-linear setting, risk prevention, risk protection and a combination of risk prevention and protection can be the optimal strategy for risk response, which depends on the effectiveness of response strategies and the initial values of the risk. For the budget allocation decision, we observe that the optimal budget amount is dependent on the different risk parameters and the interdependency degree. Our research can be extended to multiple risks by modelling the different types of interdependencies among different risks.

References

Ben-David I., T. Raz, 2001, "An integrated approach for risk response development in project planning", *Journal of the Operational Research Society*, Vol. 52, pp. 14-25.

Dey P. K., 2012, "Project risk management using multiple criteria decision-making technique and decision tree analysis: A case study of Indian oil refinery", *Production Planning & Control*, Vol. 23, pp. 903-921.

Elkjaer M., F. Felding, 1999, "Applied project risk management-introducing the project risk management loop of control", *Project Management*, Vol. 5, pp. 16-25.

Fan M., N. P. Lin and C. Sheu, 2008, "Choosing a project risk-handling strategy: An analytical model", *International Journal of Production Economics*, Vol. 112, pp. 700-713.

Guan X., T. Servranckx and M. Vanhoucke, 2021, "An analytical model for budget allocation in risk prevention and risk protection", *Computers & Industrial Engineering*, pp. 107657.

Kuo, R. J., Y. Nugroho and F. E. Zulvia, 2019, "Application of particle swarm optimization algorithm for adjusting project contingencies and response strategies under budgetary constraints", *Computers & Industrial Engineering*, Vol. 135, pp. 254-264.

Miller R., D. Lessard, 2001, "Understanding and managing risks in large engineering projects", *International Journal of Project Management*, Vol. 19, pp. 437-443.

Sato T., M. Hirao, 2013, "Optimum budget allocation method for projects with critical risks", *International Journal of Project Management*, Vol. 31, pp. 126-135.

Sherali H. D., E. Dalkiran and T. S. Glickman, 2011, "Selecting optimal alternatives and risk reduction strategies in decision trees", *Operations Research*, Vol. 59, pp. 631-647.

Sherali H. D., J. Desai and T. S. Glickman, 2008, "Optimal allocation of risk-reduction resources in event trees", *Management Science*, Vol. 54, pp. 1313-1321.

Zhang Y., X. Guan, 2018, "Selecting project risk preventive and protective strategies based on bow-tie analysis", *Journal of Management in Engineering*, Vol. 34, pp. 04018009.

Heuristic Parameter Estimation by Machine Learning

Aykut Uzunoglu[1]

[1]Faculty of Business and Economics, Augsburg University, Germany
aykut.uzunoglu@uni-a.de

Keywords: heuristic parameters, machine learning, serial batch scheduling.

1. Introduction

Whenever NP-hard problems must be solved, heuristics are a good choice for the efficient calculation of solutions. Some heuristics have parameters to control their behavior and the tuning of those parameters is crucial for their performance. One example of a tunable heuristic is the Apparent Tardiness Costs with Setups (ATCS) priority rule. This priority rule is used for scheduling different production environments like a single machine or parallel machines with (sequence-dependent) setup times and tardiness-related objectives. The ATCS priority rule determines an ordering for jobs in a schedule. For that, the ATCS function uses the two look-ahead parameters κ_1 and κ_2 (Lee & Pinedo, 1997). An extension of the ATCS heuristic, the ATCS-β heuristic, is proposed by Gahm et al. (2021) for a serial batch scheduling problem. This heuristic uses an additional parameter β to control the batch utilization. In Gahm et al. (2021), the parameter tuning of κ_1, κ_2, and β is performed online (i.e., during the solving of a single problem instance) by a full grid search (i.e., a multi-start heuristic is designed). Such a full grid search leads to low efficiency if the number of parameter combinations is high. With any additional parameter, the space of parameter combinations gets very large leading to high computation times. A naturally arising question is whether an instance-dependent single parameter combination or a reduced grid of promising parameter combinations (e.g., κ_1, κ_2, and β) can be defined beforehand to increase efficiency. To address this problem, Lee & Pinedo (1997) introduced a curve-fitting approach to estimate good parameter values offline. In contrast to online parameter tuning, offline tuning does not require a (grid) search during the solving of a single problem instance.

In this work, a machine learning model-based estimator for the offline prediction of parameters is presented. To that end, section 2 gives a detailed description of the scheduling problem and the applied ATCS-based heuristics. In section 3, the learning pipeline, the model training, and the prediction application are discussed. The results are depicted in section 4 and the conclusion in section 5.

2. Problem Description

The serial batch scheduling problem addressed in this work is common in the metal manufacturing industry whenever geometrical shapes (jobs) must be cut out from a large metal slide by a (laser) cutting machine. Besides the allocation and sequencing, a batch scheduling problem additionally has to group jobs into batches that should be processed together. In serial batching, the batch processing time is the sum of all job processing times in a batch. The set of geometrical shapes result from different customer orders and have different characteristics in terms of material type or thickness, leading to incompatible job families. Because of their different characteristics, the jobs of different families cannot be grouped in the same batch. Every job $j \in J$ has the following properties relevant for scheduling: area demand a_j, weight w_j (specifies the weight of the job tardiness on the total tardiness), processing time p_j, family f_j, and due date d_j. Furthermore, sequence-dependent setup times $s_{f,g}$ (setup time from family f to g) must be considered. The objective is the minimization of the total weighted tardiness by considering the limited area supply of the metal slide defined by the batch capacity bc. The scheduling problem at hand can be classified as $P|sb, if, b, a_j, s_{f,g}|wT$ problem (cf., Gahm et al., 2021)

This problem is NP-hard and its mixed-integer linear program formulation can only be used to solve small instances. Therefore, Gahm et al. (2021) proposed a heuristic based on the ATCS-rule with a modification to control the batch utilization, i.e., what proportion of the metal slide capacity may be used (utilizing the whole metal slide can lead to suboptimal solutions in a serial batching

problem with weighted total tardiness objective). To control the batch utilization, a modified maximum batch capacity bc^{mod} is used: $bc^{mod} \coloneqq \max\{\beta \cdot bc, \max\{a_j | \forall j \in J\}\}$.

The ATCS-rule defines job-specific urgency values each time step t in the planning horizon that will then be used in the heuristic to define which jobs are grouped as batches and in which order the batches have to be processed by the machines. For the urgency evaluation, two look-ahead parameters, κ_1 and κ_2, must be specified in the ACTS function depicted in equation (1):

$$ATCS_j\ (t,f) = \frac{w_j}{p_j} \cdot \exp\left(-\frac{\max\{d_j - p_j - t, 0\}}{\kappa_1 \cdot \bar{p}_t}\right) \cdot \exp\left(-\frac{s_{f,f_j}}{\kappa_2 \cdot \bar{s}_j}\right) \quad (1)$$

One can find reasonable value ranges for those parameters in the literature (e.g., $\kappa_1 \in [0.5, 1.0, \ldots, 5.0]$ and $\kappa_2 \in [0.1, 0.2, \ldots, 1.6]$, cf., Lee & Pinedo, 1997). For the utilization degree β, the range [0.5, 0.55, …, 1] is considered since jobs of the data set can have an area demand up to 50% of the metal slide area supply (cf., Gahm et al., 2021). The heuristic uses 1,760 combinations based on those intervals and eleven estimated parameters (of Lee & Pinedo (1997), combined with eleven β values) for the online full grid search.

Since this approach lacks efficiency particularly for large problem instances, we propose to estimate a set S_i of parameter combinations for a problem instance I_i that likely lead to good solutions. Machine learning methods showed to be successful estimators for tasks in very different areas. Park et al. (2000) introduced the application of Neural Networks (NN) for the prediction of κ_1, κ_2 for the ATCS heuristic applied to a parallel machine environment. Mönch et al. (2006) build on the work of Park et al. (2000) and successfully use, besides NNs, Inductive Decision Trees for a similar problem with dynamic job arrival times. However, those contributions only consider small instance feature vectors, simple learning pipelines, NNs with single hidden layers, and particularly do not consider the dependency between the output variables (the dependency between κ_1 and κ_2 can be easily seen in equation (1)). Therefore, the machine learning model must be capable to consider the dependency between the output values (targets) κ_1, κ_2 and β. NNs show to be a powerful machine learning model with the capability of operating on multi-dimensional output spaces (cf., Bishop, 2006). To capture the multi-dimensionality and interdependency of the output, NNs are chosen to predict promising parameter combinations.

3. Pipeline Configuration, Model Training, and Prediction Application

For the training and validation of the NN, a data set with 93,360 scheduling instances is generated. Those scheduling instances are divided into 18,672 instance classes with each class containing five instances. The instance classes are a result of different combinations of instance attributes. The most important attributes are: number of jobs $n \in \{30, 60, 100, 200, 400, 800, 1600, 3200\}$, number of machines $m \in \{1, 3, 4, 5, 10, 20\}$, and number of incompatible job families $q \in \{3, 5, 10, 20, 40\}$. Further attributes are, for example, setup time severity, tardiness factor, and due data range. After the instance generation, the instances are solved with ACTS-β by using a multi-start heuristic for all possible 1,771 parameter combinations. With that, every problem instance has a solution set with parameter combinations that results in the best (known) objective value. Since scalar- or vector-valued targets are easier to handle for machine learning models, the target set (resulting from the solution set) should be transferred to a vector-valued target by systematically "picking" a single representative of the solution set. Accordingly, problem instance I_i leads to a single sample S_i by using the single representative of the best solutions calculated by ACTS-β. The target vector $\hat{y}_i$ represents the parameter combination assigned to sample S_i. The picking process starts by calculating the mean β value ($\bar{\beta}$) and filtering the parameter combinations in the solution set that have the closest β value to $\bar{\beta}$. Those steps are repeated with κ_1 and κ_2 which in the end returns a single representative of the solution set. This single representative will then be used in the machine learning model as the target vector that has to be predicted.

To increase prediction performance, different machine learning pipeline variations have been developed and analyzed. For feature engineering, two variants of numerical representations of a problem instance I_i by a feature vector $\bar{x}_i$ were developed: a complex feature vector (CF) and an aggregated feature vector (AF). The CF-vector consists of 12 complex instance characteristics that are (partly) inspired by the literature (e.g., approximated makespan, extrapolated tardiness factor, or approximated number of jobs per batch). The AF-vector uses five simple characteristics (e.g., number of jobs) and enlarges the vector by the aggregation of job processing times, job due dates,

job weights, job capacity requirements, job capacity requirement to processing time ratios, job due date to weight ratios, setup times, and the number of jobs per family. Those basic instance characteristics are then aggregated by the following ten aggregation functions: sum, median, minimum, maximum, variance, first quartile, third quartile, 10% percentile, 90% percentile, and skewness. In total, the AF-vector contains 85 features. As preliminary experiments have shown that dimension reduction by Principal Component Analysis does not increase prediction performance, we decided to not use it in the pipeline.

For scaling of the output values, machine learning pipelines with a basic min-max-output-scaler (osc) and pipelines without any scaling (no-osc) were included in the evaluation.

The first training phase follows a grid search for the hyper-parameter tuning of the NN and 5-fold cross-validation in every step to estimate the prediction error by the root mean squared error metric. In this initial training phase, the data set is reduced to 40% of the whole data set to speed up the training. After tuning the hyper-parameters, the final model is trained on 80% of the data.

Besides the direct application of the predicted parameters in the heuristic, one can build a reduced grid based on the predicted parameters to improve the solution quality. With that, parameter combinations "next" to the predicted parameter vector $\hat{y}_i$ are also included in the reduced parameter grid. Because the reduced grid also increases the computation time (compared to single parameter configuration), the number of neighbors added to the reduced grid must be chosen wisely (to obtain a reasonable trade-off between computation time and solution quality). To control the size of the reduced grid, the exploration factor ζ is introduced. This factor defines how many neighbors are added to the reduced grid, e.g., with an exploration factor $\zeta = 1$, the eight closest neighbors (greater and smaller than the predicted value) are added to the reduced grid. For $\zeta = 2$, 64 parameter combinations, and for $\zeta = 3$, 216 combinations are added to the reduced grid. Additionally, the predicted parameter combination and also the eleven simply estimated parameter combinations (of Lee & Pinedo (1997), combined with eleven β values) are added to the reduced grid.

To control the efficiency trade-off, we propose using different grid scenarios depending on the number of jobs (as the number of jobs is the major influencing factor of the computation time). For that, three different grid scenarios (GS) are analyzed: GS1 – the exploration factor ζ is set to 2 for all instance types; GS2 – ζ is set to 1 for instances with $n \in [600, 3200]$, to 2 for $n \in [100, 600)$ and 3 for $n \in [0, 100)$; GS3 – ζ is set to 1 for instances with $n \in [1000, 3200]$, to 2 for $n \in [100, 1000)$, and to 3 for $n \in [0, 100)$.

4. Results

After training the final model on 80% of the data, the remaining 20% (test data set), are used to assess the prediction performance for the actual task – minimizing the total weighted tardiness. For that, the predictions for the test data are fed into the ATCS-β heuristic and the resulting objective value are compared to the best-known objective value of every instance found by the multi-start heuristic with 1,771 parameter combinations.

As performance metric, the "mean relative improvement to the worst objective value" (MRIW) is used since objective values may be close (or equal to) zero for some instances which could cause numerical issues (cf., Valente & Schaller, 2012). Here, larger MRIWs indicate better solution quality and lower total tardiness.

Table 1. MRIW – Performance comparison

	MRIW									Computation time							
n	30	60	100	200	400	800	1600	3200	MEAN	30 [s]	60 [s]	100 [s]	200 [s]	400 [s]	800 [m]	1600 [m]	3200 [m]
BATCS-β without ML																	
	25.0	23.8	21.5	13.0	6.0	13.7	5.3	4.6	14.1	1.5	1.6	4.1	12.8	42.4	6.1	25.3	131.5
BATCS-β with ML(CF, no-osc)																	
GS1	19.8	19.0	18.0	6.5	3.8	13.8	6.0	1.5	11.0	0.1	0.1	0.3	0.5	1.3	0.2	0.9	5.4
GS2	21.1	20.2	18.0	6.5	3.8	10.6	8.2	1.0	11.2	0.1	0.2	0.3	0.5	1.3	0.1	0.2	1.4
GS3	21.1	20.2	18.0	6.5	3.8	13.8	8.2	1.0	11.6	0.1	0.2	0.3	0.5	1.3	0.3	0.3	1.6
BATCS-β with ML(CF, osc)																	

GS1	19.8	19.3	19.1	5.8	3.9	14.8	11.8	4.7	12.4	0.1	0.1	0.3	0.6	1.7	0.3	1.1	5.5
GS2	21.1	20.2	19.1	5.8	3.9	14.1	10.7	4.7	12.4	0.1	0.2	0.3	0.5	1.3	0.1	0.3	1.5
GS3	21.1	20.2	19.1	5.8	3.9	14.8	10.7	4.7	12.5	0.1	0.2	0.3	0.5	1.4	0.2	0.3	1.5
BATCS-β with ML(AF, no-osc)																	
GS1	23.7	15.8	19.8	6.9	8.5	15.2	10.9	3.7	13.0	0.1	0.1	0.3	0.6	1.4	0.2	0.9	5.2
GS2	25.0	19.7	19.8	6.9	8.5	9.8	8.5	2.3	12.5	0.1	0.2	0.2	0.5	1.4	0.1	0.2	1.5
GS3	**25.0**	**19.7**	**19.8**	**6.9**	**8.5**	**15.2**	**8.5**	**2.3**	**13.2**	**0.1**	**0.2**	**0.3**	**0.5**	**1.3**	**0.2**	**0.3**	**1.6**
BATCS-β with ML(AF, osc)																	
GS1	19.1	15.4	18.8	7.9	10.8	15.6	6.1	2.1	12.0	0.1	0.1	0.2	0.5	1.3	0.2	0.8	5.0
GS2	21.1	19.9	18.8	7.9	10.8	11.6	6.0	1.3	12.2	0.1	0.2	0.3	0.5	1.3	0.1	0.3	1.5
GS3	21.1	19.9	18.8	7.9	10.8	15.6	6.0	1.3	12.7	0.1	0.2	0.3	0.5	1.4	0.3	0.3	1.5

Table 1 depicts the results based on the test data set grouped according to the number of jobs. The first row shows the MRIWs computed with the multi-start BATCS-β heuristic. In comparison, the following rows show the performance of the machine learning-based multi-start heuristic for four pipeline configurations: CF and AF feature vectors with and without output scaling. Regarding those configurations, the AF-vector without output scaling (no-osc) with GS3 performs best on average. For instances with $n \geq 800$, the BATCS-β heuristic with reduced grids even outperforms the full grid approach. This is because BATCS-β with ML adds the prediction of the machine learning model to the parameter space (which was not given in BATCS-β). Summarizing the results in Table 1, we can report that the BATCS-β heuristic with parameter values predicted by NNs achieves a comparable solution quality and is superior in terms of computation time.

5. Conclusion

This work demonstrates the efficiency of parameter estimation by machine learning (NNs) for the BATCS-β heuristic in the case of serial batch scheduling problems. The machine learning-based approach can reduce the initial parameter space of 1,771 combinations remarkably. Additionally, the option to control the trade-off between solution quality and computation time is provided by an exploration parameter. With that parameter, the BATCS-β heuristic enhanced by machine learning reaches 99.1% of the solution quality by simultaneously decreasing the computation time by 94.3% on average (compared to the original BATCS-β heuristic).

References

Bishop, C. M. (2006). *Pattern recognition and machine learning*. *Information Science and Statistics*. New York, NY: Springer Science+Business Media LLC.

Gahm, C., Wahl, S., & Tuma, A. (2021). Scheduling parallel serial-batch processing machines with incompatible job families, sequence-dependent setup times and arbitrary sizes. *International Journal of Production Research*, *1–24.* doi:10.1080/00207543.2021.1951446.

Lee, Y.-H., & Pinedo, M. L. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, *100*(*3)*, *464–474.* doi:10.1016/S0377-2217(95)00376-2.

Mönch, L., Zimmermann, J., & Otto, P. (2006). Machine learning techniques for scheduling jobs with incompatible families and unequal ready times on parallel batch machines. *Engineering Applications of Artificial Intelligence*, *19*(*3)*, *235–245.* doi:10.1016/j.engappai.2005.10.001.

Park, Y., Kim, S., & Lee, Y.-H. (2000). Scheduling jobs on parallel machines applying neural network and heuristic rules. *Computers & Industrial Engineering*, *38*(*1)*, *189–202.* doi:10.1016/S0360-8352(00)00038-3.

Valente, J. M., & Schaller, J. E. (2012). Dispatching heuristics for the single machine weighted quadratic tardiness scheduling problem. *Computers & Operations Research*, *39*(*9)*, *2223–2231.* doi:10.1016/j.cor.2011.11.005.

Operating rooms scheduling with a shared resource: a red-blue knapsack modeling approach

Federico Della Croce[1], Andrea Grosso[2] and Vincent T'kindt[3]

[1] DIGEP Politecnico di Torino, Italy, federico.dellacroce@polito.it
[2] D.I. Universita' di Torino, Italy, grosso@di.unito.it
[3] LIFAT University of Tours, France, tkindt@univ-tours.fr

Keywords: Operating rooms scheduling, Knapsack Problem, Exact approach.

1 Introduction

A practical operating rooms scheduling problem, which arises at a public hospital located in Northern Italy is considered. The problem consists in deriving a daily schedule for a set of operating rooms where each operating room is associated with a surgical specialty and a set of surgical interventions to be performed for the considered day. Each surgical intervention has a nominal duration and a nominal profit, the latter being a function of the intervention priority. The main requirement is that the overall intervention time in each operating room does not exceed the related operating room common time availability(e.g. 8-20).

In addition, a subset of the interventions requires the use of a shared resource among the operating rooms so that no two interventions using that shared resource can be processed simultaneously. We denote this problem as the Shared Resource Operating Rooms scheduling (SRORS) problem. Operating rooms scheduling has been widely investigated in the last 50 years. We mention here four main reference surveys on the topic published between 2010 and 2019, namely Cardoen et al. (2010), Zhu et al. (2019), Guerriero and Guido (2011) and Samudra et al. (2016).

The SRORS problem can be formally expressed as follows. We have n surgical interventions (hereafter also denoted as jobs) $1, \cdots, n$ and m operating rooms $OR_1, \cdots, OR_m$. Each job j belongs to a specific surgical specialty that is univocally assigned to operating room OR_i. For each operating room O_i, there are then n_i jobs that can be scheduled in the considered day so that $n = \sum_{i=1}^{m} n_i$. We denote by J_i the set of jobs schedulable in operating room OR_i. Each operating room OR_i is available for a common duration W for surgical interventions. Each job j may or may not require the use of a shared resource. We will denote as *red* jobs ($j \in R$) those jobs requiring the shared resource and as *blue* jobs ($j \in B$) the other jobs. Also, for each OR_i, we will denote as R_i (B_i) the set of relevant red (blue) jobs. Each job j schedulable in OR_i has a nominal duration $w_{i,j}$ and a profit $p_{i,j}$ directly derived from the job priority. The goal is to maximize the total profit of the scheduled jobs.

To avoid unnecessary moves of the shared resource from one operating romme to another, it is assumed that every day this resource can spend at most one single interval of time in every operating room. Correspondingly, for each OR_i, w.l.o.g. a sequence $\beta_i^1 \rho_i \beta_i^2$ holds, where β_i^1 is a first subset of blue jobs, ρ_i is a subset of red jobs and β_i^2 is a second subset of blue jobs. Notice that all these subsets may be empty.

2 Problem complexity

Being a generalization of the 0/1 Knapsack Problem (see, for instance, Pferschy et al. (2004)), the SRORS problem is NP-Hard, but it is of interest to determine whether it is in the weak or in the strong sense.

We consider first the special case of two operating rooms only, denoted as 2-SRORS problem. For the 2-SRORS problem, w.l.o.g. we can assume that both β_1^2 and β_2^1 are empty, namely only one subset of blue jobs is present in both operating rooms, where the blue jobs precede the red jobs in OR_1 and the red jobs precede the blue jobs in OR_2. Correspondingly, we can remove upperscripts and a sequence $\beta_1\rho_1$ holds in OR_1, a sequence $\rho_2\beta_2$ holds in OR_2. Let us denote by $f(\beta_i(t))$ ($f(\rho_i(t))$) the profit of subset β_i (ρ_i) having overall duration t. Correspondingly, by applying standard dynamic programming independently first on blue jobs and then on red jobs for each operating room with complexity $O(n_1W)$ and $O(n_2W)$ it is possible to compute $f(\beta_1(t))$, $f(\rho_1(t))$, $f(\beta_2(t))$ and $f(\rho_2(t))$ for every $t \in [0, \cdots, W]$. Then, by computing $f(\beta_1(t)) + f(\rho_1(W-t)) + f(\beta_2(W-t)) + f(\rho_2(t)), \forall t \in [0, \cdots, W]$, we get in $O(W+1)$ time the values of the best schedules in OR_1 and OR_2 for all possible values of t denoting both the completion time of β_1 and starting time of β_2. The optimal solution is then given by the schedule providing the best of the computed values always requiring $O(W+1)$ time. The overall complexity is then $O(nW)$ time, that is the 2-SRORS problem is pseudo-polynomially solvable and thus weakly NP-Hard.

Let us now consider the case when $k > 2$ is fixed. The k-SRORS problem, for each operating room OR_i, it is possible to compute and store in $O(nW)$ time the best possible solutions of the red block for every size t in $[0 \cdots, W]$. Hence, for all pairs t_1, t_2 denoting the sizes of the first and second blue blocks, respectively, $f(\rho_i(W - t_1 - t_2))$ can then be derived in constant time. But then, a dynamic programming algorithm on the blue jobs that select or not each of these jobs and in case of selection places it either at the beginning of block β_i^1 or at the end of block β_i^2, can find the optimal solution of the operating room OR_i for every interval t_1, t_2 in $[0 \cdots, W]$ in $O(nW^2)$ time. Correspondingly, it is possible to show that if the sequence of red blocks is given, then the optimal solution value can be computed in $O(knW^2)$ time. But then, as the number of red blocks sequences is bounded above by $k!$, also the k-SRORS problem is pseudo-polynomially solvable and thus weakly NP-Hard.

Finally, for general m, by reduction from 3-partition it is possible to show (proof will be provided at the Conference) that the m-SRORS problem is strongly NP-Hard.

3 ILP formulation and solution approach

A straightforward Integer Linear Programming (ILP) formulation of the problem could be derived by using a pseudopolynomial number of binary variables $z_{i,j,t}$ indicating if job j starts processing at time t in operating room i. Such formulation quickly fails to reach optimality in reasonable time already with a limited number of jobs and operating rooms. Instead, an improved knapsack-based formulation of the SRORS problem can be devised as follows.

For each job $j \in J_i$, we introduce two binary variables $x_{i,j}^1$ and $x_{i,j}^2$ if job $j \in B_i$ (blue job), or a binary variable $y_{i,j}$ if job $j \in R_i$ (red job). For blue jobs, $x_{i,j}^1 = 1$ ($x_{i,j}^2 = 1$) if job j is selected and is placed in the first (second) subset β_i^1 (β_i^2), alternatively $x_{i,j}^1 = 0$ ($x_{i,j}^2 = 0$). Similarly, for red jobs, $y_{i,j} = 1$ if job j is selected and is placed in subset ρ_i, alternatively $y_{i,j} = 0$.

Then, for each operating room OR_i, we introduce two continuous variables $s_i \geq 0$ and $e_i \geq 0$ denoting the start time and end time, respectively, of subset ρ_i of red jobs. Finally, for each pair O_i, O_k (with $i < k$) of operating rooms, we introduce a binary variable $z_{i,k}$ where $z_{i,k} = 1$ if $e_i \leq s_k$, else $z_{i,k} = 0$.

The following ILP model holds.

$$\max \sum_{i\in[1,\cdots,m],\ j\in B_i} p_{i,j}(x^1_{i,j} + x^2_{i,j}) + \sum_{i\in[1,\cdots,m],\ j\in R_i} p_{i,j}y_{i,j} \tag{1}$$

$$\sum_{j\in B_i} w_{i,j}x^1_{i,j} \leq s_i, \quad \forall i \in [1,\cdots,m] \tag{2}$$

$$\sum_{j\in R_i} w_{i,j}y_{i,j} \leq e_i - s_i, \quad \forall i \in [1,\cdots,m] \tag{3}$$

$$\sum_{j\in B_i} w_{i,j}x^2_{i,j} \leq W - e_i, \quad \forall i \in [1,\cdots,m] \tag{4}$$

$$\sum_{i\in[1,\cdots,m],\ j\in R_i} w_{i,j}y_{i,j} \leq W \tag{5}$$

$$s_i \leq e_i \quad \forall i \in [1,\cdots,m] \tag{6}$$

$$x^1_{i,j} + x^2_{i,j} \leq 1 \quad \forall j \in B_i \quad \forall i \in [1,\cdots,m] \tag{7}$$

$$e_i \leq s_k + W(1 - z_{i,k}) \quad \forall i,k \in [1,\cdots,m],\ i \neq k \tag{8}$$

$$e_k \leq s_i + Wz_{i,k} \quad \forall i,k \in [1,\cdots,m],\ i \neq k \tag{9}$$

$$x^1_{i,j} \in \{0,1\} \quad \forall i \in [1,\cdots,m], \quad \forall j \in B_i \tag{10}$$

$$x^2_{i,j} \in \{0,1\} \quad \forall i \in [1,\cdots,m], \quad \forall j \in B_i \tag{11}$$

$$y_{i,j} \in \{0,1\} \quad \forall i \in [1,\cdots,m], \quad \forall j \in R_i \tag{12}$$

$$z_{i,k} \in \{0,1\} \quad \forall i,k \in [1,\cdots,m],\ i \neq k \tag{13}$$

In the model, obj function (1) maximizes the total profit of the selected interventions. Constraint (2) states that for every operating room O_i, the total duration of the blue jobs scheduled before ρ_i is not superior to the starting time s_i. Constraint (3) states that for every operating room O_i, the total duration of the red jobs is equal to $e_i - s_i$. Constraint (4) states that for every operating room O_i, the total duration of the blue jobs scheduled after ρ_i is not superior to $W - e_i$ so that they can be operated within the time availability period W. Constraint (5) states that the total duration of the red jobs over all operating rooms is not superior to the time availability period W (we remark that this constraint is redundant but positively affects the performances of the solver). Constraint (6) states that for every operating room O_i the starting time of the red block is not superior to the end time. Constraint (7) states that for every operating room O_i, the blue jobs, if scheduled, must be processed either before or after the red block. Constraints (8-9) are the so-called disjunctive constraints on the red blocks, that is, for each pair of operating rooms O_i, O_k, either the red block of O_i precedes the red block of O_k, or, viceversa, the opposite holds. Constraints (10, $\cdots$, 13) indicate that all $x^1_{i,j}, x^2_{i,j}, y_{i,j}, z_{i,k}$ are binary variables.

The considered ILP model can be enhanced in several ways. A preprocessing step for eliminating all dominated jobs can be applied. Recalling that, given a job j, every job k with larger profit and smaller weight is preferable to j, we can show that a red job $j \in R_i$

is dominated (hence $y_{i,j} = 0$) whenever

$$\sum_{k \in R_i \cup B_i : k \neq j,\ w_{i,k} \leq w_{i,j},\ p_{i,k} > p_{i,j}} w_{i,k} > W - w_{i,j} + 1$$

namely, there exists a set of jobs in $R_i \cup B_i$ that are preferable to j such that the sum of their durations exceeds $W - w_{i,j}$.

Similarly, we can show that a blue job $j \in R_i$ is dominated (hence $x^1_{i,j} + x^2_{i,j} = 0$) whenever

$$\sum_{k \in B_i : k \neq j,\ w_{i,k} \leq w_{i,j},\ p_{i,k} > p_{i,j}} w_{i,k} > W - w_{i,j} + 1$$

namely, there exists a set of jobs in B_i that are preferable to j such that the sum of their durations exceeds $W - w_{i,j}$.

Also, an efficient knapsack-based lower bound can be computed by solving to optimality on each operating room the relevant $0/1$ knapsack problem (summing up each knapsack solution) disregarding the requirement on the shared resource.

Finally, several distinct branching schemes can be devised. A thorough computational campaign testing different bounding and branching schemes will be discussed at the Conference.

Acknowledgements

This work has been partially supported by "Ministero dell'Istruzione, dell'Università e della Ricerca" Award "TESUN-83486178370409 finanziamento dipartimenti di eccellenza CAP. 1694 TIT. 232 ART. 6".

References

Cardoen B., Demeulemeester E., Belien J. 2010, "Operating room planning and scheduling: a literature review", *European Journal of Operational Research* , Vol. 201 (3), pp. 921-932.

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

Samudra M., Van Riet C., Demeulemeester E., Cardoen B., Vansteenkiste N., Rademakers F.E. 2016, "Scheduling operating rooms: achievements, challenges and pitfalls", *Journal of Scheduling*, Vol. 19, pp.493-525.

Zhu S., Fan W., Yang S., Pei J., Pardalos P.M. 2019, "Operating room planning and surgical case scheduling: a review of literature", *Journal of Combinatorial Optimization*, Vol. 37, pp. 757-805.

Pferschy U., Pisinger D, Kellerer H. 2004, "Knapsack problems", *Springer-Verlag*.

Valid inequalities for the dynamic asset protection problem

Quentin Peña, Mehdi Serairi and Aziz Moukrim

Université de technologie de Compiègne, CNRS, Heudiasyc (Heuristics and Diagnosis of Complex Systems), CS 60 319 - 60 203 Compiègne Cedex
quentin.pena@hds.utc.fr

Keywords: team orienteering problem, bi-objective, synchronization, dynamic, wildfire

1 Introduction

In the recent years, we have seen a surge in the number and strength of wildfires around the globe. These events, whether natural or caused by human activities, can damage the wildlife, as well as people and infrastructures. When a wildfire breaks, Incident Management Teams (IMTs) need to dispatch their resources to contain the fire, evacuate the people, and protect community assets (hospitals, bridges, schools, etc.).

In this paper, we will focus more particularly on vehicle routing for asset protection. IMTs need to assign an heterogeneous fleet of vehicles to the different community assets to carry out preventive actions. Such actions include wetting the facade of a building or removing fuel material, for example. These actions effectively mitigate the damages if they are accomplished within a specified time window, not too soon or too late. Their complexity often requires the cooperation and synchronization of multiple teams and vehicles.

Unfortunately, the behaviour of wildfires and its consequences are hard to predict. Changes in wind conditions, road closures due to fallen trees, vehicles breakdowns, can make the vehicles' routes obsolete. IMTs need to react to any disruption by updating the routes of the vehicles. The new routes must protect as many assets as possible, but there is an incentive to limit the deviation from the initial routes, as some actions may require specific preparation and to limit communication issues.

The problem of routing vehicles for asset protection during a wildfire was presented as the Asset Protection Problem during escaped wildfire (APP) in Van der Merwe *et. al.* (2015). Van der Merwe *et. al.* (2017) defines the dynamic APP that aims at rerouting vehicles after a disruption occurs. As the dynamic APP is a bi-objective problem, the solution is a trade-off surface called Pareto front. The Pareto front is described by a set of solutions such that there is no feasible solution that strictly improves one objective without degrading the second. We say that these solutions are non-dominated. Multiple heuristic approaches have been studied for the mono-objective version of the APP ((Roozbeh *et al.* 2018), (Nuraiman *et. al.* 2020), (Yahiaoui *et al.* 2021)).

Our work focuses on improving the resolution of the optimal Pareto front for the dynamic APP by finding good valid inequalities that rely on the bi-objective or dynamic nature of our problem.

2 Problem presentation

The first mathematical formulation of the APP was proposed by Van der Merwe *et al.* (2015). The authors modeled the APP as a Synchronized Team Orienteering Problem with Time Windows (STOPTW). An instance is defined as a graph $G = (V, A)$. The vertices V represent the depots and the assets we seek to protect. Each asset has a resource requirement defined as a vector of integers. Resources are non-consumable. Each asset also

has a time window in which the protection action must start. To carry out the protection actions, a set of heterogeneous vehicles is available, each having a capability vector. The aim is to assign the assets to the routes of vehicles, such that the total protected value is maximized. An asset is protected if the cumulative capability vectors of the vehicles assigned to the asset covers the resource requirements of the asset, and if the protection action starts within the time window of the asset. The protection action at an asset can start only when every vehicle assigned to the asset has arrived (synchronization).

Van der Merwe *et al.* (2017) extended the mathematical formulation of the APP to the dynamic APP. On top of the assets and vehicles, the dynamic APP is based on initial routes for the vehicles before the disruption occurred. The objective is to maximize the total protected value while minimizing the deviation from the initial routes. The deviation can be represented as the number of asset/vehicle reassignments, *i.e.* for each vehicle the number of assets added to or removed from its pre-disruption route.

An important property of the dynamic APP is that an asset that does not participate to the total protected value can be visited outside of its time window. Assets that are not protected in the solution do not have to be removed from the routes of any vehicle, hence entailing no deviation. In other words, an asset that does not improve the first objective (total protected value) does not degrade the second one (deviation). It is thus always possible to build a solution with null deviation from the pre-disruption routes, even if a route became infeasible due to the disruption.

3 Valid Inequalities

We studied the problem to deduce valid inequalities based on properties specific to its bi-objective and dynamic nature. We will first present sets of valid inequalities that bound the deviation implied by the protection of an asset, based on their resource requirements. We will then present incompatibility between assets based on their time window, and deduce a set of valid inequalities. We will finally present a set of valid inequalities that combine incompatibility between assets and resource requirements.

3.1 Deviation-based inequalities

In a non-dominated solution, if an asset has been added to the route of at least one vehicle, the asset is protected in the solution. Otherwise, we could construct a feasible solution with the same protected value but strictly lower deviation (hence dominating our original solution) by simply not adding this asset to the route of the vehicle. Given upper and lower bounds $ub_v^+(i)$ and $lb_v^+(i)$ on the number of vehicles required to protect asset i, we have that:

$$ub_v^+(i)Y_i \geq \sum_{p \in \mathcal{P}} Z_{ip}^+ \geq lb_v^+(i)Y_i \tag{1}$$

where Y_i is the decision variable representing the protection status of asset i, $\mathcal{P}$ the set of available vehicles, and Z_{ip}^+ the decision variable representing the addition of asset i to the route of vehicle p.

We compute values for these bounds based on the resource requirement of the asset and the capability vector of the available vehicles.

The lower bound $lb_v^+(i)$ is the minimum number of vehicles required to cover the resource requirement of asset i. We can compute this bound by solving a Mixed Integer Program (MIP) for each asset, that can be written as a multi-dimensional knapsack problem. The size of the MIP is small enough to efficiently compute the value of the lower bound in spite of the problem being NP-hard.

The upper bound $lb_v^+(i)$ is the minimum number of vehicles required to cover the resource requirement of asset i without adding a redundant vehicle. A vehicle is redundant if the resource requirement of the asset is still covered if the vehicle is not considered. Thus, adding a redundant vehicle to the protection of an asset does not improve the total protected value but increases the deviation. We can compute this bound by solving a MIP for each asset. We believe this problem to be NP-hard, but the size of the MIP is small enough for the computation to be efficient.

Similar bounds are computed for the number of removals from routes for asset i.

3.2 Incompatibility/vehicle clique inequalities

Two assets i and j are *incompatible/vehicle* for vehicle p if vehicle p cannot visit asset i and asset j within their respective time windows. In other words, if vehicle p visits one of the asset at the opening of its time window, it always reaches the second asset after the closing of its time window.

Let $G_p^{inc/v}$ be the graph of incompatibilities between assets for vehicle p. Each node of this graph is an asset of our problem. There is an edge between two nodes i and j if assets i and j are incompatible/vehicle. A clique is a subset of nodes in an undirected graph that are pairwise adjacent. Vehicle p can visit at most one of the asset of the clique within their time window, which is necessary for the asset to be protected. Thus, we define valid inequalities to ensure that at most one asset is protected among the assets of the clique visited by vehicle p.

3.3 Incompatibility/solution clique inequalities

We call two assets i and j *incompatible/solution* if assets i and j are incompatible/vehicle for every available vehicle. Consider $\mathcal{C}$ a clique of incompatible/solution assets. Then, each vehicle can be used for the protection of at most one asset of $\mathcal{C}$.

Let $ub(\mathcal{C})$ be the maximum number of assets in $\mathcal{C}$ that can be protected using every vehicle at most once, based on the resource requirements of the assets. We thus have that:

$$\sum_{i \in \mathcal{C}} Y_i \leq ub(\mathcal{C}) \tag{2}$$

We can compute $ub(\mathcal{C})$ by solving a MIP for each clique $\mathcal{C}$. The problem is NP-hard, but the size of the MIP is small enough for the computation to be efficient.

4 Results

We generated 10 test instances following the specifications of Van der Merwe *et al.* (2015). We implemented the model in Julia. We solved the model with CPLEX 12.8, on a computer with an Intel Core i7-8550U processor and 8GB of RAM.

For every instance, we generated the extreme point allowing for maximum deviation for three different vehicle breakdowns, with a time limit of 1800 seconds. Table 1 shows the average solve time to compute this extreme point for instances based on which valid inequalities were added to the model. The number of instances solved within the time limit is shown in parenthesis.

We greatly improved the model when introducing our valid inequalities. Incompatibility/vehicle inequalities yield the best result: we solved 6 more instances with 50 assets and 5 more with 60 assets within the time limit. On average, we reduced by a third the solve time for the 4 instances solved to optimality with the initial model.

Table 1. Generation of one extreme point: solve time in seconds, number of optimal solutions

	Instance size (number of assets)			
Valid inequalities	30	40	50	60
None	42 (30/30)	29 (20/30)	58 (09/30)	458 (04/30)
Deviation	22 (30/30)	27 (20/30)	198 (09/30)	268 (03/30)
Inc/veh	1.4 (30/30)	22 (25/30)	267 (15/30)	841 (09/30)
Inc/sol	12 (30/30)	28 (24/30)	37 (11/30)	474 (04/30)
All above	1.2 (30/30)	61 (26/30)	252 (18/30)	518 (11/30)

Deviation-based inequalities greatly improve the relaxation of the model. However, as we compute the extreme point with highest deviation, the introduction of these inequalities alone did not seem to have an impact on the solve time. But, when combined with incompatibility/vehicle inequalities, they solved two more instances than incompatibility/vehicle inequalities alone, and almost halved the solve time for the 9 instances already solved within the time limit.

5 Conclusion

We deduced from the properties and structure of our problem valid inequalities that improved the resolution of our model. We improved the viability of using this model to evaluate the quality of the choices made by IMTs in retrospect. Additional inequalities or changes to the model would be needed to solve large-size instances more consistently.

However using this model for providing in real time the exact Pareto front when a disruption occurs seems out of reach. We would need to consider a heuristic approach to obtain a good approximation of the Pareto front in reasonable time.

Acknowledgements

This study was carried out within the framework of GEOSAFE (Geospatial Based Environment For Optimization Systems Addressing Fire Emergencies).

References

Nuraiman, D., Ozlen, M. and Hearne, J., 2020. A spatial decomposition based math-heuristic approach to the asset protection problem. *Operations Research Perspectives*, 7, p. 100141.

Ozlen, M. and Azizoglu, M., 2009. Multi-objective integer programming: A general approach for generating all non-dominated solutions. *European Journal of Operational Research*, 199(1), pp. 25-35.

Roozbeh, I., Ozlen, M. and Hearne, J., 2018. An Adaptive Large Neighbourhood Search for asset protection during escaped wildfires. *Computers & Operations Research*, 97, pp. 125-134.

Van der Merwe, M., Minas, J., Ozlen, M. and Hearne, J., 2015. A mixed integer programming approach for asset protection during escaped wildfires. *Canadian Journal of Forest Research*, 45(4), pp. 444-451.

Van der Merwe, M., Ozlen, M., Hearne, J. and Minas, J., 2017. Dynamic rerouting of vehicles during cooperative wildfire response operations. *Annals of Operations Research*, 254(1-2), pp. 467-480.

Yahiaoui, A-E., Moukrim, A. and Serairi, M., 2021. GRASP-ILS and set cover hybrid heuristic for the synchronized team orienteering problem with time windows. *International Transactions in Operational Research* Preprint, hal-03110595v3f.

Sequencing two classes of jobs on a machine with an external no-idle constraint

Alessandro Agnetis[1] and Marco Pranzo[1]

Universitá di Siena, Italy
{agnetis, pranzo}@diism.unisi.it

Keywords: machine scheduling, complexity, no-idle constraint.

1 Problem description

We consider a scheduling problem with two classes of jobs having distinct features and impact on the objective function.

A set J of jobs has to be performed on a two-machine flow shop. Jobs are partitioned into two classes, denoted as $\mathcal{A}$ and $\mathcal{B}$. The jobs of class $\mathcal{A}$ (called A-jobs) must be processed by M_1 only, while the jobs of class $\mathcal{B}$ (called B-jobs) must be processed by M_1 and then by M_2. The problem is to sequence the jobs so that the total completion time *of the A-jobs* is minimum, given a *no-idle constraint*, i.e., both machines must be always busy until the end of the schedule. Notice that the role of the two job classes is different. The objective only depends on the schedule of the A-jobs, but B-jobs are needed to keep machine M_2 busy. In other words, the decision maker has to balance two conflicting needs, namely to schedule the A-jobs early in the sequence (so that their contribution to the objective function is minimized) but at the same time to schedule B-jobs so that the second machine is kept busy.

The motivation for this problem comes from the testing department of an electronic firm, here corresponding to machine M_1. Such a department processes two different classes of jobs. The first class of jobs corresponds to orders which, after processing on M_1, are directly sold to the market, and the firm is therefore interested in minimizing the average time-to-market of these jobs. These jobs constitute class $\mathcal{A}$. However, an external *manufacturer* also outsources testing services to the firm, and will use the tested components to produce its own electronic devices. These testing jobs correspond to B-jobs, and the manufacturer's facility corresponds to M_2. So, M_1 is shared by "internal" A-jobs and "external" B-jobs. Whenever a certain B-job is completed on M_1, production at the manufacturer premises is guaranteed for a certain time interval. The length of such a time interval corresponds to the "processing time" on M_2 of the B-job. The agreement is that M_2 is never starving, i.e., the material flow to the manufacturer does not interrupt. The no-idle constraint may arise also from certain technical specifications. For example, in a production facility it may be unacceptable to underutilize an expensive piece of equipment, or because such facility is indeed a bottleneck resource which must be always kept busy.

2 Literature review

The problem addressed in this paper shares features from two different categories of flow shop scheduling problems, namely: (i) two-agent problems, as the two different classes of jobs play different roles, similar to two different agents owning the respective job sets, and (ii) no-idle problems, as in our problem we are not allowing to have idle time on the second machine (as there are no release dates, no need for idle time on the first machine ever arises).

Two-agent flow shop scheduling problems have been recently addressed in the literature. Fan and Cheng (2016) provide complexity, approximation and algorithmic results for various objective functions. A case in which the two agents have different objective functions is addressed by Jeong et al (2020) and solved through a branch and bound approach. However, in all the papers dealing with two-agent, two-machine flow shop problems, the two agents have individual objective functions, and all jobs, in general, require both machines. In our case, B-jobs only have the role of fulfilling a no-idle constraint on the second machine.

Problems in which machines are required to work continuously have been considered in the literature. Such problems may arise in different real world settings as described, for example, by Saadani et al (2003). Among such problems we mention the steel scheduling problem, where the continuous caster equipment requires all jobs in a lot to be scheduled without interruptions Pacciarelli and Pranzo (2004). From a more academic point of view, Adiri and Pohoryles (1982) established the hardness of $F2|\text{no-idle}|\sum C_j$, in which each job has a certain nonzero processing time on both machines. A large variety of metaheuristic approaches have been applied to no-idle m-machine flow shop problems, including Fink and Voss (2003) and Ruiz and Vallada (2009). The specificity of our problem is that in our case only A-jobs contribute to the objective function, so the complexity results established in the literature have no implications on our current problem.

3 Problem definition and complexity

The problem can be expressed as follows.

Two-Class No-Idle Scheduling Problem (2CNISP). *Two sets of jobs, $\mathcal{A}$ and $\mathcal{B}$ are given, with $|\mathcal{A}| = n_A$, $|\mathcal{B}| = n_B$ and $n = n_A + n_B$. Jobs of the two sets are called A-jobs and B-jobs respectively. Each A-job j requires a processing time p_j^1 on M_1. Each B-job k is processed sequentially on machines M_1 and M_2, requiring time p_k^1 and p_k^2 respectively, with $p_k^1 < p_k^2$. A value $\rho(0)$ is given such that Machine M_2 is busy in the interval $[0, \rho(0)]$. An unlimited buffer exists between M_1 and M_2. The problem is to find a schedule of all jobs minimizing the total completion time of the A-jobs, under the constraint that M_2 is never idle until all jobs have been processed.*

Given an overall schedule σ, let $C_s^1(\sigma)$ and $C_s^2(\sigma)$ denote the completion time of the job in s-th position on M_1 and M_2 respectively. Consider now the first s jobs in a schedule σ. The difference $C_s^2(\sigma) - C_s^1(\sigma)$ is the residual span on M_2 when the s-th job is completed on M_1, $s = 1, \ldots, n$ and it is called *runway after the s-th job*. A schedule is feasible if and only if the processing time of the job in position $s+1$ does not exceed $C_s^2(\sigma) - C_s^1(\sigma)$, $s = 0, \ldots, n-1$. Notice that since $p_k^1 < p_k^2$ for $k \in \mathcal{B}$, the problem of establishing whether a feasible solution exists can be easily solved. In fact, each B-job has the effect of increasing the runway. Consider a schedule in which we schedule all B-jobs at the beginning of a schedule (before any A-job) in nondecreasing order of their processing time on M_1. If (and only if) idle time occurs on M_2, the instance is infeasible.

A simple pairwise exchange argument shows that the following property holds.

Proposition 1. *In an optimal sequence for 2CNISP, the A-jobs are scheduled in SPT order.*

In view of Proposition 1, we can assume that the A-jobs are numbered in SPT order and we can restrict our analysis to feasible schedules in which A-jobs are SPT-ordered. Given one such feasible schedule, we call *group* j (denoted by G_j) the (possibly empty) set of B-jobs scheduled between the A-jobs j and $j+1$, for $j = 1, \ldots, n_A - 1$. Group G_0 indicates the B-jobs preceding the first A-job, G_{n_A} the B-jobs scheduled after the last A-job.

Concerning the computational complexity of 2CNISP, the following result holds.

Theorem 1. *Problem 2CNISP is NP-complete.*

The proof uses a reduction from the problem CARDINALITY CONSTRAINED PARTITION, hence only NP-hardness in the ordinary sense is proved.

4 A MIP approach for the 2CNISP problem

We address 2CNISP by means of the following idea.

- We introduce a *relaxed* problem formulation, called 2CREL, which exploits the group structure of the schedule. The optimal value of 2CREL provides a lower bound to the optimal solution of 2CNISP.
- Thereafter, 2CREL in used in a lazy-constraint fashion to solve 2CNISP. Starting from the optimal solution of 2CREL, we iteratively perform the following step. If a feasible solution for 2CNISP having the same value can be recovered such a solution is optimal for 2CNISP. Otherwise, we generate a constraint which is valid for 2CNISP and is violated by the optimal solution to 2CREL, the above constraint is added to 2CREL which is then solved again.

Although 2CREL is itself an ILP, its structure (which avoids the introduction of very large "big M" numbers) allows the model to be efficiently solved by ILP solvers. In what follows we report 2CREL in detail. In 2CREL, we relax the requirement that a B-job starts on M_2 only after completing on M_1. Precisely, we only require that the last B-job in a group *completes* on M_2 after completing on M_1, but we do not rule out that some B-job starts on M_2 before completing on M_1. In this way, a solution to 2CREL is defined by an assignment of the B-jobs into $n_A + 1$ groups G_i, where G_i is the set of *unsorted* B-jobs scheduled between $i \in \mathcal{A}$ and $i+1 \in \mathcal{A}$. No sequencing variables need to be introduced.

In 2CREL, we use binary variables $x_{ji} = 1$ if job $j \in \mathcal{B}$ is assigned to group G_i. Here we denote by ρ_i the value of the runway after the last job of group G_i, i.e., when the A-job $i+1$ starts. The formulation 2CREL is:

$$\min \quad \sum_{i \in \mathcal{A}} \left(\sum_{g=0}^{i-1} \sum_{j \in \mathcal{B}} x_{jg} \cdot p_j^1 \right) \tag{1}$$

$$\sum_{j \in \mathcal{B}} x_{ji} = 1 \quad \forall i = 0, \ldots, n_A \tag{2}$$

$$\rho_{i+1} = \rho_i - p_i^1 + \sum_{j \in \mathcal{B}} x_{ji} \cdot (p_j^2 - p_j^1) \quad \forall i = 0, \ldots, n_A - 1 \tag{3}$$

$$\rho_0 = \rho(0) \tag{4}$$

$$\rho_i \geq 0 \quad \forall i = 1, \ldots, n_A \tag{5}$$

$$x_{ji} \in \{0, 1\} \quad \forall j \in \mathcal{B}, i = 1, \ldots, n_A \tag{6}$$

The objective function (1) exploits the fact that, since the A-jobs are SPT-ordered, in the completion time of each A-job we only need to account for the total processing time of the B-jobs preceding it. Constraint (2) guarantees that each B-job $j \in \mathcal{B}$ is assigned exactly to one group. Constraints (3)–(5) define the values of the runway ρ_i. Constraint 3 relates the runway at the end of G_{i+1} with the runway at the end of the previous group G_i. Constraint (5) ensures that the last B-job of a group completes on M_2 after completing on

M_1, but does not rule out the possibility that it starts on M_2 before completing on M_1. This is why 2CREL is a relaxation of problem 2CNISP.

Computational results show that for instances with up to 50 jobs, the above MIP approach allows finding an exact solution or one having gap not exceeding 5% within a 10-minute CPU time limit on an Intel Core i9-9920X processor using Gurobi 9.1 as solver.

References

Adiri I., Pohoryles D., 1982, Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times, Naval Research Logistics Quarterly, 29(3), 495–504.

Fan B.Q., Cheng T.C.E., Two-agent scheduling in a flowshop, European Journal of Operational Research, 252(2), 376–384, 2016.

Fink A., Voss S., Solving the continuous flow-shop scheduling problem by metaheuristics, European Journal of Operational Research, 151, 400–414, 2003.

Jeong B.J., Kim Y.-D., Shim S.-O., Algorithms for a two-machine flowshop problem with jobs of two classes, International Transactions in Operations Research, 27, 3123–3143, 2020.

Pacciarelli D., Pranzo M., Production Scheduling in a Steelmaking-Continuous Casting Plant, Computers and Chemical Engineering, 28(12), 2823–2835, 2004.

Ruiz R., Vallada E., FernÃ¡ndez-MartÃnez C., (2009) Scheduling in Flowshops with No-Idle Machines. In: Chakraborty U.K. (eds) Computational Intelligence in Flow Shop and Job Shop Scheduling. Studies in Computational Intelligence, vol 230. Springer, Berlin, Heidelberg.

Saadani N.E.H., Guinet A., Moalla M., Three stage no-idle flow-shops, Computers and Industrial Engineering, 44(3), 425–434, 2003.

Carbon footprint aware resource constrained project scheduling problem in manufacturing

Humyun Fuad Rahman[1], Tom Servranckx[2], Ripon K. Chakrabortty[1], Mario Vanhoucke[2,3,4] and Sondoss El Sawah[1]

[1] Capability Systems Centre, School of Engineering & IT, UNSW Canberra, Australia
humyun.fuad@adfa.edu.au, r.chakrabortty@adfa.edu.au, s.elsawah@adfa.edu.au
[2] Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Ghent, Belgium
tom.servranckx@ugent.be, mario.vanhoucke@ugent.be
[3] Operations and Technology Management Centre, Vlerick Business School, Reep 1, 9000 Ghent, Belgium
mario.vanhoucke@vlerick.com
[4] UCL School of Management, University College London, 1 Canada Square, London E14 5AA, UK
m.vanhoucke@ucl.ac.uk

Keywords: Manufacturing project scheduling, Carbon footprints, Human factors, Memetic algorithm.

1 Introduction

Manufacturing sectors are under intense pressure to increase customer satisfaction and maintain market share in the face of fierce competition due to the globalisation of supply chains and modern industry 4.0 technologies. In order to stay competitive, manufacturers are switching from make-to-stock to make-to-order (MTO) or engineering-to-order (ETO) manufacturing processes, which allow consumers to personalise items before placing an order (Rahman *et. al.* 2015). Project managers strive to reduce the project's completion time or makespan while maintaining the temporal link between activities and meeting resource restrictions. As a result, the problem can be described as a resource constrained project scheduling problem in manufacturing (RCPSPM). Even though managers deal with project scheduling issues daily in MTO/ETO systems, there is little scheduling literature regarding such issues. Only a few independent approaches have been reported that are tailored to specific manufacturing industries such as aircraft assembly (Zhu *et. al.* 2019), shipbuilding (Wang *et. al.* 2019), and wood manufacturing (Ghiyasinasab *et. al.* 2021). Furthermore, these studies do not consider carbon footprint or energy-related objectives, which are two crucial concepts in our study.

The RCPSPM is an NP-hard issue that addresses complicated planning and scheduling challenges in industrial contexts (Ghiyasinasab *et. al.* 2021). While existing research in this field uses restricted assumptions (Oztemel and Selam 2017), we extend the RCPSPM to consider the relation between human factors and the energy use in a bi-objective optimisation problem. The type of operator used to execute activities impacts the overall completion time of a project, influencing whether it can be completed on time or not. If not, it impacts how the expenses associated with late completion of a project may be avoided. Aside from the impact on activity durations, human factors also impact an operator's pay rate per unit hour and energy usage (carbon footprint), which are the resources used to carry out operations. Most of the studies in RCPSPM, to the best of our knowledge, focus solely on reducing the project's makespan (a time-based objective) or overall cost to execute the project. However, a bi-objective approach to assess the overall cost of the manufacturing project coupled with green performance indicators (e.g. carbon footprint)

receives minimal attention in the existing literature. This gap presents an opportunity to take advantage of the advanced technology found in MTO/ETO systems by creating a unique project scheduling strategy capable of directing project managers in a variety of ways: (i) all project operations are scheduled to minimise both objectives: total project cost and carbon footprint; (ii) appropriate operators must be assigned to each task based on their human aspects while fulfilling pay restrictions and peak power load.

Carbon footprint aware RCPSPM (CRCPSPM) refers to integrating operator allocation and carbon footprint in RCPSPM. Since the CRCPSPM is a complicated scheduling problem (an extension of traditional RCPSPs), an effective optimisation approach is necessary to find (sub)optimal solutions that satisfy both objectives in an acceptable amount of time (Rahman *et. al.* 2020). Motivated by prior success in handling both single- and multi-objective scheduling problems, a genetic algorithm-based MA is proposed for solving the CRCPSPM. We test the procedure on 156 problem instances that are based on the basic RCPSP instances from the project scheduling library (PSLIB) due to the lack of standard benchmark instances for the CRCPSPM. The proposed GA-based memetic algorithm strategy is compared to the well-known and powerful non-dominated sorting genetic algorithm-II (NSGA-II) (Deb *et. al.* 2002) for validation. In summary, our contributions are twofold: (1) A novel CRCPSPM model is suggested that considers operator human variables and their influence on cost and carbon footprint (i.e. energy consumption). (2) A GA-based MA is proposed to solve the multi-objective CRCPSPM and tested on newly generated data instances as well as validated against a well-known benchmark procedure.

2 Problem description

Each project activity in a manufacturing setting uses a variety of machinery that is managed by operators. Because of the intricacy of running those devices, various operators with varying human variables can do a task quicker (slower), which results in consuming more (less) energy. As a result, project managers must handle two interconnected subproblems. First, they must allocate operators to activities while considering the impact of various salary rates, power consumption and activity duration combinations. Second, they must establish how all project operations should be scheduled to reduce total cost and carbon impact.

Operator assignment: We investigate four human factors of a set of operator types M: the skill factor α_m, the age factor β_m, the learning factor γ_m and the forgetting factor λ_m. Based on these human factors, a distinction between the operator types can be made in the production process: the best ($m = 1$), medium ($m = 2$) and worst ($m = 3$) qualified operators. The operators of type m can execute activity $i \in I$ faster and thus reduce its initial duration ($\mathbf{d_{i,m}}$) to the actual duration ($d_{i,m}$). By speeding up the activities, the total duration of the project makespan (C_{max}) can be reduced. Therefore, the probability of meeting the due date (h) will be increased and the cost of tardiness (C_t) will be decreased. In order to reduce the duration of activity i, the operator of type m uses machines that require more power $r^p_{i,m}$, which increases the cost of energy consumption (C_e). Furthermore, the wage cost of activity i depends on the type of operator m ($r^w_{i,m}$) and thus the assignment of operators will also have an impact on the total cost of wages in the project (C_w).

Project scheduling: A timetable should be constructed for all activities $i \in I$ given the precedence relations $(i, j) \in A$ between the activities ($i, j \in I$). The activities should be scheduled satisfying two types of resource constraints: a peak power load R^p and a maximum allowed wage R^w in time period t (with $t \in T$). The aim is to optimise the total cost of the project (C_{tot}), and the carbon footprint (E). First, the total cost consists of

four parts: a fixed cost (C_f), a tardiness cost (C_t), an energy cost (C_e) and a wage cost (C_w). Second, the carbon footprint depends on a carbon emission per unit of energy (ϵ).

3 Solution approach

In order to solve the CRCPSPM, a GA-based MA is proposed in this study. Goldberg and Holland (1988) proposed the GA, i.e. an evolutionary algorithm, that is traditionally started with multiple random solutions. The collection of solutions is called a *population* and each solution in the population is referred to as a *chromosome*. Natural selection through crossover and mutation develops chromosomes in a search for solutions with higher fitness values (Rahman *et. al.* 2020, Rahman *et. al.* 2015). Furthermore, the search process may improve even further by combining a local search strategy with the GA, labelled a memetic algorithm (MA) (Rahman *et. al.* 2021). The algorithm begins with non-random population initialisation and evolves through reproduction operators. A duplication method is included in the suggested MA to maintain population variety. A non-dominated sorting mechanism aids in identifying a viable set of candidate solutions for the following generations.

3.1 Results and experimental analysis

The performance of the GA is compared with the NSGA-II, which is a sophisticated algorithm for solving complicated multi-objective problems and is commonly used as a benchmark method (Deb *et. al.* 2002). Both algorithms are built in the C++ language and run in the same computational environment to ensure a fair comparison between the GA-based MA and the NSGA-II (Intel core i7 processor with 3.40 GHz clock speed and 16 GB RAM). The suggested algorithms' performance is assessed using an inverted generational distance (IGD), which combines convergence and spread (Zheng and Wang 2016). We used a pragmatic approach to construct a dataset for measuring the performance of the MA and the NSGA-II by expanding the traditional RCPSP instances from the PSPLIB dataset. This results in a total of 156 data instances: 48 J30 instances, 48 J60 instances and 60 J120 instances. The newly generated problem instances are available on `https://research.unsw.edu.au/projects/decision-support-analytics-research-group`.

This section presents a comparison between the proposed GA-based MA and NSGA-II based on the generated dataset. In order to ensure a fair comparison, each algorithm is terminated after 5,000 schedules. Table 1 shows the average $IGD(V, \delta^\star)$ values for the 156 problem instances. Since the due date for the project plays an important role in the multi-objective problem under study, the experimental evaluation is carried out with increments (0%, 2%, and 5%) in the deadline. It is shown that the proposed GA-based MA outperforms all other algorithms. More importantly, the performance of MA is robust for different increments of the project's deadline. While observing the performance of MA, it is clear that the preservation of elite solutions and its participation in the selection mechanism enhances the performance of MA.

4 Conclusion

Project scheduling methodologies are becoming more common in MTO/ETO systems for resolving scheduling issues. With more people becoming aware of global warming and energy use, lowering the carbon footprint is also becoming increasingly important to avoid negative environmental repercussions. Furthermore, the human aspects of the operators are crucial in reducing the entire project cost and carbon impact. However, project managers frequently overlook human aspects and environmental repercussions when reducing the

Deadline increment (%)	Number of activities	Avg. IGD($\mathbf{V}, \delta^\star$) MA	NSGA-II
0	30	0.210391	0.236984
	120	0.278378	0.304491
2	30	0.207589	0.202244
	120	0.247173	0.282273
5	30	0.221608	0.246423
	120	0.168162	0.236258

Table 1. Comparison between the GA-based MA and NSGA-II

project duration. As a result, this study advises to optimise the project's overall cost and carbon footprint simultaneously. Future work could investigate the impact of low-carbon scheduling on other objective functions such as net present value (NPV) optimisation. Finally, a decision support system (DSS) could be designed to help project managers make decisions by automating the decision-making process and providing a range of suboptimal options.

References

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan, 2002, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE transactions on evolutionary computation*, Vol. 06, pp. 182-197.

Ghiyasinasab M., N. Lehoux, S. Ménard and C. Cloutier, 2021, Production planning and project scheduling for engineer-to-order systems-case study for engineered wood production, *International Journal of Production Research*, Vol. 59, pp. 1068-1087.

Goldberg D. E., J.H. Holland, 1988, Genetic algorithms and machine learning, *Machine learning*, Vol. 03, pp. 95-99.

Oztemel E., A. A. Selam, 2017, Bees Algorithm for multi-mode, resource-constrained project scheduling in molding industry, *Computers & Industrial Engineering*, Vol. 112, pp. 187-196.

Rahman H.F., R.K. Chakrabortty and M.J. Ryan, 2020, Memetic algorithm for solving resource constrained project scheduling problems, *Automation in Construction*, Vol. 111, pp. 103052.

Rahman H.F., R. Sarker and D. Essam, 2015, A real-time order acceptance and scheduling approach for permutation flow shop problems, *European Journal of Operational Research*, Vol. 247, pp. 488-503.

Rahman H.F., R.K. Chakrabortty, S. El Sawah and M.J. Ryan, 2021, Energy-Efficient Project Scheduling with Supplier Selection in Manufacturing Projects, *Expert Systems with Applications*, Vol. 193, pp. 116446.

Wang W., L. Huang and J. Gu, L. Jiang, 2019, Green port project scheduling with comprehensive efficiency consideration, *Maritime Policy & Management*, Vol. 46, pp. 967-981.

Zheng X.-L., L. Wang, 2016, A collaborative multiobjective fruit fly optimisation algorithm for the resource constrained unrelated parallel machine green scheduling problem, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 48, pp. 790-800.

Zhu L., J. Lin and Z.-J. Wang, 2019, A discrete oppositional multi-verse optimisation algorithm for multi-skill resource constrained project scheduling problem, *Applied Soft Computing*, Vol. 85, pp. 105805.

Heuristic solution approaches to the multi-project scheduling problem

Dries Bredael[1], Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Ghent (Belgium)
dries.bredael@ugent.be, mario.vanhoucke@ugent.be
[2] Operations and Technology Management Centre, Vlerick Business School, Reep 1, 9000 Ghent (Belgium)
mario.vanhoucke@vlerick.com
[3] UCL School of Management, University College London, 1 Canada Square, London E14 5AA (UK)
m.vanhoucke@ucl.ac.uk

Keywords: Project Scheduling, Multi-Project, Benchmark Analysis, Metaheuristics

1 Introduction

The Resource-Constrained Multi-Project Scheduling Problem (RCMPSP) requires that a set of projects from a given project portfolio is scheduled under precedence- and resource constraints. Each project can be treated as an individual RCPSP, with the added complexity that some of the projects may be required to share certain resources. Furthermore, the presence of different projects allows for the study of a wide variety of optimisation criteria when project deadlines, also called due dates, are integrated. In this study, we analyse a set of existing (meta)heuristic solution approaches to the RCMPSP. We conduct a benchmark analysis to determine the best solution strategy for a variety of settings. The performance of the approaches is verified on a set of seven optimisation criteria and the impact of instance parameters is discussed. In literature, the majority of the proposed solution procedures are limited and only tested for a specific version of the RCMPSP where the project deadlines are set equal to the critical path length of the projects. Because the critical path length acts as a lower bound on the projects makespan, an important trade-off imposed by project earliness is removed from the problem. We reintroduce the possibility and effects of project earliness by repeating the benchmark analysis on different types of due dates. A novel dataset, presented by Van Eynde and Vanhoucke (2020), is used to this purpose. Our findings are reported in a detailed discussion.

Our analysis prompts the creation of a novel solution strategy based on the identified best elements of legacy heuristics and building on insights gained in our earlier benchmark analysis. We present a novel two-stage metaheuristic algorithm with the goal of outperforming the existing approaches on many of the previously discussed metrics.

2 Problem statement

In our literature review, we divided the studies into two groups based on their methodological approach. Centralised methologies consider a multi-project scheduling problem with one decision maker and one common goal for the entire portfolio. In contrast, decentralised versions consider the situation with multiple decision makers, one for each project, that optimise their own local objective. The problem statements of these two categories are not too widely different as the end goal is to optimise a portfolio-level objective in both cases but their solution approaches are. Centralised approaches often outperform the

decentralised approaches because they do not have to overcome an information assymetry. For this reason, our study focusses on the centralised approaches to the RCMPSP. A formal description of the (centralised) RCMPSP is presented in the following paragraph.

Table 1. Notations and formulae

General parameters	
J	The set of projects, indexed by j
N	The number of projects in the portfolio
$\overline{r}_j$	The release date of project j
$\overline{d}_j$	The due date of project j
I_j	The set of activities of project j, indexed by i
n_j	The number of non-dummy activities in project j
a_{ij}	Activity i of project j
d_{ij}	The duration of a_{ij}
a_{0j}	The dummy-start activity of project j
$a_{(n_j+1)j}$	The dummy-end activity of project j
Network parameters	
P_{ij}	The set of predecessors of a_{ij}
Resource parameters	
K	The set of renewable resource types, indexed by k
R_k	The per period availablility of resource type k
r_{ijk}	The resource requirement of a_{ij} for resource type k
Decision variables	
s_{ij}	The start time of a_{ij}
$f_{ij} = s_{ij} + a_{ij}$	The finish time of a_{ij}

Consider a multi-project portfolio that consists of a set of renewable resources K, indexed by k and a set of projects J, indexed by j. Each project j contains a set of activities Ij, indexed by i. Every activity, denoted by a_{ij}, has a duration d_{ij}, a set of renewable resource requirements r_k and a set of predecessor activities P_{ij}. It is possible that an activity has predecessor activities belonging to a different project. In the RCMPSP, the task is to determine the start times s_{ij} of each activity a_{ij} in the multi-project portfolio to optimise a certain criterion while respecting the precedence relations of each activity and the number resources of each type available to the project, R_k. The precedence relations restrict the start time of an activity a_{ij} to be no earlier than the finish time of each of its predecessor activities P_{ij}. The finish time of an activity is obtained by incrementing the start time s_{ij} of an activity with its duration d_{ij}. Furthermore, the sum of the resource requirements r_{ijk} of each activity i of each project j scheduled at time t can not be larger than the resource availability R_k, for each resource k, at any time t during the scheduling horizon T. Finally, a project can have a release date $\overline{r}_j$ that forbids the start time of any activity in the project to be earlier than $\overline{r}_j$. An overview of the notations used and other useful metrics can be found in Table 1.

3 Benchmark Analysis

We retrieved ten publications that propose a centralised metaheursitic solution procedure for the problem formulated in the previous section. These metaheuristic algorithms were implemented and rigorously tested and benchmarked on various settings. Key to our analysis is the distinction between solution strategies that focus on ordering the set of projects and those that do not. Four of the algorithms were classified as belonging to the

first group, while the remaining six did not have an inherent project ordering strategy to generate solutions. We find that this explains a significant amount of the performance differences between the procedures with the first group performing significantly better in terms of project lateness while the latter group is better able to reduce the portfolio makespan. Furthermore, performance differences between solution procedures were also explained by instance parameters, optimisation criteria and the type of due date used. We present a detailed discussion on each of these factors and review the mechanisms that underlie their impact on the performance of the metaheuristics.

4 Two-stage metaheuristic algorithm

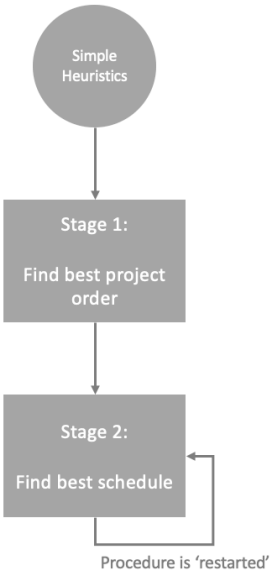


Fig. 1. Two-stage metaheuristic algorithm.

We present a novel two-stage algorithm that is briefly summarised in Figure 1. Our novel solution strategy builds upon the observation that the first step in constructing a good solution to most types of RCMPSPs is finding the appropriate project scheduling order. The first stage of our algorithm, therefore, introduces strategies similar to those proposed by Kumanan *et al.* (2006), Wauters *et al.* (2015), Perez *et al.* (2016) and Van Eynde and Vanhoucke (2020). The initial solutions are given by a set of well-performing priority rules. After the first stage algorithm concludes its search, some greedy improvements strategies are executed on the set of best-known solutions to further improve the quality and diversity of the solutions in this set. The first stage is able to outperform the majority of the existing heuristics even though its search space is limited to schedules with a strict project order. The resulting set of best solutions from the first stage is used as input to the second stage that relaxes this strict ordering restriction. The second stage algorithm builds upon some elements introduced by Homberger (2007) and Goncalves *et al.* (2008) but relies largely on

novel techniques including new local searches. Two versions of the second-stage algorithm are developed and performance differences are discussed. The second-stage algorithm is re-started using a combination of the previous best solutions and the best solutions from earlier iterations until a stop criterion is met.

Acknowledgements

We acknowledge the support provided by computational resources of the VSC (Flemish Supercomputing Center), funded by Ghent Univerisity, FWO and the Flemish Government department EWI.

References

Goncalves, J.F., J.J. Mendes and M.G. Resende, 2008, "A genetic algorithm for the resource constrained multi-project scheduling problem", *European Journal of Operational Research*, Vol. 189, pp. 1171-1190.

Homberger J., 2007, "A multi agent system for the decentralized resource constrained multi project scheduling problem", *International Transactions in Operational Research*, Vol. 14, pp. 565-589.

Kumanan, S., G.J. Jose and K. Raja, 2006, "Multi project scheduling using an heuristic and a genetic algorithm", *The International Journal of Advanced Manufacturing Technology*, Vol. 31, pp. 360-366.

Perez, E., M. Posada and A. Lorenzana, 2016, "Taking advantage of solving the resource constrained multi-project scheduling problems using multi-modal genetic algorithms", *Soft Computing*, Vol. 20, pp. 1879-1896.

Van Eynde, R. and M. Vanhoucke, 2016, "Resource constrained multi project scheduling: Benchmark datasets and decoupled scheduling", *Journal of Scheduling*, Vol. 23, pp. 301-325.

Wauters, T., K. Verbeeck, P. De Causmaecker and C. Vanden Berghe, 2015, "A learning based optimisation approach to multi-project scheduling", *Journal of Scheduling*, Vol. 18, pp. 61-74.

List of participants

- Jakob Snauwaert (jakob.snauwaert@ugent.be)
- José Coelho (jcoelho@uab.pt)
- Laurent Houssin (houssin@laas.fr)
- Avraham Shtub (shtub@ie.technion.ac.il)
- ONCU HAZIR (oncu.hazir@rennes-sb.com)
- Stefan Creemers (s.creemers@ieseg.fr)
- Gerhard Woeginger (woeginger@algo.rwth-aachen.de)
- Chris Potts (C.N.Potts@soton.ac.uk)
- Maximilian Kolter (max.kolter@hotmail.de)
- Erwin Pesch (erwin.pesch@uni-siegen.de)
- Juergen Zimmermann (juergen.zimmermann@tu-clausthal.de)
- Alessandro Hill (alessandro.hill@gmail.com)
- Max Reinke (max.reinke@tu-clausthal.de)
- Mareike Karnebogen (mareike.karnebogen@tu-clausthal.de)
- Pierre Lopez (pierre.lopez@laas.fr)
- Roel Leus (Roel.Leus@kuleuven.be)
- Nicklas Klein (nicklas.klein@unibe.ch)
- Lowell Lorenzo (Lowell.Lorenzo@up.edu.ph)
- Erik Demeulemeester (erik.demeulemeester@kuleuven.be)
- Pascale Bendotti (pascale.bendotti@lip6.fr)
- Léa Blaise (lblaise@localsolver.com)
- Aykut Uzunoglu (aykut.uzunoglu@wiwi.uni-augsburg.de)
- Hendrik Weber (hendrik.weber@tum.de)
- Jan Węglarz (jan.weglarz@cs.put.poznan.pl)
- Alain Haït (alain.hait@isae.fr)
- Karim TAMSSAOUET (karim.tamssaouet@bi.no)
- Christoph Schwindt (christoph.schwindt@tu-clausthal.de)
- Hanyu Gu (hanyu.gu@uts.edu.au)
- Julia Lange (julia.lange@wiwi.uni-kl.de)
- Vincent T'Kindt (tkindt@univ-tours.fr)
- Camilo Rodríguez-Espinosa (camiloa.rodriguez@javeriana.edu.co)
- Paz Perez-Gonzalez (pazperez@us.es)
- paula sanchez de los reyes (paulasanchezdlr.2698@gmail.com)
- Lubo Li (2382145420@qq.com)
- haohua zhang (892858495@qq.com)
- Lei LIU (lei.liu@polimi.it)
- Izack Cohen (izack.cohen@biu.ac.il)
- Marcello Urgo (marcello.urgo@polimi.it)
- OLIVIER PLOTON (olivier.ploton@univ-tours.fr)
- Christin Schumacher (christin.schumacher@tu-dortmund.de)
- Christian Stürck (christian.stuerck@hsu-hh.de)
- Lucas Berterottière (lucas.berterottiere@emse.fr)
- Itai Lishner (itailishner@hotmail.com)
- Venkataramiah Annapragada (asvenkata@azuriteproject.in)
- Mario Christian Sillus (mcs16@tu-clausthal.de)
- Jérémy Berthier (j.berthier@emse.fr)
- Quentin PENA (quentin.pena@hds.utc.fr)
- Sigrid Knust (sigrid@informatik.uni-osnabrueck.de)
- Bo Chen (b.chen@warwick.ac.uk)
- Daniel Page (drpage@pagewizardgames.com)

- Lena Wohlert (lena.sophie.wohlert@tu-clausthal.de)
- Anouck Chan (anouck.chan@onera.fr)
- Anulark Naber (anulark@gmail.com)
- Evgeny Gurevsky (evgeny.gurevsky@univ-nantes.fr)
- Ernest Foussard (ernest.foussard@grenoble-inp.fr)
- Camille Bonnin (camille.bonnin@grenoble-inp.fr)
- Feifei Li (gltfeifei@buu.edu.cn)
- Bernard Penz (bernard.penz@grenoble-inp.fr)
- Margaux Nattaf (margaux.nattaf@grenoble-inp.fr)
- Nadia Brauner (nadia.brauner@grenoble-inp.fr)
- Marie-Laure Espinouse (Marie-Laure.Espinouse@g-scop.grenoble-inp.fr)
- Rojin Nekoueian (Rojin.nekoueian@ugent.be)
- Forough Vaseghi (forough.vaseghi@ugent.be)
- Carla Juvin (cjuvin@laas.fr)
- Mahdi Fathi (mahdi.fathi@unt.edu)
- Fekadu Tolessa Gedefa (gedefaft@math.elte.hu)
- Yanfei Chen (yanfei.chen@kuleuven.be)
- YONG MA (mayong5118@outlook.com)
- Elvin Coban (elvin.coban@ozyegin.edu.tr)
- Cyril Briand (briand@laas.fr)
- Pecyna Tomasz (tpecyna@man.poznan.pl)
- Louis Riviere (louis.riviere@laas.fr)
- Izel Unsal Altuncan (izel.unsalaltuncan@ugent.be)
- Norbert Trautmann (norbert.trautmann@pqm.unibe.ch)
- Xin Guan (xin.guan@ugent.be)
- Tobias Joosten (tobias.joosten@itwm.fraunhofer.de)
- Joanna Józefowska (jjozefowska@cs.put.poznan.pl)
- Michael Moos (michael.moos@itwm.fraunhofer.de)
- Alessandro Agnetis (agnetis@diism.unisi.it)
- Liangyan Tao (lytao@nuaa.edu.cn)
- Fei Wu (fwu0966@gmail.com)
- Dominik Mäckel (dominik.maeckel@tu-dortmund.de)
- Massimiliano Caramia (caramia@dii.uniroma2.it)
- Ángeles Pérez (angeles.perez@uv.es)
- Dries Bredael (dries.bredael@ugent.be)
- weikang Guo (weikang.guo@ugent.be)
- Jingyu Luo (jingyu.luo@UGent.be)
- Rahman Torba (r.torba@emse.fr)
- Antoine Lhomme (antoine.lhomme@grenoble-inp.org)
- Stéphane Dauzère-Pérès (Dauzere-Peres@emse.fr)
- Gérémi Bridonneau (geremi.bridonneau@gmail.com)
- Christian Gahm (christian.gahm@uni-a.de)
- Salah Ahmed (salah.ahmed@usn.no)
- Stéphanie Roussel (stephanie.roussel@onera.fr)
- xi wu (xi.wu@ugent.be)
- Wanjun Liu (Wanjun.Liu@Ugent.be)
- fangfang Cao (1165600145@qq.com)
- Eliana María González-Neira (eliana.gonzalez@javeriana.edu.co)
- Tom Portoleau (tom.portoleau@gmail.com)
- Mario Flores Gomez (mario.flores@emse.fr)
- Carreira Julio (1101469@estudante.uab.pt)
- Minh-Phuoc DOAN (minh-phuoc.doan@grenoble-inp.fr)

- YAGMUR SELCUK (yagmur.selcuk@ozu.edu.tr)
- Leila Naeni (leila.mosleminaeni@uts.edu.au)
- Kian Farajkhah (kian.farajkhan@ozu.edu.tr)

List of sponsors