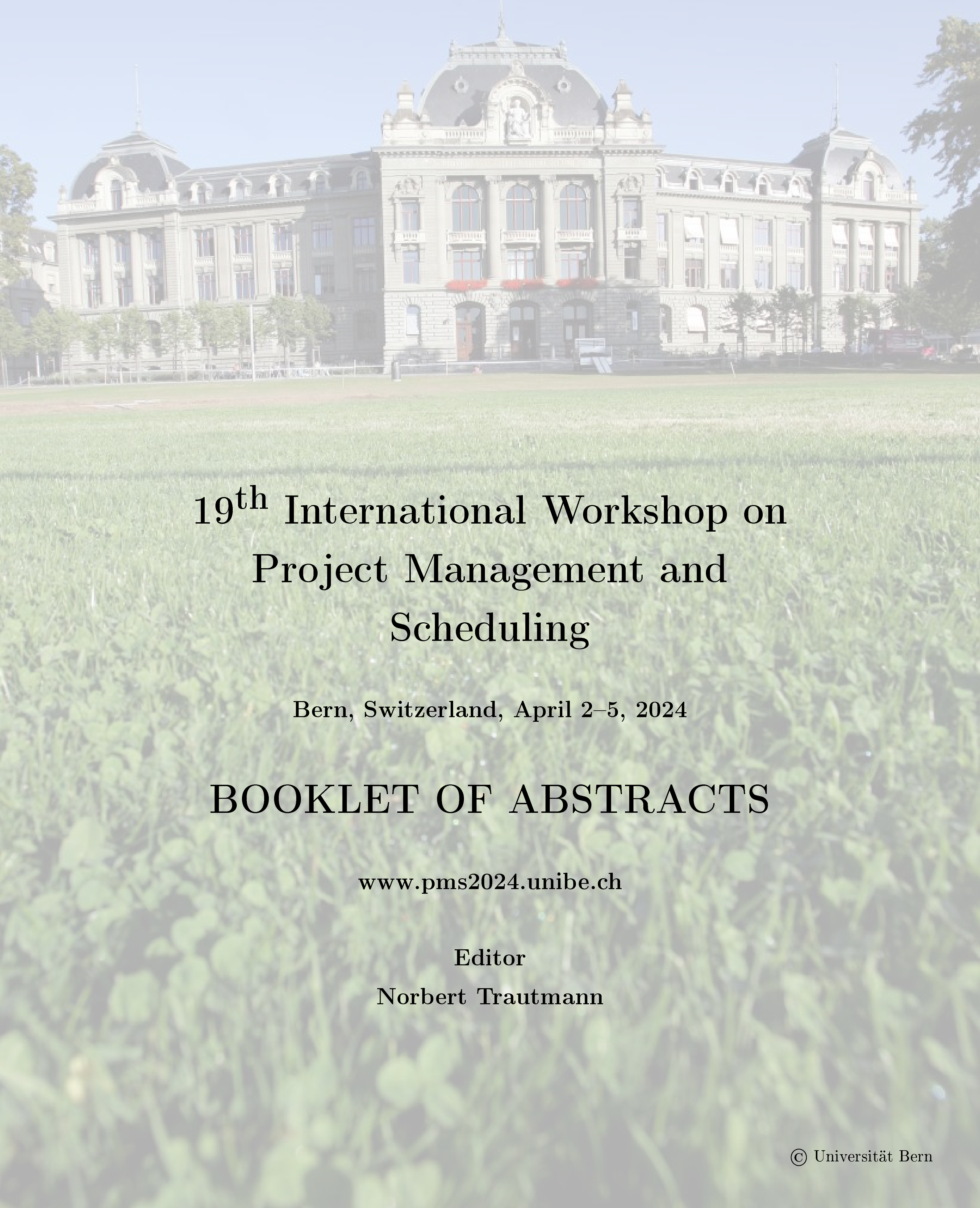
PMS2024
University of Bern



19th International Workshop on Project Management and Scheduling

Bern, Switzerland, April 2–5, 2024

BOOKLET OF ABSTRACTS

www.pms2024.unibe.ch

Editor

Norbert Trautmann

Preface

This booklet of abstracts contains the extended abstracts that will be presented at the 19th International Workshop on Project Management and Scheduling (PMS 2024), held at the University of Bern, Switzerland, from April 2–5, 2024.

PMS is an international workshop series initiated by the EURO Working Group on Project Management and Scheduling. The workshop takes place every two years at different locations in Europe and brings together researchers and industry professionals from Computer Science, Operations Research, Optimization Engineering, Mathematical Programming, and Industrial Engineering. The scientific program covers a wide range of topics related to Project Management and Scheduling, including both theoretical and applied research.

We received 54 submissions for PMS 2024. After a peer-review process with two reviews per submission, 52 extended abstracts were accepted for presentation at the workshop. The abstracts are listed below in alphabetical order based on the first author's last name. Presenting authors are indicated with ^{1}. Thank you to the dedicated reviewers, who generously contributed their time and expertise to evaluate the submitted abstracts.

It is my great pleasure to announce our esteemed plenary speakers of the PMS 2024 workshop – Nicole Megow from the University of Bremen (EURO Plenary), who will talk about recent advancements in scheduling with predictions; Federico Della Croce Di Dojola from Politecnico di Torino, who introduces a novel exact algorithm for the transportation problem; and industry experts Julien Darlay and Léa Blaise from Hexaly, who will present the mathematical optimization solver Hexaly Optimizer, formerly known as LocalSolver.

When visiting Bern, don't miss the opportunity to stroll through the charming old town and enjoy the marvelous views of the Bernese Alps. The guided tours on Wednesday and the Conference Dinner on Thursday promise unforgettable experiences at some of the unique spots our city has to offer. During the Conference Dinner, we will announce the winners of the Best Student Paper Award. We have received outstanding submissions from our young researchers and can't wait to share them with you.

I would like to extend my gratitude to our generous sponsors and partners, as well as to my colleagues from the International Program and Organizing Committee, who have invested countless hours in ensuring the success of this event. A big thank you to you, dear participants, for joining us from all over the world. Welcome to Bern!

Bern, 2nd of April — Norbert Trautmann, Conference Chair PMS 2024

Committees

Organizing Committee

International Program Committee

Sponsors & Partners

We thank our sponsors, supporters, and organizing partners:

Our special thanks go to the Faculty of Business, Economics and Social Sciences at the University of Bern for their generous support.

Remark

Contents

Part I

Invited plenary talks

Scheduling with predictions

Nicole Megow

University of Bremen

Uncertainty poses a significant challenge on scheduling and planning tasks, where jobs may have unknown processing times or machines run at unknown speeds. However, assuming a complete lack of a priori information is overly pessimistic. With the rise of machine-learning methods and data-driven applications, access to predictions about input data or algorithmic actions becomes feasible. Yet, blindly trusting these predictions might lead to very poor solutions, due to the absence of quality guarantees.

In this talk, we explore recent advancements in the popular framework of Algorithms with Predictions, which integrates such error-prone predictions into online algorithm design. We examine various prediction models and error measures, showcasing learning-augmented algorithms for non-clairvoyant scheduling with strong error-dependent performance guarantees. We demonstrate the potential of imperfect predictions to enhance scheduling efficiency and address uncertainty in real-world scenarios.

Iterated Inside Out: a new exact algorithm for the transportation problem

Federico Della Croce Di Dojola

Politecnico di Torino

We propose a novel exact algorithm for the transportation problem, one of the paradigmatic network optimization problems. The algorithm, denoted Iterated Inside Out, requires in input a basic feasible solution and is composed by two main phases that are iteratively repeated until an optimal basic feasible solution is computed. In the first "inside" phase, the algorithm progressively improves upon a given basic solution by increasing the value of several non-basic variables with negative reduced cost. This phase typically outputs a non-basic feasible solution interior to the constraint set polytope. The second "out" phase moves in the opposite direction by iteratively setting to zero several variables until a new improved basic feasible solution is reached. Extensive computational tests show that the proposed approach strongly outperforms all versions of network and linear programming algorithms available in commercial solvers such as Cplex and Gurobi and other exact algorithms available in the literature.

Scheduling with Hexaly Optimizer

Julien Darlay, Léa Blaise

Hexaly (previously LocalSolver)

Hexaly Optimizer, formerly known as LocalSolver, is a "model and run" mathematical optimization solver based on various exact and heuristic methods. This presentation will introduce the different components of Hexaly Optimizer through scheduling problems, from the user model to the algorithms enabling it to find quality solutions and lower bounds. We will first show how its modeling formalism can be used to express various academic and industrial scheduling problems using only generic operators. These models are based on interval and list decision variables and have the advantage of being very compact, which enables the solver to handle even large-scale problems. We will then give an overview of the primal and dual techniques Hexaly Optimizer uses to solve such problems: local search reinforced with a solution repair algorithm based on constraint propagation to find quality solutions, and energetic and single-machine relaxations to find quality lower bounds.

Part II

Abstracts of presented papers

Project Risk prioritisation using Monte Carlo simulation

Acebes F[1], Gonzalez-Varona JM[2], Lopez-Paredes A[2] and Pajares J[1]

[1] University of Valladolid, Spain
`fernando.acebes@uva.es, javier.pajares@uva.es`
[2] University of Malaga, Spain
`jmgonzalezva@uma.es, loppar@uma.es`

Keywords: Probability-Impact Matrix, Quantitative Risk Prioritisation, Monte Carlo Simulation, MCSimulRisk.

1 Introduction

In project risk management processes, the project manager is commonly faced with the challenge of determining the relative importance of various sources of risk to direct management efforts and maintain project profitability. Managers need help deciding which risks to address as a priority. This difficulty is due to the large number of risk sources and the complexity of assessing which are the most critical, requiring a detailed approach. In this context, any method that simplifies risk prioritisation and is accessible to practitioners involved in project management is appreciated (Ward 1999).

Risk matrices are widely accepted tools used in various industrial sectors to assess and rank risks based on the probability of risk occurrence and potential impact. However, the probability-impact matrix, in particular, has severe limitations (Cox 2008, Duijm 2015, Levine 2012). Its inability to consider the complex interrelationships between risks, the need for precise estimates of probability and impact, and its difficulty in integrating the impact of risks across multiple project objectives are questioned, often leading to conflicts.

In response to the limitations of risk matrices, this study proposes a methodology to prioritise the risks identified in the project through a quantitative analysis using Monte Carlo simulation. In line with the work of Creemers S. *et. al.* (2014), we propose a risk-driven approach whose contribution is twofold: (1) we incorporate all the risks identified in the simulation model (and not only the aleatoric uncertainty of the activities), and (2) we prioritise the risks based on the total impact of each of them on the total duration and cost objectives of the project.

2 Methodology

The flowchart following the proposed method for prioritising project risks by applying quantitative Monte Carlo simulation-based techniques is depicted in Fig. 1.

The procedure starts with 'risk identification', which entails detecting risk factors that can potentially adversely influence the project's development. The risk identification phase is carried out in the context of planning after having previously defined the precise limits of the project, established the personnel involved, delineated the tasks to be carried out and drawn up the timetable of activities. In this process, a wide range of tools are used, including but not limited to brainstorming, Delphi techniques and stakeholder consultation. The result of this first step is a list of identified risks.

Next, we estimate the probability of occurrence of each identified risk and their possible impact on the project's objectives. Following the proposal of Hillson (2014), we consider three types of uncertainty that can become project risks, which we must model according to a distribution function: aleatoric, stochastic and epistemic uncertainty. We do not consider

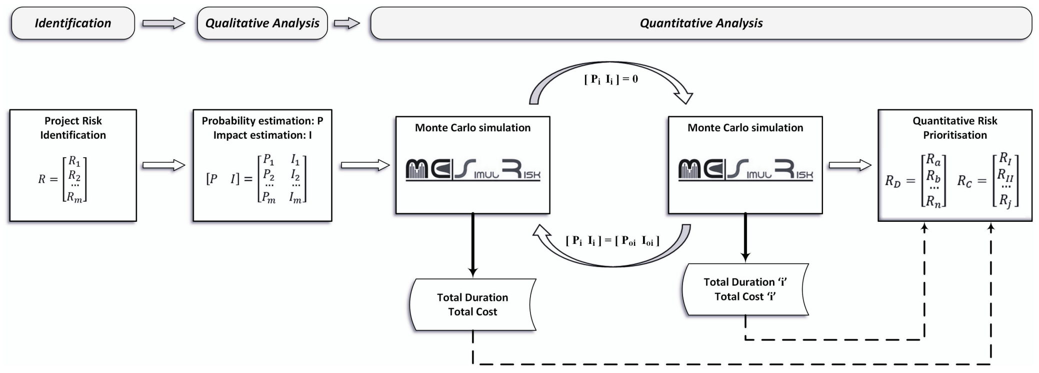

Identification
Qualitative Analysis
Quantitative Analysis
Project Risk Identification
$R = \begin{bmatrix} R_1 \\ R_2 \\ \dots \\ R_m \end{bmatrix}$
Probability estimation: P
Impact estimation: I
$[P \quad I] = \begin{bmatrix} P_1 & I_1 \\ P_2 & I_2 \\ \dots & \dots \\ P_m & I_m \end{bmatrix}$
Monte Carlo simulation
$[P_i \; I_i] = 0$
$[P_i \; I_i] = [P_{oi} \; I_{oi}]$
Monte Carlo simulation
Total Duration
Total Cost
Total Duration 'i'
Total Cost 'i'
Quantitative Risk Prioritisation
$R_D = \begin{bmatrix} R_a \\ R_b \\ \dots \\ R_n \end{bmatrix}$
$R_C = \begin{bmatrix} R_I \\ R_{II} \\ \dots \\ R_j \end{bmatrix}$

Fig. 1. Flowchart of the quantitative risk prioritisation process

ontological uncertainty, as it cannot be modelled since we do not know it at all. Each type of uncertainty will be modelled according to a distribution function representing such behaviour. Thus, for example, stochastic uncertainty will be modelled with a Bernoulli-type distribution function, a uniform distribution function can model epistemic uncertainty, and aleatoric uncertainty by normal, triangular, or Beta-Pert distribution functions (Vose 2008). The result of this second step is to obtain probability distribution functions for probabilities and impacts for each of the risks identified in the previous process.

The risk information obtained in the previous step is fed into the project model and activity information. With all the info unified in the same model, we carry out a Monte Carlo simulation using the software 'MCSimulRisk' (Acebes F. *et. al.* 2023). We carry out a first simulation, including all the information in the simulation model, and we obtain the main statistics of the simulation (mean values, standard deviations and percentiles corresponding to total duration and cost). Then, we will conduct as many simulations as risks we identified in the project. However, we will consider that the corresponding risk does not exist in each of these simulations. In other words, in each of these last simulations, we will assume that the probability of occurrence of the risk is zero (Pi=0) and its impact is also zero (Ii=0). The resulting model we introduce into the Monte Carlo simulator is the same as the initial one, except that the corresponding risk (its probability and impact) is zero. In the same way, as we have done in the first simulation, we obtain the statistics for each of these simulations in the following ones.

Finally, we obtained two prioritised lists of the identified risks, one according to the impact on the duration objective and a different one according to the impact on the project cost objective. To achieve this result, we compare the values of duration and total project cost obtained in the first simulation (which includes the complete project model) against each subsequent simulation (where the corresponding risk has been removed). For this calculation, given that we are dealing with stochastic data, we have had to choose a percentile that represents the organisation's risk appetite (Value at Risk).

3 Case study

This section briefly shows a case study where we applied our quantitative risk prioritisation proposal. We compared the results obtained with those from a traditional qualitative analysis using the risk or probability-impact matrix. With this simple example, we want to go through each of the phases of our model until the final result is achieved. The example shown can be extended to any project by following the steps outlined in the previous section.

The project used as an example consists of eight activities, the duration of which has been modelled using a normal distribution function to facilitate the reader's understanding (but a different one could have been used). The first step in the process is to identify the project risks. In this case, the project team has identified five risks: three impacting activity duration objectives (R1 to R3) and two impacting activity cost objectives (R3 and R4).

The project team must then estimate the likelihood and impact of each risk. To do this, it has all the information from the current project, information from previous projects, the opinions of all stakeholders and the team's experience. Once the probability and impact estimates for each risk are known and modelled as distribution functions, they are fed into the project model for Monte Carlo simulation. We chose P80 as the percentile representing our Value at Risk (VaR), resulting in a total project duration of 16,31 weeks and 40.389 euros. The final result of the risk prioritisation for the selected percentile is shown in Table 1.

Table 1. Quantitative risk prioritisation results

	Prob - Imp Matrix				**Duration (weeks)**			**Cost (Euros)**		
Risk	**P**	**I**	**PxI**	Rank	**Dur_Ri**	**Diff_Di**	**Rank_D**	**Cost_Ri**	**Diff_Ci**	**Rank_C**
R1	A	B	0,07	**5**	15,83	0,48	**2**	40.038	801	**4**
R2	M	A	0,2	**1**	14,84	1,47	**1**	38.959	1.879	**2**
R3	MB	MA	0,08	**4**	16,05	0,26	**3**	40.199	639	**5**
R4	B	A	0,12	**3**	16,31	0	**4**	37.856	2.982	**1**
R5	A	M	0,14	**2**	16,31	0	**4**	39.291	1.547	**3**

The first group of columns (columns 2 to 5) of Table 1 refers to the prioritisation of risks after using the risk matrix. Before this, the project team has to design a probability-impact matrix specific to the project under implementation. For each identified risk, and with the estimation of the probability and impact of each one, the result of each risk level is obtained and, finally, by ordering the values obtained, the ranking for each risk is represented in Table 1. Risk R2 is the highest priority, according to this method, and risk R1 is the one to which the least attention should be paid. The prioritisation result using the probability-impact matrix will be compared with our proposal, represented in the following group of columns.

Columns 6 to 8 include the results of prioritising risks relative to the total duration, while the last columns (9 to 11) include the prioritisation considering the cost impact. Column 8 (Rank D) includes the risks' ranking according to their importance on the total duration of the project. Column 11 (Rank C) shows the order of importance of the risks according to their final impact on the total cost. We note that the results are different from each other (impact on duration and impact on cost) and different from those obtained by applying the risk matrix.

With this proposed method, we obtain quantitative results for both duration and cost, which we use to rank the risks according to their importance (Table 1). We not only know the ranking of the risks by their importance in the project, but we also know the real quantified impact on the total cost and duration objectives. Depending on the importance of one factor or another in the project, the project manager will pay more attention to one risk. Another striking fact is that the risks that could impact cost objectives (R4 and R5) are not reflected in the impact on total duration. These risks are the least important in the ranking in terms of duration. The same is invalid for risks impacting duration objectives (R1 to R3). These impact the project's total cost and can become more critical (higher priority) than other risks directly affect the cost objective (R2 is more critical in the cost objective than R3). The project's cost depends on the activity's variable costs, which in turn depend on the duration of the activity.

4 Conclusion

This research aims to present an alternative to using the probability and impact matrix to identify the most critical risk in a project that could affect the achievement of its objectives. To this end, by maintaining a risk-driven approach, a quantitative approach based on Monte Carlo simulation has been proposed. It provides numerical results on the importance of the risks concerning their impact on the total duration and cost objectives. The proposed methodology offers notable advantages compared to other risk prioritisation methods, especially as opposed to the conventional risk matrix.

In the context of the case study, it has been observed that the ranking of risks varies significantly depending on the method used, as confirmed by our findings. Our methodology has allowed us to obtain independent numerical values for the impact of risks on the total cost and duration objectives, which is valuable for project managers, who can make decisions based on the prioritisation of risks and the predominant project objective, either duration or total cost, in case they do not coincide.

The results indicate that risks that impact the cost of the activities do not influence the project's total duration, while those that impact the duration also impact the total cost. Sometimes, this impact can be more significant than that caused by a risk that only affects activity cost. These findings suggest that this quantitative prioritisation methodology has significant potential for use by both academics wishing to extend their research into project risk and practitioners seeking to apply it to real projects on a day-to-day basis.

Funding

This research has been partially financed by Junta de Castilla y Leon (Spain) with Grant VA180P20.

References

Acebes F., J. de Anton , F. Villafanez and D. Poza , 2023, "A Matlab-Based Educational Tool for Quantitative Risk Analysis", *IoT and Data Science in Engineering Management*, Springer International Publishing, Cham, Switzerland.

Cox L.A., 2008, "What's wrong with risk matrices?", *Risk Analysis*, Vol. 28, pp. 497-512.

Creemers S., E. Demeulemeester and S. Van de Vonder , 2014, "A new approach for quantitative risk analysis", *Ann Oper Res*, Vol. 213, pp. 27-65.

Duijm N.J., 2015, "Recommendations on the use and design of risk matrices", *Safety Science*, Vol. 76, pp. 21-31.

Hillson D., 2014, "How to manage the risks you didn't know you were taking", *PMI Global Congress*, pp. 1-8.

Levine E.S., 2012, "Improving risk matrices: The advantages of logarithmically scaled axes", *Journal of Risk Research*, Vol. 15, pp. 209-222.

Vose D., 2008, "Risk Analysis: a Quantitative Guide - 3rd ed", Wiley, Chichester, U.K.

Ward S., 1999, "Assessing and managing important risks", *International Journal of Project Management*, Vol. 17, pp. 331-336.

Scenario-Based Optimization for a Multi-Skilled Resource-Constrained Project Scheduling Problem with Resource Flexibility

Niels-Fabian Baur and Julia Rieck

University of Hildesheim, Germany
Operations Research Group, Institute for Business Administration and Information Systems
`{baur;rieck}@bwl.uni-hildesheim.de`

Keywords: Multi-skill RCPSP, Resource flexibility, Stochasticity, Proactive scheduling

1 Introduction

An extension of the Resource-Constrained Project Scheduling Problem (RCPSP) is known as the Multi-Skilled RCPSP (MS-RCPSP). In the MS-RCPSP, resources are represented by employees possessing diverse skill sets, which are essential for executing various project activities. In most publications on the MS-RCPSP, the skill levels of resources do not exert an influence on the duration of activities. Here in this paper, it is assumed that these resources exhibit varying levels of proficiency in their respective skills and a correlation exists between the skill levels and activity durations. This approach is similar to Hanne and Nickel (2005), Heimerl and Kolisch (2010), Xiao *et al.* (2013), Dhib and Soukhal (2015), and Zheng *et al.* (2017). In addition, this paper endeavors to enhance the problem by introducing greater scheduling flexibility for resources. Particularly, it is possible that the number of employees allocated to an activity can fluctuate over time with no prescribed limit for resource allocation. This implies that activities may remain unstaffed even after initiation, making them preemptive in nature. Furthermore, activity durations are not predetermined but calculated based on resource allocation, similar to the RCPSP with flexible resource profiles (Naber and Kolisch 2014). Consequently, we refer to the problem addressed herein as the Resource-Flexible Multi-Skilled RCPSP (RF-MS-RCPSP) in alignment with this resource-centric perspective.

The real project environment is characterized by fluctuations and randomness. To elevate the problem to a more comprehensive and realistic level, it is crucial to introduce stochastic elements. For example, unexpected fluctuations in activity workloads can occur. By embracing stochasticity, uncertainties and their potential impact on project scheduling and resource allocation can be taken into account. Until now, only a few stochastic MS-RCPSPs have been described in the literature such as Felberbauer *et al.* (2019), Alvanchi *et al.* (2012), and Chen *et al.* (2014). The following approach captures the uncertain nature of project execution in practical scenarios, thereby enabling better decision-making. In real applications, project plans created in advance have to be adapted repeatedly to the actual conditions. Utilizing our model, it becomes possible to establish a schedule proactively, considering a variety of scenarios. This schedule is constructed in such a way that it remains largely feasible even in the presence of unexpected delays, necessitating only moderate adjustments. The fundamental idea behind this approach is that last-minute changes are susceptible to errors, especially when they impact employees with limited availability, such as during vacation periods. Therefore, the objective is to devise a schedule that anticipates various contingencies, reducing the need for significant last-minute modifications and mitigating potential disturbances.

2 Mathematical Model Formulation

The mathematical model and corresponding notation are based on the model proposed by Baur and Rieck (2020). A project is conceptualized as an activity-on-node network. Here, project activities, denoted by $i, j \in V$, serve as nodes, and the arcs $\langle i, j\rangle \in E$ connecting these nodes represent precedence relationships between the activities. Notably, the duration of activities depends on resource allocations and therefore cannot be determined in advance. Instead, an estimated processing time D_i^π is assigned to each activity i across various scenarios $\pi \in \Pi$. This parameter signifies the anticipated time required for a single worker with average proficiency to execute the activity in the corresponding scenario. Allocation of multiple resources has the potential to diminish activity duration, and there exists no restriction on the number of resources that can be assigned to an activity. For instance, when two workers (in opposite to one worker) with average skills engage in an activity, the duration is reduced by half. To formulate this, a predefined set of resources $k \in K$ and a set of skills are identified. Each activity i necessitates a distinct set of skills $s \in S_i$, and resources possess skills at predefined levels L_{ks}, denoted as $\{0, 0.5, 1, 1.5, 2\}$ for each skill and resource. Moreover, the binary parameter θ_{kt} indicates for resource k the availability at time t, given that resources are partially renewable.

The problem is formalized as a time-index-based mixed-integer linear program. It contains binary decision variables, denoted as x_{it}^π, indicating whether an activity i starts at time t in scenario π or not. Given that the duration of an activity is dependent on the resource allocation, additional decision variables, denoted as r_{ikt}^π, are introduced to signify whether a resource k is assigned to an activity i at time t for scenario π. To linearize the constraints, auxiliary binary variables y_{it}^π are introduced. These variables serve to indicate whether an activity i is continued after time t in scenario π, which makes it possible to distinguish between an interruption and the completion of an activity. Based on this information, decision-relevant durations $P_i^\pi \geq 0$ can be derived.

$$\min. \sum_{t\in T} t x_{n+1,t}^0 + \alpha \left(\sum_{\pi \in \Pi \setminus \{0\}} \sum_{t \in T} (t x_{n+1,t}^\pi - t x_{n+1,t}^0)\right) + \beta \left(\sum_{\pi\in\Pi} \sum_{i\in V} \sum_{k\in K} \sum_{t\in T} (\upsilon_{ikt}^\pi + \omega_{ikt}^\pi)\right) \quad (1)$$

Objective function (1) consists of three components. The first component aims to minimize the total project duration (PD) in scenario $\pi = 0$, equivalent to the deterministic case. The second component focuses on minimizing the extension of the project duration in other scenarios $\pi \neq 0$, relative to the deterministic case. The third component penalizes disparities in resource allocations across scenarios compared to the deterministic case. With this approach, a schedule is searched that minimizes the project duration and the possible extensions that can occur due to the scenarios. The goal is to find a single schedule a priori that requires minimal adjustments of resource allocation to compensate for fluctuations in the overall project completion times. The following constraints must be taken into account:

$$\text{s.t.} \sum_{t\in T} x_{it}^\pi = 1 \qquad \forall i \in V, \pi \in \Pi \qquad (2)$$

$$\sum_{t\in T} t x_{jt}^\pi - \sum_{t\in T} t x_{it}^\pi \geq P_i^\pi \qquad \forall \langle i,j\rangle \in E, \pi \in \Pi \qquad (3)$$

$$\sum_{k\in K}\sum_{t\in T} L_{ks} r_{ikt}^\pi \geq D_i^\pi \qquad \forall i \in V\setminus\{0, n+1\}, s \in S_i, \pi \in \Pi \qquad (4)$$

$$\sum_{\tau=0}^{t} x_{i\tau}^\pi \geq r_{ikt}^\pi \qquad \forall i \in V\setminus\{0, n+1\}, k \in K, t \in T, \pi \in \Pi \qquad (5)$$

$$\sum_{i\in V} r_{ikt}^\pi \leq \theta_{kt} \qquad \forall k \in K,\ t \in T, \pi \in \Pi \qquad (6)$$

$$\sum_{\tau=t}^{\bar{d}} \sum_{k \in K} r_{ik\tau}^{\pi} \geq y_{it}^{\pi} \qquad \forall i \in V, t \in T, \pi \in \Pi \qquad (7)$$

$$\sum_{\tau=t}^{\bar{d}} \sum_{k \in K} r_{ik\tau}^{\pi} \leq M y_{it}^{\pi} \qquad \forall i \in V, t \in T, \pi \in \Pi \qquad (8)$$

$$\sum_{t \in T} \left(y_{it}^{\pi} + \sum_{\tau=0}^{t} x_{i\tau}^{\pi} - 1 \right) \leq P_i^{\pi} \qquad \forall i \in V \setminus \{0, n+1\}, \pi \in \Pi \qquad (9)$$

$$r_{ikt}^{0} - r_{ikt}^{\pi} = \upsilon_{ikt}^{\pi},\ r_{ikt}^{\pi} - r_{ikt}^{0} = \omega_{ikt}^{\pi} \qquad \forall i \in V, k \in K, t \in T, \pi \in \Pi \setminus \{0\} \qquad (10)$$

$$r_{ikt}^{\pi} = 0 \qquad \forall i = \{0, n+1\}, k \in K, t \in T, \pi \in \Pi \qquad (11)$$

$$P_i^{\pi} \geq 0 \qquad \forall i \in V, \pi \in \Pi \qquad (12)$$

$$r_{ikt}^{\pi}, x_{it}^{\pi}, y_{it}^{\pi} \in \{0, 1\} \qquad \forall i \in V, k \in K, t \in T, \pi \in \Pi \qquad (13)$$

Equations (2) specify that each activity i must be started exactly once for each scenario. Constraints (3) guarantee adherence to precedence relationships in each scenario. With Constraints (4), it is determined that the estimated processing time of an activity i is accounted for in all scenarios for each skill $s \in S_i$ required for execution. The start of an activity in a specific scenario is defined as the moment of the first resource allocation in Conditions (5). If an activity is resumed after an interruption, this does not count as a new start. Constraints (6) ensure that each resource can be assigned to exactly one activity in a scenario at the same time, depending on the availability of the resource at that time. The auxiliary variables y_{it}^{π} are introduced in Conditions (7) and (8). If resources are assigned after time t, the activity is considered as incomplete and $y_{it}^{\pi} = 1$ holds. Variables x_{it}^{π} describe the time t at which an activity i starts and variables y_{it}^{π} indicate whether an activity is already completed in a scenario π. Leveraging this information, activity durations for specific scenarios are specified in Constraints (9), also taking potential interruptions into account. Equations (10) calculate the inequalities in resource allocations between scenarios $\pi \neq 0$ and scenario 0. Constraints (11) ensure that no resources are allocated to fictitious activities. Finally, all decision variables are defined in (12) and (13).

3 Preliminary Results and Outlook

In order to test the model outlined in Section 2, we generated 20 instances comprising $n = 10$ real activities based on the MSLIB library (Snauwaert and Vanhoucke 2022). The instances have been adjusted so that they differ in their number of precedence relationships. The first 10 instances contain on average four times as many precedence relationships between real activities. We therefore refer to them as series networks, while the other 10 instances can be regarded as parallel networks. The instances were further enhanced with problem-specific parameters, and scenarios were created for each instance by drawing symmetric triangularly distributed random numbers for the estimated processing times. These exhibit a level of uncertainty of 0.3 and 0.5 around the expected value, which is identical to the deterministic case $\pi = 0$. The tests were conducted on a server equipped with two 2.1 GHz processors and 384 GB of RAM with CPLEX 22.1 in GAMS 39.3.

The chosen modeling approach, in particular the inclusion of scenarios, is associated with a considerable computational effort, which makes it difficult to find feasible solutions even for our small instances. To alleviate this computational challenge, we introduced additional cuts and employed Benders decomposition. This involved formulating a deterministic master problem, including variables for the deterministic case ($\pi = 0$), along with sub-problems that incorporate the remaining variables corresponding to other scenarios, capturing the associated uncertainty. Furthermore, we initially generated 30 scenarios, clustered them, and selected a subset of 6 scenarios that was as representative as possible in order to reduce the number of scenarios and thus the computational effort.

Table 1. Mean results for 20 instances with $n = 10$ real activities and different levels of uncertainty

uncertainty	PD	Δ PD	# replanning	GAP [%]	CPU [s]
			Results for series networks		
0.3	35.1	2.3	1.8	38.6	10341.6
0.5	20.6	4.5	0.2	42.5	12165.7
			Results for parallel networks		
0.3	28.4	2.9	19.9	48.4	12548.3
0.5	21.7	2.5	0.2	44.7	11697.7

Table 1 summarizes mean results across all instances, presenting the objective function components (PD, Δ PD, and # replanning), the associated CPLEX gap, and the CPU runtime in seconds. These results depend on uncertainty levels and network characteristics. A 4-hour maximum runtime was set for all instances. Within this limit, eight instances were solved optimally. For all other instances, however, the resulting gaps remained remarkably high, averaging over 40%. In the future, longer runtimes should be tested for an evaluation of the generated upper and lower bounds. Tests with 20 real activities yielded a feasible solution for only one instance, emphasizing the need for heuristic solutions to handle larger instances and a higher number of scenarios.

Acknowledgments

This research is funded by the German Research Foundation (DFG), project no.: 520722132.

References

Alvanchi, A., Abourizk, S., Lee, S., 2012, "Dynamics of workforce skill evolution in construction projects", *Can. J. Civ. Eng.*, Vol. 39(9), pp. 1005-1017.

Baur, NF., Rieck, J., 2020, "Project Management with Scarce Resources in Disaster Response", *OR Proceedings 2019*, pp. 607-614.

Chen, D., Liu, S., Wang, Y., 2014, "Modeling for project scheduling with multi-skilled workforce constraints and uncertainty", *26th Chinese Control and Decision Conference*, pp. 157-160.

Dhib, C., Soukhal, A., 2015, "Mixed-integer linear programming formulation and priority-rule methods for a preemptive project staffing and scheduling", In: Schwindt, C., Zimmermann, J. (eds) *Handbook on Project Management and Scheduling*, Vol. 1., pp. 603-617

Felberbauer, T., Gutjahr, W.J., Doerner, K.F., 2019, "Stochastic project management: multiple projects with multi-skilled human resources", *J. Sched.*, Vol. 22, pp. 271-288.

Hanne, T., Nickel, S., 2005, "A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects", *Eur. J. Oper. Res*, Vol. 167(3), pp. 663-678.

Heimerl, C., Kolisch, R., 2010, "Scheduling and staffing multiple projects with a multi-skilled workforce", *OR Spectrum*, Vol. 32, pp. 343-368.

Naber, A., Kolisch, R., 2014, "MIP Models for Resource-Constrained Project Scheduling with Flexible Resource Profiles", *Eur. J. Oper. Res*, Vol. 239, pp. 335-345.

Snauwaert, J., Vanhoucke, M., 2022, "A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem". *Eur. J. Oper. Res.* Vol. 307, pp. 1-19.

Xiao, J., Ao, XT., Tang, Y., 2013, "Solving software project scheduling problems with ant colony optimization", *Comput. Oper. Res*, Vol. 40(1), pp. 33-46.

Zheng H, Wang L, Zheng X, 2017, "Teaching-learning-based optimization algorithm for multi-skill resource constrained project scheduling problem", *Soft Comput.*, Vol. 21(6), pp. 1537-1548.

Anchor-robust project scheduling with non-availability periods

Bendotti P[1,2], Brunod Indrigo L[1,2], Chrétienne P[2], Escoffier B[2,3]

[1] EDF R&D, 7 boulevard Gaspard Monge, 91120 Palaiseau, France
`pascale.bendotti, luca.brunod-indrigo@edf.fr`
[2] Sorbonne Université, CNRS, LIP6 UMR 7606, 4 place Jussieu, 75005 Paris, France
`philippe.chretienne, bruno.escoffier@lip6.fr`
[3] Institut Universitaire de France

Keywords: Anchor Robustness, Project Scheduling, Non-availability Periods.

1 Introduction

When planning large labor-intensive projects, it is convenient to rely on a *baseline* schedule, which gives a preview of logistical requirements. However, project data is often subject to uncertainty. Jobs may in particular be interrupted due to fortuitous causes of unavailability. Disruptions occurring between the planning and the actual execution of the project may cause the baseline schedule to become unfeasible. Part of the project must then be rescheduled, possibly entailing large costs. This justifies resorting to robust approaches to ensure that some jobs can keep their planned starting times in spite of disruptions. It is exactly the purpose of *anchor robustness*, a two-stage robust approach introduced in Bendotti et al. (2022): a baseline schedule is computed in which a subset of so-called *anchored* jobs have their starting times guaranteed against any disruption in a given uncertainty set. Anchor robustness is thought as a middle ground between guaranteeing all starting times, which is often overconservative, and guaranteeing only the makespan of the project, in which case the baseline schedule may change completely.

The existing literature on anchor robustness focuses on processing time uncertainty. This work aims at extending some of those results so that non-availability periods, called NA-periods in the sequel, can be taken into account. NA-periods are widespread in real-life applications: they can be known in advance, such as week-ends or holidays, or be uncertain, such as breakdowns or missing workforce. NA-periods are particularly studied in the context of machine scheduling: see Schmidt (2000) and Lee (2004) for surveys. Schedule robustness measures of the literature include the expected sum of weighted deviation from the baseline schedule (Herroelen & Leus 2004). It is used in Lambrechts et al. (2008) and Lambrechts et al. (2011) for the RCPSP with uncertain resource breakdowns.

In the case under study, a project described by a set of jobs with precedence constraints has to be scheduled. A ground set of NA-periods that may occur between the planning and the execution of the project is given. Each of those NA-periods, if it occurs, prevents a subset of jobs from being processed during a given time interval. Anchor-robustness in the context of uncertain NA-periods aims at guaranteeing the starting time of some jobs against any realization of NA-periods within an uncertainty set.

Contributions The aim of this work is to provide a study of the Anchor-Robust Project Scheduling Problem with uncertain NA-periods. An in-depth analysis is performed to devise dedicated computing tools in the case of budgeted uncertainty (Bertsimas & Sim 2003), which is a more realistic model of uncertainty from a practical point of view. Three anchoring problems with budgeted NA-uncertainty are considered. Polynomial algorithms are proposed for two of them and an inapproximability result is given for the last one.

2 NA-PERT

An instance of the classical scheduling problem known as PERT is defined by $\mathcal{I} = (J, G, p)$ where J is a set of jobs, $G = (J, A)$ is a directed acyclic precedence graph with s, t dummy source (resp. sink) nodes and $p \in \mathbb{R}_+^J$ is a vector of processing times. A feasible schedule $x \in \mathbb{R}_+^J$ of $\mathcal{I}$ must verify precedence constraints, namely $x_j \geq x_i + p_i$ for every $(i, j) \in A$. In this work, a variant of PERT is considered in which additional constraints arise in the form of NA-periods, defined as follows:

Definition 1 (NA-period). *Let J be a set of jobs. An* NA-period *for J is a pair $u = ([a_u, b_u), J^u)$ where $0 \leq a_u < b_u$ and $J^u \subseteq J$. The NA-period u means that the jobs of J^u cannot be performed during the semi open time interval $[a_u, b_u)$.*

In the sequel, an instance $\mathcal{I} = (J, G, p)$ of PERT as well as a finite set U of NA-periods for J are given. The subset of NA-periods affecting job i is denoted by $U^i = \{u \in U | i \in J^u\}$. The set U represents a ground set of NA-periods, only a subset of which will actually realize and affect the jobs. Notation δ will typically be used to designate such a subset of realized NA-periods. The notion of realization will fully make sense once NA-uncertainty is introduced in Section 3.

It is assumed that jobs satisfy the resumable execution hypothesis, meaning that a job, if interrupted, resumes as soon as it is available again. The starting and completion times of jobs when subset of NA-periods $\delta \subseteq U$ realizes can be translated as functions of time.

Definition 2 (Starting and completion time functions). *Let i be a job. Let $\delta \subseteq U$. The* starting time function *$S_i^\delta : \mathbb{R}_+ \to \mathbb{R}_+$ and the* completion time function *$C_i^\delta : \mathbb{R}_+ \to \mathbb{R}_+$ are such that, for any $\tau \in \mathbb{R}_+$:*

$$S_i^\delta(\tau) = \min\left([\tau, +\infty) \setminus \bigcup_{u \in \delta \cap U^i} [a_u, b_u)\right) \qquad C_i^\delta(\tau) = \min\left\{\tau' \geq S_i^\delta(\tau),\ \lambda^{\delta \cap U^i}(\tau, \tau') = p_i\right\}$$

where $\lambda^{\delta \cap U^i}(\tau, \tau')$ is the available time left by the NA-periods of $\delta \cap U^i$ between τ and τ'.

The variant of PERT under study can now be defined based on those functions.

Definition 3 (NA-PERT). *$\mathcal{I}^\delta = (J, G, p, \delta)$ is an instance of NA-PERT if $\mathcal{I} = (J, G, p)$ is an instance of PERT and $\delta \subseteq U$ is a subset of NA-periods. A vector $y \in \mathbb{R}_+^J$ is a schedule for $\mathcal{I}^\delta$ if it satisfies the following conditions:*

$$S_i^\delta(y_i) = y_i \qquad \text{for every } i \in J \tag{1}$$

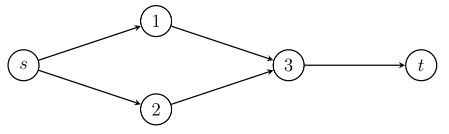
$$y_j \geq C_i^\delta(y_i) \qquad \text{for every } (i, j) \in A \tag{2}$$

Condition (1) is equivalent to $y_i \notin [a_u, b_u)$ for every $u \in \delta \cap U^i$, it prevents a job from starting during an NA-period of δ that affects it. Condition (2) means that the available time between the starting times of jobs i and j must be sufficient to perform job i entirely if $(i, j) \in A$. Without any NA-period, it leads to the usual precedence constraint $y_j \geq y_i + p_i$.

Note that NA-PERT is a special case of the variant of PERT with time-dependent completion times presented in Minoux (2007).

Since jobs s and t are dummy source and sink jobs used to represent the beginning and the end of the project respectively, it is assumed in the sequel that they are not affected by any NA-period. It is also assumed that $y_s = 0$ in any schedule.

Example 1. Consider a project with $J = \{s, 1, 2, 3, t\}$, the precedence graph in Figure 1a and the processing times in Figure 1b. Let also $U = \{u_1, u_2, u_3\}$ be a ground set of three NA-periods described by Figure 1c.
Figures 1d and 1e give feasible schedules for two subsets δ of realized NA-periods. Note that in Figure 1e, job 2 can be processed during NA-period u_2 since it only affects job 1. $\Diamond$



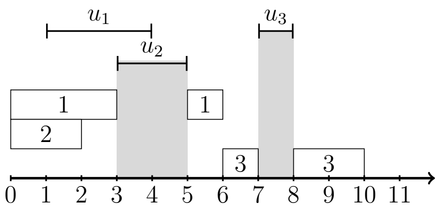
(a) Precedence graph

i	p_i
s	0
1	4
2	2
3	3
t	0

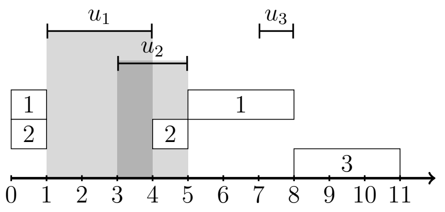
(b) Job processing times

u	$[a_u, b_u)$	J^u
u_1	$[1,4)$	$\{1,2\}$
u_2	$[3,5)$	$\{1\}$
u_3	$[7,8)$	$\{3\}$

(c) NA-periods



(d) Schedule for $\delta = \{u_2, u_3\}$



(e) Schedule for $\delta = \{u_1, u_2\}$

Fig. 1: Example of NA-PERT

3 Anchor robustness with NA-uncertainty

The aim of this work is to extend anchor robustness, introduced for processing-time uncertainty in Bendotti et al. (2022), to uncertainty on NA-periods, an uncertainty model called NA-uncertainty from now on. An NA-uncertainty realization is a subset of NA-periods $\delta \subseteq U$. The NA-uncertainty set is defined by a subset $\varDelta$ of 2^U.

Budgeted uncertainty sets (Bertsimas & Sim 2003) are widely studied due to their convenient structure and to the control offered on conservativity through the *budget* parameter. The subsequent results concern anchoring problems using the budgeted NA-uncertainty set.

Definition 4 (Budgeted NA-uncertainty set). *The* budgeted NA-uncertainty set *with* budget $\varGamma \in \{0, \dots, |U|\}$ *is* $\varDelta(U, \varGamma) = \{\delta \subseteq U \mid |\delta| \leq \varGamma\}$

Anchorage and anchored sets can now be defined.

Definition 5 (Anchored set). *Let x be a schedule of $\mathcal{I} = (J, G, p)$ and let $\varDelta$ be an NA-uncertainty set. $H \subseteq J$ is x-*anchored *if for any $\delta \in \varDelta$ there exists a schedule y^δ of (J, G, p, δ) that satisfies $x_i = y_i^\delta$ for every $i \in H$.*

In the sequel, three natural problems involving anchored sets will be considered:

Problem 1 (AnchRobCheck). **Input:** An instance $\mathcal{I} = (J, G, p)$ of PERT, an NA-uncertainty set $\varDelta$, a schedule x of $\mathcal{I}$ and a subset $H \subseteq J$.
Question: Check if H is x-anchored.

Problem 2 (AnchRobFind). **Input:** An instance $\mathcal{I} = (J, G, p)$ of PERT, an NA-uncertainty set $\varDelta$ and a subset $H \subseteq J$.
Question: Find a schedule x of $\mathcal{I}$ with minimum makespan such that H is x-anchored.

Problem 3 (AnchRobMaxWeight). **Input:** An instance $\mathcal{I} = (J, G, p)$ of PERT, an NA-uncertainty set $\varDelta$, a schedule x of $\mathcal{I}$ and weights $(w_i)_{i \in J}$.
Question: Find a subset $H \subseteq J$ with maximum weight such that H is x-anchored.

A characterization of anchored sets that can then be used to solve anchoring problems is now given. To that end, a function is introduced based on completion time functions.

Definition 6 (Worst-case longest path function). *Let Δ be an NA-uncertainty set. For any $i, j \in J$, the* worst-case longest path function $L^{\Delta}_{ij} : \mathbb{R}_+ \to \mathbb{R}_+$ *is such that, for any $\tau \in \mathbb{R}_+$:*

$$L^{\Delta}_{ij}(\tau) = \max_{\substack{(i_1,\dots,i_r)\in\mathcal{P}_{ij} \\ \delta\in\Delta}} C^{\delta}_{i_{r-1}} \circ C^{\delta}_{i_{r-2}} \circ \cdots \circ C^{\delta}_{i_1}(\tau)$$

where $\mathcal{P}_{ij}$ is the set of i-j paths in the precedence graph G.

Intuitively, achieving anchor-robustness in a schedule requires the starting times to be scattered enough to absorb uncertainty. One would then want to create sufficiently large gaps to overcome any uncertainty realization.The following theorem shows that the gaps between jobs ensuring anchor-robustness are related to functions L^{Δ}_{ij} of Definition 6.

Theorem 1 (Anchored set characterization). *Let x be a schedule of $\mathcal{I} = (J, G, p)$. Let Δ be an NA-uncertainty set. $H \subseteq J$ is x-anchored if and only if $x_j \geq L^{\Delta}_{ij}(x_i)$ for every $i, j \in H \cup \{s\}$ and $S^U_i(x_i) = x_i$ for every $i \in H$.*

Let us now focus on solving anchoring problems. First, it can be shown that the worst-case longest path function L^{Δ}_{ij} is computable in polynomial time. It follows that AnchRobCheck and AnchRobFind can be solved in polynomial time by combining the characterization of Theorem 1 and the computation of L^{Δ}_{ij}.

Theorem 2 (Polynomialility of AnchRobCheck). *AnchRobCheck under budgeted NA-uncertainty can be solved in polynomial time.*

Theorem 3 (Polynomialility of AnchRobFind). *AnchRobCheck under budgeted NA-uncertainty can be solved in polynomial time.*

Finally, it can be shown that AnchRobMaxWeight is not approximable within constant ratio using a reduction from the Maximum Independent Set problem and the inapproximability result from Håstad (1999).

Theorem 4 (Inapproximability of AnchRobMaxWeight). *AnchRobMaxWeight for a budgeted NA-uncertainty set is not approximable within constant ratio, even with $\Gamma = 1$ and binary weights, unless $P = NP$.*

Bibliography

Bendotti, P., Chrétienne, P., Fouilhoux, P. & Pass-Lanneau, A. (2022), 'The anchor-robust project scheduling problem', *Operations Research* **71**(6), 2267–2290.

Bertsimas, D. & Sim, M. (2003), 'Robust discrete optimization and network flows', *Mathematical Programming* **98**(1), 49–71.

Herroelen, W. & Leus, R. (2004), 'The construction of stable project baseline schedules', *EJOR* **156**(3), 550–565.

Håstad, J. (1999), 'Clique is hard to approximate within $n^{1-\epsilon}$', *Acta Mathematica* **182**, 105–142.

Lambrechts, O., Demeulemeester, E. & Herroelen, W. (2008), 'Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities', *Journal of scheduling* **11**(2), 121–136.

Lambrechts, O., Demeulemeester, E. & Herroelen, W. (2011), 'Time slack-based techniques for robust project scheduling subject to resource uncertainty', *ANOR* **186**, 443–464.

Lee, C.-Y. (2004), 'Machine scheduling with availability constraints', *Handbook of scheduling. Algorithms, models and perforance analysis* pp. 495–507.

Minoux, M. (2007), 'Models and algorithms for robust PERT scheduling with time-dependent task durations', *Vietnam Journal of Mathematics* **35**(4), 387–398.

Schmidt, G. (2000), 'Scheduling with limited machine availability', *EJOR* **121**, 1–15.

Analyzing Criticalities on Project Network with Variable Activity Durations

Lucio Bianco[1], Massimiliano Caramia[1], Stefano Giordani[1] and Alessio Salvatore[1,2]

[1] University of Rome "Tor Vergata", Italy
{bianco,caramia}@dii.uniroma2.it, {stefano.giordani,alessio.salvatore}@uniroma2.it
[2] Istituto di Analisi dei Sistemi ed Informatica, CNR, Italy

Keywords: Variable activity durations; Critical activities; Activity floats.

1 Introduction

Temporal analysis on project networks has been one of the most studied topics in the project management and scheduling arena. From the early study of Kelley (1963) on the Critical Path Method in the Sixties, many researchers addressed the problem of modeling project activities and constraints to minimize the project completion time (makespan). Special attention has been deserved on the relation between the minimum makespan of the project and the longest path of the project network in the quest of defining the so-called "critical" activities, i.e., those activities responsible for the increase of the project completion time when delayed from their earliest start (finish) time and/or a variation of their durations happens. When activity durations are fixed and given, the temporal analysis consists of executing a forward and a backward recursion to calculate the earliest and the latest start (finish) times of the activities. When the difference between the earliest and latest start times (total float) of an activity is zero also the difference between the earliest and latest finish time is zero and the activity is said to be critical.

Several authors addressed the concept of criticality of an activity giving different definitions based on the type of temporal relationships (Finish-to-Start precedences, Generalized Precedence Relationships (GPRs), feeding precedences) of the project (see, e.g., Bianco *et. al.* (2022), Bianco *et. al.* (2023), Elmaghraby and Kamburowski (1992), Quintanilla *et. al.* (2012), Valls and Lino (2001)). Notwithstanding, a shared definition brings together these contributions, that is, an activity is critical if it lies on a longest path (critical path) of the project network. In this work, we wish to investigate if this still applies when the activity durations are not fixed and given but are unknown variables, to be determined to minimize the project makespan. In particular, we assume that the duration d_i of an activity i ranges in a given interval, that is, $d_i \in [d_i^{\min}, d_i^{\max}]$.

Our research questions are as follows: *Are the activity floats, calculated respectively on the start and finish times, still equal? If the floats are different, is it however necessary that they should be both equal to zero to identify a critical activity? Since activity durations are variable and then the optimal duration of an activity evaluated with the forward and backward recursions could be different, are these durations equal when the activity is critical? Is an activity belonging to both the longest path obtained with the forward and backward recursions critical? Does it exist a vector of activity durations such that an activity is critical if and only if it belongs to a longest path of the project network with that activity durations?*

When activities' durations are not fixed, it is simple to show that the traditional concept of critical activity remains valid if we have only Finish-to-Start relations among activities. Oppositely, in case of GPRs, including Start-to-Start (*SS*), Start-to-Finish (*SF*), Finish-to-Start (*FS*), and Finish-to-Finish (*FF*) relationships, that traditional concept is no longer valid, as we will show, and a more general characterization of criticality is required.

2 Forward and Backward Recursions

We consider an acyclic project network and then activities $i = 1, \ldots, n$ are assumed topologically indexed. For ease of presentation of the results, but w.l.o.g., we assume zero time lags. We consider the additional dummy activities 0 (initial) and $n+1$ (final), and additional precedences $(0, i)$ of type *SS* and $(i, n+1)$ of type *FS*, for $i = 1, \ldots n$, to force activity i, if necessary, to start (finish) not earlier (later) than the project start (completion). Given any feasible duration $d_i \in [d_i^{\min}, d_i^{\max}]$, for each activity i, let us denote with $N(d)$ the GPRs project network related to that activity durations. A GPRs project network can be equivalently represented by a standardized network (Bartusch *et. al.* 1988) where only one type of precedence relations is considered. In particular, the *SS*-standardized network represents precedence relations with respect to activity starting times. Let $t_{ij}(d_i, d_j)$ be the length of arc (i, j) equal to the minimum difference between the starting times of activities j and i, according to precedence relation (i, j), i.e., $t_{ij}(d_i, d_j) = 0, d_i, -d_j, d_i - d_j$, if precedence (i, j) is of type *SS*, *FS*, *SF*, and *FF*, respectively. Let us denote with $\ell_{h \to k}^{N(d)}$ the length of the longest path from node h to node k, with $h < k$, in the *SS*-standardized project network $N(d)$, if such a path exists. Similarly, the *FF*-standardized network represents precedence relations with respect to activity finish times. In this representation, the length $\hat{t}_{ij}(d_i, d_j)$ of arc (i, j) is the minimum difference between the finish times of activities j and i, according to precedence relation (i, j): $\hat{t}_{ij}(d_i, d_j) = d_j - d_i, d_j, -d_i, 0$, if precedence (i, j) is of type *SS*, *FS*, *SF*, and *FF*, respectively. Let us denote with $\hat{\ell}_{h \to k}^{N(d)}$ the length of the longest path from node h to node k, with $h < k$, in the *FF*-standardized network of project network $N(d)$, if such a path exists. While in general $t_{ij}(d_i, d_j)$ may be different from $\hat{t}_{ij}(d_i, d_j)$, it is easy to show that the lengths of the paths between 0 and $n+1$ do not depend on the type of network standardization, therefore $\ell_{0 \to i}^{N(d)} + \ell_{i \to n+1}^{N(d)} = \hat{\ell}_{0 \to i}^{N(d)} + \hat{\ell}_{i \to n+1}^{N(d)}$, for any given (real) activity i, and $\ell_{0 \to n+1}^{N(d)} = \hat{\ell}_{0 \to n+1}^{N(d)} = C_{\max}^d$, where $C_{\max}^d$ is the minimum project length with the given activity duration $d_i \in [d_i^{\min}, d_i^{\max}]$, for each activity i. In the following, unless otherwise stated, we will refer to the *SS*-standardized network representation.

An optimal duration, denoted with d_i^{FW}, of activity i is the minimum value in the range $[d_i^{\min}, d_i^{\max}]$, that allows i to start at ES_i, assuming $ES_0 = 0$. We can compute d_i^{FW} by the following forward (FW) recursion. Since increasing d_i will not increase length $t_{hi}(d_h, d_i)$ of arc (h, i), for the calculation of ES_i we can initially consider $d_i = d_i^{\max}$ and calculate $ES_i = \max_{(h,i) \in \Gamma^-(i)} \{ES_h + t_{hi}\}$, where $\Gamma^-(i)$ is the set of incoming arcs (h, i) of i and $t_{hi} = t_{hi}(d_h^{FW}, d_i^{\max})$.Conversely, decreasing d_i will not increase $t_{ij}(d_i, d_j)$, for any outgoing arcs (i, j) of i. Then, we determine d_i^{FW} as the minimum feasible value of d_i that still allows i to start at ES_i: this can be done by looking only at the set $\Gamma^-(i)$ of incoming arcs of i. The earliest finish time of i is $EF_i = ES_i + d_i^{FW}$, and the minimum project makespan $C_{\max}^* = ES_{n+1} = \max_i\{EF_i\}$.

Analogously, another optimal duration, denoted with d_i^{BW}, of i is the maximum value in the range $[d_i^{\min}, d_i^{\max}]$, that allows i to start at LS_i, assuming $LS_{n+1} = ES_{n+1} = C_{\max}^*$. We compute d_i^{BW} by the following backward (BW) recursion. Since decreasing d_i will not increase length $t_{ij}(d_i, d_j)$ of arc (i, j), we can, initially, assume $d_i = d_i^{\min}$ and calculate LS_i, i.e., $LS_i = \min_{(i,j) \in \Gamma^+(i)}\{LS_j - t_{ij}\}$, where $\Gamma^+(i)$ is the set of outgoing arcs (i, j) of i and $t_{ij} = t_{ij}(d_i^{\min}, d_j^{BW})$. Then, we determine d_i^{BW} as the maximum feasible value of d_i that still allows i to start at LS_i: this can be done by looking only at the set $\Gamma^+(i)$ of outgoing arcs of i. The latest finish time of i is $LF_i = LS_i + d_i^{BW}$. Let us consider the project network with $n = 4$ activities, and variable activity durations, shown in Figure 1.

Table 1 lists the activity forward and backward optimal durations (d_i^{FW}, d_i^{BW}) and the earliest/latest start/finish times (ES_i, LS_i, EF_i, LF_i), calculated by applying the forward

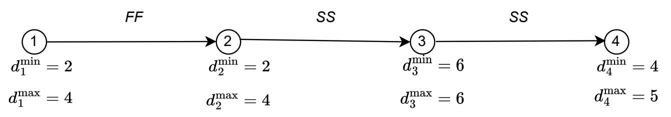


Fig. 1. Example of a project network with GPRs and variable activity durations

and backward recursions. The minimum project makespan $C^*_{\max}$ is equal to 6, and $d_i^{FW} = d_i^{BW}$, for all activities i but 2; in particular, $d_2^{FW} < d_2^{BW}$.

activity i	1	2	3	4
d_i^{FW}, d_i^{BW}	2, 2	2, 4	6, 6	4, 4
ES_i, LS_i	0, 2	0, 0	0, 0	0, 2
EF_i, LF_i	2, 4	2, 4	6, 6	4, 6

Table 1. *FW* and *BW* durations, and earliest/latest start/finish times of the example in Fig. 1

Looking at activity 2, we note that $LS_2 - ES_2 = 0$ and $LF_2 - EF_2 = 2$. Therefore, differently from the case with known and fixed activity durations where the values of these two differences are always equal and we simply refer to them as activity float, when activity durations are variable these two floats could be different. Let us call *start float* of activity i the difference $LS_i - ES_i$, and *finish float* of i the difference $LF_i - EF_i$. Since, according to the traditional concept, an activity is *critical* if it has a non-positive float, activity 3 would be classified as critical. However, we would have difficulty to characterize the criticality of activity 2, since one of its floats is positive. The above example shows the need to redefine and generalize the concept of critical activity and its relation with activity floats.

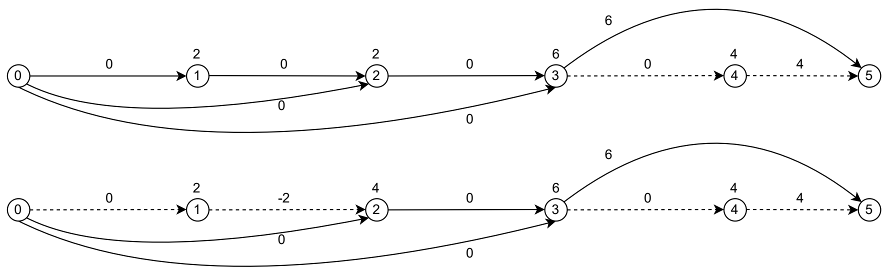


Fig. 2. The *SS*-standardized network of the *FW*-network and *BW*-network associated with Fig. 1

Figure 2 represents the *SS*-standardized networks of the *FW*-network $N(d^{FW})$ and *BW*-network $N(d^{BW})$ of the above example, respectively, with activity duration resulting from forward and backward recursions. Here, only the arcs belonging to a longest path from node 0 to node $n + 1$ traversing an activity node are shown: among them, dashed arcs do not belong to a critical path (i.e., a longest path from 0 to $n + 1$). Note that *FW*-network and *BW*-network can be different, at least for the arc lengths. Let us consider activities 2 and 3 that both result to belong to a critical path in both networks. However, while activity 3 has both the start and finish floats equal to zero, as one could expect, this is not the case for activity 2 which has only the start-float equal to zero. In addition, activity 1 is on a critical path of the *FW*-network, but not in a critical path of the *BW*-network, despite both its start and finish floats are greater than zero. Finally, we note that activity 4 has both positive floats, but on the contrary of activity 1, it does not belong to any critical path in both networks. Therefore, it is difficult to relate the criticality of an activity with its belonging to a critical path on the FW-network and/or on the BW-network. Moreover, the analysis of the example does not show all the possible issues.

Let $\ell^{FW}_{0\to i}$ and $\ell^{BW}_{0\to i}$ be the lengths of the longest path from 0 to i, and let $\ell^{FW}_{i\to n+1}$ and $\ell^{BW}_{i\to n+1}$ be the lengths of the longest path from i to $n+1$, of the SS-standardized networks of the FW-network and BW-network, respectively.

Let $\delta^{FW}_i = C^*_{\max} - (\ell^{FW}_{0\to i} + \ell^{FW}_{i\to n+1}) \geq 0$ be the *FW-gap* of activity i, i.e., the difference between $C^*_{\max}$ and the length of the longest path from 0 to $n+1$ traversing node i in the FW-network, and let $\delta^{BW}_i = C^*_{\max} - (\ell^{BW}_{0\to i} + \ell^{BW}_{i\to n+1}) \geq 0$ be the *BW-gap* of activity i (w.r.t. the BW-network). Therefore, from the definitions of FW-gap and BW-gap of activity i, it follows that $\delta^{FW}_i \leq LS_i - ES_i$, because $\ell^{FW}_{i\to n+1} \geq \ell^{BW}_{i\to n+1}$, and, analogously, $\delta^{BW}_i \leq LS_i - ES_i$, because $\ell^{BW}_{0\to i} \geq \ell^{FW}_{0\to i}$. Hence, we have $\max\{\delta^{FW}_i, \delta^{BW}_i\} \leq LS_i - ES_i$. Analogously, using the FF-standardized network representation, it can be shown that $\delta^{FW}_i \leq LF_i - EF_i$ and $\delta^{BW}_i \leq LF_i - EF_i$. Therefore, we have

$$\max\{\delta^{FW}_i, \delta^{BW}_i\} \leq \min\{LS_i - ES_i, LF_i - EF_i\}$$

and, then, it is possible that an activity could have both positive start and finish floats, despite belonging to a critical path in both the FW- and BW- networks.

3 Ongoing and future work

From the above analysis, we observe that the start and finish floats of an activity can be different. Accordingly, it can happen that just one of them is equal to zero. Therefore, we can identify as critical an activity for which $\min\{LS_i - ES_i, LF_i - EF_i\} = 0$.

The importance of this result is that it calls for a more general concept of criticality opening new perspectives on the project scheduling. In fact a float of critical activity different from zero together with a variable duration allows a degree of flexibility in the resource assignment usable to reduce the makespan of the project.

We will try to answer to the other research questions posed in the introduction, and, in particular, we will show that there exists a vector of activity durations such that an activity is critical if and only if it belongs to a critical path of the project network with that activity durations. In addition, we aim to characterize the type of activity criticalities, and to extend this analysis to the more general class of feeding precedence relationships.

References

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., Giordani, S., 2022, "Project scheduling with generalized precedence relations: a new method to analyze criticalities and flexibilities", *European Journal of Operational Research*, Vol. 298(2), pp. 451-462.

Bianco, L., Caramia, M., Giordani, S., Salvatore, A., 2023, "Scheduling activities in project network with feeding precedence relations: an earliest start forward recursion algorithm", *Annals of Operations Research*, doi 10.1007/s10479-023-05521-0.

Elmaghraby, S.E., Kamburowski, J., 1992, "The analysis of activity networks under generalized precedence relations", *Management Science*, Vol. 38(9), pp. 1245-1263.

Kelley, J.E., 1963, "The critical path method: Resource planning and scheduling", in: J.F. Muth, G.L. Thompson (Eds.) *Industrial Scheduling*, Prentice Hall, New Jersey, pp. 347-365.

Quintanilla, S., Perez, A., Lino, P., Valls, V., 2012, "Time and work generalised precedence relationships in project scheduling with preemption: An application to the management of service centres", *European Journal of Operational Research*, Vol. 219(1), pp. 59-72.

Valls, V. and Lino, P., 2001, "Criticality analysis in activity-on-node networks with minimal time lags", *Annals of Operations Research*, Vol. 102, pp. 17-37.

Operations scheduling in reconfigurable flow shops

Fayez F. Boctor[1]

Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport
Faculté des sciences de l'administration, Université Laval
`fayez.boctor@fsa.ulaval.ca`

Keywords: Reconfigurable manufacturing systems, Scheduling, Mathematical modeling.

1 Introduction

A new type of manufacturing systems, called reconfigurable manufacturing systems (RMS), is proposed to allow combining the flexibility of flexible manufacturing systems (FMS) and high capacity of dedicated manufacturing lines (DML). Although the design of RMS has received a lot of attention during the last two decades, there are few publications dealing with operations scheduling in this type of manufacturing systems. This paper deals with operations scheduling in reconfigurable flow shops and discusses three versions of the problem. It shows that the first version of this scheduling problem can be solved by separately solving two sequencing problems. The other two versions can be formulated as a single machine sequencing problem. The paper also provides a heuristic to solve large instances. A numerical experiment shows that this heuristic produces solutions remarkably close to the optimum for the tested instances involving up to 5 subfamilies of 20 jobs and 5 workstations each.

RMS are designed to admit several configurations that allow the production of a family of parts that can be divided into subfamilies. Each subfamily requires one of the possible configurations and, often, moving to manufacturing a new subfamily requires to reconfigure the system. Reconfiguring the system requires a major setup while manufacturing parts within a same subfamily requires no or minor setups. Generally, the reconfiguration setup time is sequence-dependent while it is often assumed that minor setups required to manufacture parts within a subfamily are sequence-independent and, consequently, their time can be added to parts processing time. Thus, the problem reduces to sequencing subfamilies as well as to sequence parts within each subfamily with the objective of minimizing the overall make span.

Plastic extrusion lines are an example of reconfigurable manufacturing systems. The three main components of a plastic extrusion line are: an extruder, a cooling bath, a haul-off machine. However, processing some jobs may require a co-extruder, a coiler or a cutting machine, a drill, or a printer to print some information on the extruded products. So, it is frequent to reconfigure the line by adding or removing such equipment. The subfamilies of products are characterized by their raw material (there are about 20 types of raw materials) their color (there are about 60 different colors), final shape (coil or straight pieces) and the die needed to profile the subfamily products. So, changeover time between two subfamilies may require adding or removing a co-extruder, replacing the coiler by a cutting machine, adding or removing a printer, removing or adding a drill, as well as changing the used raw material. This may require up to 4 hours. On the other hand, if a job is to be coiled while the next job of the same subfamily is to be cut into pieces, we only need to replace the coiler by a cutting machine, and this takes just 15 minutes.

The remaining of this paper is organized as follows. Section 2 reviews some related literature and section 3 define and formulate the problem. Section 4 proposes a heuristic solution method to schedule operations in a reconfigurable flow shop while section 5 reports on a numerical experiment undertaken to evaluate the proposed heuristic solution method.

2 Literature review

Studying RMS was active during the last two decades since the introduction of this concept by Koren *et al.* in 1999. Pansare *et al.* (2021) reviewed 454 articles dealing with RMS published up to the year 2020. However, most of these publications address design aspects of RMS. Very few articles deal with the problem of scheduling operations within this type of manufacturing systems.

Azab and Naderi (2015) considered a ***first version*** of the reconfigurable flow shop manufacturing system (RFSMS) where all machines should be stopped to do the reconfiguration major setup. They proposed a mathematical model for scheduling its operations. However, their model can only allow solving small instances. They solved instances with up to 5 subfamilies and 4 jobs within each subfamily. Naderi and Azab (2021) considered a ***second version*** of the RFSMS where the major setup prior to the processing of the next subfamily is not always mandatory. If the system is not reconfigured, a variable and sequence-independent extra time (called skipping time) should be added to each operation of the jobs of this subfamily. Delorme *et al.* (2023) studied a ***third version*** of the problem where some (but not all) machines do not require setup when the system is reconfigured in between subfamilies. Those machines can continue processing jobs until the other machines are reconfigured. The following deals with the first version of the problem.

3 Problem statement and formulation

The problem considered hereafter is that of scheduling the operations of a reconfigurable flow shop designed to produce S subfamilies of jobs and each subfamily s is composed of n_s jobs. Each subfamily requires a subset w_s of the available machines or workstations. All the jobs of a same subfamily follow the same processing sequence through its required workstations. To change the configuration, the line should be stopped, some workstations or machines should be moved out, others should be added. The reconfiguration setup is sequence dependent while jobs within a same subfamily requires no or minor sequence-independent setup time. The objective is to minimize the overall make span. To solve the problem, we can decompose it into two independent problems: sequencing subfamilies and sequencing jobs within each subfamily.

3.1 Sequencing subfamilies

Sequencing subfamilies can be formulated as a sequence-dependent single machine scheduling problem, or equivalently as a traveling salesman problem. Let r_{gh} be the reconfiguration setup time if we process subfamily h immediately after subfamily g and x_{gh} be a binary variable that takes the value 1 if subfamily h is processed immediately after subfamily g. Also, let r_{0h} be the reconfiguration setup time if subfamily h is in the first position of the processing sequence. Then the subfamilies sequencing problem can be formulated as follows:

$$\text{Find: } x_{gh} \in \{0,1\};\ g= 0,..,S;\ h=1,..,S \text{ and } g \neq h \text{ which} \quad (1)$$

$$\text{Minimizes: } \sum_{g=0}^{S}\sum_{h=1}^{S} r_{gh}x_{gh} \quad (2)$$

$$\text{Subject to: } \sum_{g=0}^{S} x_{gh} = 1; \qquad h=1,\ldots,S;\ g\neq h \quad (3)$$

$$\sum_{h=1}^{S} x_{gh} = 1; \qquad g=0,\ldots,S;\ g\neq h \quad (4)$$

$$u_g - u_h + (S+1)x_{gh} \leq S \qquad g=0,\ldots,S;\ h=1,\ldots,S;\ g\neq h \quad (5)$$

The variables u_g and u_h are arbitrary numbers and the third set of constraints is the subtour elimination constraint.

3.2 Jobs sequencing within a subfamily

Sequencing jobs within a subfamily s can be formulated as a permutation flow shop problem. Let p_{ij} be the processing time of job i on workstation j, y_{ip} be a binary variable that takes the value 1 if job i is in position p in the processing sequence, F_{jp} be the finish time of the job on position p on workstation j. Then the problem can be formulated as follows:

$$\text{Find:} \quad y_{ip} \in \{0,1\} \quad \text{and} \quad F_{jp} \geq 0; i = 1,..,n_s; p = 1,..,n_s; j = 1,...,w_s \text{ which} \tag{6}$$

$$\text{Minimize:} \quad F_{w_s n_s} \tag{7}$$

$$\text{Subject to:} \quad \sum_{i=1}^{n_s} y_{ip} = 1; \qquad p=1,\ldots,n_s \tag{8}$$

$$\sum_{p=1}^{n_s} y_{ip} = 1; \qquad i=1,\ldots,n_s \tag{9}$$

$$F_{jp} \geq F_{j,p-1} + \sum_{i=1}^{n_s} y_{ip} p_{ij}; \qquad j=1,\ldots,w_s;\ p=2,\ldots,n_s \tag{10}$$

$$F_{jp} \geq F_{j-1,p} + \sum_{i=1}^{n_s} y_{ip} p_{ij}; \qquad j=2,\ldots,w_s;\ p=1,\ldots,n_s \tag{11}$$

3.3 Formulation of the other two versions of the problem

In the second version of the problem, where reconfiguring the flow shop prior to processing some subfamilies is not always mandatory, we can deal with the problem as a flow shop problem with sequence-dependent setup times (see Meng *et al.* 2022). In this case, the setup time before a job is nil if the preceding job belongs to the same subfamily and equals to its skipping time, if not. However, we need to study the efficiency of this formulation compared to the one proposed in Azab and Naderi (2015).

The third version of the problem, where not all machines require reconfiguration when we move from a given subfamily to another one, can be formulated like the second version. The setup time for a job on a given machine is nil if the previous job belongs to the same subfamily and equals the required reconfiguring setup time of this machine, if not. Again, we need to study the efficiency of this way of modeling the problem compared to the model proposed in Delorme *et al.* (2023).

4 A proposed heuristic

To solve the **subfamilies sequencing** problem presented in section 3.1, a hybrid heuristic is suggested. First an initial sequence is obtained by applying the nearest neighbor heuristic S times starting by a different subfamily in each repetition, and the best solution is retained. Then, this sequence is improved by repeatedly removing one subfamily from the sequence and reinsert it in the best possible position.

Note that for small values of S, it is faster to enumerate and evaluate all the possible sequences and retain the best one. In our computational experiment, the average time to enumerate all sequences of 9 subfamilies was 0.293 seconds while instances with 10 subfamilies required 2.911 seconds in average.

To solve the problem of **sequencing jobs** within each subfamily, the well-known NEH sequential insertion heuristic is used (see Nawaz *et al.* 1983). This insertion heuristic is repeated 100 times and in each repetition the job to insert in the partial sequence is selected randomly.

5 Evaluation experiment

To evaluate the performance of the proposed heuristic, 30 medium-size test instances were randomly generated, solved to optimality using the models presented in section 3, and solved by the proposed heuristic. The number of subfamilies is 3, 4, and 5. For all test instances, the number of jobs within each subfamily is 20, and the number of machines is 5. The following table summarizes the obtained results and shows that the proposed heuristic produces solutions remarkably close to the optimal solutions.

Number of subfamilies	Number of instances	Optimal solution	Heuristic solution			
		Average make span	Average computational time (sec)	Average make span	Number of times the optimal solution was found	Average percentage deviation from the optimum
3	10	344.9	7.100	345.3	6	0.119 %
4	10	452.6	9.467	453.2	4	0.133 %
5	10	570.4	11.835	571.1	3	0.124 %
Average						0.125 %

References

Azab A., B. Naderi, 2015, “Modeling the problem of production scheduling for reconfigurable manufacturing systems", *Precedia CIRP*, Vol. 33, pp. 76-80.

Delorme X., G. Fleury, Ph. Lacomme and D. Lamy, 2023, “Modeling and solving approaches for scheduling problems in reconfigurable manufacturing systems", *International Journal of Production Research*, DOI: 10.1080/00207543.2023.2224446.

Koren Y., U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy and H. N. Brussel, 1999, “Reconfigurable manufacturing systems", *CIRP Annals - Manufacturing Technology*, Vol. 48, pp. 527-540.

Meng L., K. Gao, Y. Ren, B. Zang, H. Sang and Z. Chaoyong, 2022, “Novel MILP and CP models for distributed hybrid flow shop scheduling problem with sequence-dependent setup times", *Swarm and Evolutionary Computation*, Vol. 71, article 101058.

Naderi B., A. Azab, 2021, “Production scheduling for reconfigurable assembly systems: Mathematical models and algorithms", *Computers & Industrial Engineering*, Vol. 162, article 107741.

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

Pansare R., G. Yadav and M. R. Nagare, 2021, “Reconfigurable manufacturing system: a systematic bibliometric analysis and future research agenda", *Journal of Manufacturing Technology Management*, Vol. 33, pp. 543-574.

A risk-averse approach to payment delays

Maria Elena Bruni[1] and Öncü Hazır[2]

[1] Department of Mechanical Energy and Management Engineering, University of Calabria, Italy
mariaelena.bruni@unical.it
[2] Rennes School of Business, 2 Rue Robert d'Arbrissel, Rennes, France
oncu.hazir@rennes-sb.com

Keywords: Project Scheduling, Robust Optimization, NPV, Risk Management.

1 Introduction

Payments delays create a time lag between the expenses incurred by the contractor and the progress payments received from the client, and they are a crucial category of risk for projects. The challenges associated with obtaining continuous project finances often place undue financial strain on contractors who may seek loans from financial institutions to maintain their daily operations. These loans must be returned with interest, increasing financing costs and considerably lowering the Net Present Value. In this paper, we delve into cash flow and project scheduling strategies to mitigate late payment impacts and enhance project resilience, presenting a distributionally robust risk-averse model that minimizes the financing cost by accurately estimating the amount and timing of the expenses and revenues throughout the project life cycle and foreseeing possible cost overruns and cash flow fluctuations. The model includes the financing costs to determine the best project schedule and financing alternative, covering the cash deficit with the minimum financing cost in a two-stage stochastic program. The start time for each activity is set in the first stage when the delay of client payments is still unknown. In the second stage, loans can be borrowed based on the delays and the balance between negative and positive cash flows.

2 Mathematical formulation

The project, given in an activity-on-node (AoN) representation, is shown as a directed graph $G(V, E)$, where $V = \{0, 1, \dots, n, n+1\}$ (node 0 and $n+1$ are dummy nodes for project beginning and project completion) is a set of nodes of the activities and E is a set of arcs showing the relations between the activities of finish-start zero-lag precedence.

The time $\mathscr{T} = \{0, \dots, T\}$ is describred by a discrete set of periods and $\overline{d}$ shows the project deadline. Each activity has a known duration p_i. For each activity $i \in V$, let $T^i \subseteq \mathscr{T} = \{e_i, \dots, l_i\}$ show the time points at which activity i can start, i.e., periods between the earliest e_i and latest l_i start times for an activity i. Each activity, i, requires or brings cash: outflow $c_i^{out} < 0$ at the start time and inflow $c_i^{in} \geq 0$ at completion. At project start, some amount of cash (C_0) is invested by the contractor. The cash flows are discounted with a rate of β is the per period. In most of the cases, the time at which payments are received is not known with certainty, and subject to a random delay for the activity i represented by ξ_i^ω, $\forall \omega \in \Omega$, where Ω refers to the scenario set (with a slight abuse of notation, we will denote with ξ^ω the vector $[\xi_0^\omega, \dots \xi_{n+1}^\omega]$). Payments occur after the activity completion, possibly with a delay, but always within the planning horizon $\mathscr{T}$. Due to the payment delays, the contractor must finance the cash deficit by setting up a feasible financing plan; commonly borrowing short or long-term loans. A line of credit allows the contractors to borrow any time t. One possible way of repaying the debt is to pay off the compounded interest (with the rate $r > \beta$) and the borrowed amount at the end of the project. We integrate this borrowing possibility into our model.

Payment delays can be described using a discrete set of scenarios that can be formulated with expert opinions. However, the probabilities of occurrence for a specific scenarios are difficult to estimate. We assume the scenarios concerning the timing of payments are derived from historical data or subjective judgments, while the associated probabilities are considered ambiguous. A simple, clear, and appropriate ambiguity set structure is the box ambiguity set: ambiguity is injected considering a perturbation around the nominal probability distribution. This is equivalent to determining the bounds for the box uncertainty using the parameter Ψ to quantify the discrepancy that can be interpreted as the decision-makers confidence regarding the nominal probability distribution.

The box ambiguity set can be defined as $\Xi^{box} = \{\mathbf{p} = \mathbf{p_0} + \pi | \mathbf{e^T}\pi = 0, \|\pi\|_\infty \leq \Psi\}$ where $\mathbf{p_0} \in \mathscr{R}^{|\Omega|}$ is the nominal distribution of the discrete probability; $\mathbf{e} \in \mathscr{R}^{|\Omega|}$ is the vector of ones; $\pi \in \mathscr{R}^{|\Omega|}$ a perturbation vector; $\|\pi\|_\infty = \max_{\omega\in\Omega} |\pi_\omega|$; $\Psi \in [0,1]$ is the upper bound of the fluctuation ($\mathbf{\Psi}$ will be used to denote $\mathbf{e}\Psi$) and $\mathbf{e^T}\pi = 0$ ensures that the sum of the probabilities represented by the vector $\mathbf{p}$ is equal to 1.

We define binary decision variables $x_{it}, \forall i \in V, t \in T^i$, that take value of one if the activity i starts in period t and zero otherwise. These are first-stage variables. In the second stage, the binary variable q_{it}^{ω} takes value of one if the time at which the payment of the activity i is made at period t in scenario ω. The continuous variable $l_t^{\omega}, \forall t \in T$ shows the amount of the loan borrowed to avoid cash shortage in scenario ω. The model can be formulated as follows.

$$NPV = \max \quad \sum_{i\in V}\sum_{t\in T^i} \frac{c_i^{out}}{(1+\beta)^t} x_{it} + WCVaR[\mathrm{Q}(\mathbf{x}, \xi(\omega))] \tag{1}$$

$$\sum_{t\in T^i} x_{it} = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{t\in T^j} t x_{jt} \geq \sum_{t\in T^i} t x_{it} + p_i \qquad \forall (i,j) \in E \tag{3}$$

$$\sum_{t\in T^{n+1}} t x_{(n+1)t} + p_{n+1} \leq \bar{d} \tag{4}$$

$$x_{it} \in \{0,1\} \qquad \forall i \in V, t \in T^i \tag{5}$$

where

$$\mathrm{Q}(\mathbf{x}, \xi(\omega)) = \max \sum_{i\in V}\sum_{t\in\mathscr{T}} \frac{c_i^{in}}{(1+\beta)^t} q_{it}^{\omega} + \sum_{t\in\mathscr{T}} \frac{l_t^{\omega}}{(1+\beta)^t} - \sum_{t\in\mathscr{T}} \frac{l_t^{\omega}(1+r)^{T-t}}{(1+\beta)^T} \tag{6}$$

$$\sum_{t\in\mathscr{T}|t\geq l_i} t q_{it}^{\omega} = \sum_{t\in T^i} t x_{it} + p_i + \xi_i^{\omega} \quad \forall i \in V \tag{7}$$

$$\sum_{t\in\mathscr{T}|t\geq l_i} q_{it}^{\omega} = 1 \quad \forall i \in V \tag{8}$$

$$C_0 + \sum_{i\in V}\sum_{t'\leq t} c_i^{out} x_{it'} + \sum_{i\in V}\sum_{t'\leq t} c_i^{in} q_{it'}^{\omega} + \sum_{t'\leq t} l_t^{\omega} \geq 0 \quad \forall t \in \mathscr{T} \tag{9}$$

$$q_{it}^{\omega} \in \{0,1\} \quad \forall i \in V, t \in \mathscr{T}, \quad l_t^{\omega} \geq 0 \quad \forall t \in \mathscr{T} \tag{10}$$

First-stage constraints are shown in (2)-(5). The constraints (2) guarantee that an activity starts between its earliest and latest start time. Constraints (3) are added to respect the precedence relations. Constraints (4) guarantee that the project is completed before the deadline. Constraints (5) imposes the binary restrictions on the first-stage variables. In the second stage, constraints (7) formulate the time of cash inflows by considering the possible delay at each scenario. Constraints (8) and the binary restrictions in constraints (10) guarantee that the cash payment for a completed activity is received as lump-sum. Constraints (9) ensure that a capital deficit needs to be covered with the funds l_t^ω borrowed at time t in scenario ω. Constraints (10) impose the binary and non-negativity restrictions on the second-stage variables. The second stage recourse function under scenario ω is denoted as $\mathrm{Q}(\mathbf{x}, \xi(\omega))$. The first term of the objective function (6) represents the total payments received, the second term represents the amount of money borrowed, and the third term displays the financing cost. The money borrowed l_t^ω is considered a cash inflow at time t, but at the end of the time horizon is an outflow since the capital borrowed and the interests should be paid back. The objective function (1) maximizes the net present value under risk. In particular, we consider the average value of the $1-\alpha$ left tail of the distribution, controlling the expected value in a given percentage of worst-case realizations. Following the results presented in (Rockafellar, R. T. and Uryasev, S. 2002), $WCVaR[\mathrm{Q}(\mathbf{x}, \xi(\omega))] = \max_{\eta \in \mathscr{R}^+} \eta - \frac{E_\xi[(\eta - \mathrm{Q}(\mathbf{x},\xi(\omega)))^+]}{1-\alpha} = \max_{\eta \in \mathscr{R}^+} \eta - \frac{1}{1-\alpha} \sum_{\omega \in \Omega} p^\omega \gamma^\omega$. Now, let us focus on the term $\sum_{\omega \in \Omega} p^\omega \gamma^\omega$. Based on the definition of the ambiguity set, we can write $\max_\pi \gamma^T \mathbf{p}$, $\mathbf{p} = \mathbf{p_0} + \pi$, $\mathbf{e^T}\pi = 0$, $\|\pi\|_\infty \leq \Psi$. Using the strong duality in linear programming, we obtain $\gamma^T \mathbf{p_0} + \min \mathbf{\Psi}^T\beta + \mathbf{\Psi}^T\delta$, $\mathbf{e}\mu - \beta + \delta = \gamma$, $\beta \geq 0$, $\delta \geq 0$, μ free. We combine the latter model with model (1)-(10), and formulate the deterministic model under the box ambiguity set :

$$
\begin{aligned}
&\max \sum_{i \in V} \sum_{t \in T^i} \frac{c_i^{out}}{(1+\beta)^t} x_{it} + \eta - \frac{\gamma^T \mathbf{p_0} + \mathbf{\Psi}^T\beta + \mathbf{\Psi}^T\delta}{1-\alpha} && \\
&(2)-(5), (6)-(10) && \\
&\gamma^\omega \geq \eta - \mathrm{Q}(\mathbf{x}, \xi(\omega)) && \forall \omega \\
&\gamma^\omega \geq 0 && \forall \omega \\
&\mathbf{e}\mu - \beta + \delta = \gamma && \\
&\beta \geq 0, \delta \geq 0, \gamma \geq 0, \mu \quad \text{free} &&
\end{aligned}
$$

Here η is a free decision variable, representing the VaR, and decision variables γ^ω are used to express the deviations of $\mathrm{Q}(\mathbf{x}, \xi(\omega))$ below the VaR. By varying the value of $\alpha \in [0, 1)$, different risk preferences can be obtained. At the two extremes, $\alpha = 0$ corresponds to the risk-neutral formulation; in contrast, when $\alpha \to 1$ the decision-maker is highly risk-averse and seeks the NPV maximization under the worst case.The proposed risk-averse two-stage distributional robust model can encompass various models by changing the value of the parameters α and Ψ. In particular, for $\alpha = 0$ and $\Psi = 0$ we obtain a standard two-stage stochastic programming model. For $\alpha = 0$ and $\Psi \neq 0$, we consider distributional ambiguity within a risk-neutral approach. For $\alpha > 0$ and $\Psi = 0$, we have a two-stage CVaR-based risk-averse stochastic programming problem. This flexibility comes at a cost since the resulting model is generally complicated to solve for ambiguity and the risk-aversion.

Solving a distributionally robust problem is a challenging task, especially when distributional robustness is embedded into the realm of two-stage stochastic programs. For this class of problems, an efficient solution method is yet to be proposed. Exploiting the problem structure, a hierarchical heuristic algorithm is designed and tested. It decomposes the problem into an upper-level master problem and a sub-problem. In the upper-level problem, all the variables that belong to the second stage are projected out, and an additional artificial variable, denoting an approximation on the second-stage objective function, is added. In each iteration the master problem is amended with two different cuts coming for the information on the solution corresponding to previous iterations. The first type is known as the "no-good" cut and excludes the current solution from the search space (Balas 1979). The second type of cut requires the solution of the second stage.

3 Computational experiments

To assess the performance of the heuristic approach, we selected 10 instances from the DC2 data set of (Vanhoucke, M. 2010) and considered different combinations of parameters. In particular, we use a deadline that corresponds to an increase (Incr) of $5, 10, 15, 20$ percent of the minimum project duration, 9 combinations of the distribution of the cash flows (Leyman, P. and Vanhoucke, M. 2017), three values $(0.25, 0.50, 0.75)$ for the capital constrainedness (CC), three scenario cardinalities $|\Omega| = \{20, 40, 60\}$. The heuristic algorithm was coded in AIMMS and Gurobi 9.1 was used as a solver. The heuristic is very fast. The CPU time slightly increases for the more involved instances (see, for instance, dc1, dc3, dc10). However, as the number of scenarios increases, our algorithm outperforms the solver, finding better solutions in a few seconds.

References

Balas, E. (1979) "Disjunctive Programming", in Hammer, P. L., Johnson, E. L., and Korte, B. H. (eds) Discrete Optimization II.

Leyman, P. and Vanhoucke, M. (2017) "Capital- and resource-constrained project scheduling with net present value optimization", *European Journal of Operational Research*, 256(3), pp. 757–776.

Rockafellar, R. T. and Uryasev, S. (2002) "Conditional value-at-risk for general loss distributions", *Journal of Banking Finance*, 26(7), pp. 1443–1471.

Vanhoucke, M. (2010) "A scatter search heuristic for maximising the net present value of a resource-constrained project with fixed activity cash flows", *International Journal of Production Research*, 48(7), pp. 1983–2001.

Machine Scheduling with Shift Border Constraints Motivated by a RCPSP from Industry

Felix Buld[1], Sven O. Krumke[2], Stephan Breiter[3], Jörg Hoffmann[3] and Toma Nikolov[3]

[1] TU Munich, Germany
`felix.buld@tum.de`
[2] RPTU Kaiserslautern-Landau, Germany
`sven.krumke@math.rptu.de`
[3] Rolls-Royce Deutschland Ltd & Co KG, Germany

Keywords: Scheduling with Additional Constraints, Applications, Resource-Constrained Project Scheduling.

1 Project overview

This extended abstract centers on the master's thesis of Buld (2023). The thesis deals with scheduling problems arising in a production environment for the final assembly of aero engines. The environment consists of a low-volume production process with significant processing times of several hours for each task. Involved products consist of multiple parts with a complex structure.
Mathematical challenges arise, especially in the coupling of several production lines in time, decisions about which machine to select in a flexible machine environment, recirculation constraints of jobs returning to a machine, and shift border constraints due to the organization of the workforce. A complete model must also consider additional resources such as operators and toolings. The thesis includes the following primary research directions:

1. **Modeling** Buld (2023) studies the scheduling problem with all constraints from a machine and a resource-constrained project scheduling perspective. The full model can be captured using the three-field notation according to Graham *et. al.* (1979) as extended by Pinedo (2016) together with resource constraints $res \cdot t1$ from Blazewicz *et. al.* (1983) and reassignment and border constraints defined in Buld (2023) and Section 5 respectively. The parameter t describes time-dependent resource sizes. It reads
$$FJc \mid intree, r_j, reassign, res \cdot t1, brdrs \mid \gamma \tag{1}$$
with objective $\gamma \in \left\{C_{\max}, \sum_j w_j C_j, \sum_j w_j T_j\right\}$ or alternatively
$$PS \mid sp\text{-}graph, reassign, res \cdot t1, brdrs \mid \gamma \tag{2}$$
using an activity-on-node network and the three-field notation for project scheduling (Brucker *et. al.* 1999) and the notion of *sp-graphs* (Brucker 2007).
2. **Theoretical** The computational complexity of selected sub-problems concerning recirculation and border constraints is analyzed.
3. **Solving** Mixed integer linear programs (e. g. time-indexed, event-based, flow-based) from literature are extended and tailored to the specific scheduling problems. They are implemented for solving alongside heuristic schedule generation schemes such as list scheduling.
4. **Practical** The developed algorithms are incorporated into a prototypical, Python-based software tool and applied to practical data to shorten the time required to create adequate schedules for the machines deployed compared to manual planning.

In this extended abstract, we would like to focus on our results and insights on a specific aspect, namely machine scheduling with shift border constraints.

2 Modeling scheduling problems with border constraints

In fundamental scheduling problems, one usually regards time to be continuous. For the considered application, we relax this assumption. Let us introduce the *border constraints*. It connects scheduling with BIN PACKING problems.

Given a planning horizon $[0, T]$, let $B = \{b_0, b_1, b_2, \ldots, b_q\} \subseteq [0, T]$ with

$$0 = b_0 < b_1 < \ldots < b_q = T \tag{3}$$

be a finite set of *(shift) borders*. For a scheduling problem $\alpha \mid \beta, brdrs \mid \gamma$, constraint $brdrs$ expresses that a set of borders B is given and that no job $j \in \mathcal{J}$ crosses a border, i. e.

$$\{j \in \mathcal{J} \mid S_j < b < C_j\} = \emptyset \qquad \text{for all } b \in B. \tag{4}$$

Intervals between two borders are denoted as *bins* L, and their lengths are abbreviated by Δ_l for all $l \in L$.

For the general analysis, the bins may have different lengths. Figure 1 illustrates the notation. The depicted schedule is infeasible concerning border constraints as jobs 2 and 3 cross borders.

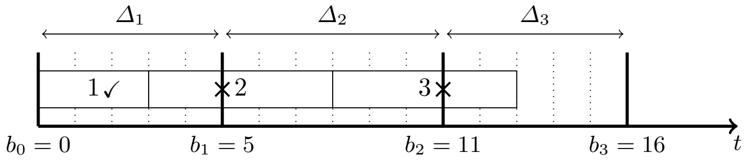


Fig. 1. Single machine instance for the introduction of notation.

3 Setting and consequences in the real-world application

In the considered application, each task needs to be conducted by exactly one operator. And every operator can complete one task at a time. The operators work in a shift-based work model. A *shift* defines a time slot. The fixed duration of every shift is called *shift time*. Imagine a task to be an alternating sequence of manual steps and machine processes on a product. Since it must be assigned to one operator, such a task must be completed within one shift. It means the tasks cannot cross a shift border, i. e. start before and finish afterward. If a task can not be accomplished in one shift, it will be postponed until the next shift begins. This causes unwanted but forced idle time. Figure 2 depicts this postponement.

Depending on the precedence-structure, border constraints may prolong the overall time from the initially scheduled start until the actual completion of a task by a significant amount. Section 5 quantifies this amount for basic settings.

The company introduced a takt scheme to pace the production line for the main products. Due to low volume production, the takt time amounts to several hours, even larger than a shift time. The challenge in this application constitutes designing a schedule that coheres with the desired takt scheme and operator availability based on shifts. The critical point is the long processing times of tasks. Even more complexity arises since the tasks for the main products share the machine park with other value streams.

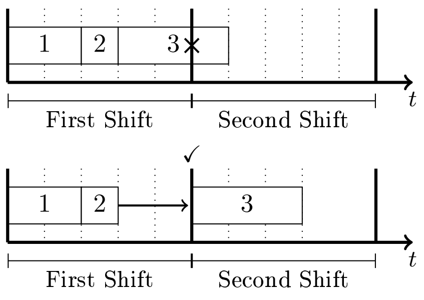


Fig. 2. Situation at shift borders for tasks in the considered application.

4 A possible solution approach for dealing with shift border constraints

Buld (2023) shows that even the single machine scheduling problems $1 \mid brdrs \mid C_{\max}$ and $1 \mid brdrs \mid \sum_j C_j$ with border constraints are $\mathcal{NP}$-hard in the strong sense.

Since including the border constraints turns a vast class of originally efficiently solvable scheduling problems $\mathcal{N}P$-hard, one can ask if one can solve the original problem and modify the obtained schedule slightly to get a "near-optimal" one for the problem with border constraints. One may compare it to changing preemptive schedules into non-preemptive schedules, as often done in approximation algorithms. The idea is to ***push*** one job after another ***to the right*** to meet the border constraints, as it is motivated in the application. This idea is similar to the *Next-Fit heuristics* for BIN PACKING (Johnson 1973).

Buld (2023) formalizes this procedure for certainly structured scheduling problems within the *pushing algorithm*. The structure one needs is the encoding of a schedule in an acyclic *graph of technical and organizational sequences* $G_{org} = (V, E \cup E_{org})$. The nodes V correspond to the tasks to be scheduled, and the set of edges comprises *given* technical sequences and precedences E. *Chosen* organizational sequences for every machine E_{org} extend the set of edges. One then sorts nodes topologically, iterates through the nodes in this order, and computes starting times concerning the border constraints. This is possible in $\mathcal{O}(|V| + |E| + |E_{org}|)$ time (Buld 2023).

5 Theoretical results on the solution approach

Assume that every job fits into every bin, i. e. $p_{\max} \leq \Delta_{\min}$. This condition ensures that each considered instance of a scheduling problem with border constraints has a feasible solution and a convenient analysis, similar to the one for BIN PACKING (Johnson 1973). Buld (2023) shows that the pushing algorithm increases the length of each path in G by at most factor two. Therefore, the makespan $C_{\max}$ and the sum of completion times $\sum_j C_j$ after pushing tasks is at most doubled. Even more, this allows to bound the gap between the optimal objective values for a scheduling problem without and with border constraints. Following (Buld 2023), the result reads:

Theorem 1. *For a scheduling problem with environment* $\alpha \in \{1, P_m, F_m, J_m, FFc, FJc\}$ *and* $p_{\max} \leq \Delta_{\min}$*, it holds*

$$OPT(\alpha \mid r_j, prec, brdrs \mid C_{\max}) \leq 2 \cdot OPT(\alpha \mid r_j, prec \mid C_{\max}), \tag{5}$$

$$OPT(\alpha \mid r_j, prec, brdrs \mid \sum_j C_j) \leq 2 \cdot OPT(\alpha \mid r_j, prec \mid \sum_j C_j). \tag{6}$$

Note that this result is independent of bins' relative sizes. This bound is tight for single machine or flow shop environments (Johnson 1973, Buld 2023). If we consider tasks that are *small* compared to the bins' sizes, we can give better bounds:

Theorem 2. *For a scheduling problem with environment* $\alpha \in \{1, P_m, F_m, J_m, FFc, FJc\}$ *and* $p_{\max} \leq q \cdot \Delta_{\min}$ *with* $q \in (0, 1]$*, it holds*

$$OPT(\alpha \mid r_j, prec, brdrs \mid C_{\max}) \leq \frac{1}{1-q} \cdot OPT(\alpha \mid r_j, prec \mid C_{\max}), \tag{7}$$

$$OPT(\alpha \mid r_j, prec, brdrs \mid \sum_j C_j) \leq \frac{1}{1-q} \cdot OPT(\alpha \mid r_j, prec \mid \sum_j C_j). \tag{8}$$

Theorem 2 improves the bound from Theorem 1 for $q \in (0, 1/2]$. Thus, we can conclude that border constraints may be negligible if tasks' processing times are relatively small compared to bins' sizes. However, it is still important to consider border constraints within the scheduling in applications with comparatively large processing times like the one at hand.

6 Conclusions

In this extended abstract based on Buld (2023), we present results on machine scheduling with border constraints. The initial use case lies in the practical project, which is to design schedules that respect takt scheme requirements and simultaneously meet shift border constraints defined in Section 2.

In Section 5, we show how much impact border constraints can have in the worst case for certainly structured scheduling problems. A numerical study on real-world data concludes this use case in Buld (2023).

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, no. 1, pp. 11-24.

Brucker P., 2007, "Scheduling Algorithms", Springer, Berlin; Heidelberg.

Brucker P., A. Drexl, R. Möhring, K. Neumann and E. Pesch, 1999, "Resource-constrained project scheduling: Notation, classification, models, and methods", *European Journal of Operations Research*, Vol. 112, no.1, pp. 3-41.

Buld F., 2023, "Efficient Algorithms for Resource-Constrained Project Scheduling in a Production Environment", *Master's Thesis*, RPTU Kaiserslautern-Landau, Department of Mathematics.

Graham R.L., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, 1979, "Optimization and approximation in deterministic sequencing and scheduling: a survey", *Annals of discrete mathematics*, Vol. 5, pp 287-326.

Johnson D.S., 1973, "Near-optimal bin packing algorithms", *PhD Thesis*, Massachusetts Institute of Technology.

Pinedo M.L., 2016, "Scheduling theory, algorithms, and systems", fifth ed., Springer, Cham; Heidelberg; New York; Dordrecht; London.

A purely buffer-based approach to the proactive and reactive resource-constrained project scheduling problem

Yanfei Chen, Erik Demeulemeester and Jannik Matuschke

Research Centre for Operations Management, KU Leuven, Belgium
`yanfei.chen, erik.demeulemeester, jannik.matuschke@kuleuven.be`

Keywords: Project scheduling, Scheduling under uncertainty, PR-RCPSP, Proactive and reactive policies, Buffer-based reactions

1 Introduction

The *proactive and reactive resource-constrained project scheduling problem* (PR-RCPSP), introduced by Davari and Demeulemeester (2019), addresses uncertainties in real-world projects in a novel manner. A PR-RCPSP solution is a *proactive and reactive policy* (PR-policy) that includes the baseline schedule and foresees potential transitions (reactions) to other schedules. In their subsequent work, reactions are categorized into *selection-based* and *buffer-based*. While both classes are crucial, buffer-based reactions hold greater significance. Acknowledging their theoretical and managerial importance, we focus on a purely buffer-based approach to solve the PR-RCPSP. Our method includes a strategy for generating a sufficient selection and innovative heuristics for proactive and reactive procedures. The synergy of these methods allows us to construct schedule pools and derive optimal PR-policies. Experimental results reveal the superiority of our solutions over existing alternatives in terms of both computational time and combined cost.

2 The PR-RCPSP

The PR-RCPSP involves activities $N = \{0, 1, \ldots, n+1\}$, with 0 and $n+1$ as dummy start and end activities. Non-dummy activities $i \in N \setminus \{0, n+1\}$ have stochastic integer durations $\tilde{p}_i$ within $[p_i^{\min}, p_i^{\max}]$. Dummy activities have $\tilde{p}_0 = \tilde{p}_{n+1} = 0$. Realizations $\mathcal{B} = \{\boldsymbol{p}^l \mid l = 1, \ldots, |\mathcal{B}|\}$ have probabilities $\pi(\tilde{\boldsymbol{p}} = \boldsymbol{p}^l)$. Instances include renewable resources $\mathcal{R}$ with availabilities R_k, and activity i requires r_{ik} units of resource k. Zero-lag finish-start precedence relations E indicate j starts after i finishes, with transitive closure $T(E)$ and transitive reduction $t(E)$.

Solutions to the PR-RCPSP are PR-policies, denoted as $\mathit{\Pi}$. At each decision moment, we update our information about project execution. All the available information collectively defines the state of execution, and each decision in PR-policy $\mathit{\Pi}$ corresponds to a certain state of execution. A schedule vector is represented as $\boldsymbol{s}$. The cost of a PR-policy has baseline and reaction components, including fixed and deviation costs. The formulation accounts for baseline and reaction costs over realizations:

$$\min_{\mathit{\Pi} \in \boldsymbol{\mathit{\Pi}}} \sum_{l=1}^{\mathcal{B}} \pi(\tilde{\boldsymbol{p}} = \boldsymbol{p}^l) \left\{ w_b s_{n+1}^{[0]_{\mathit{\Pi}}} + \sum_{k=1}^{\nu_{\mathit{\Pi},l}} \left(w_r + \sum_{i \in U\left(\boldsymbol{s}^{[k-1]_{\mathit{\Pi},l}}, t_k\right)} w_i \left| s_i^{[k]_{\mathit{\Pi},l}} - s_i^{[k-1]_{\mathit{\Pi},l}} \right| \right) \right\}. \quad (1)$$

Here, $\boldsymbol{\mathit{\Pi}}$ is the set of all PR-policies, and w_b, w_r, and w_i denote baseline, fixed reaction, and deviation costs.

3 Methodologies

Our methodology for solving the PR-RCPSP consists of three phases. In the first phase, we generate a sufficient selection that will be used to create schedules in the schedule pool. In the second phase, we first construct the initial schedule pool using various heuristics, and then, we augment the initial schedule pool by reacting to conflicts. In the last phase, we obtain the optimal PR-policy by solving the PR-RCPSP using Model 3. The flow chart of our approach is illustrated in Figure 1.

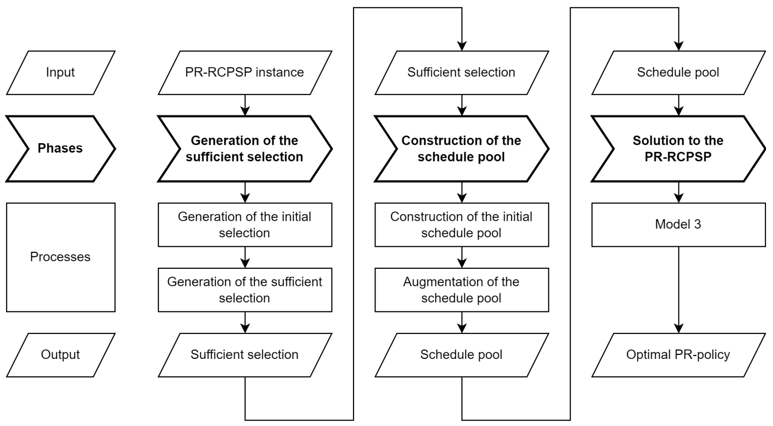

Input
Phases
Processes
Output
PR-RCPSP instance
Generation of the sufficient selection
Generation of the initial selection
Generation of the sufficient selection
Sufficient selection
Sufficient selection
Construction of the schedule pool
Construction of the initial schedule pool
Augmentation of the schedule pool
Schedule pool
Schedule pool
Solution to the PR-RCPSP
Model 3
Optimal PR-policy

Fig. 1. A purely buffer-based approach to the PR-RCPSP

3.1 Generation of the sufficient selection

The generation of the sufficient selection involves two stages. In the first stage, an initial selection, denoted as X^{init}, is created by generating the schedule $\boldsymbol{s}^{\text{init}}$ using an initial realization $\boldsymbol{p}^{\text{init}} \in \mathcal{B}$ through the branch-and-bound method by Demeulemeester and Herroelen (1992). Any implicitly added resource arcs during this process are identified by comparing the starting time of each activity $i \in N \setminus \{0, n+1\}$ with the maximum finishing time of its predecessors. The selected activities are added to X^{init}. The initial selection X^{init} is then extended to a sufficient selection X. We use the algorithm proposed by Stork and Uetz (2005) to enumerate all minimal forbidden sets and eliminates them by introducing resource arcs.

3.2 Construction of the schedule pool

In this study, all schedules in both the initial and final schedule pools are generated by applying the *earliest start policy* (ES-policy) to the sufficient selection X and a realization $\boldsymbol{p}$. This approach ensures that every reaction in the optimal PR-policy adheres to the buffer-based principle. We explore five distinct initial schedule pools in our experiments, the most important of which is a set constructed using the *reversed starting time criticality* (reversed STC) heuristic. An extension of the STC heuristic proposed by Van de Vonder *et. al.* (2008), the reversed STC heuristic inherits key concepts from STC, where the criticality of activity starting times is assessed based on probability assumptions. The reversed STC heuristic introduces the concept of reversed STC values, considering the potential impact of each activity on its direct and transitive successors. The algorithm iteratively selects activities with the highest reversed STC values, introducing buffers after them and generating new schedules. The process continues until the project makespan meets a deadline, maintaining the same sufficient selection.

Once we have obtained the initial schedule pool, we proceed to augment it by generating high-quality reactions to conflicts. This phase aims to strike a balance between diversity and similarity in the final schedule pool. Buffer-based reactions are ensured through the introduction of the *buffer-based schedule generation scheme* (buffer-based SGS). The scheme reacts to conflicts, resulting in diverse schedules while maintaining buffer-based principles. To achieve this goal, the algorithm randomly selects pairs of schedules and realizations from the initial schedule and realization pools. Buffer-based SGS reacts to conflicts in the selected schedule-realization pair, generating new schedules by adjusting ongoing activity durations. These new schedules, contributing to diversity, are added to the schedule pool if not already present. The process continues until the desired size of the final schedule pool is achieved.

3.3 Solution to the PR-RCPSP

Upon completion of constructing the schedule pool, we advance to tackle the PR-RCPSP utilizing Model 3, a Markov decision process (MDP) introduced by Davari and Demeulemeester (2019).

4 Findings

Experiments are performed on J30X_1 instances from PSPLIB (Kolisch and Sprecher 1997). The obtained results are juxtaposed with those from Davari and Demeulemeester (2019), identified as DD.

4.1 Main results

The key findings are summarized in Table 1. Across all ten scenarios, RSTC consistently establishes the lowest average computational time, with the average instance-wise margin frequently exceeding 50% relative to DD. Concerning the combined cost, RSTC exhibits notable effectiveness particularly in scenarios where the ratio of the fixed reaction cost w_r to the cost per unit time for the baseline schedule w_b is high. In cases where w_r is small, RSTC retains its competitive edge against DD, requiring only half of the computational time while delivering comparable combined costs on an instance-wise basis. The buffer-based nature of the reversed STC heuristic and the buffer-based SGS justifies these outcomes, emphasizing the advantages of preserving the resource flow of the project under uncertainty.

4.2 Supplementary experiments

We conduct supplementary experiments to further explore our purely buffer-based approach and the PR-RCPSP in general. Among the 48 generated sufficient selections, 40 are dominant, theoretically avoiding superfluous resource arcs that might reduce the solution space. We propose two variants of the reversed STC heuristic, finding that increased complexity can lead to overfitting. Meticulous experiments are conducted to address the balance between diversity and similarity within the schedule pool. We reveal that computing the optimal size of the initial schedule pool for one specific scenario is justifiable but practically impossible and oftentimes counterproductive. Compared to DD, RSTC tends to achieve a lower combined cost in scenarios with high resource factors and low resource strengths.

Table 1. Comparison of results between DD and RSTC

$\mathcal{S}^{\text{init}}$	w_b	w_r	Time	%ΔTime	Cost	%ΔCost	#Best	#States	#Cuts	#Conts
DD	25	0	245.95		1,649.21		41	3,987,309	16,740	11,656
		50	450.37		1,815.39		14	4,561,013	17,107	11,656
		100	372.60		1,931.26		5	4,181,568	16,865	11,656
		150	365.29		2,023.28		4	3,924,144	16,653	11,656
		200	380.05		2,104.30		1	3,692,974	16,462	11,656
RSTC	25	0	96.31	−54.25	1,683.65	2.07	0	2,475,690	9,075	10,792
		50	220.18	−29.54	1,812.98	−0.08	10	3,270,352	9,139	10,792
		100	252.16	−8.72	1,883.78	−2.36	22	3,031,343	9,051	10,792
		150	213.56	−26.21	1,931.76	−4.36	27	2,754,131	8,955	10,792
		200	215.60	−29.69	1,968.27	−6.22	29	2,544,078	8,865	10,792
DD	50	0	266.30		3,172.66		36	3,971,584	16,193	11,533
		50	387.07		3,384.20		35	4,332,693	16,343	11,533
		100	222.97		3,539.00		22	4,010,686	16,116	11,533
		150	393.96		3,658.44		7	3,794,780	15,928	11,533
		200	377.09		3,758.56		6	3,585,239	15,774	11,533
RSTC	50	0	98.06	−50.77	3,221.23	1.58	0	2,402,059	8,844	10,792
		50	216.13	−24.37	3,427.35	1.31	0	3,270,352	9,139	10,792
		100	222.97	−21.65	3,557.28	0.54	5	3,031,343	9,051	10,792
		150	233.41	−20.46	3,646.22	−0.30	10	2,754,131	8,955	10,792
		200	205.53	−28.67	3,716.87	−1.07	13	2,544,078	8,865	10,792

5 Conclusions

In this study, we present a purely buffer-based approach to tackle the PR-RCPSP. The introduced reversed STC heuristic yields a schedule pool that consistently requires the least average computational time, achieves a competitive combined cost when w_r is small, and dominates in scenarios where w_r is large. The key takeaway for project managers is the importance of prioritizing stability and avoiding unnecessary alterations to task assignments, particularly when uncertainty-related costs are significant.

References

Davari M., Demeulemeester E., 2019a, “Important classes of reactions for the proactive and reactive resource-constrained project scheduling problem”, *Annals of Operations Research*, Vol. 274, pp. 187-210.

Davari M., Demeulemeester E., 2019b, “The proactive and reactive resource-constrained project scheduling problem”, *Journal of Scheduling*, Vol. 22, pp. 211-237.

Demeulemeester E., Herroelen W., 1992, “A branch-and-bound procedure for the multiple resource-constrained project scheduling problem”, *Management Science*, Vol. 38, pp. 1803-1818.

Kolisch, R., Sprecher A., 1997, “PSPLIB-A project scheduling problem library: OR software-ORSEP operations research software exchange program”, *European Journal of Operational Research*, Vol. 96, pp. 205-216.

Stork F., Uetz M., 2005, “On the generation of circuits and minimal forbidden sets”, *Mathematical Programming*, Vol. 102, pp. 185-203.

Van de Vonder S., Demeulemeester E. and Herroelen W., 2008, “Proactive heuristic procedures for robust project scheduling: An experimental analysis”, *European Journal of Operational Research*, Vol. 189, pp. 723-733.

Exact algorithms for a shift scheduling problem in intra-hospital patient transfer

Marco Claps[1], Maxence Delorme[2], Manuel Iori[1] and Giorgio Zucchi[3]

[1] DISMI, University of Modena and Reggio Emilia, Italy
marco.claps@unimore.it, manuel.iori@unimore.it
[2] TISEM, Tilburg University, Netherlands
m.delorme@tilburguniversity.edu
[3] R&D Department, Coopservice S.coop.p.a, Reggio Emilia, Italy
giorgio.zucchi@coopservice.it

Keywords: Shift Scheduling, Integer Linear Programming, Hierarchical Models, Decomposition Algorithm

1 Introduction

Intra-hospital patient transfer (IHPT) refers to the transfer of a patient from one department to another in the same hospital and has become particularly relevant after the Covid-19 pandemic. Our research concerns the homogeneous scheduling of a crew composed of identical employees who must perform a given IHPT task (i.e., carrying a patient) over a given time horizon. The problem is to determine the shifts of the employees such that the demand in each time period is covered, while the constraints imposed by public healthcare, labour law, and the company that provides the IHPT service are respected. An important objective for the company (which is not often studied in the literature) is to increase schedule homogeneity. Two shifts are said to be part of a homogeneous schedule if they start and end at the same time period but take place on different days. In this work, we introduce three tailored IHPT solution methods, two using *integer linear programming* (ILP) and one using a decomposition algorithm, and evaluate them on real instances provided by hospitals in the region of Tuscany. We demonstrate that our algorithms can decrease the total schedule costs, computed as the number of worked hours, while also increasing the schedule homogeneity compared to the solution currently adopted by the company.

2 Literature Review

The existing literature on *personnel scheduling problems* (PSP) is very large and covers different areas such as logistics, healthcare, retail and manufacturing. A comprehensive review of PSP was provided by Ernst et al. 2004, according to which it is possible to classify our problem as a "flexible-demand PSP", because the number of employees required may vary from one time period to another. In our work, the time horizon corresponds to a week, a time period corresponds to an hour and the demand for each hour is obtained by using forecasting techniques based on historical data.

Following the literature, there are three main strategies when it comes to building a PSP schedule. The first, and most intuitive strategy, consists of finding a suitable task-employee matching. For example, Krishnamoorthy et al. 2001 proposed ILP models for a PSP in which tasks with fixed starting and ending times were assigned to a heterogeneous workforce. Whereas those ILP models could be solved with a state-of-the-art ILP solver, the authors recognized that not all relevant PSP constraints could easily be modeled using ILP. The second strategy consists of enumerating every feasible sequence of tasks that can be performed by each employee forming a so called stint (see Ernst et al. 2004) and solving a set covering type model where each employee is assigned to exactly one stint. For example, Mehrotra et al. 2000 used such a strategy to solve multiple staff scheduling problems with breaks. Even though such a strategy can be effective in solving some PSPs, the number of feasible stints grows exponentially with the number of periods considered in the time horizon, which quickly becomes an issue for state-of-the-art ILP solvers. The last strategy is a mix between the other two and consists of dividing the time horizon into a set of rotations, enumerating every feasible sequence of tasks that can be performed by each employee for each rotation, and assigning each employee to exactly one shift per rotation. For example, Legrain et al. 2020 used such a strategy to solve a nurse scheduling problem. The nurse scheduling problem, which has been widely studied in the literature, has many similarities with our problem. We refer the reader to the work ofBurke et al. 2004 for a survey on the topic. Even though most practical nurse scheduling problems are solved through non-exact approaches (see, e.g., Millar et al. 1998) because of the complexity induced by real-world case studies, promising algorithms based on decomposition methods were also proposed in the recent literature. For example, Brucker et al. 2010 suggested a decomposition algorithm where the stints are constructed in a first phase and then assigned to the nurses in a second phase. In this work, we mix the strategies of Legrain et al. 2020 and Brucker et al. 2010 to develop a shift-based decomposition approach for our IHPT problem.

3 Problem Definition and mathematical model

We consider a set $T = D \times H$ of 168 time periods that can be decomposed into 7 days (or rotations) of 24 hours each. For each time period $t \in T$, we are given a staffing requirement (or demand) d_t that must be covered by a set of shifts. Every shift $s \in S$ has a starting time t_s ($t_s \in T$), a duration of $r_s \in \{4, 5, 6, 7, 8\}$ consecutive time periods, and an ending time $t_s + r_s$ ($t_s + r_s \in T$). Every day, each employee $e \in E$ can be assigned to at most one shift $s \in S$, provided that (i) there is a minimum period of $r^{min} = 16$ resting hours between the moment an employee ends a shift s and the moment they start their next shift s', (ii) an employee performs at most $s^{max} = 5$ shifts in total, and (iii) an employee does not work more than $h^{max} = 40$ hours in total. To model schedule homogeneity, we also define the concept of a shift family. We state that two non-identical shifts s and s' belong to the same shift family $f \in F$ if both shifts have the same duration (i.e., $r_s = r_{s'}$) and the same starting hour, but not the same starting day (i.e., $t_s \equiv t_{s'} \mod 24$). For modeling purposes, we define the following sets: (i) all the shifts with family f are gathered in $S(f)$, (ii) all the shifts that include time period t as a working time (i.e., all the shifts s such that $t_s \leq t < t_s + r_s$) are gathered in $S(t)$, and (iii) all the shifts that include time period t as a working or resting time (i.e., all the shifts s such that $t_s \leq t < t_s + r_s + r^{min}$) are gathered in $I(t)$. By introducing binary decision variables x_{es} taking the value 1 if employee e is assigned to shift s and binary decision variables y_f taking the value 1 if the shift family f is used in the solution, the IHPT problem that we face can be modeled as follows:

$$\min \quad k \sum_{e \in E} \sum_{s \in S} r_s x_{es} + \sum_{f \in F} y_f \tag{1}$$

$$\text{s.t.} \quad \sum_{e \in E} \sum_{s \in S(t)} x_{es} \geq d_t \qquad \forall t \in T, \tag{2}$$

$$\sum_{s \in S} r_s x_{es} \leq h^{max} \qquad \forall e \in E, \tag{3}$$

$$\sum_{s \in S} x_{es} \leq s^{max} \qquad \forall e \in E, \tag{4}$$

$$\sum_{s\in I(t)} x_{es} \leq 1 \qquad \forall e \in E, \forall t \in T, \tag{5}$$

$$\sum_{e\in E}\sum_{s\in S(f)} x_{es} \leq M y_f \qquad \forall f \in F, \tag{6}$$

$$x_{es} \in \{0,1\} \qquad \forall e \in E, \forall s \in S, \tag{7}$$

$$y_f \in \{0,1\} \qquad \forall f \in F. \tag{8}$$

The objective function (1) minimizes the total number of worked hours and the number of shift families used. A suitable coefficient k to establish a hierarchy a lexicographic ordering of objectives) between the two components is $k = 24 \times 5 = 120$ (as there are 24 possible starting times and 5 possible durations for a shift). Constraints (2) ensure that the demand is covered in each time period. Constraints (3) and (4) impose that the workload of each employee satisfies the requirements. Constraints (5) ensure that the minimum resting period between two consecutive shifts is respected. Finally, big-M constraints (6) link the two sets of variables x_{ed} and y_f, where a suitable big-M coefficient is $7 \times |E|$ (assuming that 7 is the number of days). Model (1)-(8), referred to as **M1** hereafter, does not perform very well empirically even though it possesses a polynomial number of variables ($O(|E||S|)$) and constraints ($O(|E||T|)$).
Inspired by the recent work of Delorme et al. 2022 on hierarchical optimization, our second approach, referred to as **M2** hereafter, consists of solving two successive ILP models. First, the number of worked hours, $z_1 = \sum_{e\in E}\sum_{s\in S} r_s x_{es}$, is minimized using constraints (2)-(5) and (7). Second, the number of shift families used, $z_2 = \sum_{f\in F} y_f$, is minimized using constraints (2)-(8) together with supplementary constraint $\sum_{e\in E}\sum_{s\in S(t)} x_{es} \leq z_1$. Even though experiments showed that **M2** outperformed **M1** in terms of number of instances solved to optimality, large hospitals (with a large number of employees $|E|$) remained a challenge. Consequently, we also developed a decomposition method suitable for large-size instances.

4 Decomposition method

Following the idea developed by Brucker et al. 2010, we decomposed the IHPT problem in two phases: in the first phase we determine the shifts that minimize the number of worked hours and the number of shift families used, and in the second phase we determine whether there exists a feasible shift allocation of the employees. By introducing integer decision variables x_s indicating the number of times shift s is used in the solution and re-using binary decision variables y_f, the first phase of the IHPT problem can be modeled as follows:

$$\min \qquad k\sum_{s\in S} r_s x_s + \sum_{f\in F} y_f \tag{9}$$

$$\text{s.t.} \qquad \sum_{s\in S(t)} x_s \geq d_t \qquad \forall t \in T, \tag{10}$$

$$\sum_{s\in S} r_s x_s \leq h^{max}|E|, \tag{11}$$

$$\sum_{s\in S} x_s \leq s^{max}|E|, \tag{12}$$

$$\sum_{s\in I(t)} x_s \leq |E| \qquad \forall t \in T, \tag{13}$$

$$\sum_{s \in S(f)} x_s \leq M y_f \qquad \forall f \in F, \tag{14}$$

$$x_s \in \mathbb{N}_0 \qquad \forall s \in S, \tag{15}$$

$$y_f \in \{0,1\} \qquad \forall f \in F, \tag{16}$$

which is a direct adaptation of model (1)-(8) in which $x_s = \sum_{e \in E} x_{es}$. Note that, like **M2**, model (9)-(16) can also be solved hierarchically. If we denote by $\bar{x}_s (s \in S)$ a solution of model (9)-(16), the second phase of the IHPT problem can be modeled with constraints (2)-(8) plus $\sum_{e \in E} x_{es} = \bar{x}_s, \forall s \in S$. This second phase consists of assigning the shifts found in the first phase such that the workload of each employee and the resting period between two consecutive shifts are respected. If the second phase is infeasible, one may decide to go back to the first phase after adding a so-called "no-good cut".

5 Results and Conclusions

The three models were coded in Python and executed on an MSI GF63 Thin with Intel core i7-11800H 2.5 GHz with 16 GB of RAM on Windows 11. Gurobi 10.0.3 was used as a solver. A time limit of 300 seconds was imposed for each execution of the model. We ran the three models on a set of 16 real instances and reported the computation time in the following table:

	H 01	**H 02**	**H 03**	**H 04**	**H 05**	**H 06**	**H 07**	**H 08**	**H 09**	**H 10**	**H 11**	**H 12**	**H 13**	**H 14**	**H 15**	**H 16**
M1	8.02	121.56	17.04	T.L.	13.33	T.L.	4.72	12.29	T.L.	10.99	9.90	T.L.	5.24	T.L.	T.L.	T.L.
M2	28.21	34.75	62.80	T.L.	39.42	T.L.	18.70	24.17	235.51	42.54	29.54	76.64	15.87	63.08	T.L.	64.95
M3	4.80	4.29	6.78	9.05	6.85	137.35	4.49	5.38	24.70	4.19	5.13	8.14	4.83	9.37	13.14	8.00

We observe that **M3** could solve all the instances to optimality in 12.65 seconds on average, which is significantly better than **M2** (13 instances solved to optimality in 86.36 seconds on average) and **M1** (9 instances solved to optimality in 140.56 seconds on average). Also, the solution provided by M3 uses 4.25 less hours on average when compared to the solution currently adopted by the company.

Future research consists of studying larger instances in which the second phase of **M3** is infeasible and evaluating the performance of the model when no-good cuts are required. We are also interested in the stochastic aspect of our IHPT problem and in developing scenarios for the demand instead of considering a deterministic measure. Another direction could be to focus on the workload balance; in this scenario, our formulation could be easily adapted, for example by using variables y_{ef} instead of y_f.

References

Brucker, Peter et al. (2010). "A shift sequence-based approach for nurse scheduling and a new benchmark dataset". In: *Journal of Heuristics* 16, pp. 559–573.

Burke, Edmund K et al. (2004). "The state of the art of nurse rostering". In: *Journal of scheduling* 7, pp. 441–499.

Delorme, Maxence et al. (2022). "Improved instance generation for kidney exchange programmes". In: *Computers & Operations Research* 141, p. 105707.

Ernst, Andreas T et al. (2004). "Staff scheduling and rostering: A review of applications, methods and models". In: *European journal of operational research* 153.1, pp. 3–27.

Krishnamoorthy, Mohan and Andreas T Ernst (2001). "The personnel task scheduling problem". In: *Optimization methods and applications*, pp. 343–368.

Legrain, Antoine, Jérémy Omer, and Samuel Rosat (2020). "A rotation-based branch-and-price approach for the nurse scheduling problem". In: *Mathematical Programming Computation* 12, pp. 417–450.

Mehrotra, Anuj, Kenneth E Murphy, and Michael A Trick (2000). "Optimal shift scheduling: A branch-and-price approach". In: *Naval Research Logistics (NRL)* 47.3, pp. 185–200.

Millar, Harvey H and Mona Kiragu (1998). "Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming". In: *European journal of operational research* 104.3, pp. 582–592.

A Systems Thinking Theoretical Foundation for Project Uncertainty Management and Strategies.

Author Paul W.M. Cuypers

HYVES Project Research
`paul@hyves.no`

Keywords: risk management, systems thinking, uncertainty management.

1 Introduction

Uncertainty permeates every aspect of project work. As one cannot control what is not defined, and the definition of uncertainty is uncertain, this paper presents a generic project uncertainty model using a systems thinking approach. Elemental strategies for managing project uncertainty are identified based on the model, and techniques are suggested for strategy implementation. The fundamental differences between risk and uncertainty management and the need for a scientifically transcendent approach are highlighted.

2 Literature

The PMBOK (PMI, 2017) does not explicitly define uncertainty. Still, it describes risk as an uncertain event or condition that, if it occurs, has a positive or negative effect on project objectives. Ward and Chapman (2002) argue that focusing on uncertainty provides an essential difference in perspective through an enhanced focus on both opportunity and risk. Another view on the difference between risk and uncertainty is the ability to quantify the probability of an event occurring, implying that risk and uncertainty are mutually exclusive (Perminova et al., 2008; Migilinskas and Ustinovicius, 2008). Others, like Meyer, Loch, and Pich (2002), perceive uncertainty as variability, classified into stages starting with variation, foreseeable uncertainty, unforeseeable uncertainty, and chaos. Different approaches for dealing with uncertainty are suggested. Pitch, Loch, and De Meyer (2002) identify three fundamental project management strategies: instructionism, learning, and selectionism. Johansen et al. (2014) present a process-oriented nine-step framework for identifying, analyzing, and managing project uncertainty involving objectives, threats, opportunities, decisions, and stakeholders, resulting in a combined risk and opportunity matrix. Hubbard and Evans (2010) dispute the effectiveness of such matrices. Raadgever et al. (2011) confirm the validity of the classic strategies for managing uncertainty: Ignoring, knowledge generation, interaction, and coping. Ackermann (2023) promotes systems thinking to induce broad, systematic, and regular thinking and evaluate risk events using causal loop diagrams.

3 Methodology

In this research, a project is perceived as a conceptual system, where a *system* is defined as a whole that contains two or more parts, where each part potentially affects the properties or the behavior of the whole. None of the parts has an independent effect; how any part influences the whole depends on how others act. The elements of a system are always connected; between any parts, there is always a direct and an indirect path. Compared to functional and process analysis, the systems thinking approach makes sense of the real-world complexities by looking at the whole picture rather than splitting them into parts.

4 Project uncertainty

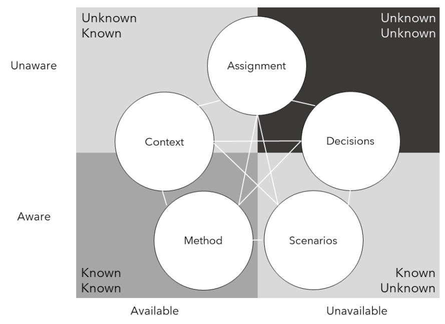


Fig. 1. The project system with the uncertainty matrix projected.

Uncertainty can be defined as a lack of information awareness or information availability. The uncertainty matrix (figure 1) visualizes this principle forming the quadrants:

- Known-known refers to information that is available and its relevance is understood
- Known-unknown is desired, but unavailable information
- Unknown-known, information is present, but one is unaware of its existence
- Unknown-unknown, relates to unavailable information and unawareness of its pertinence.

A project can be modeled as a conceptual system, with principal elements and subcomponents being the:

- Assignment (stakeholders, benefits, deliverables, activities, resources)
- Context (prerequisites, inter-dependencies, constraints, threats, opportunities)
- Decisions (qualifiers, rationale, alternatives, effects, confirmation)
- Method (approach, techniques, tools, coordination mechanisms, communication channels)
- Scenario (plan, risks, problems, crises, posture).

Uncertainty exists regarding the elements of the system (*static perspective*), the interactions between elements, outside forces, and the effects of interventions on natural variation (*dynamic perspective*). The *primary strategies* to systematically minimize uncertainty are raising information awareness, increasing information availability, improving information effectiveness, and maximizing information efficiency. The scenario elements consist of the components *plan* (intended order of events), *risk* (potential for problems and crisis), *problems* (unwanted or harmful situations), and *crises* (unsolvable problems) and posture. During the secure stage, risks are conceptual. A risk event (RE) occurs when an actual risk is introduced. Launching a ship opens the risk of sinking. Being a unique endeavor, the risk stage coincides with the normal operating mode of the project. A *problem event* (PE) occurs when the actual state deviates from the intended target or status quo. When all options to remedy the problem have failed, and the only alternative is to reduce the damage,

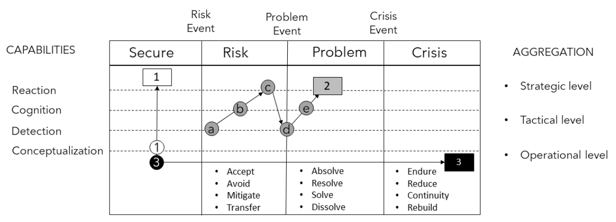


Fig. 2. The stages of the risk continuum.

a *crisis event* (CE) is a fact. Projects must acquire adequate conceptualization, detection, cognition, and reaction capabilities to detect the *tell tales* of risk, problem *symptoms*, and crisis *indicators*. In the best case, a risk is avoided in the secure stage by design (1); in the worst case, a risk transgresses into a problem and crisis undetected (3). One possible scenario is risk detection (a), cognition (b), mitigation (c), problem event due to residual risk, detection (d), cognition (e), and problem-solving (2). For each state, measures can be envisioned, prepared, and deployed. *Pre-emptive measures* may include risk acceptance, avoidance, mitigation, or transfer. Problem *countermeasures* involve solving, resolving, dissolving, or absolution. *Contingency measures* for dealing with crises are to endure, reduce, rebuild, or provide continuity. The *posture* is a coherent set of measures per stage and the decision to commit resources as a precaution. Eight posture combinations (2*2*2) exist, ranging from entirely passive (N,N,N) to fully proactive (Y,Y,Y).

5 Discussion

Models are a means to an end; the presented model can be adjusted for specific project domains, features, or situational factors. In theory, project uncertainty is a property of the observer; acquiring more information does not fundamentally change the system. Becoming aware of a risk or problem neither prevents nor solves it. This is true, provided the observation itself does not change the system. A fundamental issue with the conventional project risk management paradigm is the presumption that problems can be prevented during the risk stage. First, given the uncertain nature of projects, a scenario exists where undetected problems are present at project inception, implying that risk responses are no longer relevant as the horse has escaped the stable. Launching a ship with a hole effectively bypasses the risk stage. Second, any overlooked prerequisites, constraints, or dependencies will cause the plan to fail; no probability is involved. The project is past the risk stage; the question becomes when and how the problem will be discovered, if at all. Becoming aware is not a problem event occurring; the problem is there all the time. Discriminating between separate risk, problem, and crisis stages with individual measures prevents *false mitigation*: actions designed to solve the problem are perceived as risk mitigation. Carrying a spare tire does not reduce the probability of a flat or enable one to keep driving. False mitigation precludes a conscious decision about the desirability and economics to address the risk, solve the problem, or prepare for a crisis. A third issue is a lack of granularity; any source of trouble is attributed to risk. Unidentified stakeholders, deliverables or requirements, wrong decisions, and unsound methods are uniformly dubbed risks, like a modern-day evil spirit creating havoc. Business analysis, methodology, problem-solving, and decision-making are all separate fields supported by multiple sciences. From a systems thinking perspective, making a wrong decision can cause a problem or increase risk in a scenario, they are separate though related elements. Similar logic goes for threats and opportunities; causal loop diagrams help visualize these relationships. Uncertainty reduction implies maximizing the known-known area by raising information awareness, for example, using the Project Metadata Technique and Power Kanban; and increasing availability through investigation, prototyping, calculation, simulation, and forecasting. The generic project hypothesis can be supported using models such as Cynefin to assess the nature of the domain (Kurz/Snowden, 2003), the Four Phase model regarding the strategic contribution (Hardjono, 1995), or the level of uniqueness using the Diamond of Innovation (Shenhar/Dvir, 2007).

6 Conclusions

Project uncertainty can be generically defined by projecting the uncertainty matrix on a conceptual project system. Uncertainty management aims to maximize the known-known area by improving information availability and awareness while maximizing information effectiveness and efficiency. In addition, projects must acquire robust conceptualization, detection, cognition, and reaction capabilities to handle scenario-related uncertainty. The conventional risk responses alone (avoid, transfer, mitigate) are insufficient to secure project success; a conscious decision regarding preemptive and contingency measures is necessary to define a coherent posture given time and resource constraints. Uncertainty management is a crossroads of scientific fields, requiring a trans-disciplinary approach. Although founded in the project management realm, the presented uncertainty approach is universal and applicable in other domains.

References

Ackermann, F., 2023, “Systems Thinking for Project Management? Risky Not To.", *Journal of Systems Thinking*, Vol 3, pp. 3:1-13.

De Meyer A., Loch C.H. and Pich M.T., 2002, “From variation to chaos.", *MIT Sloan Management Review*, 43, pp. 60-67.

Hardjono, T., 2023, “Ritmiek en organisatiedynamiek : vierfasenmodel : met aangrijpingspunten voor organisatorische interventies ter vergroting van de effectiviteit, efficiency, flexibiliteit en creativiteit.", *Phd. Technische Universiteit Eindhoven. DOI: 10.6100/IR449363*

Hubbard D. and Evans D., 2010, “Problems with scoring methods and ordinal scales in risk assessment.", *IBM Research and development*, Vol 54 no 3, pp. 2-1 to 2-10.

Johansen A. et al, 2014, “Uncertainty Management, A Methodological Framework Beyond The Six Ws", *Procedia Social and Behavioral Sciences* , 119, pp. 566-57.

Kurz, F., Snowden, J., 2003, “The new dynamics of strategy: Sense-making in a complex and complicated world" (PDF), *IBM Systems Journal 42(3).*, pp. 462â€“483.

Migilinskas, D., Ustinovicius, L., 2008, “Methodology of risk and uncertainty management in constructions technological and economical problems", *The 25th International Symposium on Automation and Robotics in Construction Selected papers.*, pp. 789-794.

Perminova, O., Gustafsson M., and Wikstrom, K., 2008, “Defining uncertainty in projects, a new perspective.", *International Journal of Project Management*, 26, pp. 73-79.

Pich M.T., Loch C.H., and De Meyer A., 2002, “On uncertainty, ambiguity, and complexity in project management.", *Management Science*, 48, pp. 1008-1023.

Project Management Institute, 2017, “A Guide to the Project Management Body of Knowledge: PMBOK Guide.", *PMI*, version 6.

Raadgever G. T. et al, 2011, “Uncertainty management strategies Lessons from the regional implementation of the water framework directive in the Netherlands", *Environmental Science and Policy*, 14, pp. 64-75.

Shenhar, A, Dvir, D. 2007, “Reinventing Project Management: The Diamond Approach to Successful Growth and Innovation", *Boston: Harvard Business School Press.*

Ward, D. and Chapman, A., 2003, “Transforming project risk management into project uncertainty management.", *International Journal of Project Management*, 21, pp. 97-105.

Lock scheduling with non-identical parallel chambers

Morteza Davari[1], Mohammad Ranjbar[2] and Dirk Briskorn[3]

[1] Skema Business School, France
morteza.davari@skema.edu
[2] Ferdowsi University of Mashhad, Iran
[3] University of Wuppertal, Germany

Keywords: Lock scheduling, polynomial-time algorithm, time-indexed formulation.

1 Problem definition

In addition to road, rail, and air transport, rivers and waterways serve as natural infrastructures well-suited for freight transport. Despite being an age-old method, transporting goods over inland waterways proves reliable, cost-effective, and environmentally friendly within logistics and supply chains. According to the European Commission, the energy consumption for water transport is approximately 17% of that for road transport and 50% of rail transport. Eurostat reported in 2014 that waterways contribute to around 12.3% of Germany's total freight transport infrastructure, and this figure is approximately 6.6% for the European Union (EU). Navigating through key waterways such as the Panama Canal, the Kiel Canal, the Albert Canal, and others necessitates the use of locks due to variations in water levels at specific points.

A lockage or lock movement denotes a singular operation of a chamber, encompassing the entry of one or more ships into the chamber, the adjustment of water levels from downstream to upstream or vice versa, and the subsequent exit of the ships from the chamber. Note that ships approach from either the upstream or downstream side. The term *lockage time* corresponds to the duration required to execute this operation. The chamber's capacity signifies the maximum number of ships that can be concurrently accommodated during a single lockage. Locks are comprised of either a single chamber or multiple parallel chambers. For example, the Wijnegem lock, situated in Belgium, linking the Albert canal to the Antwerp port, comprises three non-identical parallel chambers.

Having many applications in real-world situations, the lock scheduling problem (LSP) in several variations has attracted far many concerns during the last years. In this context, several papers consider the LSP in a single-lock single chamber setting. Petersen and Taylor (1988) publish one of the early LSP papers where they consider the Welland Canal in North America. Nauss (2008) presents an integer programming model to determine the sequence of a queue at a lock. Moreover, Smith, Nauss, Mattfeld, Li, Ehmke, and Reindl (2011) consider a single chamber as a two-stage queue, combine optimization and simulation methods to analyze it, and develop a mixed integer programming (MIP) along with a heuristic solution approach. Hermans (2014) studies an LSP consisting of one chamber with unit capacity. Passchyn, Coene, Briskorn, Hurink, Spieksma, and Berghe (2016) describe a polynomial algorithm to minimize total waiting time for a bidirectional LSP including a single lock chamber.

Formally, we consider a single lock consisting of $M = \{1, \ldots, m\}$ non-identical parallel chambers. The chambers operate independently of each other, and each is characterized by two numbers: a one-dimensional length denoted by L_j that represents the length of the chamber and a lockage time (or simply a duration), denoted by d_j. Moreover, a set $N = \{1, \ldots, n\}$ of non-identical ships arrive on either side of the lock. Each ship $i \in N$ is characterized by an arrival time a_i, a length l_i, a waiting cost per unit time w_i and a position p_i, where $p_i = 0$ for a downstream arrival, and $p_i = 1$ for an upstream arrival. Also without loss of generality and only for convenience, we assume $a_1 \leq a_2 \leq \ldots \leq a_n$. We aim to assign ships to chambers and schedule each lockage of chambers such that the total waiting cost is minimized. The problem is denoted by LSTW. Apart from LSTW, we study the special case where ships are identical, i.e., $l_i = 1$ and $w_i = 1$. This special case is denoted by $\mathrm{LSTW} - \mathrm{IS}$.

For the ease of reference and in the remainder of this paper, we use the following notations. We divide set N into two disjoint subsets D and U where $D = \{i \in N | p_i = 0\}$ and $U = \{i \in N | p_i = 1\}$. Let U_t (respectively, D_t) be the set of ships that have already arrived upstream (respectively, downstream) before or at time t. Thus, U_t and D_t are computed as follows:

$$U_t = \{i \in U | a_i \leq t\} \text{ and } D_t = \{i \in D | a_i \leq t\}.$$

Note that ships in both U_t and D_t are sorted according to non-decreasing order of their arrival times. We define notations $\omega_k(U_t)$ (respectively, $\omega_k(D_t)$) which is the kth ship in U_t (respectively, D_t), and $\mathit{\Gamma}^U_t = |U_t|$ (respectively, $\mathit{\Gamma}^D_t = |D_t|$) that is the number of ships arrived upstream (respectively, downstream) at time t.

2 Complexity

Following a straightforward reduction from 3-partition, we conclude LSTW is strongly NP-hard. Also, the complexity of $\mathrm{LSTW} - \mathrm{IS}$ for an arbitrary number of chambers (m is part of the input) is strongly NP-hard since its no-wait counterpart is already known to be strongly NP-complete (Passchyn, Briskorn, and Spieksma, 2019). We provide a polynomial-time algorithm to solve $\mathrm{LSTW} - \mathrm{IS}$ when the number of chambers (m) is constant.

Theorem 1. *Assuming m being constant,* $\mathrm{LSTW} - \mathrm{IS}$ *is solvable in* $\mathrm{O}(n^{m+3})$ *time.*

Note that when $m = 1$, the $\mathrm{LSTW} - \mathrm{IS}$ is equivalent to the lockmaster's problem (Passchyn et al., 2016) with a bound on the number of ships in the lock. Interestingly, our algorithm for such a special case runs in $\mathrm{O}(n^4)$ which is computationally comparable with the algorithm introduced in Passchyn et al. (2016).

3 A time-indexed formulation

In this section, we propose mixed-integer linear programming formulations for both LSTW and $\mathrm{LSTW} - \mathrm{IS}$. We define variables $x^U_{jt}(x^D_{jt})$ which is one if chamber j loads ships (if any) and leaves upstream (downstream) at time t and variables y_{ijt} which is one if ship i is serviced by chamber j at time t and zero otherwise. The following time-indexed formulation solves LSTW.

$$\text{TIF} : \min \sum_{i\in N}\sum_{j\in M}\sum_{t=0}^{T} y_{ijt} w_i (t-a_i)^+$$

s.t.

$$\sum_{s=t}^{\min(T;t+d_j-1)} (x^U_{js} + x^D_{js}) \leq 1 \qquad \forall j \in M, t \in \{0,...,T\} \tag{1}$$

$$\sum_{s=t}^{\min(T;t+2d_j-1)} x^U_{js} \leq 1 \qquad \forall j \in M, t \in \{0,...,T\} \tag{2}$$

$$\sum_{s=t}^{\min(T;t+2d_j-1)} x^D_{js} \leq 1 \qquad \forall j \in M, t \in \{0,...,T\} \tag{3}$$

$$\sum_{j\in M}\sum_{t=0}^{T} y_{ijt} = 1 \qquad \forall i \in N \tag{4}$$

$$\sum_{i\in U} l_i y_{ijt} \leq L_j x^U_{jt} \qquad \forall j \in M, t \in \{0,...,T\} \tag{5}$$

$$\sum_{i\in D} l_i y_{ijt} \leq L_j x^D_{jt} \qquad \forall j \in M, t \in \{0,...,T\} \tag{6}$$

$$y_{ijt} \leq x^U_{jt} \qquad \forall i \in U, \forall j \in M, t \in \{0,...,T\} \tag{7}$$

$$y_{ijt} \leq x^D_{jt} \qquad \forall i \in D, \forall j \in M, t \in \{0,...,T\} \tag{8}$$

$$y_{ijt} = 0 \qquad \forall i \in N, \forall j \in M, t < a_i \tag{9}$$

$$x^U_{jt}, x^D_{jt} \in \{0,1\} \qquad \forall j \in M, t \in \{0,...,T\} \tag{10}$$

$$y_{ijt} \in \{0,1\} \qquad \forall i \in N, \forall j \in M, t \in \{0,...,T\} \tag{11}$$

For the above formulation, constraints (1) ensures that each chamber, while starting a movement, leaves either upstream or downstream and that once a movement of a chamber is started at time t, another movement of the same chamber (regardless of its direction) cannot be started before $t + d_j$. Constraints (2) (constraints (3)) guarantee that when a chamber movement starts from downstream (upstream) at time t, another movement of the same chamber from downstream (upstream) is not allowed before $t + 2d_j$. Constraints (4) enforces assignment of each ship to exactly one movement of a chamber. Constraints (5)-(6) satisfy the capacity requirement for each movement. And finally, constraints (7)-(8) enforce movements of chambers whenever there is at least one ship assigned to such movements.

4 Valid inequalities

We borrow the following property from Passchyn et al. (2019).

Property 1. Each chamber movement either directly follows upon a previous movement or starts upon some a_i while containing ship i.

Following the above property, any of the two TIF formulations introduced above, can be improved by including the following set of constraints:

$$x^U_{jt} \leq x^D_{j,t-d_j} \qquad \forall j \in M, t \in \{d_j, ..., T\} \text{ with } n^U_t = 0 \tag{12}$$

$$x^D_{jt} \leq x^U_{j,t-d_j} \qquad \forall j \in M, t \in \{d_j, ..., T\} \text{ with } n^D_t = 0 \tag{13}$$

$$x^U_{jt} \leq 0 \qquad \forall j \in M, t \in \{0, ..., d_j - 1\} \text{ with } n^U_t = 0 \tag{14}$$

$$x^D_{jt} \leq 0 \qquad \forall j \in M, t \in \{0, ..., d_j - 1\} \text{ with } n^D_t = 0 \tag{15}$$

where n^D_t is the number of ships that arrive downstream at time t, n^U_t is the number of ships that arrive upstream at time t. Based on the above-described property, when $n^D_t + n^U_t = 0$, any allowed chamber's movement at time t must coincide with the end of a previous movement, which is guaranteed by constraints (12)-(15).

5 Initial results

Preliminary findings indicate that the TIF formulation, incorporating additional inequalities, exhibits effectiveness in solving instances of LSTW and LSTW − IS involving up to 100 jobs. The absence of these additional inequalities significantly hampers the performance of TIF. In the case of LSTW − IS and for small m, an early implementation of the proposed polynomial algorithm demonstrates a slightly superior performance compared to TIF. For the case where $m = 1$, a dedicated iterative dynamic programming approach is performant. Additionally, early results suggest that an associated set-covering formulation may not yield computational advantages, while a Benders decomposition approach exhibits some potential.

Bibliography

Hermans, J. Optimization of inland shipping. *Journal of Scheduling*, 17(4):305–319, 2014.

Nauss, R. M. Optimal sequencing in the presence of setup times for tow/barge traffic through a river lock. *European Journal of Operational Research*, 187(3):1268–1281, 2008.

Passchyn, W., Coene, S., Briskorn, D., Hurink, J. L., Spieksma, F. C., and Berghe, G. V. The lockmasterâ€™s problem. *European journal of operational research*, 251(2):432–441, 2016.

Passchyn, W., Briskorn, D., and Spieksma, F. C. No-wait scheduling for locks. *INFORMS Journal on Computing*, 31(3):413–428, 2019.

Petersen, E. R. and Taylor, A. J. An optimal scheduling system for the welland canal. *Transportation science*, 22(3):173–185, 1988.

Smith, L. D., Nauss, R. M., Mattfeld, D. C., Li, J., Ehmke, J. F., and Reindl, M. Scheduling operations at system choke points with sequence-dependent delays and processing times. *Transportation Research Part E: Logistics and Transportation Review*, 47(5):669–680, 2011.

Modeling the process of animal food production as a hybrid flow shop scheduling problem

Ron Elias[1], Tal Grinshpoun[2], Hagai Ilani[1] and Elad Shufan[1]

[1] SCE – Shamoon College of Engineering, Israel
`ronel8@ac.sce.ac.il, hagai@sce.ac.il, elads@sce.ac.il`
[2] Ariel University, Israel
`talgr@ariel.ac.il`

Keywords: animal food production, hybrid flow shop, job families, lot streaming.

1 Introduction

Hybrid flow shop (HFS) represents a specialized manufacturing environment, diverging from traditional flow shops due to the presence of multiple parallel machines at one or more stages. Over the past decades, various HFS problems have been extensively studied, shedding light on their complexities and nuances across different industries. Applications of HFS have been used in various manufacturing processes, including electronics (Jin, Ohno, Ito and Elmaghraby 2002), paper (Sherali, Sarin and Kodialam 1990), textile (Grabowski and Pempera 2000), and concrete (Guinet 1991). Differently from the above domains, food production may involve unique restrictions regarding the order in which different products are produced, e.g., due to allergens. Recently, Mendoza-Mendoza, Ospino-Castro and Romero-Martínez (2021) modeled a food production problem as an HFS, but the process that they studied did not contain such ordering restrictions.

In this paper, we model an HFS problem found at Tadmir Mixture Institute, which is a large institute for the production of animal feed mixtures. The institute provides clients with three different types of mixtures depending on their need for medication: dirty, neutral, and clean. A *dirty* mixture includes drugs, such as antibiotics, whereas a *clean* mixture does not contain any drugs. A *neutral* mixture does not contain drugs in its recipe but may contain drug residues. Accordingly, a clean mixture cannot be produced after a dirty one without an intervening neutral mixture; this precaution is taken to prevent scenarios in which a clean mixture might contain drug residues, which could be harmful to some animals. For each mixture type, there is a wide range of recipes, denoted by mixture codes. In total, there are several hundreds of distinct mixture codes tailored for a variety of animals.

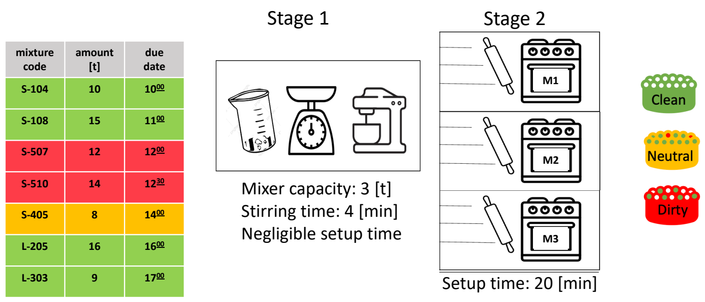
The production process includes other aspects, which are detailed in Section 2. The problem is formally modeled in Section 3. A discussion in Section 4 concludes the paper.

2 Description of the production process

The main production process includes two stages: mixing and steam pressing. In the first stage, raw materials of various types – liquid, flour, or grains – enter a mixer. The mixer blends the materials to achieve a uniform mixture. In the second stage, the mixture is transferred to a steam press, which forms dumplings of varying shapes and sizes tailored to the clients' specifications.

The first stage has a single mixer machine. The mixer can contain a maximal amount of three tonnes of raw material; following Wang, Zhao, Gao and Sutherland (2019), we denote this by $SZ_{\max}{=}3\,\mathrm{t}$. A fixed processing time ($p_1{=}4\,\mathrm{min}$) is required to complete the mixing, even when the mixer is not full. Therefore, the processing time of a job j, for which an amount SZ_j of raw material is needed, is given by $\lceil \frac{SZ_j}{SZ_{\max}} \rceil \cdot p_1$. The setup times for operations of the first stage are negligible. The second stage consists of three identical steam-press machines that work in parallel. There is a fixed setup time (20 min) for preparing any second-stage machine between jobs of different mixture codes.

An illustration of the production process is given in Figure 1. The institute receives job orders from various clients. Each job order specifies the required mixture code, amount, and due date. Recall that each mixture code is either dirty, neutral, or clean; a clean mixture may be produced after a clean or neutral mixture, but not after a dirty one. The objective is to complete the processing of all jobs with a minimal number of tardy jobs. If it is possible to complete all the jobs within their deadlines, the objective is to complete the overall production process as soon as possible, i.e., minimize the makespan.



mixture code	amount [t]	due date
S-104	10	$10^{\underline{00}}$
S-108	15	$11^{\underline{00}}$
S-507	12	$12^{\underline{00}}$
S-510	14	$12^{\underline{30}}$
S-405	8	$14^{\underline{00}}$
L-205	16	$16^{\underline{00}}$
L-303	9	$17^{\underline{00}}$

Fig. 1. Illustration of the production process.

Another aspect of the production scheduling problem regards concurrency. As noted, the mixer can handle up to three tonnes at a time. Therefore, large-capacity jobs (lots) are divided into *sublots* of up to three tonnes each. When a sublot leaves the mixer, it can enter one of the steam press machines (when a machine is available). Contemporaneously, the next sublot of that job can start processing in the mixer. The division into sublots yields an interesting property in which a job may be concurrently processed in the two stages. Nevertheless, all sublots of a job must be sequentially processed on the same machine in stage 2. Figure 2 presents a Gantt chart demonstrating the concurrency.

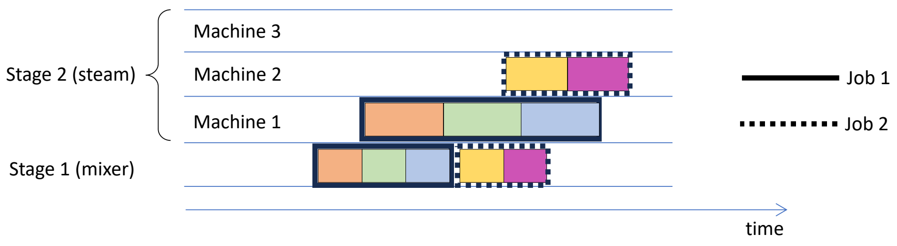


Fig. 2. An example of a Gantt chart showing the division of a job into sublots (different sublots are shown by different colors).

3 Formal modeling of the problem

We model the problem at focus using the $\alpha|\beta|\gamma$ notation (Graham, Lawler, Lenstra and Kan 1979):

$$F(1,3)\,|\,fmls, s_{lgh}, split, NI, SZ_{\max}, p_{1j} = \lceil \frac{SZ_j}{SZ_{\max}} \rceil \cdot p_1\,|\,Lex(\sum U_j, C_{\max}) \tag{1}$$

The first field, α, characterizes the machine environment. In this problem, the environment is a hybrid flow shop with two stages: a single machine at the first stage and three parallel identical machines in the second stage. We follow the notation of Emmons and Vairaktarakis (2013) for hybrid flow shops. The stages are denoted by $l = \{1, 2\}$.

The second field, β, delineates the characteristics of the jobs, i.e., the constraints stemming from different mixture codes and the division into sublots. The mixture codes and types dictate the setup times. Two consecutive jobs of different mixture codes require a non-negligible, yet constant, setup time in the second stage. Additionally, a clean job cannot succeed a dirty job; this constraint on mixture types can also be represented in the form of setup times by applying infinity as the setup time between all mixture codes of dirty type and all mixture codes of clean type. Consequently, every mixture code can be referred to as a job *family*, with sequence-dependent setup times between families (constant or infinity, depending on the mixture types). This is denoted in the literature as $fmls, s_{gh}$ (Pinedo 2022). Since, in this problem, setup times differ in the two stages (negligible vs. 20 min, respectively, for valid consecutive mixtures), we revise the notation to s_{lgh} to account for the stage dependency.

The term *lot streaming*, which refers to splitting a lot into sublots to enable their parallel production, was first coined by Reiter (1966). Most studies on lot streaming use a designated notation system rather than the standard $\alpha|\beta|\gamma$ to express the variety of problem features (Chang and Chiu 2005, Cheng, Mukherjee and Sarin 2013). Nonetheless, *split* has been previously used for denoting lot streaming in the $\alpha|\beta|\gamma$ notation (Sethanan, Wisittipanich, Wisittipanit, Nitisiri and Moonsri 2019). Some of the specific lot-streaming characteristics are relevant to our problem: *consistent* sublots are such that their sizes remain the same over all the machines, and *no intermittent idling*, denoted *NI* (Cheng et al. 2013), means that all sublots of the same lot must be processed sequentially on the same machine. Note that the consistency characteristic is redundant when no intermittent idling is applied on a two-stage flow shop.

There is another lot-related characteristic to consider that is absent from the lot-streaming literature: the maximal bound on the size of a sublot (3 t). This sublot-size constraint is unique to our problem because the tendency in lot-streaming scenarios is to use small-sized sublots to enhance the advantages of parallelism. However, the capacity bound of the mixer requires consideration of the constraint. Although we are not aware of existing literature that explicitly models a maximal bound on the size of a sublot, we adopt for this purpose the notation $SZ_{\max}$ of Wang et al. (2019) who use it as a parameter in their model. Additionally, the processing time in the mixer of each sublot is constant (p_1=4 min). Therefore, as explained in Section 2, the processing time in stage 1 of a job j is given by $p_{1j} = \lceil \frac{SZ_j}{SZ_{\max}} \rceil \cdot p_1$.

The third field, γ, represents the objective function. The problem herein comprises two objectives: the primary objective is to minimize the number of tardy jobs, denoted ΣU_j, whereas a secondary objective is to minimize the makespan, denoted C_{max}. T'kindt and Billaut (2006) studied various types of multicriteria objectives in scheduling. They generalized the concept of primary/secondary objectives and termed such scenarios as having a *lexicographical* order between the criteria, denoted *Lex* and followed by the ordered list of criteria.

4 Discussion

We have presented the problem of animal food production at Tadmir Mixture Institute as an HFS, which is known to be NP-hard even for two stages (Emmons and Vairaktarakis 2013). Our problem differs from standard HFS due to additional aspects, particularly by applying the combination of setup times and division into sublots. Existing studies on HFS, with or without setup times, usually employ evolved meta-heuristics (Allahverdi, Ng, Cheng and Kovalyov 2008). But in our case, the simple structure of constant family-derived setup times and practically constant sublot size suggest the direction of developing dedicated heuristics. As a preliminary idea, we advised the institute to unite job orders from clients with the same mixture code and close deadlines. The job union leverages two important characteristics of the problem – the constant processing time of the mixer and the relatively high setup time of steam pressers. Implementing this simple idea proved highly effective compared to the institute's scheduling routine. The actual production process includes additional details that we omitted for clarity. For example, mixtures containing only flour ingredients do not require the steam pressing stage. However, such jobs can be represented in our model as having zero setup and processing times in stage 2. The provided modeling of this unique problem is a first step toward the development of a full optimization solution.

References

Allahverdi, A., Ng, C. T., Cheng, T. E. and Kovalyov, M. Y.: 2008, A survey of scheduling problems with setup times or costs, *European journal of operational research* **187**(3), 985–1032.

Chang, J. H. and Chiu, H. N.: 2005, A comprehensive review of lot streaming, *International Journal of Production Research* **43**(8), 1515–1536.

Cheng, M., Mukherjee, N. and Sarin, S.: 2013, A review of lot streaming, *International Journal of Production Research* **51**(23-24), 7023–7046.

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

Grabowski, J. and Pempera, J.: 2000, Sequencing of jobs in some production system, *European Journal of Operational Research* **125**(3), 535–550.

Graham, R. L., Lawler, E. L., Lenstra, J. K. and Kan, A. R.: 1979, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, Vol. 5, Elsevier, pp. 287–326.

Guinet, A.: 1991, Textile production systems: a succession of non-identical parallel processor shops, *Journal of the Operational Research Society* **42**(8), 655–671.

Jin, Z., Ohno, K., Ito, T. and Elmaghraby, S. E.: 2002, Scheduling hybrid flowshops in printed circuit board assembly lines, *Production and Operations Management* **11**(2), 216–230.

Mendoza-Mendoza, A., Ospino-Castro, W. and Romero-Martínez, D.: 2021, Production scheduling in a flexible hybrid flow shop in the food industry based on the theory of constraints, *International Journal of Engineering Research in Africa* **52**, 124–136.

Pinedo, M. L.: 2022, *Scheduling: Theory, Algorithms, and Systems*, sixth edn, Springer Nature.

Reiter, S.: 1966, A system for managing job-shop production, *The Journal of Business* **39**(3), 371–393.

Sethanan, K., Wisittipanich, W., Wisittipanit, N., Nitisiri, K. and Moonsri, K.: 2019, Integrating scheduling with optimal sublot for parallel machine with job splitting and dependent setup times, *Computers & Industrial Engineering* **137**, 106095.

Sherali, H. D., Sarin, S. C. and Kodialam, M. S.: 1990, Models and algorithms for a two-stage production process, *Production Planning & Control* **1**(1), 27–39.

T'kindt, V. and Billaut, J.-C.: 2006, *Multicriteria Scheduling: Theory, Models and Algorithms*, second edn, Springer Science & Business Media.

Wang, H.-y., Zhao, F., Gao, H.-m. and Sutherland, J. W.: 2019, A three-stage method with efficient calculation for lot streaming flow-shop scheduling, *Frontiers of Information Technology & Electronic Engineering* **20**(7), 1002–1020.

Makespan service level for the flexible job shop scheduling problem under machine-related uncertainty

Mario Flores-Gómez[1], Stéphane Dauzère-Pérès[1,2]

[1] Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, Gardanne, France
`{mario.flores, 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 processed in a specific order (routing) on a set of available machines. A machine can only perform one operation at a time, and operations cannot be interrupted once started. 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 means determining both an assignment of operations to machines, and the sequence the operations on the machines, while respecting the routing of every job.

The Stochastic FJSP (SFJSP), where the processing times of operations in the machines are stochastic, 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 (Flores-Gómez *et. al.* 2021) and (Flores-Gómez *et. al.* 2023), we pursue the study on the relevance of the notion of *makespan service level* as defined in (Dauzère-Pérès *et. al.* 2008) but, instead of considering job-related processing time uncertainty, as in the aforementioned publications, we focus on the case where the processing times of all operations that can be processed by one machine are stochastic, i.e. the processing time uncertainty is machine related. The notion of *makespan service level* and our solution approach are recalled in Section 2. Numerical results are presented and discussed in Sections 3 and 4. Conclusions and perspectives are given in Section 5.

2 Problem statement and solution approach

As processing times are random variables, 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. Let us recall the notion of *makespan service level*:

$$\alpha(S,T) = \mathbb{P}(C_{max}(S,\xi) \leq T),$$

where ξ is a multivariate random variable of dimension n.

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 (Flores-Gómez *et. al.* 2023). 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 (Flores-Gómez *et. al.* 2023). Note that minimizing the makespan does not mean maximizing the service level.

3 Relevance of *makespan service level*

In (Flores-Gómez *et. al.* 2023), the case where the variability of the processing time of an operation is related to the job of the operation, is taken into account. However, for several reasons, the uncertainty of the processing times can be related to the machines the operations are assigned to. The level of machine flexibility (i.e. the number of eligible machines to which an operation can be assigned) has more impact since the machines that are more prone to variability should typically not be used for the operations with the largest processing times. To illustrate this comment, we generated instances where the processing times of all operations that can be processed by one machine are stochastic. The set of scenarios Ω is randomly generated for every machine $k \in \mathcal{M}$ according to the beta probability distribution (Marshall and Olkin 2007), by extending the FJSP benchmark instances from Hurink *et. al.* (1994). In this abstract, $|\Omega|$ is set to 5,000. 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_{init} be the sequence found by the tabu search in Dauzère-Pérès and Paulli (1997) for mean values of the random processing times.

Table 1. Characteristics of instances mt06, mt10 and mt20 with three lines per instance type (edata, rdata, vdata) for each level of machine flexibility.

Instance	$\|\mathcal{M}\|$	$\|\mathcal{J}\|$	Operations per job	Average $\|\mathcal{M}_i\|$	$C_{max}(S_{init})$
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

To solve the SFJSP, the tabu search in (Flores-Gómez *et. al.* 2023) is applied to the instances described in Table 1 using 500 scenarios for each run. Let S^* be the sequence found by the tabu search. Different results might be obtained depending on the representativeness of the set of selected scenarios. To assess the online evaluation of the makespan service level, 10 runs of the tabu search are conducted using 10 mutually-exclusive subsets $\Omega_{500}^{b} \subset \Omega$, $b \in \{1, \ldots, 10\}$. Each subset Ω_{500}^{b} represents the b^{th} batch of 500 scenarios in Ω. Since $|\Omega| = 5,000$, there are 10 such subsets in Ω.

4 On the improvement gap

Let us now focus on the improvement gap. A first general remark is that the larger $\frac{1}{|\mathcal{M}|}\sum_{k\in\mathcal{M}}[\alpha(S^*,T,\Omega)-\alpha(S_{init},T,\Omega)]$, the more effective the proposed solution approach. A large improvement of the service level indicates that, using the mean values of the processing times to find a sequence minimizing the makespan is not as effective for the SFJSP as maximizing the makespan service level. The results in Column "$\frac{100\%}{|\mathcal{M}|}\sum_{k\in\mathcal{M}}[\alpha(S^*,T,\Omega)-\alpha(S_{init},T,\Omega)]$" of Table 2 show the improvements for every batch of scenarios and every instance. Column "$\min_{\mathcal{M}}, b$" (resp. "$\max_{\mathcal{M}}, b$") presents the smallest (resp. largest) improvement $\forall b \in \{1,\ldots,10\}$ and the associated argument (batch b) per machine and instance. If the minimum improvement gap is equal to 0, batch b in Column "$\min_{\mathcal{M}}, b$" corresponds to the first batch detected. The improvement gap is very consistent on average regardless of the subset Ω^b_{500} as shown in Figure 1 and regardless of the level of machine flexibility, where $\alpha(S^*_{\Omega^b_{500}},T,\Omega)-\alpha(S_{init},T,\Omega)$ is plotted for instances $mt10$-$edata$ and $mt10$-$vdata$ $\forall b, \forall k \in \mathcal{M}$. The improvement is between 0% and 15% for $mt10-edata$, and between 0% and 22% for $mt10-vdata$. These improvements are larger than the ones presented in our previous publications (Flores-Gómez *et. al.* 2021, Flores-Gómez *et. al.* 2023). Please note that a small increase of the mean improvement does not necessarily mean that the solution approach is ineffective, but rather that $\alpha(S_{init},T,\Omega)$ is on average already large.

Table 2. Improvement gap between initial and optimized sequences based on subsets $\Omega^b_{500} \subset \Omega$, $b \in \{1,\ldots,10\}$ for instances from (Hurink *et. al.* 1994). One row per level of machine flexibility (edata, rdata and vdata). $\alpha(S) = \alpha(S,T,\Omega)$.

Inst.	$\frac{100\%}{\lvert\mathcal{M}\rvert}\sum_{\mathcal{M}}[\alpha(S^*_{\Omega^n_{500}},T,\Omega)-\alpha(S_{init},T,\Omega)]$										$\alpha(S^*_{\Omega^b_{500}})-\alpha(S_{init})$ (%)	
	$b=1$	$b=2$	$b=3$	$b=4$	$b=5$	$b=6$	$b=7$	$b=8$	$b=9$	$b=10$	$\min_{\mathcal{M}}, b$	$\max_{\mathcal{M}}, b$
	13.1	12.46	13.6	12.9	13.5	13.16	13.36	14.1	13.74	14.92	3.4 , 3	54.0 , 10
mt06	6.92	8.66	7.76	8.84	6.7	8.56	7.14	7.04	7.48	6.66	0.0 , 1	51.4 , 9
	10.62	10.52	10.56	10.54	10.82	10.34	10.08	10.32	10.58	10.56	0.0 , 1	53.6 , 4
	3.58	3.92	3.98	4.06	4.34	4.34	4.14	3.2	4.0	4.56	0.0 , 1	15.6 , 1
mt10	7.82	9.08	8.54	8.26	9.72	8.04	8.68	7.1	8.14	9.12	1.8 , 8	17.4 , 5
	2.28	1.98	2.7	4.42	3.42	4.04	2.0	2.88	2.88	3.88	0.0 , 1	21.6 , 6
	0.44	0.48	0.74	0.7	0.56	0.56	0.64	0.68	0.88	0.56	0.0 , 1	3.8 , 9
mt20	0.36	0.46	0.02	0.06	0.3	0.08	0.24	0.36	0.3	0.0	0.0 , 1	4.0 , 2
	3.16	3.32	3.0	3.04	2.96	2.92	3.26	3.04	3.24	3.24	0.0 , 1	19.8 , 2

5 Conclusions

In this abstract, in continuation of our work in (Flores-Gómez *et. al.* 2021) and (Flores-Gómez *et. al.* 2023), we extend the study on the relevance of the notion of *makespan service level* when the processing time uncertainty is machine related. In our previous work, the considered uncertainty was exclusively job related. The numerical results on a reduced set of instances are presented, further motivating the relevance of the *makespan service level.* Additional computational results obtained using other service levels on regular scheduling criteria such as the sum of the weighted tardiness will be presented in the conference. Scenarios obtained from historical data from an industrial partner will also be used to

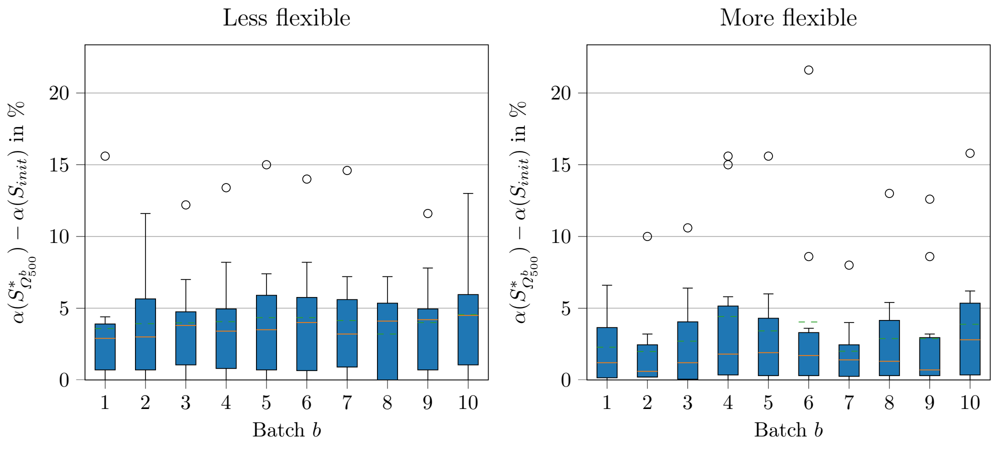


Fig. 1. Box-plot associated with the estimation error (in %) for subset $\Omega^b_{500} \subset \Omega$, $\forall b \in \{1, \ldots, 10\}$ for instances $mt10$-*edata* and $mt10$-*vdata* from (Hurink *et. al.* 1994). One point per machine. $\alpha(S, \Omega) = \alpha(S, T, \Omega), T = \{923, 653\}$.

conduct new computational experiments, and to further motivate the relevance of service level criteria in scheduling.

Acknowledgments (Final heading style)

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.

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.

Flores-Gómez M., V. Borodin and S. Dauzère-Pérès, 2023, "Maximizing the service level on the makespan in the stochastic flexible job-shop scheduling problem", *Computers & Operations Research*, Vol. 157.

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.

Improved Benders Decomposition for Project Scheduling with Monotone Objectives

Leonie Gallois[1 2], Jannik Matuschke[2] and Morteza Davari[1]

[1] Skema Business School, France
`morteza.davari@skema.edu`
[2] KU Leuven, Belgium
`leonie.gallois@kuleuven.be, jannik.matuschke@kuleuven.be`

Keywords: project scheduling, min-cost flow, Benders decomposition, shortest path.

1 Introduction

We have a project characterized by precedence constraints, denoted by $\mathcal{G}(\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ is the set of jobs and $\mathcal{A}$ is the set of precedence arcs. A subset of these jobs, denoted by $\mathcal{N}^R$, need a resource for processing. At any point in time, we can process an infinite number of regular jobs but only one resource job. We consider a discrete time horizon T. The processing time of job i is denoted by p_i. Each job i imposes a cost depending on its completion time t that we denote by c_{it}. We want to minimize the sum of these costs over all the jobs. Note that this generic cost function generalizes well-known scheduling objectives such as *makespan* and *weighted total completion time*. Already with the last objective function, the problem is strongly NP-hard and this can be shown by a reduction from the 3-partition problem.

We apply a Benders decomposition to solve this problem. We show that under the assumption that c_{it} is non-decreasing in t for each fixed i, the corresponding sub-problem can be solved by a single run of Dijkstra's shortest path algorithm in a network with $|\mathcal{N}| \cdot T$ nodes (note that when the costs are given explicitly in the input, this is polynomial in the input size). Previous studies in the literature have addressed similar problems without considering a resource. Such problems required the solution of a more involved maximum flow computation with arc-flow lower bounds as seen in Möhring (1984).

2 The dedicated algorithm

We can decompose the problem as follows: one can first define the order in which the resource jobs are processed and then schedule all jobs considering the additional precedence constraints imposed by the order of the resource jobs. This allows us to devise a Benders decomposition approach to solve the problem. We introduce variables $x_{ij} \in \{0, 1\}$ which indicate the order between two resource jobs i and j and *step* variables $y_{it} \in \{0, 1\}$ which determine if job i finishes at time t as explained by Artigues (2017).

The master problem is thus given as follows:

$$
\begin{aligned}
\min\quad & \phi && \forall i \in \mathcal{N}\\
\text{s.t.}\quad & x_{ij} + x_{ji} = 1 && \forall i, j \in \mathcal{N}^R\\
& x \text{ and } \phi \text{ fulfil optimality and feasibility cuts} && \\
& x_{ij} \in \{0, 1\} && \forall i, j \in \mathcal{N}^R
\end{aligned}
$$

The cuts are generated by solving the dual of the following sub-problem for a given solution $\hat{\mathbf{x}}$ to the master problem. The primal sub-problem is stated as follows:

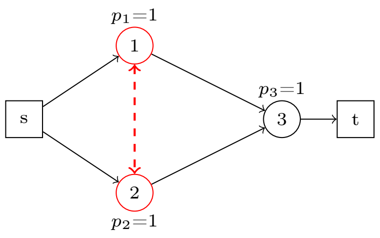


Fig. 1: A precedence graph with disjunctive arcs between two resource jobs, representing the possible choices for the additional precedence constraints

$$
\begin{array}{lll}
\phi = \min & \sum_{i\in\mathcal{N}} \sum_{t=0}^{T} c'_{it}\, y_{it} & \\
\text{s.t.} & y_{iT} = 1 & \forall i \in \mathcal{N} \\
& y_{it+1} - y_{it} \geq 0 & \forall i \in \mathcal{N}, \quad t \in [T-1] \\
& y_{it-p_j} - y_{jt} \geq -\alpha^{\hat{\mathbf{x}}}_{ij} & \forall i,j \in \mathcal{N}, \quad i \neq j, \quad t \in\ ]p_j, \ldots, T] \\
& y_{ip_i-1} = 0 & \forall i \in \mathcal{N} \\
& y_{it} \geq 0 & \forall i \in \mathcal{N}, \quad t \in [T]
\end{array}
$$

Where, for notational convenience, we define $c'_{it} = c_{it} - c_{it+1}$ with $c_{iT+1} = 0$ and $\alpha^{\hat{\mathbf{x}}}_{ij}$ as

$$
\alpha^{\hat{\mathbf{x}}}_{ij} := \begin{cases} 0 & \text{if } (i,j) \in \mathcal{A}, \\ 1 - \hat{x}_{i,j} & \text{if } i,j \in \mathcal{N}^R, \\ 1 & \text{otherwise} \end{cases}
$$

To obtain a cut, one can solve the dual of the sub-problem, which is given as follows.

$$
\begin{array}{lll}
\max & \sum_{i\in\mathcal{N}} \gamma_i - \sum_{i,j\in\mathcal{N}, i\neq j} \sum_{s=p_j}^{T} \alpha^{\hat{\mathbf{x}}}_{ij}\, v_{ijs} & \\
\text{s.t.} & \theta_i + u_{is-1} - u_{is} & \\
& \quad + \displaystyle\sum_{j\in\mathcal{N}\setminus\{i\}, s+p_j\leq T} v_{ijs+p_j} - \sum_{j\in\mathcal{N}, j\neq i} v_{jis} \leq c'_{it} & \forall i \in \mathcal{N}, \quad s = p_i - 1 \\
& u_{is-1} - u_{is} & \\
& \quad + \displaystyle\sum_{j\in\mathcal{N}\setminus\{i\}, s+p_j\leq T} v_{ijs+p_j} - \sum_{j\in\mathcal{N}, j\neq i} v_{jis} \leq c'_{it} & \forall i \in \mathcal{N}, \quad s \in [T-1] \setminus \{p_i - 1\} \\
& \gamma_i + u_{is-1} - u_{is} & \\
& \quad + \displaystyle\sum_{j\in\mathcal{N}\setminus\{i\}, s+p_j\leq T} v_{ijs+p_j} - \sum_{j\in\mathcal{N}, j\neq i} v_{jis} \leq c'_{it} & \forall i \in \mathcal{N}, \quad s = T \\
& u_{is}, v_{ijs} \geq 0 & \forall i \in \mathcal{N}, \quad s \in [T]
\end{array}
$$

Note that the dual is equivalent to an uncapacitated min-cost flow problem in a graph with nodes (i,s) for $i \in \mathcal{N}$ and $s \in [T]$. For a non-decreasing cost function, we derive $c'_{it} \leq 0$ for all $i \in \mathcal{N}$ and all $t < T$, which indicates a demand, and $c'_{iT} \geq 0$ for all $i \in \mathcal{N}$, which indicates a supply. In this flow problem, demands may be oversaturated and supplies may be underutilized.

Based on the precedence graph of Figure 1, we construct the flow graph of the dual of the sub-problem in Figure 2. The flow graph has a source denoted by s, which via the arcs

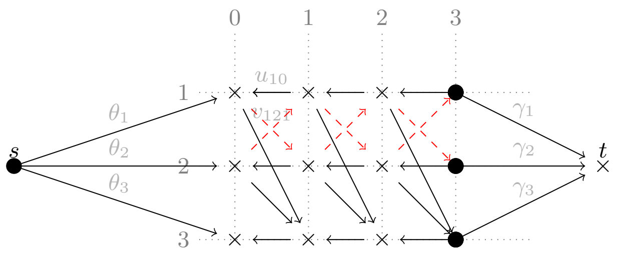


Fig. 2: Related flow graph of the dual where node (i, t) has a supply/demand of c'_{it}, the crosses stand for sinks and the dots for sources

representing variables θ can provide infinite free supply to nodes $(i, p_i - 1)$. The variables γ are represented by the arcs going to the sink t with a weight of -1 per unit of flow. The diagonal arcs have a weight of $\alpha^{\hat{\mathbf{x}}}$ per unit of flow.

Lemma 1. *An optimal solution to the dual sub-problem can be computed in time $\mathcal{O}(n \log n + m)$, where n and m are the number of nodes and arcs in the graph induced by the master problem solution $\hat{\mathbf{x}}$.*

Proof (Proof sketch). There are two possibilities to satisfy the demand of a node (i, t) for $t < T$: either from s or from (i, T). Note that, every unit of supply of (i, T) we do not use in this way decreases the objective of the min-cost flow problem by 1 by sending it to t. Hence, an optimal solution to the sub-problem can be computed as follows. For every node, we compute a shortest path from s with respect to arc lenghts $\alpha^{\hat{\mathbf{x}}}$. If the shortest-path distance of a node (i, t) is less than 1, we satisfy its demand by sending flow along the shortest path from s; otherwise, we satisfy the demand by sending flow from (i, T) via the arcs corresponding to variables u_{it}. Note that that these shortest paths can be computed for all nodes by one run of Dijkstra's algorithm.

References

Artigues C., 2017, “On the strength of time-indexed formulations for the resource-constrained project scheduling problem”, *Operations Research Letters*, Vol. 45, pp. 154-159.

Möhring R., 1984, “Minimizing costs of resource requirements in project networks subject to a fixed completion time”, *Operations Research Letters*, Vol. 32.1, pp. 89-120.

A faster FPTAS for makespan minimization with time-dependent agreeable V-shaped processing times

Nir Halman[1] and Helmut A. Sedding[2]

[1] Bar-Ilan University Ramat-Gan, Israel
`nir.halman@biu.ac.il`
[2] ZHAW, Institute of Data Analysis and Process Design, Switzerland
`helmut.sedding@zhaw.ch`

Keywords: time-dependent scheduling, FPTAS, K-approximation sets and functions, monotone dynamic programming.

1 Introduction

Scheduling jobs with time-dependent V-shaped processing times on a single machine minimizing makespan $C_{\max}$ is an NP-hard scheduling problem even if all jobs have the same slopes, but it permits a fully polynomial time approximation scheme (FPTAS), even for job-dependent slopes if they are agreeable (Sedding 2020*a*). We improve the FPTAS runtime by factor $1/\varepsilon$ (up to log terms) by giving an alternative dynamic programming (DP) formulation and employing the recent advances in Alon & Halman (2021) that let us apply the technique of K-approximation sets and functions.

Job deterioration is a common theme in time-dependent scheduling (Gawiejnowicz 2020, Sedding 2020*a*): The processing time is linearly dependent on the job's start time t, defined by function $p_j(t) = \ell_j + b_j t$ for basic processing time $\ell_j \geq 0$ and slope $b_j \geq 0$. Minimization of $C_{\max}$ requires to order the jobs by ℓ_j/b_j nondecreasingly (jobs with $b_j = 0$ last).

We study the extension to shortening as well as deteriorating processing times in a V-shape (Sedding 2020*a*), which model walking time for assembly operations, from a moving assembly line to a statically positioned supply at the line side. Constant times can only upper bound that, and are 51% higher in an extended case in Sedding (2023).

An instance specifies rational-valued $\ell_j \geq 0$, $b_j \geq 0$, a shortening slope $0 \leq a_j \leq 1$, and a common ideal start time τ. Then, a job's processing time is defined by

$$p_j(t) = \ell_j + \max\{-a_j\,(t-\tau),\, b_j\,(t-\tau)\}. \tag{1}$$

A solution is then structured into three parts (in that order): a first sequence S_1 that completes strictly before τ, a straddler job χ that starts no later than at τ and completes not earlier than at τ, and a second sequence S_2. Interestingly, the straddler job is not necessarily the shortest job. Within sequence S_2, jobs are sorted nondecreasingly by ℓ_j/b_j; while in S_1, they are sorted nonincreasingly by ℓ_j/a_j. Hence, solving an instance involves to select a straddler job, and a partition of the other jobs into the two sorted sequences. This problem is NP-hard already for common slopes ($a_j = a$ and $b_j = b$), which is shown in Sedding (2020*a*), as well as for the all-zero a_j case (Kononov 1997, Kubiak & van de Velde 1998), and the for all-zero b_j case (Cheng, Ding, Kovalyov, Bachman & Janiak 2003).

Let us only consider the special case where the basic processing times and slopes have *agreeable ratios*: $\ell_i/a_i \leq \ell_j/a_j \iff \ell_i/b_i \leq \ell_j/b_j$ for any pair of two jobs i, j. This case still permits an FPTAS, which is shown in Sedding (2020*a*). We give a faster FPTAS based on the technique of K-approximation sets and functions, introduced in Halman, Klabjan, Mostagir, Orlin & Simchi-Levi (2009), by reformulating the problem as a certain *monotone dynamic program* (DP) that falls into the FPTAS framework of Alon & Halman (2021).

Employing it allows us to omit an explicit algorithm statement, and directly conclude running time and approximation error. The resulting FPTAS's runtime dependency on n is in $\mathcal{O}(n^6)$, and is linear in $1/\varepsilon$ up to log terms, i.e., faster by a factor of $1/\varepsilon$ up to log terms than the tailor-made FPTAS in Sedding (2020*a*). As the considered problem includes special cases like all-zero a_j slopes, or all-zero b_j slopes, specialized FPTASes (Cai, Cai & Zhu 1998, Halman 2020, Kovalyov & Kubiak 1998, Kovalyov & Kubiak 2012, Ji & Cheng 2007, Sedding 2020*b*) can be substituted with our approach.

2 Dynamic Program

Based on the DP in Sedding (2020*a*), we introduce an alternative formulation as a certain monotone DP. First of all, reindex the jobs such that the straddler job χ is job $n+1$. Let state x of the n-stages dynamic program denote the (exact) completion time of the sequence of jobs that are executed before time τ, assuming that it can start being processed as early as time 0 (to be called first sequence S_1^j, containing jobs from set $\{1,\dots,j\}$). Because the straddler job starts by τ, the range of the state space for x is the rational valued interval $[0,\tau]$. For every stage $j=1,\dots,n$, we define two functions of the state x.

The first function, denoted by $z_j(x)$, refers to a second sequence $S_2^j(x)$ scheduled to start exactly at τ that minimizes the objective for all jobs $1,\dots,j$, and contains only the jobs $\{1,\dots,j\}\setminus S_1^j$. The value of $z_j(x)$ is explicitly not the objective value, but rather the makespan of $S_2^j(x)$ from τ to the completion time of the last job in the sequence. This makespan can be derived from Sedding (2020*a*, Eq. (6)) and, given state x and defining $F(i,S_2^j(x))\subset S_2^j(x)$ as the set of jobs that follow job i in $S_2^j(x)$, is equivalent to

$$z_j(x)=\textstyle\sum_{i\in S_2^j(x)}\Big(\ell_i\cdot\prod_{f\in F(i,S_2^j(x))}(1+b_f)\Big). \tag{2}$$

After stage n, the straddler job χ is appended to the end of the first sequence S_1^n, then the second sequence S_2^n follows, and the resulting completion time is the objective value. To calculate it, we use a second function, denoted by $y_j(x)$, that describes the proportional increase of sequence $S_2^j(x)$'s makespan $z_j(x)$ if increasing its start time, when having the jobs in $S_1^j(x)$ and $S_2^j(x)$ as determined by the state variable x (see below).

Algorithm 1 (DP). The alternative dynamic programming algorithm's steps are as follows.

1. Initialize functions $z_0(\cdot)\equiv 0$ and $y_0(\cdot)\equiv 1$ over their entire domain $[0,\tau]$.
2. For all j from 1 to n:
   (a) To append job j to the end of S_1, define

$$z'(x)=z_{j-1}\Big(\tfrac{x-\ell_j-a_j\tau}{1-a_j}\Big),\qquad \ell_j+a_j\tau\le x\le\tau. \tag{3a}$$

   To prepend job j to the beginning of S_2, define

$$z''(x)=\ell_j\cdot y_{j-1}(x)+z_{j-1}(x),\qquad 0\le x\le\tau. \tag{3b}$$

   (b) For S_2's makespan, define

$$z_j(x)=\begin{cases}z''(x), & \text{if } 0\le x<\ell_j+a_j\tau,\\ \min\{z'(x),z''(x)\}, & \text{if } \ell_j+a_j\tau\le x\le\tau,\end{cases}\qquad 0\le x\le\tau. \tag{3c}$$

   (c) For S_2's start-time-dependent makespan increase, define, for $0\le x\le\tau$,

$$y_j(x)=\begin{cases}(1+b_j)\cdot y_{j-1}(x), & \text{if } 0\le x<\ell_j+a_j\tau,\\ (1+b_j)\cdot y_{j-1}(x), & \text{if } \ell_j+a_j\tau\le x\le\tau \text{ and } z_j(x)=z''(x),\\ y_{j-1}\Big(\tfrac{x-\ell_j-a_j\tau}{1-a_j}\Big), & \text{if } \ell_j+a_j\tau\le x\le\tau \text{ and } z_j(x)=z'(x).\end{cases} \tag{3d}$$

3. Return completion time

$$C^{\chi}_{\max} = \inf_{0 \le x \le \tau} \{\tau + y_n(x) \cdot \max\{x + p_\chi(x) - \tau,\, 0\} + z_n(x)\}. \tag{3e}$$

Let us explain the DP recursion. Regarding function $z'(\cdot)$ in Step 2(a), to attain that job j finishes at time x, we need job $j-1$ to finish at time $\frac{x-\ell_j-a_j\tau}{1-a_j}$. If $\ell_j + a_j\tau > \tau$, then the domain of the function is empty, i.e., job j cannot be processed as a job of the first sequence because even if starting at time zero, it will finish after time τ. In this case, function z' is undefined, in Step 2(b) we set $z_j(x) \equiv z''(x)$, and in Step 2(c) we set $y_j(x) \equiv (1+b_j) \cdot y_{j-1}(x)$.

Suppose the infimum in Step 3 is reached at a point x^*. The overall job sequence then is $S = S_1^n(x^*), (\chi), S_2^n(x^*)$ and can be found by backtracking. If in stage j the state's value when performing backtracking from $z_n(x^*)$ (i.e., the corresponding value of x_j^* in $z_j(x_j^*)$) was generated by assigning the value of $z_j'(x_j^*)$ to $z_j(x_j^*)$ (hence $x_j^* = x_{j-1}^* + \ell_j + a_j \cdot (\tau - x_{j-1}^*)$), then we append job j to the end of S_1^{j-1}. If it instead was generated by assigning the value of $z''(x^*)$ to $z_j(x^*)$ (hence $x_j^* = x_{j-1}^*$), we insert job j to the beginning of S_2^{j-1}. In Step 3, the straddler job χ is appended to the end of $S_1^n(x^*)$, and $S_2^n(x^*)$ is started at $\max\{x^* + p_\chi(x^*), \tau\}$ with p_χ as in (1). If χ completes strictly before τ, idle time is inserted before starting the second sequence such that it starts precisely at τ; in this case the result is dominated by a solution for another straddler job.

3 Fully Polynomial Time Approximation Scheme

Alon & Halman's (2021) framework is used to derive an FPTAS for the DP. For the ease of presentation, instead of referring directly to Alon & Halman (2021), we cite from the concise summary available in Gawiejnowicz, Halman & Kellerer (2023, Appendix A).

We convert the DP (Algorithm 1) to integer values by multiplying all input numbers $\tau = q_\tau/r_\tau$, $\ell_j = q_{\ell_j}/r_{\ell_j}$, $a_j = q_{a_j}/r_{a_j}$, $b_j = q_{b_j}/r_{b_j}$ by $M := r_\tau \prod_{j=1}^n (r_{\ell_j} \cdot r_{a_j} \cdot (r_{a_j} - q_{a_j}) \cdot r_{b_j})$. Note that the factor $(r_{a_j} - q_{a_j})$ turns $1/(1-a_j)$ in (3a) into an integer, since $1/(1-a_j) = 1/(1 - q_{a_j}/r_{a_j}) = r_{a_j}/(r_{a_j} - q_{a_j})$. Doing so, the state space of x becomes the integer interval $[0, 1, \ldots, \tau M]$. Moreover, we divide the output value in (3e) by M to get back the unscaled objective value. Therefore, the DP is solved in $\mathcal{O}(n \cdot \tau M)$ time, which is to be repeated for each possible straddler job. Observing that M is in $\mathcal{O}(N^{4n+1})$ for

$$N := \max_{j=1,\ldots,n} \{q_\tau, r_\tau, q_{\ell_j}, r_{\ell_j}, q_{a_j}, r_{a_j}, q_{b_j}, r_{b_j}\}, \tag{4}$$

we conclude that the DP runtime is exponential in the number of input items (and is therefore not pseudo-polynomial).

Nevertheless, the DP (Algorithm 1) can be seen as a special case of Gawiejnowicz et al. (2023, Eq. (12)) in the following sense: (i) we set the level index t to be the index j and therefore the number of levels is $T = n$, i.e., the number of jobs to schedule; (ii) regarding DP equation (12) in Gawiejnowicz et al. (2023), we set $f_{t,1} = z_t$ and $f_{t,2} = y_t$, therefore the other index i is either 1 or 2, and $m = 2$; (iii) we set the state variable $I_{t,i}$ to be x, i.e., the completion time of the sequence of jobs that are executed before time τ; (iv) for every pair of levels t and i we set the additional information $A_{t,i}(x)$ to be the conditions stated in step 2 of the DP, which determine the values of $f_{t,i}$ in each case; (v) when considering level t in Gawiejnowicz et al. (2023, Eq. (12)), instead of using all previously calculated $\{z_{r,j}\}_{0 \le r < t, 1 \le j \le 2}$, we use only $\{z_{r,j}\}_{r=t-1, 1 \le j \le 2}$; (vi) we set the boundary functions to be $f_{0,1} \equiv 0$ and $f_{0,2} \equiv 1$. Thus, from (i)–(vi), we conclude that DP (Algorithm 1) is indeed a special case of Gawiejnowicz et al. (2023, Eq. (12)).

Next, we set a bound U_z on the ratio between the maximal value of functions $z_j(\cdot), y_j(\cdot)$ and their minimal non-zero value to be $U_z \leq (MN)^n$ (e.g., the product $\prod_{j=1}^{n}(1+b_j)$ after the scaling), and a bound U_S on the cardinality of the state space to be $U_S = MN$.

Functions $y_j(x)$ and $z_j(x)$, for $j = 1, \dots, n$, are monotone non-increasing, since as x grows the problem becomes less constrained, i.e., there is more space available for scheduling jobs between 0 and x. Therefore, the DP (Algorithm 1) is monotone and Condition A.1 in Gawiejnowicz et al. (2023) is satisfied; as well, it can be shown that Conditions 2–4(i) are satisfied, which grant us an approximated DP. Its last step is polynomial since computing the infimum in (3e) corresponds to calculating the minimum in the pseudo-polynomial state space $[0, 1, \dots, M\tau]$ while the approximated functions y_n, z_n are step functions with a polynomial number of steps. Finally, we use parameter value $\tau_f \in \mathcal{O}(n)$ to apply Theorem 4 in Gawiejnowicz et al. (2023) for each possible straddler job, and obtain an FPTAS.

Theorem 1. *There exists an FPTAS to minimize the makespan $C_{\max}$ on a single machine with time-dependent V-shaped processing times that runs in $\mathcal{O}\Big(\frac{n^6}{\varepsilon} \cdot \log^2 N \cdot \log \frac{n \log N}{\varepsilon}\Big)$ time, where N is the maximal value of the numbers in the input, as defined in equation* (4).

This runtime is by $1/\varepsilon$ (up to log terms) lower than Sedding's (2020*a*) FPTAS runtime, which is in $\mathcal{O}\Big(\frac{n^5}{\varepsilon^2} \cdot \log(1+b_{\max}) \cdot (\log(1+b_{\max}) + n \cdot \log(1+b_{\max}))\Big)$.

References

Alon, T. & Halman, N. (2021), ‘Automatic generation of FPTASes for stochastic monotone dynamic programs made easier’, *SIAM Journal on Discrete Mathematics* **35**(4), 2679–2722.

Cai, J.-Y., Cai, P. & Zhu, Y. (1998), ‘On a scheduling problem of time deteriorating jobs’, *Journal of Complexity* **14**(2), 190–209.

Cheng, T. C. E., Ding, Q., Kovalyov, M. Y., Bachman, A. & Janiak, A. (2003), ‘Scheduling jobs with piecewise linear decreasing processing times’, *Naval Research Logistics* **50**(6), 531–554.

Gawiejnowicz, S. (2020), ‘A review of four decades of time-dependent scheduling: Main results, new topics, and open problems’, *Journal of Scheduling* **23**(1), 3–47.

Gawiejnowicz, S., Halman, N. & Kellerer, H. (2023), ‘Knapsack problems with position-dependent item weights or profits’, *Annals of Operations Research* **326**(1), 137–156.

Halman, N. (2020), ‘A technical note: Fully polynomial time approximation schemes for minimizing the makespan of deteriorating jobs with nonlinear processing times’, *Journal of Scheduling* **23**(6), 643–648.

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

Ji, M. & Cheng, T. C. E. (2007), ‘An FPTAS for scheduling jobs with piecewise linear decreasing processing times to minimize makespan’, *Information Processing Letters* **102**(2-3), 41–47.

Kononov, A. V. (1997), ‘On schedules of a single machine jobs with processing times nonlinear in time’, *Discrete Analysis and Operational Research* **391**, 109–122.

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

Kovalyov, M. Y. & Kubiak, W. (2012), ‘A generic FPTAS for partition type optimisation problems’, *International Journal of Planning and Scheduling* **1**(3), 209–233.

Kubiak, W. & van de Velde, S. L. (1998), ‘Scheduling deteriorating jobs to minimize makespan’, *Naval Research Logistics* **45**(5), 511–523.

Sedding, H. A. (2020*a*), ‘Scheduling jobs with a V-shaped time-dependent processing time’, *Journal of Scheduling* **23**(6), 751–768.

Sedding, H. A. (2020*b*), *Time-Dependent Path Scheduling: Algorithmic Minimization of Walking Time at the Moving Assembly Line*, Springer Vieweg, Wiesbaden.

Sedding, H. A. (2023), ‘Mixed-model moving assembly line material placement optimization for a shorter time-dependent worker walking time’, *Journal of Scheduling.*

Two-stage approaches to solving the robust job-shop problem with uncertainty budget

Carla Juvin[1,2], Laurent Houssin[1,3], Pierre Lopez[1]

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

Keywords: Job-Shop Scheduling, Robust Optimization, Uncertainty Budget, Mixed Integer Linear Programming, Constraint Programming.

1 Introduction

The job-shop scheduling problem (JSSP) is a well studied and NP-hard problem where a set of jobs are to be processed on a set of machines. Each job is composed of a sequence of operations that must be processed on machines with given processing times in a given job-dependent order, and each machine can process only one operation at a time. The JSSP has received considerable attention and both metaheuristics and exact methods have been developed to solve the problem, the majority of them with the assumption that the parameters are deterministically known. However, in the real world, many sources of uncertainty can affect the quality and even the feasibility of a schedule.

There exist two major approaches to deal with data uncertainty: stochastic optimization and robust optimization. While stochastic optimization considers probability distribution, robust optimization assumes that uncertain data belong to a given uncertainty set and aims to optimize performance considering the worst-case scenario within that set.

In this paper, we propose exact solution methods to solve the robust job-shop scheduling problem. A two-stage robust optimization approach is used to deal with processing times uncertainty, where the first stage fixes the sequence of operations on machines whilst the second stage sets the operation start times.

2 Problem statement

An instance of the JSSP implies a set of jobs $\mathcal{J}$ and a set of machines $\mathcal{M}$. Each job $i \in \mathcal{J}$ consists of a sequence of n_i operations. The j^{th} operation $O_{i,j} \in \mathcal{O}_i$ of a job i must be performed by machine $\mu_{i,j} \in \mathcal{M}$ (with $\mu_{i,j} = m \iff O_{i,j} \in \mathcal{I}_m$, where $\mathcal{I}_m$ is the set of operations processed by machine m) and $p_{i,j}$ denotes its processing time. Each machine can process at most one operation $O_{i,j} \in \mathcal{I}_m$ at a time, each job can only be processed on one machine at a time, and preemption is not allowed: once an operation is started, it must be processed without any interruption.

We consider that the processing times of operations are uncertain. Each processing time $p_{i,j}$ of an operation $O_{i,j} \in \mathcal{O}_i, i \in \mathcal{J}$, belongs to the interval $[\bar{p}_{i,j}, \bar{p}_{i,j} + \hat{p}_{i,j}]$, where $\bar{p}_{i,j}$ is the nominal value and $\hat{p}_{i,j}$ the maximum deviation of the processing time from its nominal value.

The traditional robust optimization approach consists in protecting against the case when all parameters can deviate at the same time, which makes the solution overly conservative. Indeed, there is a very low probability that all parameters take their worst value all together. To overcome this limitation, Bertsimas (2004) introduces an uncertainty budget approach that allows a restriction on the number of deviations that can occur simultaneously to a given budget. In order to reach a trade-off between robustness and solution quality, we exploit this approach to define the uncertainty set.

Let Γ be the budget of uncertainty, *i.e.*, the maximum number of operations whose processing time can deviate simultaneously. For each scenario ξ, the processing time of operation $O_{i,j}$ is given by:

$$p_{i,j}(\xi) = \bar{p}_{i,j} + \xi_{i,j}\hat{p}_{i,j} \tag{1}$$

where $\xi_{i,j}$ is equal to 1 if the processing time of the operation deviates, 0 otherwise

We then define the uncertainty set $\mathcal{U}^\Gamma$ as:

$$\mathcal{U}^\Gamma = \{(\xi_{i,j})_{i\in\mathcal{J},1\leq j\leq n_i} \mid \sum_{i\in\mathcal{J}}\sum_{j=1}^{n_i}\xi_{i,j} \leq \Gamma\}. \tag{2}$$

The robust multi-stage optimization, introduced by Ben-Tal (2004), considers that a part of the decision variables must be instantiated before the uncertainty is revealed, while the other variables can be adjusted to the uncertainty realization. In our problem, we consider that the purpose is to find the sequence on the machines (first stage decisions), allowing to define a start time for each operation and each scenario (second stage decisions), minimizing the makespan in the worst-case scenario.

3 Solution methods

A robust problem can be solved using an extended formulation, which consists in duplicating the set of constraints involving uncertain parameters (in the present case, operation processing times) for all possible scenarios $\xi \in \mathcal{U}^\Gamma$. Usually formulated as a linear programming problem, it is also possible to adopt a constraint programming approach (Juvin 2023).

However, according to the structure of our uncertainty set, the number of scenarios increases exponentially with the number of operations, which quickly makes these models intractable. Therefore, in this section we deal with the evaluation of a worst-case scenario. This study then allows us to propose a compact formulation and decomposition methods of the problem.

3.1 Worst-case evaluation

In this section, it is assumed that a first-stage solution σ is given. Considering an uncertainty budget Γ, the worst-case evaluation is to identify a scenario, with at most Γ operations whose duration deviates, and that leads to the largest possible makespan. This problem can also be treated as the evaluation of a longest path in an directed acyclic graph (DAG). Such a method is actually used by Bold (2021) in the context of a robust resource-constrained project scheduling problem (RCPSP).

3.2 Compact model

As Bold (2021) for the robust RCPSP, we propose a compact formulation of the robust JSSP, based on the dual of the worst-case evaluation subproblem. We introduce the variables $C^\gamma_{i,j}$, which represent the end date of operation $O_{i,j}$ in the worst case, taking into account at most γ deviations. The compact model is as follows:

$$\min C_{\max} \tag{3}$$

$$s.t. \qquad C_{\max} \geq C^\Gamma_{i,n_i} \quad \forall i \in \mathcal{J} \tag{4}$$

$$C^{\gamma}_{i,j} \geq C^{\gamma}_{i,j-1} + \bar{p}_{i,j} \quad \forall i \in \mathcal{J}, \forall j \in \{2, \ldots, n_i\}, \forall \gamma \in \{0, \ldots \varGamma\} \tag{5}$$

$$C^{\gamma}_{i,j} \geq C^{\gamma-1}_{i,j-1} + \bar{p}_{i,j} + \hat{p}_{i,j} \quad \forall i \in \mathcal{J}, \forall j \in \{2, \ldots, n_i\}, \forall \gamma \in \{1, \ldots \varGamma\} \tag{6}$$

$$C^{\gamma}_{i,j} \geq C^{\gamma}_{i',j'} + \bar{p}_{i,j} - y_{i,j,i',j'} \cdot H \quad \forall m \in \mathcal{M}, \forall (O_{i,j}, O_{i,j'}) \in \mathcal{I}^2_m, \forall \gamma \in \{0, \ldots \varGamma\} \tag{7}$$

$$C^{\gamma}_{i,j} \geq C^{\gamma-1}_{i',j'} + \bar{p}_{i,j} + \hat{p}_{i,j} - y_{i,j,i',j'} \cdot H \quad \forall m \in \mathcal{M}, \forall (O_{i,j}, O_{i,j'}) \in \mathcal{I}^2_m, \forall \gamma \in \{1, \ldots \varGamma\} \tag{8}$$

$$C^{\gamma}_{i',j'} \geq C^{\gamma}_{i,j} + \hat{p}_{i,j} - (1 - y_{i,j,i',j'}) \cdot H \quad \forall m \in \mathcal{M}, \forall (O_{i,j}, O_{i,j'}) \in \mathcal{I}^2_m, \forall \gamma \in \{0, \ldots \varGamma\} \tag{9}$$

$$C^{\gamma}_{i',j'} \geq C^{\gamma-1}_{i,j} + \hat{p}_{i,j} + \hat{p}_{i,j} - (1 - y_{i,j,i',j'}) \cdot H \quad \forall m \in \mathcal{M}, \forall (O_{i,j}, O_{i,j'}) \in \mathcal{I}^2_m, \forall \gamma \in \{1, \ldots \varGamma\} \tag{10}$$

$$C^{0}_{i,1} \geq \bar{p}_{i,1} \quad \forall i \in \mathcal{J} \tag{11}$$

$$C^{\gamma}_{i,1} \geq \bar{p}_{i,1} + \hat{p}_{i,1} \quad \forall i \in \mathcal{J}, \ \forall \gamma \in \{1, \ldots \varGamma\} \tag{12}$$

3.3 Decomposition methods

We present a logic-based Benders decomposition method (Hooker 2000) as well as a column and constraint generation method (Zeng 2013). These two iterative approaches aim to decompose the problem into a master problem, and an adversarial subproblem, and share the same pattern. The master problem is formulated with an extended model for the robust job-shop problem (using MILP or CP) considering only a subset of scenarios and the subproblem evaluates the worst-case scenario. At each iteration, information relating to this worst-case scenario is added to the master problem.

For the Benders decomposition method, adding information about the violated scenarios consists in adding cuts in the master problem. Note that only the MILP formulation of the master is considered. The added cuts are as follows:

$$C_{\max} \geq \psi^*_h \cdot (1 - NumberOfChanges_h) \tag{13}$$

where ψ^*_h is the worst-case makespan obtained by the adversarial subproblem at iteration h and $NumberOfChanges_h$ is the number of changes, compared with the decisions made at the first stage of iteration h, that could affect the value of makespan. If no influential changes occur, then the makespan is at least equal to ψ^*_h; otherwise, the constraint is inactive.

For the column and constraint generation procedure, adding information about the violated scenario consists in generating the corresponding second-stage decision variables and the associated constraints. This is simply adding the worst-case scenario to the set of scenarios considered in the next iteration of the master:

$$\mathcal{U}^{k+1} = \mathcal{U}^{k} \cup \{\xi^*_k\}. \tag{14}$$

4 Numerical results

For computational experiments, we consider 58 classical instances of the job-shop problem from the literature, adapted to the robust context by randomly generating deviation values. We test all the models by varying the uncertainty budget according to four ratios: 5,%, 10,%, 15,% and 20,%, i.e. a total of 232 experiments per method.

The results of Table 1 are presented in terms of the number of best solutions found compared to the other methods ("best"), as well as the optimality gap ("gap (%)") obtained by each method.

For the smallest instances (6×6), all the methods succeed in finding the optimal solution (except one for the CCG_{MILP} method). For 10-machine instances, the *Benders* method

$\|\mathcal{J}\|$	$\|\mathcal{M}\|$	#	*Compact*		*Benders*		CCG_{MILP}		CCG_{CP}	
			best	gap (%)	best	gap (%)	best	gap (%)	best	gap (%)
6	6	4	4	0	4	0	3	0.5	4	0
10	5	20	17	13.8	16	34	7	8.45	15	3.5
10	10	72	28	20.64	43	45.15	5	27.85	9	24.49
15	5	20	12	43.75	6	57.3	6	57.4	20	1.05
15	10	20	1	44	7	58.3	6	58.45	10	33.8
15	15	20	0	40.8	6	60.25	6	60.1	11	42.4
20	5	24	2	61.63	4	70.54	4	70.58	24	3.88
20	10	20	0	50.8	1	67.5	2	67.45	18	31.85
20	15	12	0	–	1	65.75	0	65.67	11	47.25
30	10	20	0	–	0	79.65	0	79.25	20	18.3

Table 1. Number of best solutions found and average optimality gap for job-shop instances from the literature, categorized according to the instance size.

obtains the largest number of best solutions, but with relatively high optimality gaps. Finally, the CCG_{CP} method obtains the highest number of best solutions for the largest instances.

5 Conclusion

In this paper, we study the robust job-shop scheduling problem where operation processing times are uncertain and modeled by an uncertainty budget. We consider a two-stage decision process, where the sequences of operations must be decided before knowing the realization of the uncertainty, in order to be feasible for all scenarios, but where the processing dates of the operations can be adapted according to the observed durations. We propose a compact formulation and two decomposition methods based on solving a relaxed master problem and finding violated constraints at each iteration. For the largest instances, decomposition methods, in particular the column and constraint generation method with a master problem solved using constraint programming, yields better quality solutions.

References

Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovsk, 2004, "Adjustable robust solutions of uncertain linear programs", *Mathematical Programming*, Vol. 99, pp. 351–376.

Dimitris Bertsimas and Melvyn Sim, 2004, "The price of robustness", *Operations Research*, Vol. 52, pp. 35–53.

Matthew Bold and Marc Goerigk, 2021, "A compact reformulation of the two-stage robust resource-constrained project scheduling problem", *Computers & Operations Research*, Vol. 130, pp. 105232.

John N. Hooker, 2000, "Logic based methods for optimization: Combining optimization and constraint satisfaction", John Wiley & Sons, New York, 2000.

Carla Juvin, Laurent Houssin, and Pierre Lopez, 2023, "Constraint programming for the robust two-machine flow-shop scheduling problem with budgeted uncertainty", *CPAIOR 2023.*

Bo Zeng and Long Zhao, 2013, "Solving two-stage robust optimization problems using a column-and-constraint generation method", *Operations Research Letters*, Vol. 41, pp. 457–461.

Flow shops with reentry: Total completion time optimal schedules

Nicklas Klein[1], Michael Pinedo[2]

[1] Department of Business Administration, University of Bern, Switzerland
nicklas.klein@unibe.ch
[2] New York University, Leonard N. Stern School of Business, New York, NY, USA
mlp5@stern.nyu.edu

Keywords: Machine Scheduling, Flow Shop Scheduling, Priority Rules, Job Reentry.

1 Introduction

Flow shops are widely studied machine environments in which all jobs must visit all machines in the same order (Emmons and Vairaktarakis 2012). While conventional flow shops assume that each job traverses the shop only once, many industrial environments require jobs to pass through the shop multiple times before completion. This setting is called a flow shop with reentry and has numerous applications, such as integrated circuit manufacturing where multiple layers of patterns are placed on the same wafer (Graves *et al.* 1983) or manufacturing environments that require frequent job repair/rework (Yu and Pinedo 2020).

We consider a flow shop with m machines in series and n jobs that have to make multiple loops through the shop, i.e., after traversing the shop and completing its processing on the last machine, a job must return to machine 1 and traverse the shop again until it has completed all of its loops. The planning problem is to schedule all loops of all jobs while minimizing the total (unweighted) completion time.

Flow shops with reentry have received considerable attention in the literature (Choi and Kim 2008, Danping and Lee 2011, Shufan *et al.* 2023). Most publications in this area focus on minimizing the makespan. Yu and Pinedo (2020) propose the "Most Remaining Loops first" (MRL) rule and show that it minimizes the makespan under certain conditions, including unit processing times. The total completion time objective is examined by Jing *et al.* (2011), who propose a k-insertion heuristic to solve the problem. However, to the best of our knowledge, there is no analysis of optimal scheduling policies for the total completion time objective in the literature.

In this paper, we analyze the total completion time objective for flow shops with reentry. Since this problem is already strongly NP-hard for conventional flow shops with two or more machines (Garey *et al.* 1976), we consider the special case with unit processing times. We introduce so-called "non-interruptive" schedules, in which the next loop of any job starts on machine 1 as soon as its previous loop completes on the last machine, and show that there is a non-interruptive schedule that minimizes the total completion time. Additionally, we introduce the "Least Remaining Loops first" (LRL) rule and show that it minimizes the total completion time.

The remainder of this paper is organized as follows. In Section 2, we describe the planning problem in detail and provide an example. In Section 3, we analyze total completion time optimal schedules. In Section 4, we conclude the paper and give an outlook on future research.

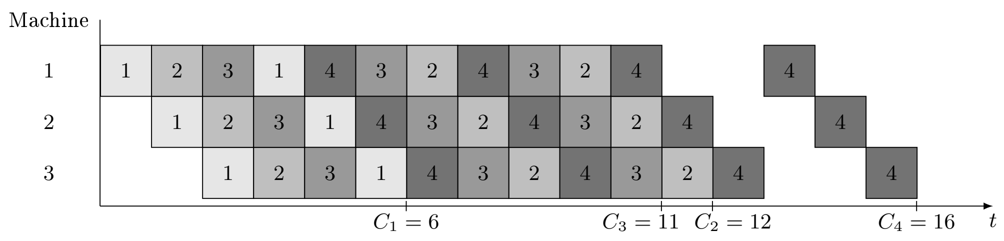
2 Problem description

Consider a flow shop environment with m machines in series and n jobs. Each job $j = 1, \ldots, n$ traverses all machines $\mathcal{L}_j$ times; after completing its processing on the last machine, a job must return to machine 1 to begin its next loop until it completes its last loop and exits the system. Let p_{ijk} be the processing time of job j on machine i in loop k; we assume $p_{ijk} = p = 1$. Let $\ell(j,k)$ refer to loop k of job j. In what follows, we only consider permutation schedules. For reentrant flow shops, these schedules are defined by considering each loop of a job as a sub-job. In total, $\sum_{j=1,\ldots,n} \mathcal{L}_j$ loops must be sequenced while adhering to precedence constraints between successive loops of the same job. These precedence constraints take the form $S_{jk} \geq C_{j,k-1}$ for all jobs $j = 1, \ldots, n$ and all loops $k = 2, \ldots, \mathcal{L}_j$, where we denote the start time of loop $\ell(j,k)$ on machine 1 as S_{jk} and its completion time on machine m as C_{jk}. A job is completed once its last loop is completed, i.e., $C_j = C_{j\mathcal{L}_j}$. Our objective is to find a permutation schedule σ that minimizes the total completion time $\sum_{j=1,\ldots,n} C_j$.

We illustrate the planning problem through the following example.

Example 1. Consider a reentrant flow shop with $m = 3$ machines and $n = 4$ jobs. The jobs require $\mathcal{L}_1 = 2$, $\mathcal{L}_2 = 3$, $\mathcal{L}_3 = 3$, and $\mathcal{L}_4 = 4$ loops. The Gantt chart for permutation sequence $\sigma = [\ell(1,1); \ell(2,1); \ell(3,1); \ell(1,2); \ell(4,1); \ell(3,2); \ell(2,2); \ell(4,2); \ell(3,3); \ell(2,3); \ell(4,3); \ell(4,4)]$ is displayed in Figure 1. We observe that machine 1 is idle for two time units as loop $\ell(4,4)$ can only start on machine 1 after loop $\ell(4,3)$ is completed on machine 3. The total completion time of this schedule is $C_1 + C_2 + C_3 + C_4 = 45$.

Fig. 1: Gantt chart for the schedule of Example 1



3 Total completion time optimal schedules

In this section, we first introduce non-interruptive schedules and show that there is a non-interruptive schedule that minimizes the total completion time. Then, we introduce the priority rule "Least Remaining Loops first" (LRL) and show that it minimizes the total completion time.

Definition 1. *In a non-interruptive schedule, the next loop of any job starts on machine 1 as soon as its previous loop completes on machine m, i.e.,*

$$S_{jk} = C_{j,k-1} \ (j = 1, \ldots, n, \ k = 2, \ldots, \mathcal{L}_j)$$

The following theorem establishes the relationship between non-interruptive schedules and total completion time optimal schedules for reentrant flow shops with unit processing times.

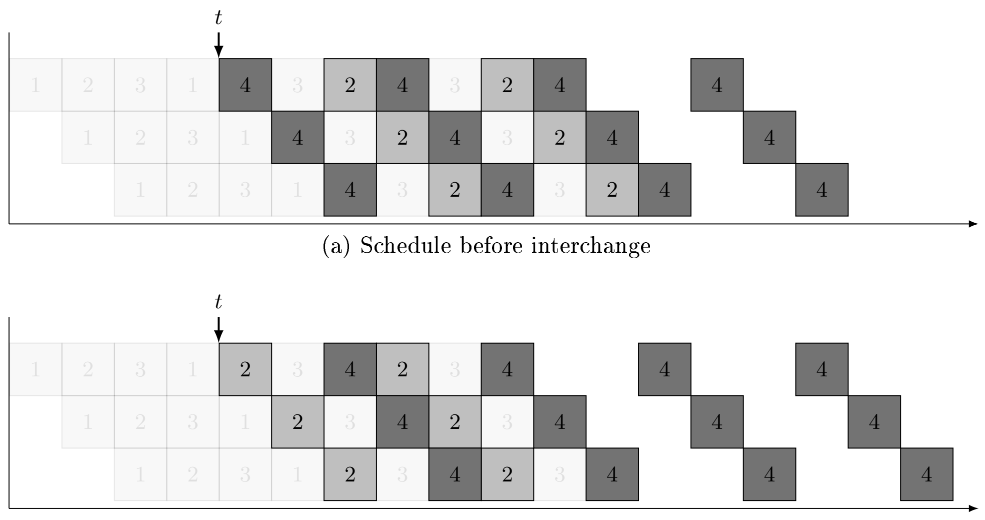
Theorem 1. *There is a non-interruptive schedule that minimizes the total completion time.*

Proof. The proof is by contradiction. Suppose that no non-interruptive schedule has minimal total completion time. Then an optimal schedule must have at least one interruption. We consider the last interruption, i.e., the last time t at which a loop $\ell := \ell(j,k)$ with $k < \mathcal{L}_j$ completes on machine m, but loop $\ell' := \ell(j',k')$ starts on machine 1. As t is the last interruption, there are no interruptions from time $t+1$ on.

We create a schedule without interruption at time t by interchanging loop ℓ' and all loops that are processed consecutively after ℓ' with loop ℓ and all loops that are processed consecutively after ℓ. We illustrate this interchange via the schedule of Example 1 in Figure 2, where $\ell' = \ell(4,1)$ and $\ell = \ell(2,2)$.

By analyzing the impact of this interchange on the total completion time, we can show that either the original schedule is not optimal or that the new schedule has the same total completion time; both contradict our assumptions.

Fig. 2: Demonstration of the interchange



(a) Schedule before interchange

(b) Schedule after interchange

We now introduce the priority rule LRL.

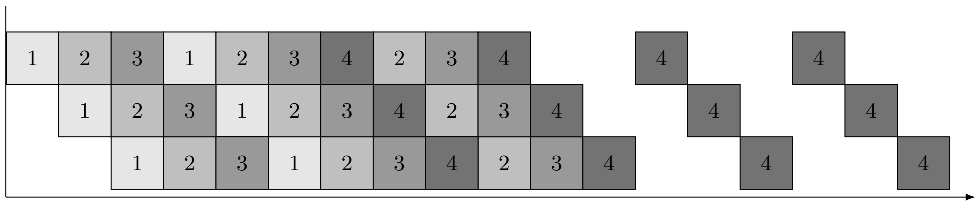
Definition 2. *The "Least Remaining Loops first" (LRL) priority rule schedules, whenever machine 1 becomes available, the next loop of a job that has, among all available jobs, the least loops remaining.*

We display an LRL schedule for the instance of Example 1 in Figure 3.

We now prove that LRL minimizes the total completion time for reentrant flow shops with unit processing times. The proof uses a similar interchange argument as the previous proof.

Theorem 2. *LRL minimizes the total completion time.*

Fig. 3: Schedule generated via priority rule LRL



Proof. The proof is by contradiction. Suppose LRL does not minimize the total completion time. This means every optimal schedule does not act according to LRL at least once. According to Theorem 1, we can assume that an optimal schedule has no interruptions. Let time t be the last time the optimal schedule does not act according to LRL with loop ℓ' being scheduled while loop ℓ would have had priority according to LRL. With a similar interchange as in the proof of Theorem 1, we can show that either the original schedule is not optimal or that the new schedule has the same total completion time; both contradict our assumptions.

4 Conclusion and outlook

In this paper, we considered scheduling a set of jobs that must go through multiple loops in a flow shop with as objective the minimization of the total completion time. We introduced non-interruptive schedules and showed that there is a non-interruptive schedule that minimizes the total completion time. We introduced the priority rule LRL and showed that it minimizes the total completion time.

For future research, we propose to analyze the conditions under which LRL minimizes the total completion time in proportionate or machine-ordered flow shops. Additionally, we propose to investigate the total weighted completion time objective. For this problem, an analysis of the "Weighted Least Remaining Loops first" (WLRL) rule would be a promising research direction. Preliminary tests have shown an average performance ratio of 1.01 and a worst-case ratio of about 1.2.

References

Choi S. and Y. Kim, 2008, "Minimizing makespan on an m-machine re-entrant flowshop", *Computers & Operations Research*, Vol. 35(5), pp. 1684-1696.

Danping L. and C. Lee, 2011, "A review of the research methodology for the re-entrant scheduling problem", *International Journal of Production Research*, Vol. 49(8), pp. 2221-2242.

Emmons H., G. Vairaktarakis, 2012, "Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications", *Springer New York, NY*

Garey M. R., D. S. Johnson and R. Sethi, 1976, "The Complexity of Flowshop and Jobshop Scheduling", *Mathematics of Operations Research*, Vol. 1(2), pp. 117-129.

Graves S., H. Meal, D. Stefek and A. H. Zeghmi, 1983, "Scheduling of re-entrant flow shops", *Journal of Operations Management*, Vol. 3(4), pp. 197-207.

Jing C., W. Huang and G. Tang, 2011, "Minimizing total completion time for re-entrant flow shop scheduling problems", *Theoretical Computer Science*, Vol. 412(48), pp. 6712-6719.

Shufan E., T. Grinshpoun, E. Ikar and H. Ilani, 2023, "Reentrant flow shop with identical jobs and makespan criterion", *International Journal of Production Research*, Vol. 61(1), pp. 183-197.

Yu T., M. Pinedo, 2020, "Flow shops with reentry: Reversibility properties and makespan optimal schedules", *European Journal of Operational Research*, Vol. 282(2), pp. 478-490.

On Branch-and-Price for Multi-Project Scheduling

Maximilian Kolter[1], Rainer Kolisch[1] and Martin Grunow[1]

Technical University of Munich, Germany
max.kolter@tum.de

Keywords: Project Scheduling, Multi-Project, Column Generation, Branch-and-Price.

1 Introduction

The resource-constrained multi-project scheduling (RCMPSP) has been subject to research for decades. First introduced by Pritsker *et. al.* (1969), the RCMPSP considers a central decision-maker scheduling several projects $p \in \mathcal{P}$ competing for a shared pool of resources $\mathcal{R}$ over discrete-time horizon $\mathcal{T} = 1, \dots, |\mathcal{T}|$. Each project p can be represented by an activity-on-the-node (AON) network $\mathcal{G}_p = \{\mathcal{V}_p, \mathcal{E}_p\}$; where the node set $\mathcal{V}_p$ represents the activities of project p, and the edges $\mathcal{E}_p$ represent precedence relationships between pairs of activities. Each activity j has to be processed for a duration of d_j periods, requiring r_{jk} units of resource k during processing. The resources $k \in \mathcal{R}$ to process activities are scarce and have a limited capacity of R_k units. Using, pulse variables x_{jt} that take the value 1, if activity j is started in period t, and 0 otherwise, the RCMPSP reads as follows:

$$\min \ f(x) \tag{1}$$

$$s.t. \ \sum_{t \in \mathcal{T}} x_{jt} = 1 \qquad \forall p \in \mathcal{P},\ j \in \mathcal{V}_p \tag{2}$$

$$\sum_{\tau = t-d_i+1}^{|\mathcal{T}|} x_{i\tau} + \sum_{\tau=0}^{t} x_{j\tau} \leq 1 \qquad \forall p \in \mathcal{P},\ (i,j) \in \mathcal{E}_p \tag{3}$$

$$\sum_{p \in \mathcal{P}} \sum_{j \in \mathcal{V}_p} \sum_{\tau = t-d_j}^{t} r_{jk} x_{j\tau} \leq R_k \qquad \forall k \in \mathcal{R},\ t \in \mathcal{T} \tag{4}$$

$$x_{jt} \in \{0,1\} \qquad \forall p \in \mathcal{P},\ j \in \mathcal{V}_p,\ t \in \mathcal{T} \tag{5}$$

Objective (1) seeks to minimize some linear function, e.g., the sum of weighted project completion times. Constraints (2) ensure that each activity starts exactly once. Constraints (3) are precedence constraints. Constraints (4) limit the resource usage by their capacity, and Constraints (5) define the variable domains. For conciseness, we represent Constraints (2), and (3) by $A\bar{x} \leq b$. Note that we use $\bar{x}$ to indicate that we refer to a vector of decision variables rather than a single variable. The matrix A has a block-angular structure:

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_{|\mathcal{P}|} \end{bmatrix}$$

Each block A_p corresponds to one project $p \in \mathcal{P}$, which opens the door for decomposing the problem by projects. However, most solution approaches for the RCMPSP merge all projects into a super-project, resulting in a single-project scheduling problem. While it is convenient to use a single-project approach, it neglects the opportunity to exploit the block-angular structure. An exception is the work of Deckro *et. al.* (1991), which uses column generation (CG) techniques for decomposing the RCMPSP by projects.

However, CG has not been adopted by the project scheduling community, and only a few attempts have been made to use CG for project scheduling problems. Presumably, CG is not competitive for project scheduling compared to other solution algorithms. However, in other domains (e.g., vehicle routing), branch-and-price (B&P) approaches, which embed CG algorithms in a branching framework, have become the standard. Hence, the question arises as to why CG and B&P do not seem to work for project scheduling. In this work, we seek to answer this question. For this purpose, in Section 2, we provide a CG formulation of the RCMPSP and analyze its lower bounds. In Section 3, we evaluate different branching strategies. Finally, Section 3.2 concludes this paper by summarizing our initial findings.

2 Column Generation

CG is used to solve linear programs (LPs) with a huge number of variables (Barnhart *et. al.* 2001). Such problems often arise from reformulations, which may have tighter relaxations and thus give better bounds. To solve such problems, CG considers only a subset of all variables (columns) in a so-called restricted master problem (RMP). After solving the RMP, its dual values are used to check if a column with negative-reduced costs is not considered yet. A so-called pricing problem, which seeks to generate a new column with minimal reduced costs, is used for this check. If a new column with negative reduced cost is found, the column is added to the RMP, and the process is repeated. Often, only a few columns are sufficient to solve the problem to optimality. Furthermore, for problems with a block-angular matrix, the pricing problem can be decomposed into several smaller problems. To apply CG to the RCMPSP, we reformulate the problem by providing the RMP in Section 2.1 and the pricing problems in Section 2.2.

2.1 Restricted Master Problem

For the RMP, we use additional notation. First, let the set $\mathcal{S}_p$ contain all schedules of project $p \in \mathcal{P}$, and let $\tilde{\mathcal{S}_p}$ be a subset of these schedules, i.e., $\tilde{\mathcal{S}_p} \subseteq \mathcal{S}_p$. The amount of resource k demanded by project p using schedule s in period t is denoted by r_{kpst}. Furthermore, we introduce binary variables Φ_{ps}, which take the value 1 if project p uses schedule s and 0 otherwise. Using this notation, we state the RMP as (6) - (9).

$$\min \; f(\Phi) \tag{6}$$

$$s.t. \sum_{s \in \tilde{\mathcal{S}_p}} \Phi_{ps} = 1 \qquad \forall p \in \mathcal{P} \qquad [\pi_p] \tag{7}$$

$$\sum_{p \in \mathcal{P}} \sum_{s \in \tilde{\mathcal{S}_p}} r_{kpst} \Phi_{ps} \leq R_k \qquad \forall k \in \mathcal{R},\ t \in \mathcal{T} \qquad [\pi_{kt}] \tag{8}$$

$$\Phi_{ps} \in [0, 1] \qquad \forall p \in \mathcal{P},\ s \in \tilde{\mathcal{S}_p} \tag{9}$$

Objective (6) is a reformulation of the original objective function. Constraints (7) ensure that we select a schedule for each project, also known as convexity constraints. We denote the dual variable associated with Constraints (7) by π_p. Constraints (8) are reformulated resource constraints associated with the dual variables π_{kt}. Finally, Constraints (9) define the variable domains.

Proposition 1. *The RMP is prone to primal degeneracy.*

Proof. Primal degeneracy occurs if several bases represent the same solution. A basis for the RMP consists of $|\mathcal{P}| + |\mathcal{R}| \times |\mathcal{T}|$ columns. However, due to Constraints (7), an optimal solution might have only $|\mathcal{P}|$ non-zero and but $|\mathcal{R}| \times |\mathcal{T}|$ zero columns. Replacing one of the zero basic columns with a non-basic column results in a degenerate solution.

Degeneracy is an undesirable property, leading to degenerate pivots in the simplex algorithm, slowing down the solving process.

2.2 Pricing Problem

The pricing problem seeks to generate new schedules with negative reduced costs. Because of the block-angular matrix, we can decompose the pricing problem into $|\mathcal{P}|$ pricing problems, one for each project p:

$$\min\ rc_p = f(x_p) - \sum_{\forall k \in \mathcal{R}} \sum_{t \in \mathcal{T}} \pi_{kt} \sum_{j \in \mathcal{V}_p} \sum_{\tau = t - d_j}^{t} r_{jk} x_{j\tau} - \pi_p \tag{10}$$

$$s.t.\ A_p \bar{x}_p \leq b_p \tag{11}$$

$$\bar{x}_p \in \{0, 1\} \tag{12}$$

Objective (10) seeks to minimize the schedule's reduced cost rc_p. Constraints (11) are the subset of the original constraints $A\bar{x} \leq b$ corresponding to project p. Lastly, constraints (12) define the decision variables for project p.

Proposition 2. *The pricing problem has the integrality property, i.e., each vertex of the polytope described by $A_p\bar{x}_p \leq b_p$ is integral.*

The proof for this observation is provided in (Möhring *et. al.* 2001) and stems from the fact that the constraint matrix A_p is totally unimodular (TU), and the right-hand side b_p is integer.

2.3 Lower Bounds

As mentioned, CG is often used because the reformulated problem may have tighter relaxations and thus give better bounds. However, our analysis shows the following:

Proposition 3. *The lower bounds obtained from solving the RMP via CG are equivalent to the lower bounds obtained from solving the LP relaxation of the original problem.*

Proof. From CG theory (Lübbecke and Desrosiers 2005), it is known that if the pricing problem has the integrality property, then the feasible regions of the CG and the LP relaxation of the original problem are equivalent. Hence, it follows from Proposition 2 that the lower bounds are equivalent.

3 Branch-and-Price

CG solves the RMP, a linear relaxation of the reformulated problem. CG can be embedded into a branch-and-bound framework to obtain integer solutions, resulting in a B&P algorithm. The literature on B&P for project scheduling proposes two branching rules. First, (Montoya *et. al.* 2014) and (Coughlan *et. al.* 2015) propose branching on start times. Second, (Coughlan *et. al.* 2015) and (Van Den Eeckhout *et. al.* 2020) propose branching on resource demands. Both branching rules are enforced by removing violating columns from the RMP and adding branching constraints to the pricing problems. In the following, we formalize the branching rules and analyze their impact on the lower bounds.

3.1 Branching on Start Times

Let S^*_j be the start time of activity j obtained from solving the RMP. If S^*_j is fractional, we create two child nodes by adding the branching constraints:

$$\sum_{t \in \mathcal{T}} t x_{jmt} \leq \lfloor S^*_j \rfloor \text{ (left branch)} \qquad \sum_{t \in \mathcal{T}} t x_{jmt} \geq \lceil S^*_j \rceil \text{ (right branch)} \tag{13}$$

Proposition 4. *Branching on start times does not improve the lower bounds obtained from the RMP compared to bounds obtained from the LP relaxation of the original problem.*

Due to space limitations, we will only sketch the proof: First, we reformulate Constraints (13) such that the integrality property is maintained. Then, we apply the same reasoning we used for proving Proposition 3.

3.2 Branching on Resource Demands

Let r^*_{kpst} be the resource demand of project p obtained from solving the RMP. If r^*_{kpst} is fractional, we create two child nodes by adding the branching constraints:

$$r_{kpt} \leq \lfloor r^*_{kpt} \rfloor \text{ (left branch)} \qquad\qquad r_{kpt} \geq \lceil r^*_{kpt} \rceil \text{ (right branch)} \tag{14}$$

Proposition 5. *Branching on resource demands may improve the lower bounds obtained from the RMP compared to bounds from the LP relaxation of the original problem.*

Proof. Adding constraints of type (14) to the pricing problem results in a resource-constraint project scheduling problem, well-known to be NP-hard. From CG theory (Lübbecke and Desrosiers 2005), it follows that, due to the NP-hardness of the pricing problem, the relaxation of the RMP is tighter than the LP relaxation of the original problem. Thus, the RMP's lower bounds are at least as good as the bounds of the LP relaxation and potentially stronger.

4 Future Research

We studied lower bounds and other properties of B&P algorithms for the RCMPSP. Our next step is an experimental evaluation of the branching rules. This includes analyzing the impact on the size of the branching tree, lower bounds, and runtimes. Further, we will evaluate the impact of solving the RMP via Lagrangian relaxation as a remedy for its degeneracy. Finally, we will benchmark the B&P approach against a commercial solver.

References

Barnhart C., Johnson E. L., Nemhauser G. L., Savelsbergh M. W., Vance P. H., 1998, "Branch-and-price: Column generation for solving huge integer programs", *Operations Research*, Vol. 46(3), pp. 316-329.

Deckro R. F., Winkofsky E. P., Hebert J. E., and Gagnon R. (1991). "A decomposition approach to multi-project scheduling" *European Journal of Operational Research*, Vol. 51(1), pp. 110-118.

Lübbecke M.E., Desrosiers J., 2005, "Selected topics in column generation", *Operations Research*, Vol. 53(6), pp. 1007-1023.

Montoya C., Bellenguez-Morineau O., Pinson E., and Rivreau D. (2014). "Branch-and-price approach for the multi-skill project scheduling problem" *Optimization Letters*, Vol. 8, pp. 1721-1734.

Möhring R. H., Schulz A. S., Stork F., and M. Uetz, 2001, "On project scheduling with irregular starting time costs", *Operations Research Letters*, Vol. 28(4), pp. 149-154.

Pritsker A. A. B., Waiters L. J., and Wolfe P. M., 1996, "Multiproject scheduling with limited resources: A zero-one programming approach", *Management Science*, Vol. 16(1), pp. 93-108.

Coughlan E. T., Lübbecke, M. E., and Schulz, J. (2015). "A branch-price-and-cut algorithm for multi-mode resource leveling", *European Journal of Operational Research*, Vol 245(1), 70-80.

Van Den Eeckhout M., Vanhoucke M., and Maenhout B. (2020). "A decomposed branch-and-price procedure for integrating demand planning in personnel staffing problems", *European Journal of Operational Research*, Vol. 280(3), pp. 845-859.

Volland J., Fügener A., and Brunner J. O. (2017). "A column generation approach for the integrated shift and task scheduling problem of logistics assistants in hospitals", *European Journal of Operational Research*, Vol. 260(1), pp. 316-334.

A more powerful checker for the cumulative scheduling problem

Kristina Kumbria[1], Jacques Carlier[1], Antoine Jouglet[1] and Abderrahim Sahli[2]

[1] Université de Technologie de Compiègne, France
`kristina.kumbria@hds.utc.fr, jacques.carlier@hds.utc.fr, antoine.jouglet@hds.utc.fr`
[2] Univ Gustave Eiffel,ESIEE Paris, France
`abderrahim.sahli@esiee.fr`

Keywords: Scheduling; resource; tripartition; complexity; dynamic programming

1 Introduction

The Cumulative Scheduling Problem (CuSP) involves scheduling a set $I = \{1, \ldots, n\}$ of n tasks, on a resource with a given capacity of m units. Each task i is characterized by a release date r_i, a duration p_i, a deadline d_i, and a resource requirement of c_i units. Energetic Reasoning (ER), originally introduced in (J. Erschler and P. Lopez), is proving to be a powerful tool for tackling the CuSP. It focuses on the development of feasibility tests, commonly referred to as ER checkers, and adjustments concerning time constraints(L. Hidri, A. Gharbi and M. Haouari), (A. Tesch). Several ER checkers have been proposed in the existing literature. In 1999, (P. Baptiste, C. Le Pape, and W. Nuijten) proposed an $O(n^2)$ checker that evaluates the energy balance of $O(n^2)$ intervals. In 2018, (Y. Ouellet and C-G. Quimper) introduced an $O(n \log^2 n)$ checker based on the Monge Matrix and Range trees. In (J. Carlier, A. Sahli, A. Jouglet and E. Pinson), we presented an $O(\alpha(n)n \log n)$ checker, reducing the number of necessary intervals and following the methodology of Ouellet and Quimper, where $\alpha(n)$ is the inverse Ackermann function.

The purpose of this presentation is to introduce a new definition of the energetic reasoning method for the checker, using an Integer Linear Programming (ILP) model outlined in Section 2. This ILP program takes into account the integer constraints of c_i, which makes our new checker more powerful than the traditional one. By maximizing the proportion of tasks executed outside the interval, we minimize the proportion executed inside the interval, thereby obtaining a lower-bound evaluation. The ILP can be solved efficiently using a dynamic programming approach, allowing us to evaluate the energy balance in the general case.

2 Tripartition problem: Dynamic programming approach

Here, we introduce a tripartition problem involving a subset of tasks, denoted as $\mathcal{J} \subseteq I$, with each task having three integer values: a_i, b_i, and c_i. Additionally, two values, $m_{\mathcal{A}}$ and $m_{\mathcal{B}}$, are provided, which are smaller than the overall capacity m. We aim to solve the following mathematical program:

$$\max_{\mathcal{A} \subset \mathcal{J}, \mathcal{B} \subset \mathcal{J}} \left(\sum_{i \in \mathcal{A}} a_i c_i + \sum_{i \in \mathcal{B}} b_i c_i \right) \quad \textbf{subject to:} \quad \sum_{i \in \mathcal{A}} c_i \leq m_{\mathcal{A}} \quad \textbf{and} \quad \sum_{i \in \mathcal{B}} c_i \leq m_{\mathcal{B}} \qquad (1)$$

where $\mathcal{A}$ and $\mathcal{B}$ represent disjoint subsets of $\mathcal{J}$. Furthermore, let $\mathcal{M}$ denote the set $\mathcal{J}$ minus the elements in $\mathcal{A}$ and $\mathcal{B}$. This optimization problem can be expressed as the following integer linear program (ILP):

$$P(\mathcal{J}, m_{\mathcal{A}}, m_{\mathcal{B}}) = \max \sum_{i \in \mathcal{J}} (a_i c_i x_i + b_i c_i y_i) \tag{2}$$

$$\textbf{subject to: } x_i + y_i \leq 1, \forall i \in \mathcal{J} \quad \textbf{and} \quad \sum_{i \in \mathcal{J}} c_i x_i \leq m_{\mathcal{A}} \quad \textbf{and} \quad \sum_{i \in \mathcal{J}} c_i y_i \leq m_{\mathcal{B}} \tag{3}$$

where $x_i \in \{0,1\}$ (resp. $y_i \in \{0,1\}$) are binary variables indicating whether task i is in $\mathcal{A}$ (resp. $\mathcal{B}$). This problem can be seen as a variant of the knapsack problem, specifically a multiple-knapsack problem. In this scenario, we are provided with a set $\mathcal{J}$ of items to be allocated into two knapsacks, each with its respective capacity: $m_{\mathcal{A}}$ and $m_{\mathcal{B}}$. Each item $i \in \mathcal{J}$ is characterized by its weight c_i and its profit per unit weight b_i (resp. a_i) when placed in the first (resp. second) knapsack. The goal is to determine two disjoint subsets $\mathcal{A}$ and $\mathcal{B}$ of $\mathcal{J}$ corresponding to the contents of the two knapsacks in a way that maximizes the total profit, represented by the sum of the selected items. To address this problem, we employ a dynamic programming approach by constructing a table denoted as F. The entries in this table are computed as follows:

$$F(0, m_1, m_2) = 0 \qquad \forall m_1 \in \{0, ..., m_{\mathcal{B}}\}, \forall m_2 \in \{0, ..., m_{\mathcal{A}}\} \tag{4}$$

$$F(i, m_1, m_2) = \max \left\{ \begin{array}{r} F(i-1, m_1, m_2), \\ F(i-1, m_1 - c_i, m_2) + b_i c_i, \text{ iff } m_1 \geq c_i \\ F(i-1, m_1, m_2 - c_i) + a_i c_i, \text{ iff } m_2 \geq c_i \end{array} \right\} \begin{array}{r} \forall m_1 \in \{0, ..., m_{\mathcal{B}}\}, \\ \forall m_2 \in \{0, ..., m_{\mathcal{A}}\}, \\ \forall i \in \mathcal{J} \end{array} \tag{5}$$

The terms in the expressions correspond to different cases: $F(i-1, m_1, m_2)$ represents the case where item i is not included in the optimal solution. $F(i-1, m_1 - c_i, m_2) + b_i c_i$ corresponds to the situation where item i is placed in the first knapsack(iff $m_1 \geq c_i$). $F(i-1, m_1, m_2 - c_i) + a_i c_i$ corresponds to the case where item i is placed in the second knapsack(iff $m_2 \geq c_i$). The complexity of this dynamic programming approach is $O(n \times m_{\mathcal{A}} \times m_{\mathcal{B}})$.

3 Energy evaluation of an interval

The objective here is to establish a lower bound on energy requirement within a given interval $[\alpha, \delta]$ using a dynamic programming model. Minimizing this energy requirement is equivalent to maximizing the parts of tasks that do not fall within the specified interval. Let $\mathcal{J}(\alpha, \delta)$ be the set of tasks that consistently interact with interval $[\alpha, \delta]$, formally defined as $\mathcal{J}(\alpha, \delta) = \{i \in I \mid r_i + p_i \geq \alpha \text{ and } d_i - p_i \leq \delta\}$. In this context, b_i (resp. a_i) denotes the maximum part of task i that can be processed strictly before α (reps. after δ). We introduce the concepts of *crossing tasks* and *semi-crossing tasks*. These definitions allow a more precise assessment of the energy balance within an interval.

Definition 1.
- *A task i is called a crossing task (mandatory part defined in (A. Lahrichi)) if* $(d_i - p_i) < \alpha$ *and* $(r_i + p_i) > \delta$. *Its minimum energy demand is equal to* $w_i = (\delta - \alpha) \times c_i$.
- *A task i is called a plus-semi-crossing task if* $(r_i + p_i) > \delta$ *and* $\alpha \leq (d_i - p_i) \leq \delta$. *Its minimum energy demand is equal to* $w_i = (\delta - d_i + p_i) \times c_i$*, and it is achieved by scheduling task i at time* $d_i - p_i$ *(i.e., it finishes after* δ*).*
- *A task is called a minus-semi-crossing task if* $(d_i - p_i) < \alpha$ *and* $\alpha \leq (r_i + p_i) \leq \delta$. *Its minimum energy demand is equal to* $w_i = (r_i + p_i - \alpha) \times c_i$*, and it is achieved by scheduling task i at time* r_i *(i.e., it starts before* α*).*

There are a total of m' crossing tasks, leaving (m - m') resources available. However, some of these resources are still required by semi-crossing tasks. This leads to $m_{\mathcal{B}}$ available resources before α and $m_{\mathcal{A}}$ available resources after δ. Define $\mathcal{C}(\alpha, \delta)$ (resp. $\mathcal{S}(\alpha, \delta)$) to be the set of crossing (resp. semi-crossing) tasks. Then, let $\mathcal{J}'(\alpha, \delta)$ be the set of tasks in $\mathcal{J}(\alpha, \delta)$ that do not belong to $\mathcal{C}(\alpha, \delta) \cup \mathcal{S}(\alpha, \delta)$. The total energy over the time interval $[\alpha, \delta]$, denoted as $W(\alpha, \delta)$, can be computed by determining a tripartition $(\mathcal{A}, \mathcal{B}, \mathcal{M})$ of $\mathcal{J}'(\alpha, \delta)$.

$$W(\alpha, \delta) = \sum_{i \in \mathcal{C}(\alpha,\delta) \cup \mathcal{S}(\alpha,\delta)} w_i + \sum_{i \in \mathcal{J}'(\alpha,\delta)} p_i - P(\mathcal{J}'(\alpha, \delta), m_{\mathcal{A}}, m_{\mathcal{B}}) \tag{6}$$

4 New checker

Given an instance of CuSP, if the condition

$$\forall \alpha, \delta \in \mathbf{N}^+, \alpha < \delta \quad (\delta - \alpha) \times m - W(\alpha, \delta) \geq 0 \tag{7}$$

is violated then the considered instance of CuSP is infeasible. We have proved the following proposition by using non-trivial properties of ILP.

Proposition 1. *To ensure that condition (7) holds, it is sufficient to check intervals* $[\alpha, \delta] \in \Omega = \{[\alpha, \delta] \mid \alpha \in \mathcal{O}_1, \delta \in \mathcal{O}_2, \alpha < \delta\}$ *with:* $\mathcal{O}_1 = \{r_i, r_i + p_i, d_i - p_i \mid i \in I\}$ *and* $\mathcal{O}_2 = \{r_i + p_i, d_i, d_i - p_i \mid i \in I\}$.

This checker is more efficient than the one based on classical ER. It can be used to establish an improved lower bound for the CuSP optimization problem. We conducted experiments to assess its performance, testing it with randomly generated data using uniform distributions. Specifically, we generated data with different numbers of tasks ($n = 10, 20, 50, 100$) and resource capacities ($m = 5, 6, 7, 8, 9, 10$). The instances were generated as follows:

- The resource requirements c_i are generated between 1 and m.
- For 80% of tasks, the processing times are generated between 1 and 20. For the remaining tasks, they are generated between 20 and 50.
- The initial value of C_{max} is set to $\max(\max_{\forall i} p_i, 1.1 \times \frac{\sum_i p_i c_i}{m})$.
- The release dates r_i are generated between 0 and $(C_{max} - p_i)$.
- The deadlines d_i are generated between $p_i + r_i$ and C_{max}.

Table 1. Results

n	m	% of instances with $LB > LB_{class}$	*gap*(%)
20	7	80	1.439
20	9	87	1.643
20	10	85	1.707
50	7	77	0.587
50	9	83	0.565
50	10	86	0.706
100	7	79	0.314
100	9	78	0.272
100	10	84	0.301

Table 1 provides a summary of the computational results obtained in the generated instances. In this table, LB represents the lower bound based on the new checker, LB_{class}

represents for the classical ER lower bound (represented as LB_3 in (J. Carlier, A. Jouglet and A. Sahli)), and *gap* denotes the average difference calculated as $\frac{LB-LB_{class}}{LB}$. Both $(LB-1)$ and $(LB_{class}-1)$ correspond to the largest values at which each checker detects infeasibility.

Based on the obtained results, it is evident that the new checker is more efficient than the classical one. Specifically, there was a significant improvement in the lower bound in around 80% of the cases examined, despite the relatively small average deviation between the two lower bounds.

5 Conclusion

We have proposed a new ER checker for the CuSP based on dynamic programming. Recognizing the inefficiency of considering all intervals, we have identified specific relevant intervals $[\alpha, \delta]$, where $\alpha \in \{r_i, r_i + p_i, d_i - p_i\}$ and $\delta \in \{r_i + p_i, d_i, d_i - p_i\}$. Our implementation of the algorithm uses these intervals. The results demonstrate its superior efficiency compared to the classical checker, consequently enhancing the ER lower bound for the CuSP.

One perspective of this work is to improve the efficiency of the method by introducing redundant resources as in (J. Carlier and E. Néron). We are also studying the particular case of the m-machine scheduling problem. The perspectives of industrials could be to integrate this checker in their optimization constraint programming software.

References

Baptiste P., Le Pape C., and Nuijten W. 1999, "Satisfiability tests and time-bound adjustments for cumulative scheduling problems", *Annals of Operations Research*, Vol. 92, pp. 305-333.

Carlier J., Jouglet A., and Sahli A. 2023, "Algorithms to compute the energetic lower bounds of the cumulative scheduling problem", *Annals of Operations Research*, https://doi.org/10.1007/s10479-023-05596-9

Carlier J. and Néron E. 2007, "Computing redundant resources for the resource constrained project scheduling problem", *European Journal of Operational Research*, Vol. 176, Issue 3, pp. 1452-1463.

Carlier J., Sahli A., Jouglet A., and Pinson E. 2022, "A faster checker of the energetic reasoning for the cumulative scheduling problem", *International Journal of Production Research*, Vol. 60.

Erschler J., and Lopez P. 1990, "Energy-based approach for task scheduling under time and resources constraints", In Proceedings of the 2nd international workshop on project management and scheduling, pages 115–121, 1990. Release Dates and Delivery Times", *Journal of Scheduling*, Vol. 05, pp. 329-355.

Hidri L., Gharbi A., and Haouari M. 2008, "Energetic Reasoning Revisited: Application to Parallel Machine Scheduling", *Journal of Scheduling*, Vol. 11, pp. 239-252.

Lahrichi A. 1982, "Ordonnancements : La notion de parties obligatoires et son application aux problèmes cumulatifs", *RAIRO* no. 3, pp. 241-262

Ouellet, Y., and Quimper, C-G. 2018, "An $O(n \log^2 n)$ Checker and $O(n^2 \log n)$ Filtering Algorithm for the Energetic Reasoning," *The 15th International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pp. 477-494.

Tesch A. 2018, "Improving energetic propagations for cumulative scheduling", *International Conference on Principles and Practice of Constraint Programming*, pp. 629-645.

Fast Lower Bounds for Scheduling Problems in Hexaly Optimizer

Philippe Laborie

Hexaly, 251 Boulevard Pereire, Paris, France
`plaborie@hexaly.com`

Keywords: Lower Bounds, Resource-Constrained Project Scheduling, Job-Shop Scheduling, Machine Assignment and Scheduling, Constraint Programming, Software.

1 Introduction

Hexaly Optimizer (formerly LocalSolver) is a mathematical optimization solver used to address industrial problems in routing, scheduling, packing, and network design, among other areas (Gardi *et. al.* 2014). As such, its main purpose is to produce good primal solutions in a limited amount of computation time. In spite of this focus on primal solutions, the production of good dual bounds (and proofs of optimality) is interesting for several reasons (reassuring the user about the solver's capabilities, making sure that the formulation is consistent with the reality, etc.).

In this context, we describe some recent work that aims to improve the dual bounds calculated by Hexaly Optimizer on scheduling problems without affecting its performance on primal solutions. This paper focuses on *makespan* minimization. Our goal is to compute *as fast as possible*, at the start of the resolution, a decent lower bound on the *schedule makespan* in addition to the more expensive bounds calculated during the search. By *fast*, we mean a low calculation time compared to that allocated to solving the problem, that is to say, typically less than a second for problems of a few thousand tasks.

2 Context and notations

To simplify the notations, we assume that the problem is transcribed into the canonical representation of scheduling problems proposed in (Laborie *et. al.* 2009). In particular:

– Activities are represented by interval variables x_i which can be optional. An optional interval variable is a decision variable whose possible values are either an interval of integers $[s, e)$ or a particular value : *absent*.
– For an interval variable x_i, the expressions $p(x_i)$, $s(x_i)$, $e(x_i)$, $l(x_i)$ respectively represent the presence (Boolean), the start, the end and the length $(e(x_i) - s(x_i))$ of the interval when present.
– Temporal constraints are represented by constraints of type: $e(x_i) \leq s(x_j)$.
– Presence constraints are expressed as implications of type $p(x_i) \Rightarrow p(x_j)$.
– The constraint $alternative(x, \{x_1, ..., x_n\})$ means that when x is present, one and only one of the x_i ($i \in [1, n]$) is present and it is then equal to x. This constraint is commonly employed to handle machine allocation where activity x must be allocated to one out of n possible resources.
– The constraint $disjoint(\{x_1, ..., x_n\})$ means that the present intervals x_i are pairwise disjoint. Usually, this constraint is used to describe disjunctive resources like single-capacity machines. It can be complemented with a sequence-dependent setup time matrix.

– The constraint $cumul(\{x_1, ..., x_n\}, \{y_1, ..., y_n\}) \leq C$, where the y_i are integer variables, means that the accumulation of contributions from intervals x_i with value y_i do not exceed C. This constraint is typically used to represent renewable resources.

If z is an integer variable, we denote z^{min} and z^{max} respectively as the lower and upper bounds of the domain of z after the initial propagation of the problem. We also assume a dummy integer variable h such that $\forall i, e(x_i) \leq h$ (value of the end of the schedule).

3 Lower bounds computation

The calculated lower bounds are based on energetic relaxations of the problem and implement the components below.

Reformulation of quasi-disjunctive cumuls. As we will see later, some components of the calculation of lower bounds consider redundant reformulations of cumul constraints inspired by (Baptiste and Bonifas 2018). More precisely, for a constraint $cumul(\{x_1, ..., x_n\}, \{y_1, ..., y_n\}) \leq C$, when there exist 3 intervals i, j, k such that $y_i^{min} + y_j^{min} + y_k^{min} > C$, an additional (redundant) cumul constraint of capacity 2 is used with contributions $y \in \{0, 1, 2\}$, which is optimal from an energy point of view. This reformulation does not rule out any feasible solution. It can be computed in $O(n \log(n))$ for each cumul. An example of such a reformulation is illustrated on Figure 1.

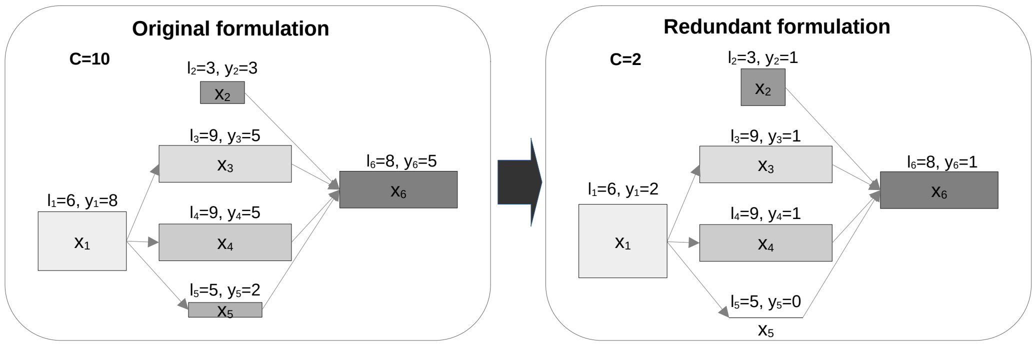


Fig. 1. Example of quasi-disjunctive cumul reformulation

Energy precedence propagation. The propagation of *energy precedence* (Laborie 2003) for an interval variable x_i participating in a resource constraint (of type *disjoint* or *cumul*) guarantees that for any subset Ω of x_i's predecessors in the temporal constraint graph, the resource contains enough energy to execute all intervals of Ω between the minimal start time of Ω and the start of x_i. This propagation makes it possible to increase the values of $s^{min}(x_i)$ even when there is no constraint on the latest end time of the schedule h. It can be performed for all interval variables and for all subsets Ω with a worst-case complexity of $O(n(p + \log(n)))$ where n denotes the number of interval variables and p the maximum number of predecessors of a given variable in the precedence graph ($p < n$). In our framework, this propagation is executed at the root node of the resolution on all the *disjoint* and *cumul* constraints until a global fix point is reached. Besides the original formulation of cumul constraints this propagation also considers the reformulated cumuls. For example considering the reformulated cumul on the right side of Figure 1 and the subset $\Omega = \{x_1, x_2, x_3, x_4, x_5\}$ of predecessors of x_6, the energy precedence propagation would infer that interval x_6 cannot start before $0 + \lceil \frac{(6 \cdot 2)+(3 \cdot 1)+(9 \cdot 1)+(9 \cdot 1)+(5 \cdot 0)}{2} \rceil = 17$.

Linear energy relaxation. This relaxation is particularly useful in case of optional interval variables (resource allocation problems). It consists of a linear program (LP) whose only decision variables are the Boolean presence variables $p(x_i)$. The presence and alternative constraints are directly translated in the LP. For example a constraint $alternative(x, \{x_1, ..., x_n\})$ is formulated as:

$$\sum_{i \in [1,n]} p(x_i) = p(x)$$

An energetic relaxation is introduced for the resources. For a constraint $disjoint(\{x_1, ..., x_n\})$ we will have:

$$\min_{i \in [1,n]} s^{min}(x_i) + \sum_{i \in [1,n]} p(x_i) \cdot l^{min}(x_i) \leq h$$

For a constraint $cumul(\{x_1, ..., x_n\}, \{y_1, ..., y_n\}) \leq C$ we will have:

$$C \cdot \min_{i \in [1,n]} s^{min}(x_i) + \sum_{i \in [1,n]} p(x_i) \cdot l^{min}(x_i) \cdot y_i^{min} \leq C \cdot h$$

Reformulated cumul constraints are also considered in the linear relaxation.

This basic formulation is improved by considering the latency durations of the intervals (minimum length of the longest path after a given interval in the precedence graph) as well as the inaccessible energy due to the $s^{min}(x_i)$ at the start of schedule.

The objective of the linear program is to minimize the end date h. Although the decision variables $p(x_i)$ are Boolean and the problem is an ILP, in practice for performance reasons, only the LP linear relaxation is solved.The optimal solution of the LP provides a lower-bound h^* on the makespan h.

Dichotomy on bounds. The components presented above permit to find new bounds for the start and end of interval variables and for the makespan. Given these bounds, the last component of the proposed lower bound computation consists in performing a dichotomy on the objective value in order to find the smallest value $h_{LB} \in [h^{min}, h^{max}]$ such that no inconsistency is detected when propagating the constraint $h \leq h_{LB}$. By construction, h_{LB} is a valid lower bound on the makespan. This step uses classical propagation algorithms like the *disjunctive constraint* for disjoint and *timetable constraint* for cumul (Baptiste *et. al.* 2006).

4 Results

Table 1 summarizes the average deviation of our lower bounds compared to the best-known primal solutions on some classic scheduling benchmarks. For each of these benchmarks, the problem was stated using the Hexaly Optimizer formulation provided in the example tour (Hexaly 2024). The calculation times of bounds are negligible (less than 0.5s). On the RCPSP, among the 2520 instances tested, 121 lower bounds from the literature are improved, notably on the largest instances comprising 300 tasks for which more than 25% of the best known lower bounds are improved (RG300, see (Vanhoucke *et. al.* 2016)).

[1] Flexible jobshop instances: *Barnes, Behnke, Brandimarte, Dauzere, Fattahi, Hurink, Kacem.*

Table 1. Average deviation of the lower bounds from the best known solutions

	Instances	Problem size	Deviation
Jobshop (*ft*, la*, orb*, tai**)	133	36-2000	6.1%
Flexible jobshop ([1])	330	50-500	5.9%
Flexible jobshop with setup times (*SDST-HUdata*)	20	50-100	15.0%
Openshop (*tai**)	60	16-400	0.7%
RCPSP (*j30, j60, j90, j120, RG300*)	2520	30-300	4.6%

5 Conclusion and future work

The lower bounds described in this article are available in Hexaly Optimizer since version 12.0. Although the approach proposed in this paper focuses on minimizing the makespan, additional lower bounding techniques have also been implemented for different classical scheduling objectives like the (weighted) sum of lateness or tardiness (Petit-Jean Genat and Laborie 2024). These techniques exploit well-known polynomial algorithms for single-machine problems[2].

The planned extensions aim to improve these fast bounds in the following directions :

– Extend the linear relaxation in particular when interval presence and length are involved in the objective or in side constraints.
– Strengthen constraint propagation (for instance using edge-finding algorithms).
– *Improve the management of the interactions between the single-machine relaxations.*

Another idea is to exploit the by-products of the different components to guide the search toward good primal solutions. These by-products being:

– The solution of the linear relaxation.
– The domain of the variables after the cut $h \leq h_{LB}$ in the dichotomy on bounds.
– *The solutions of the single-machine relaxations.*

References

P. Baptiste, P. Laborie, C. Le Pape and W. Nuijten, 2006, “Constraint-Based Scheduling and Planning", *Handbook of Constraint Programming*, Chapt. 22, pp.761-799.

P. Baptiste, N. Bonifas, 2018, “Redundant cumulative constraints to compute preemptive bounds", *Discrete Applied Mathematics*, Vol. 234, pp. 68-177.

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

P. Laborie, 2003, “Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results", *Artificial Intelligence*, Vol. 143, No 2, pp. 151-188.

P. Laborie, J. Rogerie, P. Shaw and P. Vilím, 2009, “Reasoning with Conditional Time-Intervals. Part II: An Algebraical Model for Resources", *Proc. of the 22nd International Florida Artificial Intelligence Research Society Conference, FLAIRS-09.*

Hexaly, Example tour, `https://www.hexaly.com/docs/last/exampletour`, Accessed: 2024-02-20.

L. Petit-Jean Genat and P. Laborie, “Bornes rapides dans Hexaly Optimizer basées sur les problèmes d'ordonnancement à une machine", *Proc. ROADEF 2024.*

M. Vanhoucke, J. Coelho and J. Batselier. 2016, “An overview of project data for integrated project management and control", *Journal of Modern Project Management*, Vol. 3, No 2, pp. 6-21. `http://solutionsupdate.ugent.be/dataset/rg300`, Accessed: 2024-02-20.

[2] This work is beyond the scope of the present paper. Future work related to these techniques appears in *italic.*

PyJobShop: Solving machine scheduling problems with constraint programming in Python

Leon Lan[1], Joost Berkhout[1]

Vrije Universiteit Amsterdam, The Netherlands
`{l.lan,joost.berkhout}@vu.nl`

Keywords: scheduling, flexible job shop problem, constraint programming, benchmark library, software

1 Introduction

Machine scheduling has a long history in the field of operations research (Potts & Strusevich 2009). While many great achievements have been made, there have been critical voices about scheduling research being too far from real-world scheduling needs (McKay & Wiers 1999). Pinedo (2016) outlines a variety of factors that differ in practical scheduling settings compared to the ones studied in the literature, among which the dynamic and uncertain nature of real-world applications, but also the complexity and specificity of real-world problems that are hard to capture in models. The recent decades show that increasingly more complex scheduling models are being studied in isolation, however, Dauzère-Pérès, Ding, Shen & Tamssaouet (2024) conclude in their survey on flexible job shop problems (FJSP) that there is need for more ready-to-use approaches that can be used to address generic extensions to the FJSP. Most importantly, by making such an approach publicly available, researchers and practitioners can benchmark new and tailored approaches, and build on top of this approach to address even more complex problem variants.

The main contribution of our work is to provide an open-source and easy-to-use Python software implementation named PyJobShop for modeling and solving scheduling problems with constraint programming. Our contributions are further divided as follows:

1. We describe a general variant of the classical FJSP that encompasses many real-world constraints and objectives, building on the work of Dauzère-Pérès et al. (2024).
2. We implement a constraint programming (CP) model using IBM ILOG CP Optimizer (Laborie, Rogerie, Shaw & Vilím 2018), which is known to consistently outperform mixed-integer linear programming on benchmark instances of many scheduling problems (Naderi, Ruiz & Roshanaei 2023).
3. We maintain a library of benchmark instances and their best-known solutions for many scheduling problem variants, providing a stable reference benchmark library to support the development of better and tailored solution approaches.

The outline of this abstract is as follows. Section 2 describes the problem. In Section 3, the PyJobShop software package is introduced, and Section 4 introduces the benchmark instance library. Section 5 discusses future research.

2 Problem description

In this section, we describe the general variant of the FJSP. We begin by presenting the classical description of the FJSP in Subsection 2.1 and subsequently introduce extensions in Subsection 2.2.

2.1 The classical flexible job shop problem (FJSP)

The FJSP considers a set of jobs $\mathcal{J}$, a set of machines $\mathcal{M}$, and a set of operations $\mathcal{O}$. Each job $j \in \mathcal{J}$ consists of a set of operations $\mathcal{O}_j \subseteq \mathcal{O}$ that must be processed in sequence, and each operation $i \in \mathcal{O}$ can be processed on any machine in its subset $\mathcal{M}_i \subseteq \mathcal{M}$. Processing operation i on one of its eligible machines $m \in \mathcal{M}_i$ takes $p_{im} \geq 0$ time. Furthermore, it is assumed that the time starts at zero (without losing on generality), machines can only process one operation at a time, and once an operation has started processing, it must be completed without interruption. A solution to the FJSP determines for each operation to which machine it is assigned and when its processing starts and ends, i.e., the sequence in which the operations are processed for each machine is determined. The goal is to minimize the makespan.

2.2 General FJSP

In this section, we describe a general variant of the FJSP that PyJobShop currently supports. This general FJSP includes many constraints that can be found in real-world scheduling applications. First, we discuss the decision variables of our model, followed by constraints, and we end this section by discussing the objective functions.

Decision variables. For every job $j \in \mathcal{J}$, let $\overline{S}_j \geq 0$ denote the starting time and let $\overline{C}_j \geq \overline{S}_j$ denote the completion time. Similarly, for every operation $i \in \mathcal{O}$, let $S_i \geq 0$ denote the starting time and let $C_i \geq S_i$ denote the completion time. We denote the machine assignment of operation i by $A_i \in \mathcal{M}_i$. Furthermore, the job decision variables are related to their operation variables, namely $\overline{S}_j = \min_{i \in \mathcal{O}_j} S_i$ and $\overline{C}_j = \max_{i \in \mathcal{O}_j} C_i$.

Timing constraints. Timing constraints restrict the domain of timing variables based on the input data. For job $j \in \mathcal{J}$, commonly known timing constraints are release times $r_j \geq 0$ and deadlines $d_j \geq 0$. A release time is the earliest time that a job (and its operations) are available for processing, whereas a deadline is the latest time that a job should be completed. There is also a job due date, not to be confused with a deadline, which is the latest time by which a job must be completed before incurring lateness or tardiness penalties. One can define timing constraints on the operation timing variables as well, e.g., the earliest and latest time that an operation can start or end.

Arbitrary precedence graph. In the classical FJSP, it is assumed that the processing order of operations is sequential, meaning that the $(i+1)$-th operation can only start when the i-th operation of a job has finished processing. We consider here the case of arbitrary precedence graphs on all operations, where a directed acyclic graph dictates the processing order of the operations $\mathcal{O}$. Each arc (i,k) in the graph defines a precedence relationship for operation pair (i,k), which do not necessarily need to be part of the same job.

Generalized precedence relationships. Classically, a precedence relationship (i,k) means that operation k must start not earlier than the completion time of operation i, i.e., $C_i \leq S_k$. We consider here a more general form of precedence relationships, which restricts when one operation starts or ends in relation to another. Using the inequality operator $(\leq)$ and equality operator $(=)$, eight combinations can be defined that relate the starting and completion times of two operations. For brevity, we do not list them here. As an example, the precedence relation $C_i = S_k$ states that operation k starts the moment that operation i is completed, which can be used to model no-wait or blocking problems. Moreover, all precedence relationships can also be extended with a *delay* $l \geq 0$ on the left-hand side, e.g.,

$C_i + l = S_k$. These delays can be used to model cases with minimum or maximum time lags.

Machine connection graph. In practical applications with flow shop characteristics, it is common for machines to be physically connected such that only some machines are accessible from one another. The machine connection graph defines the interrelationships between machines. An arc (m, l) in this graph means that machine m is connected to machine l. This machine connection graph will be used in the next paragraph to define assignment relations.

Assignment relationships. Recall that A_i denotes the machine assignment of operation i. Similar to the precedence relationships, we can define constraints that relate the assignment decisions of two operations rather than the timing ones. The simplest assignment relations are to assign operations to the same machine ($A_i = A_k$) or to different machines ($A_i \neq A_k$). Another more common assignment relationship is a machine connection restriction in hybrid flow shops. For example, consider two operations i and k that represent the same physical batch and assume that i precedes k. When scheduling operation i on machine m, operation k should only be scheduled on machines that are reachable from machine m.

Sequence-dependent setup times. A common extension to many scheduling problems is the inclusion of sequence-dependent setup times. The parameter $st_{ikm} \geq 0$ represents the time that is incurred between the processing of operation i and operation k on machine m. The setup times are assumed to be anticipatory or non-anticipatory: anticipatory means that the setup can start before the corresponding operation is available at the machine (if the machine is idle), whereas non-anticipatory means that the setup can only start when the next operation has arrived at the machine.

Objectives. Objective functions are computed based on the job completion times. Common objective functions such as the makespan, total flow time, and total tardiness are all defined as linear combinations of the job completion times. We plan to support less common objective functions such as earliness, throughput, or sequence-dependent setup costs in later versions of PyJobShop.

3 Software implementation

Our software package is available at `https://github.com/PyJobShop/PyJobShop/`. The codebase follows good software engineering practices, such as version control and unit-testing, and provides a simple and generic user modeling interface as illustrated in Listing 1.1. This interface can be used to define and solve scheduling problems without specific knowledge of CP technology. In this example, we first define a `Model` object and then the jobs and machines. For each job, we define the set of corresponding operations and assign them to the corresponding job. Then, we define for each operation and machine pair the corresponding processing time, as well as the precedence relation for each subsequent operation pair. Finally, calling the `solve` method solves the defined scheduling problem.

4 Benchmark instance library

We are curating a library of benchmark instances that are sufficiently hard to solve (i.e., not solved to optimality within seconds by CP solvers), and we plan to use those instances to evaluate our models while also providing a stable benchmark library for other researchers to compare against. All instances are stored at `https://github.com/PyJobShop/Instances` in FJSPLIB format, and we have currently stored all known FJSP benchmark instances.

Code Listing 1.1. Using PyJobShop's modeling interface to solve a scheduling instance.

```
import random
from pyjobshop import Model

m = Model()
jobs = [m.add_job() for _ in range(4)]
machines = [m.add_machine() for _ in range(2)]

for job in jobs:
    operations = [m.add_operation(job=job) for _ in range(4)]

    for op in operations:
        for machine in machines:
            duration = random.randint(1, 10)
            m.add_processing_time(machine, op, duration)

    for idx in range(len(operations) - 1):
        op1, op2 = operations[idx], operations[idx + 1]
        m.add_timing_precedence(op1, op2)

result = m.solve()
```

5 Future work

Our research is currently being actively developed. The first step is to extend the model from Section 2, for example to settings with multi-resource considerations or multiple factories (distributed FJSP). The second step is to implement a CP model with Google OR-Tools (Perron & Furnon 2023), which is a free and open-source solver, making our software more accessible. Finally, we are validating the user interface of PyJobShop by implementing a large-scale scheduling problem for a Dutch compound feed manufacturer.

References

Dauzère-Pérès, S., Ding, J., Shen, L. & Tamssaouet, K. (2024). The flexible job shop scheduling problem: A review, *European Journal of Operational Research* **314**(2): 409–432.

Laborie, P., Rogerie, J., Shaw, P. & Vilím, P. (2018). IBM ILOG CP optimizer for scheduling, *Constraints* **23**(2): 210–250.

McKay, K. N. & Wiers, V. C. (1999). Unifying the theory and practice of production scheduling, *Journal of Manufacturing Systems* **18**(4): 241–255.

Naderi, B., Ruiz, R. & Roshanaei, V. (2023). Mixed-Integer Programming vs. Constraint Programming for Shop Scheduling Problems: New Results and Outlook, *INFORMS Journal on Computing* **35**(4): 817–843.

Perron, L. & Furnon, V. (2023). OR-Tools. v9.7. https://developers.google.com/optimization/.

Pinedo, M. (2016). *Scheduling: theory, algorithms, and systems*, fifth edn, Springer.

Potts, C. N. & Strusevich, V. A. (2009). Fifty years of scheduling: a survey of milestones, *Journal of the Operational Research Society* **60**: S41–S68.

Scheduling Direct Deliveries in Make-to-Order Manufacturing with Intermediate Storage

Julia Lange[1] and Timo Gschwind[1]

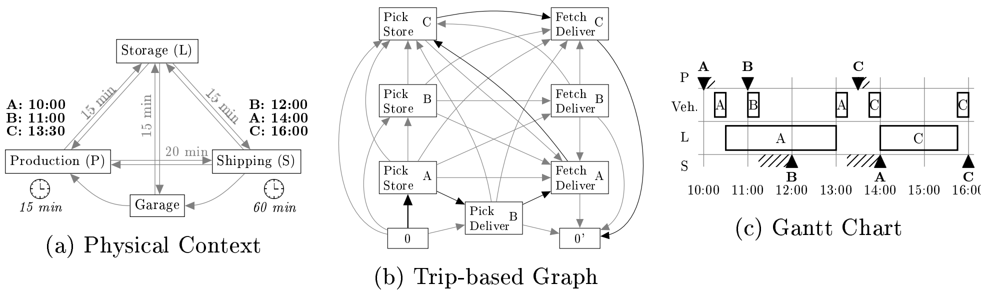
University of Kaiserslautern-Landau, Germany
julia.lange@rptu.de, timo.gschwind@rptu.de

Keywords: Intralogistics, Scheduling and Routing, Mixed-integer programming model.

1 Introduction

In make-to-order manufacturing, individual products are produced based on incoming orders with flexible lead time agreements. Considering the operational planning level, the production is completed at certain times on the manufacturing side, while customer orders require timely shipment on the distribution side. Inspired by a real-world case of customized floor board production, this work focuses on operating transport activities between the production (P) and the shipping (S) area with a given fleet of homogeneous vehicles located in an imaginary garage, see Figure 1a. Since completion times and shipping deadlines are not fully synchronized, see times for products A, B and C in Figure 1a, and storage capacity in both areas is temporally limited, here by 15 and 60 min, intermediate storage places (L) are available. Relevant travel links and times are indicated by arcs in Figure 1a.

The intralogistics planning problem involves a set of customer orders to be delivered to S before a given due date and a set of production orders to be picked up in P after their ready times. Every production order is uniquely paired with one customer order by an individual product. P and S feature a limited allowed order waiting time. Orders can be fulfilled either (i) by one vehicle trip from P to S if the corresponding product is available just in time or (ii) by two trips from P to a place in L and from there to S. Thus, a warehousing decision needs to be made for each product and if so, the concrete storage place must be selected. Order volumes are given in a way so that the capacity of each vehicle and each storage place is one order and no consolidation is possible. Therefore, each trip is a direct shipment including pickup, travel and delivery with a given start location, end location, travel time, earliest starting and latest ending time. Following Emde and Zehtabian (2019) and Gschwind *et. al.* (2020), the problem can be rephrased into finding a sequence of trips for each vehicle, namely a tour, in a trip-based graph so that all orders are fulfilled and the total travel time of the vehicles is minimal. These works are extended by allowing intermediate storage, which leads to alternative trips with different processing times for



(a) Physical Context (b) Trip-based Graph (c) Gantt Chart

Fig. 1: Example of the DDSP-S

each order, and assignment and sequencing decisions at storage places. Figure 1b shows the trip-based graph for one existing vehicle, where the nodes represent seven trips and dummy start and end depots for the tour. Since every node has a trip-induced time window and products cannot be fetched before being stored, the arcs indicate feasible sequences.

Based on the trip-based graph representation, we present a mixed-integer programming (MIP) formulation for the *direct delivery scheduling problem with intermediate storage* (DDSP-S), which can be classified as NP-hard (see Emde and Zehtabian (2019)). A computational study using a general MIP solver yields methodological and managerial insights as well as conclusions on promising future research directions.

2 Mathematical Formulation

In the DDSP-S, a set of trips $i \in \mathcal{I}$ and a set of orders $o \in \mathcal{O}$ are defined, where a subset $\mathcal{I}(o)$ of effective trips exists for each order. For each trip, a time window $[a_i, b_i]$ is derived from ready times of production orders and due dates of customer orders, and a processing time p_i summarizes handling and traveling activities. In case two trips i and j refer to the same product being stored and fetched, the pair (i, j) is part of the set of related trip pairs $\mathcal{R}$. The trip-based graph is set up with node set $\mathcal{I} \cup \{0, 0'\}$, as indicated in Figure 1b, and arc set $\mathcal{A}$ containing feasible physical route arcs between pairs of trips i and j, and dummy start arcs $(0, i)$ and end arcs $(j, 0')$ for each vehicle's tour. Physical route arcs are weighted by their travel time $t_{i,j} > 0$, while dummy arcs feature $t_{i,j} = 0$.

All orders must be satisfied by using a given number of vehicles K and a set of storage places $l \in \mathcal{L}$. The goal is to find a sequence of trips i with starting times $T_i \geq 0$ for each vehicle so that all orders are satisfied and the total travel and processing time is minimized. In Figure 1b, the black arcs indicate a feasible tour for the case $K = 1$ and $\mid \mathcal{L} \mid= 1$, and Figure 1c shows the resulting schedule of transport and storage activities in a Gantt Chart.

In the mathematical formulation, sequences of trips are represented by binary decision variables $x_{i,j}$ taking a value of 1, if trip i is performed directly before trip j by the same vehicle, and 0 else. Pairs of fetching trips (from L to S) that are related to different customer orders but the same storage place are summarized in the set of competing trips $\mathcal{C}$. In case both trips of a pair are selected, and thus, products of different orders are stored at the same storage place, the binary variable $y_{i,j}$ determines a proper sequence of the fetching trips. $y_{i,j} = 1$ indicates that the product related to trip i precedes the product related to trip j at the storage place. In line with the results of Lange and Werner (2018) for a job shop problem, preliminary tests indicate that this type of sequencing variables is favorable against other modeling approaches when applying a general MIP solver. The DDSP-S can be described as follows:

$$\min \sum_{(i,j)\in\mathcal{A}} t_{i,j} \cdot x_{i,j} + \sum_{(i,j)\in\mathcal{A}:j\neq 0'} p_j \cdot x_{i,j} \tag{1}$$

s.t.

$$\sum_{(i,j)\in\delta^+(0)} x_{i,j} \leq K \tag{2}$$

$$\sum_{(i,j)\in\delta^+(i)} x_{i,j} - \sum_{(j,i)\in\delta^-(i)} x_{j,i} = 0 \qquad \forall\, i \in \mathcal{I} \tag{3}$$

$$\sum_{j\in\mathcal{I}(o)} \sum_{(i,j)\in\delta^-(j)} x_{i,j} = 1 \qquad \forall\, o \in \mathcal{O} \tag{4}$$

$$\sum_{(k,i)\in\delta^-(i)} x_{k,i} - \sum_{(k,j)\in\delta^-(j)} x_{k,j} = 0 \qquad \forall\, (i,j) \in \mathcal{R} \tag{5}$$

$$T_i + p_i + t_{i,j} + M(x_{i,j} - 1) \leq T_j \qquad \forall\,(i,j) \in \mathcal{A} : i \neq 0, j \neq 0' \tag{6}$$

$$T_i + p_i + t^{\text{store}} \leq T_j \qquad \forall\,(i,j) \in \mathcal{R} \tag{7}$$

$$a_i \leq T_i \leq b_i \qquad \forall\, i \in \mathcal{I} \tag{8}$$

$$\sum_{(k,i)\in\delta^-(i)} x_{k,i} + \sum_{(k,j)\in\delta^-(j)} x_{k,j} \leq y_{i,j} + y_{j,i} + 1 \qquad \forall\,(i,j) \in \mathcal{C} : i < j \tag{9}$$

$$\big(|\mathcal{I}^{\text{out}}(l)| - 1\big) \cdot \sum_{(k,i)\in\delta^-(i)} x_{k,i} \geq \sum_{(i,j)\in\mathcal{C}:i<j} (y_{i,j} + y_{j,i}) \quad \forall\, i \in \mathcal{I}^{\text{out}}(l) : \exists\,(i,j) \in \mathcal{C}, l \in \mathcal{L} \tag{10}$$

$$T_i - p_{s(j)} + M(y_{i,j} - 1) \leq T_{s(j)} \qquad \forall\,(i,j) \in \mathcal{C} \tag{11}$$

$$x_{i,j} \in \{0,1\} \qquad \forall\,(i,j) \in \mathcal{A} \tag{12}$$

$$y_{i,j} \in \{0,1\} \qquad \forall\,(i,j) \in \mathcal{C} \tag{13}$$

The objective function (1) minimizes the sum of the total inter-trip travel time of all vehicles and the total processing time of all selected trips. Constraint (2) restricts the number of vehicles, while flow conservation Constraints (3) assure a proper construction of the tours. Oder fulfillment is guaranteed by Constraints (4). Constraints (5) make sure that both, store and fetch, are executed whenever a product is selected for storing. Constraints (6) determine a feasible starting time sequence for trips operated by the same vehicle and eliminate subtours. For all related store and fetch trips, Constraints (7) assure a proper starting time sequence additionally accounting for the minimum reasonable storage time t^{store}. The start time window for each trip is implemented by Constraints (8). Constraints (9) guarantee a unique sequence for each two fetch trips competing for the same storage place, i.e. $(i,j) \in \mathcal{C}$, while Constraints (10) assure that no sequence is determined for fetch trips not operated. Here, $\mathcal{I}^{\text{out}}(l)$ denotes the set of all fetch trips related to storage place l. The starting time sequence of competing fetch trips is adjusted according to the sequencing decision by Constraints (11), where $s(j)$ describes the related store trip for a given fetch trip j. Constraints (12) and (13) define the domain of the corresponding decision variables.

3 Computational Study

Computational experiments are conducted on randomly generated instances with 50 and 100 orders based on a shop floor layout as shown in Figure 1a. The planning horizon covers six hours, and the waiting time limits are 15 and 60 min for P and S. The minimum storage time t^{store} is set to 30 min. The production order ready times are uniformly distributed over the first 2 h of the planning horizon, while the customer order due dates are spread over the remaining 4 h. The number of vehicles K and the number of storage places $|\mathcal{L}|$ are varied in two parameter settings each. K is set to 10 for 50 orders and to 20 for 100 orders, while an increase by 30% is considered. $|\mathcal{L}|$ is varied between 5 and 6 for 50 orders and between 7 and 8 for 100 orders. 5 instances are generated for each of the eight combinations of parameter values. Preliminary tests have shown that such values avoid instance infeasibility caused by unrealizable time restrictions, and insufficient numbers of vehicles or storage places. The solution method is implemented in Python 3.8. The academic version of Gurobi 10.0 is used as a solver with 2.5% MIP gap tolerance. The numerical tests are conducted on a standard laptop with 8 GB RAM and Intel Core i5 with 1.6 GHz.

Table 1 summarizes the results for all parameter settings, where an increased numbers of vehicles and storage places is indicated by '+'. For both instance sizes, the average number of nodes and arcs of the trip-based graph, the average runtime to optimality (in seconds), and the average optimal objective function value (in minutes) are given.

Table 1: Computational Results

	50 Orders				**100 Orders**			
Setting	Nodes	Arcs	Time	Obj	Nodes	Arcs	Time	Obj
$K\ \ \lvert\mathcal{L}\rvert$	345.6	68,948.6	25.2	1848.0	945.2	518,646.0	949.1	3,692.8
$K{+}\ \lvert\mathcal{L}\rvert$	354.8	72,780.0	30.0	1780.6	930.4	501,972.4	475.7	3,570.8
$K\ \ \lvert\mathcal{L}\rvert{+}$	391.8	88,063.6	28.2	1832.4	1058.0	655,852.8	626.0	3,689.2
$K{+}\ \lvert\mathcal{L}\rvert{+}$	395.0	89,922.0	28.5	1770.0	1014.0	597,085.6	429.5	3,556.0

Since the number of vehicles is only used as a parameter in Constraint (2) of the model, it has no effect on the size of the trip-based graph. In contrast, a higher number of storage places leads to more trips and increases the number of nodes and arcs. However, this does not lead to a systematic increase in runtimes. For the smaller instances, no significant difference in runtime among instances with different parameter settings can be observed. For larger instances, the planning flexibility gained a higher number of vehicles and storage places even decreases the average runtimes, while changes in the number of vehicles show a higher effect. Regarding the solving behavior, it should be noted that the root LP bound differs by at most 2.5% from the optimal objective function value for all instances, and the optimal solution is always the first feasible solution found.

From a managerial perspective, it can be seen that additional storage spaces do not lead to a significant reduction in the total travel and processing times of the vehicles, while a larger fleet decreases the objective function value. However, note that fixed costs of vehicles and congestion effects are not considered here. Regarding the applicability of the model and a general solver in practice, the average runtimes of less than 30 s for smaller instances might be appropriate for planning on the operational level, while the average runtimes of more than 10 min for instances with 100 orders are not reasonable.

4 Conclusion

The DDSP-S can be described by a MIP formulation based on a trip-based graph. First numerical results applying a general MIP solver indicate that computation times need to be reduced for larger instances and practical applications, while increasing planning flexibility by adding vehicles and storage places seems to work in favor of this. In our ongoing work, we conduct a broader analysis on effects of instance parameters to gain deeper methodological and managerial insights. In the future, extending the problem to fit make-to-stock environments with long-term storage options and multiple orders with the same product seems as interesting and relevant as developing construction and improvement heuristics to meet practical requirements on operational planning.

References

Emde S., S. Zehtabian, 2019, “Scheduling direct deliveries with time windows to minimise truck fleet size and customer waiting times", *Int. J. Prod. Res.*, Vol. 57(5), pp. 1315-1330.

Gschwind T., S. Irnich, C. Tilk and S. Emde, 2020, “Branch-cut-and-price for scheduling deliveries with time windows in a direct shipping network", *J. Sched.*, Vol. 23, pp. 363-377.

Lange J., F. Werner, 2018, “Approaches to modeling train scheduling problems as job-shop problems with blocking constraints", *J. Sched.*, Vol. 21, pp. 191-207.

Selecting the Right Product Development Methodology for New Product Development Projects

Itai Lishner[1] and Avraham Shtub[1]

[1]Faculty of Data and Decision Sciences, Technion—Israel Institute of Technology, Haifa, Israel
`itailishner@hotmail.com, shtub@technion.ac.il`

Keywords: : methodologies, project management, modeling and simulation, system dynamics, agile, waterfall.

1 Introduction

Various methodologies exist in product development. Certain methodologies may be better suited for specific types of projects compared to others, as there is no universal approach that applies to all projects. When managing a New Product Development (NPD) project, a crucial strategic decision revolves around selecting the appropriate development methodology for the team to adhere to. This choice carries significant implications; its impacting resource requirements, the quality of the deliverables, the overall duration of the project, and even the project scope. The selection of a methodology for the development team is heavily influenced by the organization's culture and the team's past experiences. Consequently, the chosen methodology can also determine the specific team that will be assigned to the project. It is challenging to accurately forecast the impact of the development methodologies on project duration, since there is a lack of scientific methods or dedicated tools to facilitate such decision-making (Andrei *et. al.* 2019). The emergence of agile methodologies can be attributed to the absence of a reliable method for accurately estimating product development duration and scope. Agile approaches (Agile Manifesto 2001) diverge from the traditional mindset of predicting strict deadlines and instead advocate for initiating the development process without extensive upfront planning or knowledge of the eventual outcomes. However, despite this flexibility, managers and organizations still require some level of predictability to construct their organizational roadmap, allocate budgets and resources, plan for growth, and more. This demand for predictability presents a challenge for those implementing agile practices within organizations (VersionOne 2019, Vijayasarathy and Turk 2008).

Accurately estimating project durations is highly challenging task, and existing popular models like CPM (Critical Path Method) and PERT (Program Evaluation and Review Technique) often fall short in providing reliable estimation capabilities for project duration. These models have a fundamental weakness in that they do not adequately consider the specific product development methodology employed, the uncertainties associated with task durations, resource availability, and scope changes, which are prevalent in many projects (Lishner and Shtub 2022).

Several studies have focused on exploring the relationship between the strategy and methodologies of NPD management and their impact on the project's outcome (Joslin and Müller 2015, Papke-Shields and Boyer-Wright 2017, Ciric *et. al.* 2021, Ahmed *et. al.* 2022, Joslin and Müller 2016, Ika and Pinto 2022). However, most of these studies have primarily relied on theoretical constructs rather than real-world project data. As a consequence, the existing models for selecting development methodologies and strategies suffer from poor validation based on empirical evidence. The current limitations of available tools, impede the understanding of how various product development methodologies impact project duration. This understanding is crucial for formulating an effective project management strategy.

This paper addresses a simulation model designed to test various product development strategies. The research methodology focuses on enhancing a validated model of the project life cycle, adapting and updating it to align with advanced development methodologies. Subsequently, a verification and validation process were conducted using data from real projects.

By employing this model, it becomes possible to comprehend the anticipated effects of different strategies on project timelines. Project managers can utilize this tool to determine the most effective strategy for executing a specific project, gaining a more comprehensive understanding of the tradeoffs associated with different methodologies.

2 Model Description

The proposed NPD model was developed using the Vensim PLE system dynamics modeling tool. The model core is built upon the published model of Ford and Sterman (1998) with an additional view of NPD project management studies in which characteristics of NPD projects from different methodologies have been examined. Ford and Sterman based their model on Pugh-Roberts' well-known models and utilized objects from the Program Management Modelling System (PMMS) model. Their model's core is centered on the "Rework Cycle" paradigm, which was first published by Cooper (1980). This paradigm identifies undiscovered errors in the current work, which leads to more work in correcting these errors. As more work is done, more errors are produced, leading to a recursive nature of the project where rework generates more rework, resulting in the creation of more tasks that need to be done to complete the project or a phase of the project. The following improvements were implemented to the basic model:

2.1 Stochastic Variables Instead of Deterministic Variables

The original model uses deterministic variables for setting the rate of task execution, for example: "average completion duration" and "resource constraint". In the proposed NPD model, more variables were added, such as: "no. of resources", "resource show-up probability", "task mean duration", and more. In addition, the deterministic variables such as "average task duration" and more were replaced with stochastic variables according to a distribution function allowing one to control the mean and the std. deviation of the function. Moreover, the "resource constraint" is now relying on the "resource allocation" and "show-up probability", which is a random variable that represents the probability of a resource being available to perform the tasks, since not all resource types can have an assured 100% availability rate during the project life cycle.

2.2 Risk Management

Any NPD project has risks associated with it. There is a probability of risk events happening that may affect the duration of some of the project's tasks and cause a delay in the project's scheduling. The proposed NPD model includes a function of "Task Duration Risks" which contains the average probability that tasks have of taking longer due to the expected risk; this probability is being considered with each task duration calculation. The Risk function impacts the duration of tasks, utilizing a Normal distribution with a standard deviation determined by the project risk assessment. For each task associated with risks, the probability of an extension in its duration is calculated. This probability is also linked to the project's progress and the number of tasks completed successfully. This is based on the assumption that as the project progresses, risks are typically reduced until the project concludes and finally all risks are eliminated.

2.3 Dynamic Allocation of Resources

In the original model, the allocation of resources is fixed, meaning that if a resource is not being used due to lack of tasks in the backlog, the resource is not exploited by other tasks as may happen in reality. In the proposed NPD model, the option for automatic resource allocation was added and it can be configured according to the project definition. For example, if the "Tasks to Change Backlog" is empty, meaning there are no tasks to change, the resources which were allocated to that task's backlog can be reallocated to help complete tasks from another backlog, such as the "Project Tasks Backlog". The model allows one to choose the automatic allocation rules.

2.4 Apply Different NPD Methodology

The original model did not consider the methodology employed in the project. In the proposed model, one can choose among waterfall, agile, and hybrid methodologies. The Gates mode (Waterfall) in the model permits work solely on the tasks available before approval of the next gate. Approval of a gate occurs when all tasks associated with it are completed, tested, and approved by the testing team.

The Iterations methodology (agile and spiral) is based on prior studies modeling agile methodology (Van Oorschot *et. al.* 2009, Glaiel *et. al.* 2014). The Iterations mode utilizes parameters like Sprint Length to determine the duration of each sprint. The overhead of sprint planning is set by the Planning Length parameter, and another influential factor in Iteration mode is Schedule Pressure (Van Oorschot *et. al.* 2009). This factor dynamically alters the average productivity of the team as the end of the iteration approaches, impacting both the quality of tasks performed and the likelihood of identifying errors.

Recognizing that many organizations don't strictly adhere to agile or waterfall methodologies, the necessity to model a hybrid mode becomes evident. The hybrid mode retains the gate options while enabling work on short or long iterations between gates. Parameters such as gate types, length, and iteration scope, along with other relevant factors for both Gates and Iterations modes, are available to tailor the model to suit the unique hybrid approach of each project and organization.

3 Results and Conclusions

To validate the model, we conducted tests on two real multidisciplinary projects that employed waterfall and agile methodologies. The results demonstrate alignment between the model's predictions and the actual project implementation. By employing stochastic parameters and Monte Carlo simulation, the proposed model enables multiple simulation runs, generating histograms of project results. This facilitates robustness testing of strategies and provides insights into the probability of project completion within specific timeframes. The model's application of Monte Carlo simulation has potential for broader utilization and future research.

Despite its utility, the model has limitations. The averaging approach enhances understanding of the project's end result but may yield less accurate results during execution. Additionally, project strategy cannot be detached from organizational and team capabilities, meaning not all methodologies are universally applicable. Project managers must consider the project's limitations, team dynamics, and organizational context when selecting and adapting methodologies.

To effectively utilize our model, we recommend a systematic workflow. First, select a methodology based on the project's constraints. Then, perform a Monte Carlo analysis, setting boundaries for fixed parameters. Choose the best-performing strategy based on the analysis results. Conduct sensitivity analysis to assess project robustness. If results are unsatisfactory, adjust the project plan, strategy, or parameters until optimal results are achieved. Utilize Monte Carlo simulation and statistical measurements to assess the probability of achieving better project performance with each methodology, enhancing decision-making.

In conclusion, the presented model provides a valuable framework for choosing and evaluating NPD project strategies. Its ability to simulate scenarios and align with real-world implementations demonstrates its effectiveness. However, it is essential to acknowledge the limitations associated with the averaging assumption and contextual factors influencing project strategy.

References

Agile Manifesto: Manifesto for Agile Software Development, 2001, [online] access 5 January 2024, https://agilemanifesto.org/.

Ahmed, R., Shaheen, S. and Philbin, S. P., 2022, "The role of big data analytics and decision-making in achieving project success", *Journal of Engineering and Technology Management*, 65, 101697.

Andrei, B. A., Casu-Pop, A. C., Gheorghe, S. C. and Boiangiu, C. A., 2019, "A study on using Waterfall and Agile methods in software project management", *Journal of Information Systems and Operations Management*, 125-135.

Cooper, K. G., 1980, "Naval ship production: A claim settled and a framework built", *Interfaces*, 20-36.

Ciric, D., Delic, M., Lalic, B., Gracanin, D. and Lolic, T., 2021, "Exploring the link between project management approach and project success dimensions: A structural model approach", *Advances in Production Engineering and Management*, 16(1).

Ford, D. N., Sterman, J. D., 1998, "Dynamic modeling of product development processes", *System Dynamics Review: The Journal of the System Dynamics Society*, 14(1), 31-68.

Glaiel, F. S., Moulton, A. and Madnick, S. E. , 2014, "Agile project dynamics: A system dynamics investigation of agile software development methods."

Ika, L. A., Pinto, J. K., 2022, "The re-meaning of project success: Updating and recalibrating for a modern project management", *International Journal of Project Management*, 40(7), 835-848.

Joslin, R., Müller, R., 2015, "Relationships between a project management methodology and project success in different project governance contexts", *International Journal of Project Management*, 33(6), 1377-1392.

Joslin, R., Müller, R., 2016, "The relationship between project governance and project success", *International Journal of Project Management*, 34(4), 613-626.

Lishner, I., Shtub, A., 2022, "Using an artificial neural network for improving the prediction of project duration", *Mathematics*, 10(22), 4189.

Papke-Shields, K. E., Boyer-Wright, K. M., "Strategic planning characteristics applied to project management", 2017 *International Journal of Project Management*, 35(2), 169-179.

Summers, G. J., Scherpereel, C. M., "Flawed decision models and flexibility in product development", 2023 *Journal of Engineering and Technology Management*, 67, 101728.

Tignor, W., 2009, "Agile ProjecProc.", *Proc. Int'l Conf. of the System Dynamics Society.*

Van Oorschot, K. E., Sengupta, K. and van Wassenhove, L., 2009, "Dynamics of Agile software development", *Proc. International Conf. of the System Dynamics Society.*

VersionOne, 2019, "13th annual state of Agile report", retrieved from: https://www.stateofAgile.com/#ufh-i-521251909-13th-annual-state-of-Agile-report/473508

Vijayasarathy, L. E. O. R., Turk, D., 2002, "Agile software development: A survey of early adopters", *Journal of Information Technology Management*, 19(2), 1-8.

A local search tree heuristic approach for the stochastic 2-machine flow shop scheduling problem

Lei Liu[1], Marcello Urgo[2]

[1] Nottingham University Business School China, University of Nottingham, Ningbo, China
`lei.liu@nottingham.edu.cn`
[2] Department of Mechanical Engineering, Politecnico di Milano, Milano, Italy
`marcello.urgo@polimi.it`

Keywords: Stochastic scheduling, Flow shop, Local search tree heuristic.

1 Introduction and problem statement

In the manufacturing sector, effective planning and scheduling are crucial due to the sector's inherent unpredictability. This unpredictability is characterised by incomplete information and a spectrum of unforeseen events, including variability in production times, the emergence of urgent tasks such as expedited orders, rework demands, cancellation of orders, breakdowns of machinery, and material deficits.

This paper delves into the intricacies of scheduling for remanufacturing operations organised in a two-machine permutation flow shop, wherein components are processed in batches. One of the key challenges identified is the unpredictability of batch sizes, as some parts may be deemed non-repairable and consequently need replacement with new ones. Furthermore, the processing time required for these batches is also subject to uncertainty, as the extent of damage to the parts dictates the duration of the repair process. Thus, more severe damage leads to longer processing times. This paper presents a framework that employs stochastic modelling of processing times of batches of remanufacturing parts, represented by probability distributions, to facilitate more robust scheduling under uncertainty. Liu and Urgo (2023a) addressed the same problem using an exact branch-and-bound algorithm, while Liu and Urgo (2023b) tackled the revised version of the problem employing both exact method and two heuristic approaches, specifically the Iterated Greedy algorithm and the NEH heuristic, which have been recognised as the most promising heuristics for deterministic flow shop scheduling problems.

The described problem can be modelled through a stochastic two-machine permutation flow shop scheduling problem. 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. Phase-type distributions are used to model stochastic processing times due to their capability to approximate general distributions. The objective function under consideration seeks to minimise the Value-at-Risk (VaR) of the makespan, thereby aiming to derive a robust scheduling solution. Specifically, the proposed schedule is designed to ensure that the likelihood of the makespan exceeding the VaR is confined to a predefined risk level, denoted as α, which reflects the decision-maker's risk aversion. This approach guarantees that the probability of the makespan surpassing the VaR threshold is no greater than $1 - \alpha$, thus aligning the solution with the decision-maker's specified level of risk tolerance.

2 Local search tree approach

In this paper, we proposed a local tree search approach with a two-stage procedure (Della Croce *et. al.* 2014) for the described stochastic two-machine flow shop scheduling problem: a first heuristic procedure is applied to the problem for generating a starting solution, and then a post-processing procedure is applied exploiting of a branch-and-bound analysis and a rolling horizon local search procedure.

The initial schedule can be obtained by arranging all the jobs according to the decreasing order of $(\frac{1}{E(j_1)}$- $\frac{1}{E(j_2)})$ 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 (Liu and Urgo 2023a).

For the second-stage procedure, a rolling horizon approach together with a *window reoptimisation* problem is exploited. A window size parameter h is set, and then, grounding on the first-stage heuristic schedule $\boldsymbol{x}_{ini}$, the first h jobs will be sequenced as a subschedule $\boldsymbol{x}_{s0}$. Hence, a branch-and-bound algorithm (Liu and Urgo 2023a) is exploited to find the optimal solution for $\boldsymbol{x}_{s0}$, denoted as $\boldsymbol{x}^*_{s0}$. Given that the position of the first job in $\boldsymbol{x}^*_{s0}$ is fixed and will be kept in the final solution, the next h jobs in the initial schedule $\boldsymbol{x}_{ini}$ are considered. The branch-and-bound approach is used to identify the optimal sequence for these jobs, generating a new subschedule $\boldsymbol{x}_{s1}$. Hence, the first one of these will be fixed, and a new subproblem with h jobs will be considered until a first-stage heuristic schedule is reached.

The outline of the proposed Local search tree algorithm is provided in Algorithm 1, and an example instance with 7 jobs and size parameter equals to 4, is presented in Fig. 1.

```
Algorithm 1: Local search tree algorithm
Input: initial schedule $\boldsymbol{x}_{ini}$, size $h$, final schedule $\boldsymbol{x}^* = \varnothing$, $i$=0
Step 1: Choose the first $h$ jobs of $\boldsymbol{x}_{ini}$ and taken as subschedule $\boldsymbol{x}_{s0}$.
Step 2: The branch-and-bound algorithm is exploited to find the optimal solution of $\boldsymbol{x}_{si}$,
  say $\boldsymbol{x}^*_{si}$; $i = i + 1$.
Step 3: Put the first location job $k$ of $\boldsymbol{x}^*_{si}$ into $\boldsymbol{x}^*$.
Step 4: Pop out the first location job $k$ of $\boldsymbol{x}^*_{si}$, and take the $h + i$ job of $\boldsymbol{x}_{ini}$ and put it
  into the last location of $\boldsymbol{x}^*_{si}$, a new subschedule with $h$ jobs is generated.
Step 5: Run step 2 and step 3, until the last job of $\boldsymbol{x}_1$ is put into the final schedule,
  append the last optimised subschedule into the final schedule $\boldsymbol{x}^*$.
```

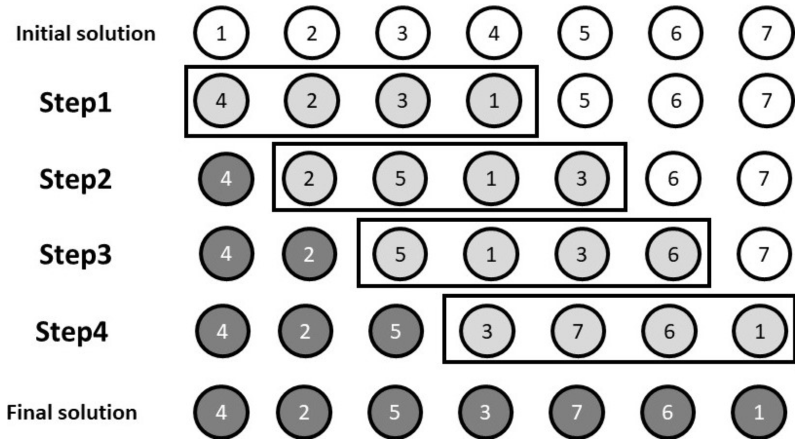


Fig. 1. Example of Local search tree algorithm

The branch-and-bound algorithm used for the window re-optimisation is based on a Markov activity network model, as proposed in(Liu and Urgo 2023a). Both full and partial schedules are formalised as an AoA network, supporting the definition of a continuous-time Markov chain (Fig. 2) modelling the processing of the jobs, whose time to absorption corresponds to the makespan of the schedule. This enables the calculation of the VaR by Eq. 1 where β and T are the vector and matrix of the CTMC, α is the considered risk level, and ζ is the VaR_α value to be estimated.

$$1 - \alpha = 1 - \beta e^{\zeta * T}\mathbf{1} \tag{1}$$

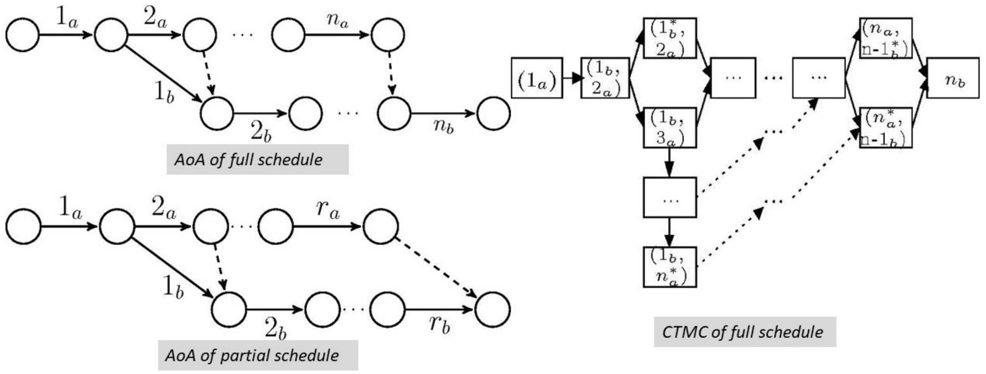


Fig. 2. AoA of full and partial schedule and CTMC figure

3 Numerical experiments

A set of test instances has been generated considering n = 10, 20, 30 and 50 jobs. Processing times are modelled through phase-type distributions, randomly generated by the BuTools library by providing values for the mean and number of phases. The mean value of the random processing times is adapted from the deterministic processing times of the Taillard dataset, specifically from the 50-job and 5-machine instances, by considering two machines only. The number of phases is randomly sampled between 1 and 4, and risk level $\alpha = 20\%$ is considered for the optimisation. The window size parameter, h, is selected to be 6 to strike a trade-off, ensuring that the branch-and-bound algorithm can be executed in a timely manner for the subschedule of this size, while still being large enough to optimise the subschedule effectively.

The experimental outcomes are presented in Table 1 and Fig. 3, which illustrate the efficacy of the local search tree heuristic algorithm. These results highlight the enhancements made from the initial solution, the discrepancy from the optimal solution for instances with 10 and 20 jobs (or the global lower bound for instances with 30 and 50 jobs), and the CPU time taken to arrive at the solution, respectively.

Grounding on the experiments, the proposed algorithm is able to find the final schedule in about one minute. With respect to the performance, the local search tree heuristic can improve the initial heuristic solution by an average of 7.7% and with a gap of 2.2% towards the optimal solution or global lower bound.

Numerical experiments indicate that the proposed local search tree heuristic algorithm is highly effective in solving the stochastic two-machine flow shop scheduling problem. Critical factors contributing to the success of this algorithm include both the window size parameter, denoted as (h), and the quality of the initial schedule. Future research could profitably focus on the optimal selection of the parameter (h) and the development of initial heuristic schedules. Moreover, the integration of advanced local search techniques may further improve the solution quality of the proposed approach. Additionally, the proposed method provides a flexible framework that could be adapted to various stochastic scheduling problems, e.g., m-machine flow shop scheduling problem, representing a valuable direction for future research.

Table 1. Results

Job No.	Δ% *vs* initial			Δ% *vs* optimal/GLB			solution time		
	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max
10	12.4	0.29	27.3	0.4	0	3.0	3.4	3	5
20	6.5	0	13.5	2.1	0	8.6	41.8	18	90
30	7.2	2.0	14.5	3.1	0.9	5.7	92.1	32	201
50	4.8	2.9	8.3	3.2	1.0	6.6	174.6	78	307
All	7.7	0	27.3	2.2	0	8.6	77.9	3	307

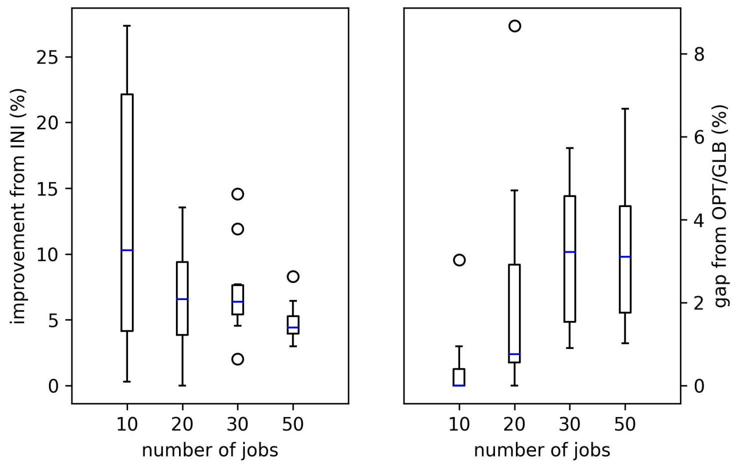


Fig. 3. Performance of local search tree heuristic algorithm

Acknowledgments

This research was partially funded by the H2020 MSCA European ITN DIGIMAN4.0 (Project ID: 814225).

References

Della Croce, F., Grosso, A. and Salassa, F., 2014, "A matheuristic approach for the two-machine total completion time flow shop problem", *Annals of Operations Research*, 213(1), pp.67-78.

Liu, L., Urgo, M., 2023a, "A branch-and-bound approach to minimise the value-at-risk of the makespan in a stochastic two-machine flow shop", *International Journal of Production Research*, pp.1-17.

Liu L., Urgo M., 2023b, "Robust scheduling in a two-machine re-entrant flow shop to minimise the value-at-risk of the makespan: branch-and-bound and heuristic algorithms based on Markovian activity networks and phase-type distributions", *Annals of Operations Research*, pp.1-24.

Solving the Stochastic Resource-Constrained Project Scheduling Problem with Flexible Resource Profiles using a Sample Average Approximation Approach

Ann-Kathrin Mendl and Julia Rieck

University of Hildesheim, Universitätsplatz 1, Germany,
Operations Research Group, Institute for Business Administration and Information Systems
`{mendl;rieck}@bwl.uni-hildesheim.de`

Keywords: Stochastic RCPSP, Flexible resources, Sample Average Approximation.

1 Introduction

In today's competitive environment with short product life cycles and high customer expectations, working in projects is a predominant form of work organization. As part of the planning process, which is carried out before project realization, individual activities of a project are scheduled according to an objective function and the available renewable resources (e.g., employees) are used efficiently to carry out the activities.

We consider projects represented as activity-on-node networks $N = (V, E)$. Node set $V = \{0, 1, \dots, n, n+1\}$ consists of real activities $i = 1, \dots, n$ and two fictitious activities, 0 and $n+1$, which represent the start and completion of the project. Set E contains arcs for all precedence constraints between activities. If the goal is to minimize the project duration, then the problem can be formulated as a deterministic *resource-constrained project scheduling problem* (RCPSP). This problem assume a constant duration $d_i \geq 0$ of the activities $i \in V$ and a constant resource requirement $r_{ik} \geq 0$ of resources $k \in K$ during the execution of activities; K is the set of all renewable resources. The RCPSP ignores uncertainties that lie in the future. In careful and structured planning, however, it should be assumed that uncertainties will arise, e.g., in the duration of activities or in the number of resources required over time. Otherwise, there may be considerable disruptions in the schedule and the solutions generated may be too restrictive for implementation and unsuitable for many practical applications (Hartmann and Briskorn 2010).

In order to integrate these uncertainties, there are also approaches in the literature. These include the RCPSP with stochastic aspects (in particular with stochastic activity durations) and the RCPSP with flexible resource allocation over time or with flexible resource profiles. In the latter case, the number of employees can be varied during the execution, thus leveraging flexibility potentials. As far as we know, there is no approach that offers both advantages of stochastic activity durations and those of flexible resource profiles, respectively. This leads to a relevant research gap, which is addressed in this contribution. Based on a chance-constrained formulation, a mathematical model for flexible and stochastic project scheduling problems is presented and evaluated in a performance analysis. Moreover, a suitable serial schedule generation scheme is introduced.

2 Stochastic and Flexible Resource-Constrained Project Scheduling

The problem under consideration is a stochastic RCPSP with flexible resource profiles (abbreviation: FSRCPSP). Therefore, it represents a combination of the *stochastic RCPSP* (SRCPSP) and the *flexible RCPSP* (FRCPSP). We developed the FSRCPSP model based on the SRCPSP by Möhring *et. al.* (1984) and the FRCPSP by Kolisch *et. al.* (2003). The following characteristics are important for the model formulation in Section 3:

– The SRCPSP takes into account that the durations of the activities are defined as random variables $\mathbf{d} = (\mathbf{d_0}, ..., \mathbf{d_{n+1}}) \sim \mathbf{P}$; $\mathbf{P}$ is a chosen probability distribution. The objective is to minimize the expected project duration (Möhring *et. al.* 1984).
– The FRCPSP considers a deterministic total workload for individual activities, which can, e. g., be specified in person hours. The workload $w_{ik} \geq 0$ of activity $i \in V$ for resource $k \in K$ can be distributed over time with varying resource requirements. This results in a different resource allocation from period to period and in decision-relevant activity durations that depend on the resource allocation (Naber and Kolisch 2014).

In what follows, the FSRCPSP is specified by a stochastic workload defined as a vector $\mathbf{w}$ of random variables. Moreover, the formulation requires a discretization of the time horizon $T = \{0, \dots, |T|\}$. Then, the workload results from the sum of resource requirements r_{ikt} over all periods $t = 0, \dots, |T|-1$ in the planning horizon. The flexible r_{ikt} can not be chosen arbitrarily, but must be adjusted within the predefined limits $[\underline{r}_{ik}, \overline{r}_{ik}]$ (Schramme 2014).

3 Solution Methodology and Mathematical Model Formulation

The FSRCPSP can be solved in two different ways. A first approach is to find a policy (i. e., a set of decision rules) that dynamically prescribes the start of certain activities at certain decision times (Ballestín and Leus 2009). Alternatively, a two-stage procedure can be used (Davari and Demeulemeester 2019). The first stage provides the generation of a proactive, robust *baseline schedule* S^B that is protected as much as possible against any uncertainties that may arise. In the second stage, the aim is to define a reactive strategy which is always used if a disruption occurs during project execution. The two schedules (proactive: S^B and reactive: S^R) are combined by using a robustness measure; e. g., a *confidence level* (CL). The CL measures the probability that each activity starts exactly at the start time of the equivalent activity in S^B (Lamas and Demeulemeester 2016).

Consequently, the problem of determining a baseline schedule S^B with a minimum project duration ($C_{\max}$) arises in such a way that all constraints of the FSRCPSP are fulfilled and a specified CL is satisfied. This problem can be formulated as a *chance-constrained* (CC) model. For this purpose, we choose the mixed-integer programming (MIP) models inspired by Möhring *et. al.* (1984) and Kolisch *et. al.* (2003) as a basis and integrate the uncertainties with chance-constraints such as Lamas and Demeulemeester (2016) for the SRCPSP. For the formulation, we introduce binary decision variables $x_{it} \in \{0, 1\}$ for activities $i \in V$ and time periods $t \in T$, which take the value 1 if i is started at time t. Moreover, let $(1 - \alpha)$ be the CL set in advance by the project managers. The precedence and the resource constraints are adjusted and formulated as follows:

$$(1-\alpha) \quad \leq Pr\left(\sum_{t\in T} t\, x_{jt} - \sum_{t\in T} t\, x_{it} \geq d_i,\ \forall \langle i,j\rangle \in E\right) \tag{1}$$

$$(1-\alpha) \quad \leq Pr\left(\sum_{i\in V} r_{ikt} \leq R_{kt},\ \forall k \in K, t = 0, \dots, |T|-1\right) \tag{2}$$

Constraints (1) and (2) ensure that a solution of the CC-FSRCPSP is only feasible if the probability $Pr(*)$ of satisfying all precedence constraints and resource capacities R_{kt} for resources $k \in K$ and each point in time $t \in T$ are greater than or equal to the defined CL.

Since the resulting MIP-model for determining the baseline schedule S^B can not be solved in a direct way, we have to reformulate the CC-FSRCPSP by using *sample average approximation* (SAA). Here, the uncertainty is considered by looking at scenarios $\pi \in \mathcal{S}$ in which w_{ik} is drawn according to the underlying probability distribution. Furthermore, let $r^{\pi}_{ikt} \geq 0$ and $d^{\pi}_i \geq 0$ be real-valued decision variables for the resource requirements and the activity durations, which are dependent on the scenario π. With $\overline{W}_i$ being the set of all possible start times of activity $i \in V$, we obtain the following constraints:

$$\sum_{t \in \overline{W}_j} tx_{jt} - \sum_{t \in \overline{W}_i} tx_{it} \geq d^{\pi}_i - (M \cdot \omega_{\pi}) \qquad \forall \langle i,j \rangle \in E, \pi \in \mathcal{S} \tag{3}$$

$$\sum_{i \in V} r^{\pi}_{ikt} - (M \cdot \nu_{\pi}) \leq R_{kt} \qquad \forall k \in K, t \in T, \pi \in \mathcal{S} \tag{4}$$

$$\omega_{\pi} + \nu_{\pi} \leq 2 \cdot \rho_{\pi} \qquad \forall \pi \in \mathcal{S} \tag{5}$$

$$\sum_{\pi \in \mathcal{S}} \rho_{\pi} \leq |\mathcal{S}| \cdot \varepsilon \tag{6}$$

Precedence constraints are ensured with Conditions (3). The binary decision variable $\omega_{\pi} \in \{0,1\}$ receives the value 1 if any precedence condition is violated in the corresponding scenario $\pi \in \mathcal{S}$. Conditions (4) belong to the resource constraints and the decision variables ν_{π} check whether the solution is feasible under scenario π with regard to the resource capacities. If more resource units are allocated to a scenario, $\nu_{\pi} \in \{0,1\}$ takes the value 1. Inequalities (5) and (6) count the number of violations of scenarios. As soon as the sum of binary decision variables $\rho_{\pi} \in \{0,1\}$ is greater than $(|\mathcal{S}| \cdot \epsilon)$, the observed solution is infeasible. The resulting SAA-FSRCPSP enables the minimization of the project duration $C_{\max}$ over all scenarios $\pi \in \mathcal{S}$ under consideration of a confidence level $(1-\epsilon)$.

Since the FSRCPSP is a generalization of the RCPSP, it is $\mathcal{NP}$-hard. Therefore, the use of exact solution methods is only possible for small instances with a low number of activities and scenarios. In order to solve larger problem instances, we have implemented a *serial schedule generation scheme* (SSGS) in the programming language C++ (see Section 4).

4 Serial Schedule Generation Scheme for the SAA-FSRCPSP

Starting from a sub-schedule that only contains activity 0, the other activities are scheduled one after the other in our SSGS. In each iteration, the set of activities that can be scheduled (the eligible set $\mathcal{E}$) is determined on the basis of the underlying network N.

An activity j with the highest priority is selected from set $\mathcal{E}$ according to a priority rule, e.g., earliest start time first (EST-rule). Then, the start time S_j for activity j has to be identified. To do this, we first calculate the earliest start time ES_j of activity j by considering the predecessor activities $i \in Pred(j)$. In particular, the calculation principle $ES_j = \max_{i \in Pred(j), \pi \in \mathcal{S}}(S_i + d^{\pi}_i)$ applies here. Consequently, all durations d^{π}_i of the different scenarios $\pi \in \mathcal{S}$ are explicitly included. In order to specify the durations d^{π}_i, the corresponding workloads w^{π}_{ik} are distributed over the time axis (e. g., evenly or taking into account the maximum available residual capacity) within the limits $\underline{r}_{ik}$ and $\overline{r}_{ik}$. Once the ES_j is calculated, the second step is to determine the time $t^* \geq ES_j$ at which a resource-feasible scheduling of j can take place. For this purpose, we do not select the scenario with the maximum workload, but we select a scenario $\pi \in \mathcal{S}$ with the corresponding workload w^{π}_{jk} for activity j. Accordingly, a roulette wheel selection is used, which is defined by the probability density of $\mathbf{w}$ for each activity. After selecting a workload, it is distributed over the time axis starting from ES_j within the limits $[\underline{r}_{ik}, \overline{r}_{ik}]$. The earliest point in time at which a feasible distribution of w^{π}_{ik} is possible is $t^* \geq ES_j$. We set $S_j = t^*$.

When all activities are scheduled, a vector of start times $S = (S_0, \ldots, S_{n+1})$ is created. The feasibility of this solution must be checked for all scenarios with regard to CL. As soon as a scenario is not able to fulfill the specified start times, a conflict arises. If the number of conflicts defined in Constraints (6) is not exceeded, the schedule is a proactive, robust and feasible schedule with regard to the predefined confidence level $(1-\epsilon)$.

5 Computational Results and Outlook

Based on the PSPLIB library provided by Kolisch and Sprecher (1996), we created 100 instances with $n = \{10, 30\}$ real activities. Thereby, flexible and stochastic characteristics were added to the instances, e. g., a set $\mathcal{S}$ of scenarios was generated using the beta(2,5) distribution from Lamas and Demeulemeester (2016) and resource limits $[\underline{r}_{ik}, \overline{r}_{ik}]$ were included for each activity. In the performance analysis, the quality of the SSGS was examined in comparison to optimal solutions achieved with GAMS 39.3 and CPLEX 22.1. Table 1 shows the average objective function values $\varnothing_{C_{\max}}$ (i. e., the project durations) of the instances, the number of optimally and feasibly solved instances, the average runtimes $\varnothing_{tcpn}$ of optimally solved instances as well as the average gap for all feasibly solved instances.

Table 1. Computational results ($\epsilon = 0.0$)

	$\varnothing_{C_{\max}}$	# optimal	$\varnothing_{tcpn}$	# feasible	$\varnothing_{\text{GAP}}$
Results achieved by GAMS and CPLEX					
$n = 10$	23.36	50	115.82 s.	0	0 %
$n = 30$	70.62	47	3660.52 s.	3	3.66 %
Results achieved by SSGS					
$n = 10$	25.18	19	1.43 s.	31	6.70 %
$n = 30$	72.66	13	2.12 s.	37	7.04 %

The results obtained show that the SSGS provides a good approximation of the average project duration. In addition, significantly shorter run times are recorded. The SSGS reliably provides a solution for instances with 30 activities in approximately 2 seconds. In order to improve the solution quality of the SSGS, the next step is to integrate it into a suitable metaheuristic. A population-based method is planned in the future.

Acknowledgments

This research is funded by the German Research Foundation (DFG), project no.: 520722132.

References

Ballestín, F., R. Leus, 2009, “Resource-Constrained Project Scheduling for Timely Project Completion with Stochastic Activity Durations”, *Prod. Oper. Manage.*, Vol. 18, pp. 459-474.

Davari, M., E. Demeulemeester, 2019, “The Proactive and Reactive Resource-Constrained Project Scheduling Problem”, *J. Sched.*, Vol. 22, pp. 211-237.

Hartmann, S., D. Briskorn, 2010, “Survey of Variants and Extensions of the Resource-Constrained Project Scheduling Problem”, *Eur. J. Oper. Res.*, Vol. 207, pp. 1-14.

Kolisch, R., K. Meyer, R. Mohr, C. Schwindt and M. Urmann, 2003, “Ablaufplanung für die Leitstrukturoptimierung in der Pharmaforschung”, *J. Bus. Econ.*, Vol. 78, pp. 825-848.

Kolisch, R., A. Sprecher, 1996, “PSPLIB - A Project Scheduling Problem Library”, *Eur. J. Oper. Res.*, Vol. 96, pp. 205-216.

Lamas, P., E. Demeulemeester, 2016, “A Purely Proactive Scheduling Procedure for the Resource-Constrained Project Scheduling Problem with Stochastic Activity Durations”, *J. Sched.*, Vol. 19, pp. 409-428.

Möhring, R., F. Radermacher and G. Weiss, 1984, “Stochastic Scheduling Problems I: General Strategies”, *Math. Methods Oper. Res.*, Vol. 28, pp. 193-260.

Naber, A., R. Kolisch, 2014, “MIP Models for Resource-Constrained Project Scheduling with Flexible Resource Profiles”, *Eur. J. Oper. Res.*, Vol. 239, pp. 335-345.

Schramme T., 2014, “Modelle und Methoden zur Lösung des ressourcenbeschränkten Projektablaufplanungsproblems unter Berücksichtigung praxisrelevanter Aspekte”, *PhD Thesis*, Paderborn: Paderborn University.

Variants of a genetic algorithm for the resource-constrained project scheduling with alternative subgraphs

Rojin Nekoueian[1], Tom Servranckx[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Ghent (Belgium)
`rojin.nekoueian@ugent.be, tom.servranckx@ugent.be, mario.vanhoucke@ugent.be`
[2] Operations and Technology Management Centre, Vlerick Business School, Reep 1, 9000 Ghent (Belgium)
[3] UCL School of Management, University College London, 1 Canada Square, London E14 5AA (UK)

Keywords: Resource-constrained project scheduling, Alternative subgraphs, Genetic algorithm.

1 Introduction

The problem of scheduling projects with limited resources has been extensively studied in the literature and is referred to as the resource-constrained project scheduling problem (RCPSP). The RCPSP involves organizing activities while considering both resource availability and activity dependencies to minimize the overall project duration. This problem is known to be NP-hard (Blazewicz *et. al.* 1983). Although the majority of research studies typically assume a predetermined and entirely known project structure, after consulting with industry professionals, researchers noted that intricate projects often involve interchangeable work packages rather than a fixed project network (Servranckx and Vanhoucke 2019). In this research, an extention of the basic RCPSP with alternatives is studied. The RCPSP with alternative subgraphs (RCPSP-AS) consists of two subproblems of selection and scheduling. In the selection subproblem, among several alternatives existing in a subgraph, one alternative is selected and in the scheduling subproblem, the selected alternative activities together with fixed activities are scheduled using schedule generation schemes (SGS). Servranckx and Vanhoucke (2019) found the first benchmark solution for the RCPSP-AS employing a tabu search algorithm. Constructive heuristics including the RCPSP-AS selection and scheduling priority rules were developed by Nekoueian *et. al.* (2023). Moreover, Servranckx *et. al.* (2022) discuss extensions of the basic RCPSP-AS. Another extension of the RCPSP with flexible project structure can be found in Kellenbrink and Helber (2015) where choosing optional activities determines the execution of other activities and/or causes other choices. In the RCPSP-AS, the project network is entirely reconstructed to obtain an alternative project network, while the project networks to solve the RCPSP-PS are directly obtained from RCPSP studies. Moreover, the RCPSP-PS allows logical dependencies between activities that are not connected by a direct precedence relation, where the logical relations always align with the precedence relations in the RCPSP-AS.

Similar to the RCPSP, the RCPSP-AS is an NP-hard problem and hence finding an optimal solution for this problem is challenging. Consequently, metaheuristic methods are ineteresting since they provide near optimal solutions and frequently surpass heuristic methods. Among metaheuristics, the genetic algorithm (GA) has proven to be exceptionally effective in addressing scheduling problems (Hartmann and Briskor 2022). A GA is inspired by the natural evolution of populations and is a versatile search algorithm used to generate solutions for intricate problems and was first developed by Holland (1975). In this study, new variants of GA are proposed for the RCPSP-AS taking into account problem specific local searches for the selection subproblem.

2 Problem description

In this section, we briefly explain the RCPSP-AS specifications originally developed by Servranckx and Vanhoucke (2019). We present the RCPSP-AS project networks with a directed acyclic graph and the activity-on-the-node (AoN) project network, where activities should be scheduled using renewable resources with known and constant availability over project makespan. Activities in the RCPSP-AS are divided into two subsets of *fixed* and *alternative activities.* The *fixed activities* are essential for project completion, and *alternative activities* are optional and should be selected. When scheduling activities, precedence and resource constraints are satisfied and activities are scheduled in a way that the project makespan is minimised. As stated, there exists several alternative subgraphs in the RCPSP-AS project which include multiple alternatives. A *principal activity* is a *fixed activity* that causes the decision of selecting one alternative among several alternatives in a subgraph and a *terminal activity* terminates the decision and is placed at the end of an alternative subgraph. An alternative activity that is the direct successor of a principal activity is a *branching activity.* An *alternative subgraph* is a subgraph consisting of multiple *branching activities* and alternative activities that originate from the same *principal activity* and end with the same *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.

There exist two relations between alternative subgraphs in the RCPSP-AS project network. A *nested alternative* subgraph is an alternative subgraph that is embeded in another alternative subgraph. It might be that an alternative for one work package is related or linked to an alternative of the same or a different work package. If an alternative with a link to another alternative activities is selected, the *linked alternative activities* should also be selected.

These defenitions are applied to an illustrative project example and can be found in the legend of Figure 1. In Figure 1, activity S and E are dummy start and end activities and the project includes three *alternative subgraphs* that each includes two *branching activities.* One possible *alternative path* is $\{S, 1, 2, 4, 9, 10, 12, E\}$. The set of *principal activities* is $\{S, 1, 9\}$ and the set of *terminal activities* is $\{4, 9, 12\}$. *Alternative subgraph l=2* is *nested* in *alternative subgraph l=1* and there are *links* between activities 3 and 6 as well as activities 7 and 11. This means that if *branching activity* 1 is selected, a decision should be made whether to select activity 2 or 3. If activity 3 is seleted then activity 6 and its successor (activity 8) should be selected.

3 Methodology

Generally, a GA starts with the population initialisation and subsequently, new individuals are generated by crossover, mutation and selection over different generations until a pre-determined stopping criterion is achieved. In this study, we develop and examine the performance of different local searches for the selection subproblem resulting in three versions of a GA for the RCPSP-AS.

Since the RCPSP-AS consists of two subproblems, we consider two activity lists for the solution representation, i.e. one list for the selection subproblem and another list for the scheduling subproblem. The selection list selects a *branching activity*, its successors

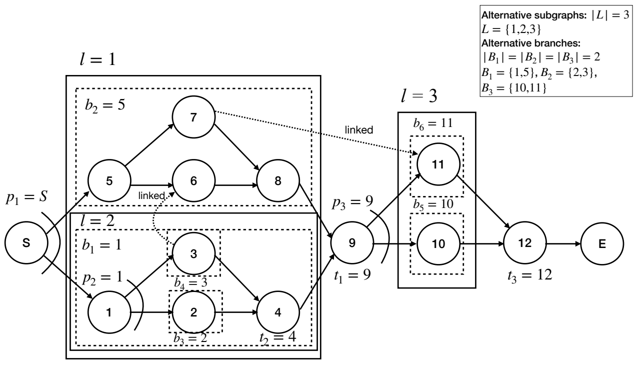


Fig. 1. Project network with alternatives mentioned in Nekoueian *et. al.* (2023)

and *linked activities* based on the priority of *branching activities* while the scheduling list provides the priority of activities for the serial scheduling generation scheme (SSGS).

We introduce two different learning-based local searches for the selection subproblem. In order to identify the impact of learning on our GA, a no learning local search is proposed and all the operators of the GA variants are considered to be the same except the local search for the selection subproblem. These local searches are explained as follows:

No learning In this GA variant, a random local search for the selection subproblem is implemented such that two *branching activities* are randomly chosen and swapped in the selection list. Therefore, the selected *branching activities* change after employing a no learning selection local search. This random local search does not include any information from previous branch selections and hence it is called no learning.

Learning 1 and 2 Each *branching activity* has a known selection probability which is changed throughout the evolutionary process of the GA. Specifically, the selection probability of the selected *branching activity* will be increased, while the selection probability of the non-selected *branching activities* will be decreased. This adjustment is implemented in two different ways. Learning 1, starts with an equal probability of selection for all the *branching activities* and one branch is selected randomly when this local serach is implemented. Subsequently, the selection probability of *branching activity* increases by a predefined value whereas for the other branches, the selection probability decreases. In learning 2, the probability of selecting a *branching activity* depends on the frequency of individuals in the GA populations that prioritise a specific *branching activity.* As the number of elite solutions in the population increases, the selection likelihood of the *branching activities* that provide the minimum project makespan increases. In summary, the distinction between learning 1 and learning 2 lies in how the selection probability of *branching activities* is affected. In learning 1, this probability changes by a predefined value while the changes in the probabilities are determined by the individuals in the population in learning 2.

4 Results and experiments

The performance of newly developed GA variants are examined using a dataset including 360 instances with different project properties provided by Servranckx and Vanhoucke (2019). Table 1 shows the average project makespan of the GAs and significant difference (two-sided p value in paired sample t-test) of the results when implementing learning in the selection local search. From Table 1, it can be concluded that all the newly developed GAs outperform the existing constructive heuristic (CONH-2) proposed by Nekoueian *et. al.* (2023). Moreover, the GA with learning 1 generally outperforms the GA with no learning for the selection subproblem. We also see an improved average makespan when learning 2 was implemenetd on complex projects in our GA framework, however, the difference with the no learning and learning 1 in general is not significant. In our extended experiments on projects with different network complexities, we found that the no learning and learning 2 outperform learning 1 for complex projects.

Table 1. Comparison among developed methodologies for the RCPSP-AS

Research	**Methodology**	**Avg. makespan**	**P value comapred to**		
			no learning	**learning 1**	**learning 2**
This study	GA no learning	93.54	-	**0.03**	0.27
	GA learning 1	**93.46**	**0.03**	-	0.22
	GA learning 2	93.51	0.27	0.22	-
Nekoueian *et. al.* (2023)	CONH-2	94.08	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**

5 Conclusion

In this study, we investigated new versions of a GA for the RCPSP-AS which is the problem of selection and scheduling activities in work packages that can be executed in alternative ways. We developed variants of a GA including and excluding learning and tested the significant differences between the results of the GAs and the previously developed constructive heuristic (CONH-2) proposed by Nekoueian *et. al.* (2023). The results show that all the GA variants outperform the existing CONH-2. Furthermore, based on the results, learning 1 which includes gradual increase in the selection probability of branches outperforms no learning and learning based on population.

Acknowledgments

This work was supported by the Fonds of Wetenschappelijk Onderzoek (FWO) under Grant 12A4222N and by the Bijzonder Onderzoeksfonds (BOF) under Grant BOF24Y2019 000101.

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.

Hartmann S. and Briskor D., 2022, “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.

Holland J., 1975, “Adaptation in natural and artificial systems", *University of Michigan Press, Ann Arbor*, 1975, 2009.01.07.

Kellenbrink C. and S. Helber, 2015, “Scheduling resource-constrained projects with a flexible project structure", *European Journal of Operational Research*, Vol. 246, pp. 379 - 391.

Nekoueian R., T. Servranckx and M. Vanhoucke, 2023, “Constructive heuristics for selecting and scheduling alternative subgraphs in resource-constrained projects", *Computers & Industrial Engineering*, 109399.

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.

Servranckx T., J. Coelho and M. Vanhoucke, 2022, “Various extensions in resource-constrained project scheduling with alternative subgraphs". *International Journal of Production Research*, Vol. 60(11), pp.3501-3520.

Applying Quantum Computing for the Aircraft Deconfliction Problem

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

[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: Tactical Aircraft Deconfliction, Quantum Approximate Management, Optimization, Quantum Computing, Heuristic, Approximation.

1 Introduction

Daily, air transportation grapples with various challenges, encompassing passenger and aircraft service procedures, coordination of vehicles on the airport apron, and the delineation of optimal flight trajectories. Despite the abrupt decline in boarded passengers during the coronavirus pandemic, air travel appears to be regaining its former appeal. An area of interest for us, closely tied to the density of aircraft in the airspace, is reducing the risk of conflicts emerging at the tactical level (5 to 30 minutes before conflicts occur). A conflict is defined as a situation where two aircraft are within 5 nautical miles of each other. While this problem has been extensively explored, numerous proposed solutions exhibit limitations, as pointed out by Pelegrín & d'Ambrosio (2022). In our study, we present the aircraft tactical deconfliction problem formulation as a quantum Hamiltonian and apply the Quantum Approximate Optimization Algorithm (QAOA) (Farhi et al. 2014) to solve it. Additionally, we conduct a comparative analysis of probabilities, circuit length and number of solutions on two IBM physical quantum computers: the 127-qubit *ibm_ sherbrooke* with the Eagle processor and the latest 133-qubit *ibm_ torino* with the Heron processor.

To date, only one paper explores the deconfliction using quantum computing (Stollenwerk et al. 2019). However, this study addresses strategic deconfliction using departure delays only and employs non-universal quantum annealing device. Our research builds upon foundational articles that address the problem classically. In particular, we utilize the geometrical interpretation to detect conflicts (Bilimoria 2000). Additionally, we limit potential aircraft maneuvers to velocity and heading changes only, recognized for being more manageable for human operators while leading to a negligible increase in fuel consumption (Omer 2015). We benchmark our method on artificial instances proposed by Rey & Hijazi (2017), which were subsequently scaled down to fit the current quantum computers in size.

2 Formulation

Given a set of n aircraft, for each of them we propose m different maneuvers, including the option of no maneuver, i.e., maintaining the original trajectory. We define a set of variables in an injective relation with the available qubits, $|X| = R$,

$$X = \{x_{ij} :\ 1 < i \leq n,\ 1 < j \leq m,\ x_{ij} \in \{0, 1\}\}. \tag{1}$$

If the variable x_{ij} is assigned the value 0 it indicates that the aircraft i is not performing maneuver j, whereas a value of 1 indicates the opposite. We assume that maneuvers for an aircraft are disjoint, meaning no aircraft can perform two maneuvers simultaneously.

Solving any combinatorial optimization problem involves defining an appropriate cost function. In case of QAOA, the function needs to be represented as a quantum cost Hamiltonian, which is a linear operator that represents the total energy of a quantum system. Such Hamiltonians can be constructed using Pauli matrices, $I = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, $Z = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$. We write Z_{ij} to denote Z acting on a qubit that is mapped from a variable x_{ij}. Also, we write $Z_{ij}Z_{i'j'}$ as a shorthand for a tensor product $Z_{ij} \otimes Z_{i'j'}$.

We include two constraints into the cost Hamiltonian: a constraint ensuring that an aircraft performs one and only one maneuver and a constraint ensuring that no conflict occurs between two aircrafts. We can express the first condition by:

$$H_1 = \sum_{i=1}^{n} I - \sum_{j=1}^{m} \left(H_x(x_{ij}) \prod_{j'=1, j' \neq j}^{m} (H_{\text{not}}(x_{ij'}))) \right), \tag{2}$$

where $H_x(x_{ij}) = \frac{1}{2}(I - Z_{ij})$ specifies that the maneuver j is to be performed by aircraft i, and $H_{\text{not}}(x_{ij'}) = \frac{1}{2}(I + Z_{ij'})$ says that any other maneuver cannot be performed. We sum over all possible maneuvers for all aircraft. Since QAOA is a minimization algorithm, we need to negate the inner sum. Similarly, the second constraint can be described as:

$$H_2 = \sum_{i,j,i',j' : \text{CM}(i,j,i',j')=1} H_{\text{and}}(x_{ij}, x_{i'j'}). \tag{3}$$

Here, CM represents a conflict matrix filled with potential pairwise conflicts calculated geometrically. For any such conflict, we add the term $H_{\text{and}}(x_{ij}, x_{i'j'}) = \frac{1}{4}I - \frac{1}{4}(Z_{ij} + Z_{i'j'} - Z_{ij}Z_{i'j'})$ to count the conflicts. Having a feasible solution, this sum yields 0.

One could define an additional Hamiltonian $H_{\text{opt}} = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} H_x(x_{ij})$ responsible for an optimization criterion with partial costs represented by the weights w_{ij}. We, however, consider the decision version of the problem, therefore our final Hamiltonian is

$$H_c = H_1 + H_2. \tag{4}$$

QAOA is a hybrid quantum-classical optimizaton algorithm that takes its origins from quantum adiabatic evolution. Given R qubits, the algorithm alternately applies the cost Hamiltonian H_c and the mixer Hamiltonian H_M, typically composed of Pauli-X, to the equal superposition state $|+\rangle^{\otimes R}$ state p times, $p \in \mathbb{Z}^+$. The role of H_c is to distinguish our desired problem solution in phase, while H_M aims to cancel out amplitudes of states having different phases, increasing the probability of measuring the desired solution. This is achieved by optimizing sequences of variational parameters $\overrightarrow{\gamma}$ and $\overrightarrow{\beta}$ using a classical optimizer to minimize the expectation $\langle\psi|H_c|\psi\rangle$. The full quantum circuit looks as follows:

$$|\psi_p(\overrightarrow{\gamma}, \overrightarrow{\beta})\rangle = e^{-i\beta_p H_M} e^{-i\gamma_p H_C} \dots e^{-i\beta_1 H_M} e^{-i\gamma_1 H_C} |+\rangle^{\otimes R}. \tag{5}$$

3 Experiments and Results

As mentioned earlier, the experiments were conducted on instances proposed by Rey & Hijazi (2017), which underwent a reduction in size. Our focus was put on instances of the Random Circle Problem (RCP), and as such, we denote an instance as RCP $n \times m$ where there are n aircraft, each capable of executing m maneuvers.

Theoretically, a higher value of p (indicating a longer circuit) should yield better results. However, two challenges arise. Firstly, longer circuits introduce more variational parameters, intensifying the complexity of the optimization process. Secondly, current quantum computers are susceptible to noise, and the cumulative impact of it significantly worsens

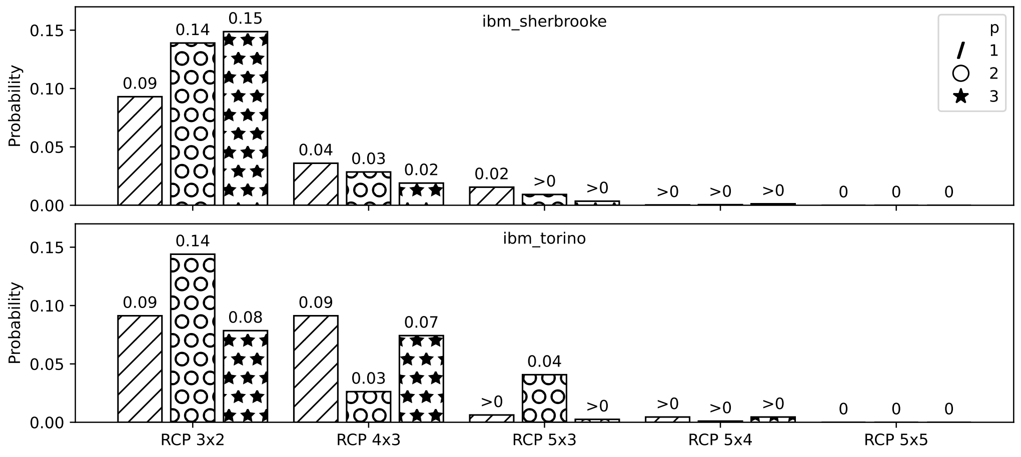


Fig. 1: Probability of finding a conflict-free solution as a function of circuit length and instance size. The upper plot shows results for the older *ibm_ sherbrooke* quantum computer, while the bottom one shows results for the new *ibm_ torino* quantum computer.

the results. The presence of noise and imperfections also restricts us to using only a couple of qubits, even when the quantum device offers over 100 of them. The initial set of experiments, as illustrated in Figure 1, presents the probability of measuring a feasible solution as a function of instance difficulty, circuit length, and quantum processor type. It is worth noting that the probability is calculated based on multiple thousands of quantum circuit sampling. Given that circuit sampling is computationally efficient, even a probability as low as 0.001 for finding a feasible solution is considered successful for solving the instance.

It is evident that the probability of measuring a conflict-free solution decreases with increasing instance difficulty. Notably, an increase in circuit length improves results on *ibm_ sherbrooke* for the RCP 3×2 instance. As more challenging instances require additional quantum entanglements, the circuit must also become longer. Consequently, we observe a breakpoint in the trend from $p = 2$ in the instances RCP 4×3 and RCP 5×3, leading to a decrease in probability, aligning with our expectations.

The observed dependencies are also visible in the case of the *ibm_ torino* quantum computer, although to a lesser extent. The variability in results could stem from various factors, including the influence of initial variational parameters or the inherent nature of randomness of quantum computing. Further investigation is required to minimize the observed variability. Despite the fact *ibm_ torino* is considered to be a more powerful machine compared to *ibm_ sherbrooke* it is more apparent only in specific instances, such as RCP 5×3 or RCP 5×4. Unfortunately, none of the quantum machines succeeded in finding any correct solution for the RCP 5×5 instance.

We also explored the correlation between the number of conflict-free solutions in the instance, and the probability of measuring one. Instances with fewer potential solutions pose increased challenges, not only due to their intrinsic complexity but also because they entail a higher number of aircraft conflicts. Modeling these conflicts with a quantum computer involves establishing error-prone entanglements between qubits, possibly leading to worse results. The chart in Figure 2 validates our assumptions. For the examined instance, RCP 3×2, there would be $2^3 = 8$ feasible solutions if there were no conflicts. By introducing a single conflict scenario, the probability of measuring the correct solution becomes 0.5. The probability gradually decreases as more conflicts are added, reaching 0.03 with only one

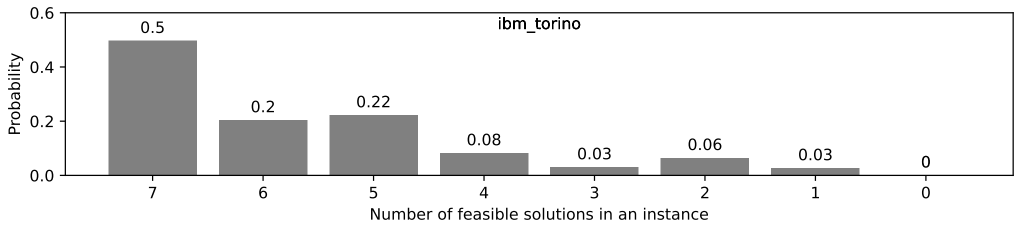


Fig. 2: Probability of finding a conflict-free solution as a function of number of feasible solutions in an instance containing 3 aircraft capable of performing 2 maneuvers.

conflict-free scenario. Even the smallest error during circuit execution on noisy hardware can lead to varied optimization processes, resulting in non-monotonic probability decrease.

4 Conclusions and future work

In this paper, we introduced the quantum Hamiltonian for tactical aircraft deconfliction problem and showed how to leverage the two quantum phenomena — superposition and entanglement — to address the exponential nature of this problem. We used two real quantum computers solve well-known problem instances with varying circuit lengths of the QAOA algorithm. Additionally, we investigated the relationship between the number of solutions in an instance and the probability of measuring a conflict-free solution. Our approach is versatile and can be readily improved by incorporating additional constraints and optimization criteria. For instance, investigating the impact of our approach with the introduction of additional weights to favor minimal changes to the original plan or to reduce overall fuel consumption would be beneficial. We believe, that in the future our approach will be used to solve real-world instances on large-scale fault-tolerant quantum computers.

This research was supported by Poznan Supercomputing and Networking Center (PSNC), project no DOB-SZAFIR/01/B/023/01/2021 and partially funded by Poznan University of Technology, project no 0311/SBAD/0734. The access to quantum devices was possible through IBM Quantum Innovation Center established at PSNC.

Bibliography

Bilimoria, K. (2000), A geometric optimization approach to aircraft conflict resolution, *in* '18th Applied aerodynamics conference', p. 4265.

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

Omer, J. (2015), 'A space-discretized mixed-integer linear model for air-conflict resolution with speed and heading maneuvers', *Computers & Operations Research* **58**, 75–86.

Pelegrín, M. & d'Ambrosio, C. (2022), 'Aircraft deconfliction via mathematical programming: Review and insights', *Transportation science* **56**(1), 118–140.

Rey, D. & Hijazi, H. (2017), Complex number formulation and convex relaxations for aircraft conflict resolution, *in* '2017 IEEE 56th annual conference on decision and control (cdc)', IEEE, pp. 88–93.

Stollenwerk, T., O'Gorman, B., Venturelli, D., Mandra, S., Rodionova, O., Ng, H., Sridhar, B., Rieffel, E. G. & Biswas, R. (2019), 'Quantum annealing applied to de-conflicting optimal trajectories for air traffic management', *IEEE transactions on intelligent transportation systems* **21**(1), 285–297.

Sequencing a single machine with no idle times to minimize total cost: a lower bound derived from permutation constraint relaxation

Olivier Ploton and Vincent T'kindt

Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée (LIFAT), France
`{olivier.ploton,vincent.tkindt}@univ-tours.fr`

Keywords: Scheduling, Relaxation, Lagrangian, Lower Bound.

1 Introduction

We consider a sequencing problem in which n jobs have to be scheduled on a single machine with no idle times. Each job j is defined by a processing time p_j and by a single cost function f_j. The objective is to minimize the total cost $\sum_j f_j(C_j)$, where C_j is the completion time of job j. This problem is denoted by $1|\textit{no-idle}|\sum_j f_j$ using Graham's notation (Graham et al. 1979).

The $1|\textit{no-idle}|\sum_j f_j$ problem generalizes all single-machine problems with regular objectives and deadlines. Indeed, semi-active schedules form a dominant set for regular objectives, and semi-active schedules generate no idle times in the absence of release times. Deadlines are taken into account by setting an infinite cost $f_j(C_j)$ for any deadline violation ($C_j > \tilde{d}_j$). On a theoretical point of view, the $1|\textit{no-idle}|\sum_j f_j$ problem is strongly NP-hard, because several of its sub-problems are strongly NP-hard, as, for example, the $1|\tilde{d}_j|\sum w_j C_j$ problem (Lenstra et al. 1977).

We now define the notion of permutation constraint relaxation. From now on, the original $1|\textit{no-idle}|\sum_j f_j$ problem is refered to as the non-relaxed or strict problem, and a plain schedule, solution of the original problem, is refered to as a strict schedule. Obviously, in a strict schedule, each job must appear exactly once. So, a strict schedule is a permutation of $(1, \ldots, n)$. We relax this constraint and consider relaxed schedules in which each job may be absent or may appear one or several times.

The main advantage of a relaxed problem is to be solved in pseudo-polynomial time and space using dynamic programming. To solve a strict problem, a traditional dynamic programming scheme needs to consider the set of already scheduled jobs in each state. This implies an exponential number of states (at least one per job subset), and thus an exponential worst-case time and space complexity. On the opposite, to solve a relaxed problem, the same dynamic programming scheme only needs to consider some temporal information, usually the makespan of already scheduled jobs. This way, the number of states and thus the worst-case time and space complexity are only pseudo-polynomial.

In this paper we study a new way of indirectly solving a strict problem by directly solving only relaxed problems. Notice that, for a strongly NP-hard strict problem, the number of direct relaxed problem resolutions needed is necessarily exponential in the worst case. As far as we know, only two ways of solving a strict problem through its relaxed counterparts are described in the literature: Inclusion-Exclusion and Lagrangian relaxation.

Inclusion-Exclusion is a mathematical formula used to count the number of solutions of a strict problem instance, given the number of solutions of an exponential number of corresponding relaxed problems instances. On a computational point of view, the Inclusion-Exclusion principle (Fomin and Kratsch 2010, Nederlof 2008) consists in reducing an optimization problem instance to a polynomial number of decision problems instances, and solving them using the Inclusion-Exclusion formula. Inclusion-Exclusion is conceived as a theoretical, rather than practical tool. On a practical point of view, it is outperformed by other exact algorithms such as Branch-and-Bound. But on a theoretical point of view, it outperforms them and achieves a moderately exponential worst-case time complexity along with a pseudo-polynomial worst-case space complexity (Ploton and T'kindt 2022a, Ploton and T'kindt 2022b).

Lagrangian relaxation consists in trading a constraint for penalties. Each job j is assigned a penalty $\lambda_j \in \mathbb{R}$ and the penalties form together a vector $\lambda = (\lambda_1, \ldots, \lambda_n)$. Then, a penalized objective value is computed and normalized such as to eliminate any penalty in a strict schedule. This penalized objective value is a lower bound of the Lagrangian dual optimum, itself a lower bound of the objective value of any strict schedule.

Any method to update λ and derive the closest possible lower bound of the Lagrangian dual optimum is suitable. Abdul-Razaq and Potts (1988) used a subgradient descent and derived a lower bound usable in a Branch-and-Bound procedure. Their work has been integrated by Tanaka et al. (2009), whose algorithm constitutes, as far as we know, the current state of the art in practice for the weighted tardiness minimization problem.

2 Permutation constraint relaxation and permutation classes

We now introduce a new iterative method to closely approximate the Lagrangian dual optimum, using the notion of permutation class and concepts from Inclusion-Exclusion.

For any relaxed schedule S solution of the $1|\textit{no-idle}|\sum_j f_j$ problem, and for any job j, we define $n_j(S)$ as the number of occurrences of j in S. Thus, a schedule is strict when $n_j(S) = 1$ for all j. We denote the objective function as $\gamma = \sum_j f_j$, and we define the normalized penalized objective γ_λ as $\gamma_\lambda(S) = \gamma(S) + \sum_{j=1}^{n}(n_j - 1)\lambda_j$. The Lagrangian $L(\lambda)$ associated to a particular λ is the minimum of $\gamma_\lambda(S)$ among all relaxed schedules S.

Following Abdul-Razaq and Potts (1988), $L(\lambda)$ can be computed using the following dynamic programming scheme: we define $opt_\lambda[C]$ as the minimum $\gamma_\lambda(S)$ among relaxed schedules S whose completion time is C. Then, we have $\gamma_\lambda(S) = opt_\lambda[\sum_{j=1}^{n} p_j]$, and:

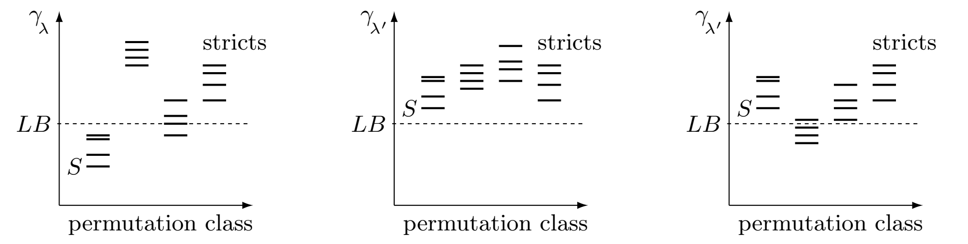
$$opt_\lambda[\,0\,] = \gamma_\lambda(\textit{empty relaxed schedule}) = -\sum_{j=1}^{n} \lambda_j \tag{1}$$

$$opt_\lambda[C] = \min_{j \in \{1 \ldots n\}\,|\,p_j \le C} (opt_\lambda[C - p_j] + f_j(C) + \lambda_j) \qquad \text{for } C > 0 \tag{2}$$

A corresponding optimal schedule $S^*(\lambda)$ can be derived from a backtrace of this scheme.

We now cope with permutation classes. We define two schedules S and S' as equivalent up to a permutation when we can derive S' by permuting the jobs of S or, equivalently, when $n_j(S) = n_j(S')$ for all j. We define the permutation class $Cl(S)$ of a schedule S as the set of schedules S' which are equivalent to S up to a permutation. Notice that all strict schedules are equivalent to each other and form a single permutation class. Moreover, permutation classes form a partition of the set of all relaxed schedules.

Our aim is to iteratively find a lower bound LB that approaches the Lagrangian dual optimum $\sup_\lambda L(\lambda)$. We now describe a way to update penalties and enhance the lower bound of the dual optimal Lagrangian. Suppose that a vector of penalties λ and a lower bound LB are already known. We conceptually partition the set of relaxed schedules in permutation classes, and we represent schedules, as in Figure 1a, with classes on the horizontal axis (4 classes in the example) and objectives on the vertical axis. Notice that modifying λ shifts each class vertically without changing its shape. This is because the difference between penalized objectives of two equivalent schedules S and S' does not depend on λ: $\gamma_\lambda(S') - \gamma_\lambda(S) = \gamma(S') - \gamma(S)$.



(a) Initially: S optimal for λ (b) Case S optimal for λ' (c) Case S not optimal for λ'

Fig. 1: Refining a search for a dual Lagrangian optimum

We solve the relaxed problem for λ, so we find an optimal schedule $S = S^*(\lambda)$ with objective $L(\lambda)$. For now, we assume that we can choose a new penalty vector λ' which improves the bound, i.e. such that $\gamma_{\lambda'}(S) > LB$. An efficient way of choosing λ' will be described later on. There are two cases: either S remains optimal for λ', or it is not anymore.

In the case where S remains optimal for λ' (Figure 1b), we have $L(\lambda') = \gamma_{\lambda'}(S) > LB$ so, we increase the bound by taking $LB \leftarrow \gamma_{\lambda'}(S)$, which improves it.

In the case where S is no longer optimal for λ' (Figure 1c), it is guaranteed that the whole class $Cl(S)$ is dominated. This class is eliminated of the search, which simplifies the problem.

We derive an iterative method (Algorithm 1) to compute a lower bound of the Lagrangian dual optimum. This algorithm is very close in structure to the sub-gradient descent of Abdul-Razaq and Potts (1988) but the way to update the penalties in statement (3) is radically different.

Algorithm 1: Computation of a lower bound by elimination of permutation classes.

```
function PermutationClassLB:
    $\forall j,\ \lambda_j \leftarrow 0$
    $LB \leftarrow LB_0$        // any lower bound
    for $k \leftarrow 1, 2, 3, \dots$ do
        $S_k \leftarrow S^*(\lambda)$ // relaxed problem with penalties solved using Dynamic Programming
        $LB \leftarrow \max(LB, \gamma_\lambda(S_k))$
        if $S_k$ is strict then: strict problem solved, $S_k$ optimal. stop.
        Find a new $\lambda$ such that $\forall k' \le k, \gamma_\lambda(S_{k'}) > LB$          (3)
        if there is no such $\lambda$ then: stop.
    result: $LB$ is a lower bound of $\sup_\lambda L(\lambda)$, itself lower bound of the optimum objective.
```

We now describe how to implement statement (3) in polynomial time. It is convenient to define LB_k as the value of LB at iteration k. Expanding the definition of γ_λ, we derive that each condition in statement (3) is a linear inequality involving $\lambda_1, \dots, \lambda_n$ and LB.

Taking into account that $LB \geq LB_{k'}$ for each $k' \leq k$, we derive this linear inequality system:

$$\gamma(S_1) + \sum_{j=1}^{n}(n_j(S_1) - 1)\lambda_j > LB \qquad (E_1) \qquad \text{and} \qquad LB \geq LB_1 \qquad (E'_1)$$

$$\cdots$$

$$\gamma(S_k) + \sum_{j=1}^{n}(n_j(S_k) - 1)\lambda_j > LB \qquad (E_k) \qquad \text{and} \qquad LB \geq LB_k \qquad (E'_k)$$

This is a system with a fixed set of real variables (λ_j and LB) and no integer variable. Moreover, this system is incremental: at iteration k, both equations (E_k) and (E'_k) are added, and previous equations are neither modified nor withdrawn. As a consequence, this system is guaranteed to be solvable in polynomial time and has a chance to be efficiently solved by standard Linear Programming solvers.

3 Computational experiments on the $1|\tilde{d}_j|\sum_j w_j C_j$ problem

We have implemented and compared the performance of our algorithm with the reference algorithm (ARP) of Abdul-Razaq and Potts (1988), in the context of the $1|\tilde{d}_j|\sum_j w_j C_j$ strongly NP-hard problem. We have compared speeds and bound quality, i.e. deviation from the optimum objective value, provided by the exact branching algorithm of Shang et al. (2021). According to our results, both algorithms share the same time complexity: our algorithm is around 10 times slower independently of the instance size, and memory consumption is not a limiting factor. But our algorithm has a better bound quality: around 0.2% compared to 0.6% for (ARP). We plan to replace (ARP) with our new algorithm in the algorithm of Tanaka et al. (2009), and compare efficiencies.

In conclusion, permutation constraint relaxation appears as a valuable tool to solve scheduling problems, but there are still many open questions ahead. On a theoretical point of view, is there a way to bound the number of permutation classes and derive a worst-case complexity bound ? On a practical point of view, is there a way of accelerating the convergence of the iterations, e.g. by a clever choice of a particular solution of the linear system ? These questions are of great importance to widen the application domain of the permutation constraint relaxation technique.

References

Abdul-Razaq T.S., C. Potts, 1988, "Dynamic Programming State-Space Relaxation for Single-Machine Scheduling", *Journal of the Operational Research Society*, 39(2):141–152.

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.

Lenstra J.K., A.H.G. Rinnooy Kan, and P. Brucker, 1977, "Complexity of machine scheduling problems", *Annals of Discrete Mathematics*, 1:343–362.

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

Ploton O., V. T'kindt, 2022a, "Exponential-time algorithms for parallel machine scheduling problems", *Journal of Combinatorial Optimization*, 44:3405–3418.

Ploton O., V. T'kindt, 2022b, "Moderate worst-case complexity bounds for the permutation flowshop scheduling problem using Inclusion–Exclusion", *Journal of Scheduling*, 26(2):137–145.

Shang L., V. T'Kindt, F. Della Croce, 2021, "Branch and Memorize exact algorithms for sequencing problems: efficient embedding of memorization into search trees", *Computers and Operations Research*, 128:105–171.

Tanaka S., S. Fujikuma, M. Araki, 2009, "An exact algorithm for single-machine scheduling without machine idle time", *Journal of Scheduling*, 12(6):575–593.

Solver based heuristics for rolling stocks corrective maintenance scheduling

Tom Ray[1,2], Ronan Bocquillon[1] and Vincent T'kindt[1]

[1] Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée de Tours (LIFAT), France
[2] SNCF Voyageurs, ingénierie du matériel cluster ouest, France

Keywords: Rolling Stock maintenance, Mathematical Programming, Constraint Programming, Heuristics.

1 Introduction

Passenger trains have a very precise schedule due to the transportation demand and railway systems aim to exploit rolling stocks to their maximum capacity. Maintaining a healthy network of rolling stocks can be really difficult because it must rely on an effective maintenance schedule that does not impact the transportation plan. But while some maintenance operations are known beforehand, some repairing that could not have been predicted still needs to be done. These jobs are brought to our knowledge through the train itself. The time allowed to fix these malfunctions is relatively short (from a few hours to a few days). It is allowed to schedule a complete repair, or a partial repair named diagnosis that ensures that the train can be used in normal condition even if the operation is not completely done. The aim of our study is to find an efficient way to schedule the starting times of the maintenance jobs, completely or not, so that their due dates are met.

Section 2 defines the considered problem, next presents a Mixed Integer Linear Programming (MILP) model and a Constraint Programming (CP) model. Section 3 introduces two local search heuristics based on these models and Section 4 provides an overview of the computational results.

2 Problem definition and associated models

2.1 Problem definition

Let $I = \{1, ..., n\}$ be the set of jobs to schedule and $J = \{1, ..., m\}$ be the set of tracks available for the maintenance. Each job i has a repair duration p_i, a diagnosis duration p_i^d, a due date d_i, a tardiness cost w_i, a diagnosis cost u_i and a need for a specific infrastructure. Each track j has one or more infrastructures. We define V_i as the vector of tracks j on which job i can be assigned. Starting from the compatibility between the infrastructure requirements of the jobs, those available on the tracks and the availabilities of trains and tracks, we define $\mathcal{T}_{ij}^d$ as the set of time intervals at which job i can start on track j. The starting times values are in $[0, H]$ with H being the planning horizon. We define T_i as the tardiness of job i based on its starting time. Let S_i be the starting time of job i in a given schedule, then we have: $T_i = S_i - d_i$.

In this problem, we aim to minimize the sum of the weighted tardiness of jobs while limiting the number of performed diagnosis, especially on highly important repairs. We denote by ε the total cost allowed for the performed diagnosis. This problem is noted $P|\mathcal{T}_{ij}^d|\sum w_i T_i$ and is strongly NP-Hard.

2.2 A Mixed Integer Linear Programming Model

We present a time-indexed model based on binary variables representing the times t at which jobs start. We have:

- x_{ijt} : 1 if job i starts on track j at the time t, 0 otherwise, $\forall i \in I$, $\forall j \in V_i$, $\forall t \in \mathcal{T}^d_{ij}$;
- y_i : 1 if job i is scheduled in diagnosis mode, 0 if it is scheduled in repair mode, $\forall i \in I$.

We also introduce integer variables to model the starting times and tardiness that are used to compute the objective function:

- S_i : the starting time of job i, $\forall i \in I$;
- T_i : the tardiness of job i based on its due date, $\forall i \in I$.

The model is given as follows :

$$\text{Minimize } z_1 = \sum_{i=1}^{n} w_i \times T_i \tag{1}$$

$$\sum_{i=1}^{n} u_i \times y_i \leq \varepsilon \tag{2}$$

$$\sum_{j=1}^{m} \sum_{[t^b,t^e]\in\mathcal{T}^d_{ij}} \sum_{t=t^b}^{t^e-p^d_i} x_{ijt} \leq 1, \forall i \in I \tag{3}$$

$$\sum_{j=1}^{m} \sum_{[t^b,t^e]\in\mathcal{T}^d_{ij}} \sum_{t=t^b}^{t^e-p_i} x_{ijt} \geq 1 - y_i, \forall i \in I \tag{4}$$

$$\sum_{j=1}^{m} \sum_{[t^b,t^e]\in\mathcal{T}^d_{ij}} \sum_{t=t^b}^{t^e-p^d_i} x_{ijt} \geq y_i, \forall i \in I \tag{5}$$

$$\sum_{\substack{i'=1\\ i'\neq i}}^{n} \sum_{t'=t}^{t+p^d_i} x_{i'jt'} \leq (1 - x_{ijt}) \times n, \begin{array}{r}\forall i \in I\\ \forall j \in V_i\\ \forall t \in \mathcal{T}^d_{ij}\end{array} \tag{6}$$

$$\sum_{\substack{i'=1\\ i'\neq i}}^{n} \sum_{t'=t+p^d_i}^{t+p_i-1} x_{i'jt'} \leq (y_i - x_{ijt} + 1) \times n, \begin{array}{r}\forall i \in I\\ \forall j \in V_i\\ \forall t \in \mathcal{T}^d_{ij}\end{array} \tag{7}$$

$$S_i \geq \sum_{j=1}^{m} \sum_{[t^b,t^e]\in\mathcal{T}^d_{ij}} \sum_{t=t^b}^{t^e-p^d_i} x_{ijt} \times t, \forall i \in I \tag{8}$$

$$\begin{array}{l} T_i \geq 0 \\ T_i \geq S_i - d_i \end{array}, \forall i \in I \tag{9}$$

The objective function (1) represents the total weighted tardiness that has to be minimized. The constraint (2) gives a limit on the cost induced by the performed diagnosis. Constraints (3), (4) and (5) are used to schedule each job at a single time point, taking into account the possibility of a diagnosis. Constraints (6) and (7) guarantee that a track can host only one job at a time, whether it is a diagnosis or a complete repair. Constraints (8) and (9) are used to compute the starting times of jobs and their tardiness.

2.3 A Constraint Programming Model

In this section, we present a constraint programming (CP) model based on interval variables. For this model we define J_i as the interval variable associated with job i, J_{ij} as the interval variable representing the possibility of job i being scheduled on track j and J^c_{ij} (resp. J^d_{ij}) as the interval variables representing the possibility of job i being scheduled on track j in complete repair mode (resp. in diagnosis mode).

To guarantee that exactly one track and one mode (complete repair or diagnosis) is selected for each job, we use two levels of alternatives: the first one ensures that exactly one track is selected for each job, while the second ensures that exactly one mode is selected for each job. For each track, we use a disjonctive constraint to ensure that the jobs do not overlap. Due to a lack of space, the complete model is not reported in the paper but it will be presented during the conference.

3 Heuristics

In this section, we introduce two local search heuristics. We chose to use matheuristics because they proved to be efficient for solving scheduling problems (see e.g. (T'kindt 2023)). We chose to use two different matheuristic frameworks to leverage at best these two models.

The concept used is the same for both as they are local search heuristics: we define a neighbourhood of solutions to explore and we try to iteratively improve our current solution, step by step, until we are stuck into a local optimum or we have reach a given time limit. Each heuristic exploits two procedures: the first one, called intensification, explores the neighbourhood of a solution. The second one, called diversification, is used in case we are stuck into a local optimum to try to jump to another neighbourhood that may be more interesting.

3.1 Local Branching

In this section we describe a Local Branching (LB) heuristic (Fischetti and Lodi 2003) that exploits the MILP formulation given in section 2.2. For this heuristic, we use a Hamming distance constraint to define the neighbourhood of the current solution. This distance counts every change between two consecutive iterations. Let s^b be the solution at the current iteration, and let x^b and y^b be its associated variables. We define the following sets:

$$X^b_0 = \{x_{ijt} | \forall i \in I, j \in V_i, t \in \mathcal{T}^d_{ij} \text{ and } x^b_{ijt} = 0\} \quad Y^b_0 = \{y_i | \forall i \in I \text{ and } y^b_i = 0\}$$

$$X^b_1 = \{x_{ijt} | \forall i \in I, j \in V_i, t \in \mathcal{T}^d_{ij} \text{ and } x^b_{ijt} = 1\} \quad Y^b_1 = \{y_i | \forall i \in I \text{ and } y^b_i = 1\}$$

and then the Hamming distances:

$$D^x(x, x^b) = \sum_{x_{ijt} \in X^b_0} x_{ijt} + \sum_{x_{ijt} \in X^b_1} (1 - x_{ijt})$$

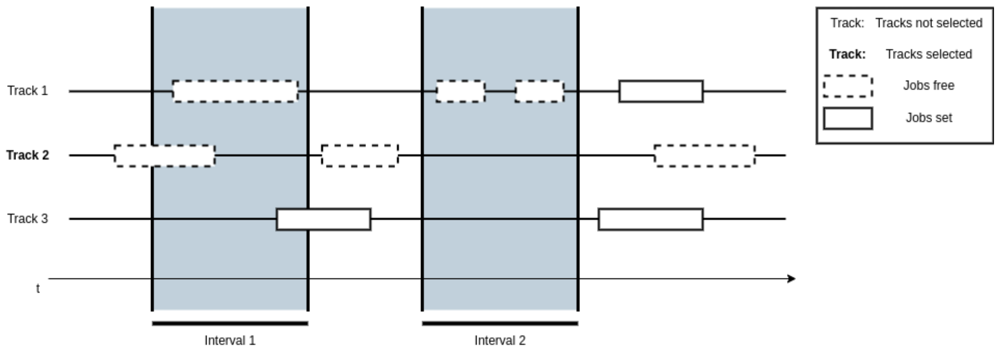
$$D^y(y, y^b) = \sum_{y_i \in Y^b_0} y_i + \sum_{y_i \in Y^b_1} (1 - y_i)$$

Then, the intensification and the diversification processes limit the number of changes in the current solution by using these distances and a parameter K_x.

3.2 Variable Partitioning Local Search

In this section we describe a variable partitioning local search (VPLS) heuristic (Della Croce *et. al.* 2013) that exploits the constraint programming formulation sketched in section 2.3. The neighbourhood is defined by randomly selecting multiple non-overlapping intervals and a set of tracks. At each iteration, we free everything that has been scheduled in the selected intervals and tracks. Everything else is set exactly as it is in the current solution (Figure 1). The corresponding subproblem is solved by CP. Then, we repeat as many iterations as possible within a given time limit.

Fig. 1. Exemple of neighboorhood for the intensification process in the VPLS heuristic



4 Experiments

We conducted experiments on a set of randomly generated instances. This set contains 105 different types of instances defined by a triplet (s, m, n). The parameter s determines the way availabilities between trains and tracks are distributed through simulating different maintenance sites. All the generated instances follow a structure similar to the case of rolling stocks fleets and maintenance sites located in Paris, more specifically in the northern part of the city, and represent scenarios that the planners may encounter during their work.

After comparing the two exact models, we see that for difficult instances the CP model is, on the average, worse than the MILP model. In some cases both models find an optimal solution but the MILP model struggles to prove its optimality. The use of the CP model is very time efficient for infeasible and easy instances but for big and difficult instances, while both model reach the given time limit, the MILP shows better deviations. After evaluating the two heuristics, we see that LB and VPLS improve the results of their respective parent model alone but also that on the average LB improves the results of both models. The VPLS heuristic is faster than LB in most cases but not necessarily more efficient. Therefore, VPLS is more interesting to use with a reduced time budget. But as long as efficiency is considered, LB outperforms VPLS. More detailed results will be discussed during the conference.

References

Fischetti M. and Lodi A., 2003, “Local branching“, *Mathematical Programming*, vol. 98, pp. 23-47.

Della Croce, F. and Grosso, AC. and Salassa, F. and others, 2013, “Matheuristics: embedding MILP solvers into heuristic algorithms for combinatorial optimization problems“, *Heuristics: theory and applications*, pp. 31-52

T'kindt, V., 2023, “The marriage of Matheuristics and Scheduling“, *Scheduling seminar*, `https://schedulingseminar.com/presentations/SchedulingSeminar_VincentTkindt.pdf`

Artificial Intelligence (AI) Application in Construction Scheduling and Project Management: A Theoretical Framework

Saeed Rokooei[1] and Mahdi Ghafoori[1]

Mississippi State University, U.S.
srokooei@caad.msstate.edu, mghafoori@caad.msstate.edu

Keywords: Artificial Intelligence, Construction, Project Management.

1 Abstract

The application of Artificial Intelligence (AI) has witnessed a notable surge across diverse industries, with a parallel emergence of numerous AI applications tailored for the Architecture, Engineering, and Construction (AEC) sector. Despite this, untapped potential in construction scheduling and project management presents opportunities for further AI advancements. This paper introduces a theoretical model for an AI-based construction scheduling approach that leverages Building Information Modeling (BIM), addressing unexplored potential in the AEC industry. The model's high-level hierarchy, components, development process, implementation approach, and outputs are outlined. The model inspires project management scholars to utilize the BIM model along with other legacy information to design and develop AI-based construction schedules.

2 Background

Artificial intelligence (AI) applications are experiencing exponential growth across various fields and industries. Companies are actively pursuing the latest developments in AI applications to align their operations and approaches with cutting-edge technologies. Concurrently, the construction industry has initiated the exploration and utilization of emerging technologies, although progress and implementation face typical delays inherent to the nature of the industry. The expectation is that AI-based tools will find extensive use in project management. These tools can contribute to process automation, data analysis, as well as team communication and collaboration. However, several challenges, such as data privacy and security, ethical considerations, adoption barriers, and change management, must be effectively addressed (Weng 2023). Savio and Ali (2023) reviewed the application of AI in project management, noting that despite challenges such as data quality, integration with existing systems, and uncertainties in initial investments, AI can aid project managers in data-driven decision-making, risk management enhancement, and optimization of resource allocation. Various studies, such as those by Hamada *et al.* (2021), Sree and SNSVSC (2016), Han *et al.* (2015), and Crawford *et al.* (2015), discuss the use of AI in project management, employing techniques such as fuzzy models, machine learning, and optimization algorithms. Within the construction sector, project management stands out as a rich context for AI applications (Aljebory and QaisIssam 2019). Taboada *et al.* (2023) conducted a systematic literature review, concluding that AI-enabled project management applications have significantly increased over the last decade. However, their classification did not produce a category for planning and scheduling. Hajdasz (2014) developed a monolithic construction computer-aided system (MoCCAS) as a comprehensive decision support tool for construction site management in repetitive projects. Wang *et al.* (2012) utilized artificial neural networks and support vector machines to predict project Key Performance Indicators (KPIs), including scheduling performance. Cheng and Hoang (2018) fused the least squares support vector machine (LS-SVM) and the firefly algorithm (FA) to estimate the duration of diaphragm walls in construction projects, achieving accurate forecasts with low prediction deviation (<10%). Faghihi *et al.* (2015) conducted a comprehensive review of various automation applications in construction scheduling over three decades, revealing that genetic algorithms, expert systems, case-based reasoning, model-based approaches, and neural networks were the most commonly used methods in construction scheduling, in that order.

3 Problem statement

The accurate prediction of potential delays in construction projects remains a critical challenge that significantly impacts project timelines and resource allocation. The vast amount of historical project data, coupled with external factors such as weather conditions, regulatory changes, and supply chain disruptions, presents a complex landscape for project managers to navigate. Traditional project scheduling methods often fall short in accounting for these dynamic variables, leading to unforeseen delays and cost overruns. To address this issue, there is a growing need to harness the power of artificial intelligence (AI) and predictive analytics to analyze historical project data and external factors systematically. The aim of this research is to investigate and develop a theoretical framework for an AI-driven construction scheduling methodology. This approach harnesses the capabilities of Building Information Modeling (BIM) to analyze both historical project data and external factors. By incorporating BIM Models and other relevant data, the research aims to deliver a robust tool that empowers construction project managers to predict potential delays accurately and make optimized scheduling decisions. This would enable project managers to proactively identify and mitigate risks, optimize resource allocation, and make informed scheduling decisions. However, the development and implementation of such a predictive analytics system poses various technical, methodological, and practical challenges that need to be addressed. These challenges include but are not limited to data integration, model accuracy, real-time data updates, and user-friendly interfaces for seamless adoption by construction professionals.

4 Model development and implementation

The model development encompasses several key components, as illustrated in Figure 1. The model begins by collecting historical project data, including project schedules and timelines, resource allocations, relevant performance metrics, and construction BIM models. This data serves as the foundation for training and validating the predictive model. Subsequently, in the next step, relevant features are extracted from the collected data. These features may include project size, complexity, historical weather patterns, economic indicators, and regulatory changes. Importantly, feature engineering plays a crucial role in enhancing the model's ability to capture the diverse factors influencing project delays. Moreover, the feature-engineered data is used to train the predictive model. Depending on the complexity and nature of the data, machine learning algorithms such as regression, decision trees, or neural networks are employed to learn the patterns and relationships within the dataset. Finally, after the training phase, the trained model is validated using additional historical data not used during the training phase. This step ensures the model's generalizability and reliability in predicting project delays across various scenarios. Once validated, the predictive model is integrated into the project management system. Consequently, it becomes a tool for real-time decision-making, offering insights into potential delays and aiding in schedule optimization.

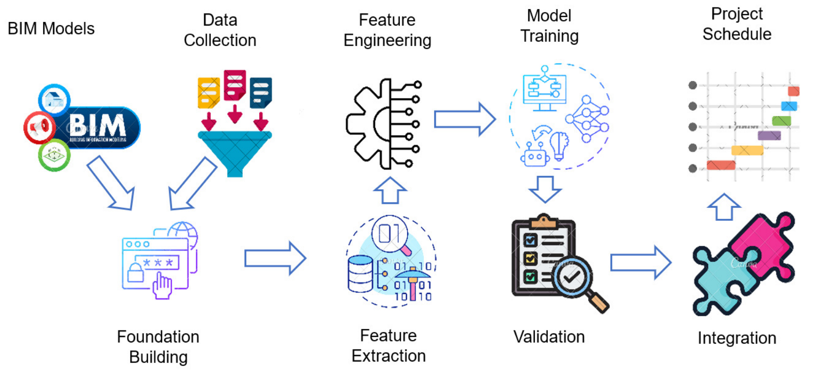


Fig. 1. Predictive Analytics Model Flowchart

To successfully implement the predictive analytics model, the following types of input data are necessary: (1) Historical Project Data, wherein project timelines, milestones, and resource allocations from completed projects are crucial for training the model; (2) External Factors, encompassing data on external factors such as weather conditions, economic indicators, and regulatory changes that are required to capture the broader context influencing project delays; (3) BIM models, which include all construction elements and components, and hence, quantity take offs (QTO) and (4) Real-Time Project Data, involving real-time data on progress, resource usage, and any unexpected events for ongoing projects, which are necessary to continually update the predictive model and enhance its accuracy.

5 Model Output

The output of the predictive analytics model is a forecast of potential project delays, which can be presented in various forms, including: (1) Prediction of activity durations, forming the project schedule through activity sequencing and duration estimates; (2) probability of delay, indicating the likelihood of a project delay based on the current project status and external factors; (3) timeframe of delay, providing an estimate of the expected delay duration, enabling project managers to proactively address potential issues; and (4) suggested mitigation strategies, offering recommendations for actions to mitigate or minimize the predicted delays, such as resource reallocation or schedule adjustments. One example of a mitigation strategy could be resource reallocation. If the predictive analytics model forecasts a potential delay in a certain phase of the project due to resource constraints, the suggested mitigation strategy could involve reallocating resources from less critical tasks to the ones causing the delay. Beyond delivering forecasts of potential project delays, the predictive analytics model offers a dynamic tool for project management optimization. The probability of delay metric provides a nuanced understanding, allowing project teams to allocate resources judiciously and prioritize tasks effectively. The timeframe of delay estimation empowers project managers to formulate contingency plans and adjust project timelines strategically. Moreover, the suggested mitigation strategies furnish actionable insights, guiding decision-makers in implementing proactive measures to circumvent or mitigate delays. This comprehensive utilization of the predictive analytics model not only enhances decision-making precision but also contributes to overall project success by minimizing disruptions and maximizing resource efficiency.

6 Summary and Conclusion

The paper addresses the critical challenge of accurately predicting delays in construction projects through the development of an AI-driven predictive analytics model. Traditional scheduling methods often fail to account for dynamic variables, leading to unforeseen delays and cost overruns. To mitigate this, the research proposes a framework that utilizes Building Information Modeling (BIM) and historical project data to systematically analyze and predict potential delays. By leveraging historical project data and external factors, the model enables construction professionals to make more informed decisions, minimize disruptions, and maximize resource efficiency, ultimately contributing to the overall success of construction projects. However, the implementation of such a model presents various technical, methodological, and practical challenges that need to be addressed for effective utilization in real-world construction scenarios.

References

Aljebory, K.M. and QaisIssam, M., 2019, March. Developing AI Based Scheme for Project Planning by Expert Merging Revit and Primavera Software. In *16th International Multi-Conference on Systems, Signals & Devices (SSD)* Istanbul, Turkey: IEEE, pp. 404-412.

Cheng, M.Y. and Hoang, N.D., 2018, 'Estimating construction duration of diaphragm wall using firefly-tuned least squares support vector machine', *Neural Computing and Applications*, 30(8), pp. 2489-2497.

Crawford, B., Soto, R., Johnson, F., Misra, S., Paredes, F. and OlguÃn, E., 2015, 'Software Project Scheduling using the Hyper-Cube Ant Colony Optimization algorithm', *Tehnicki vjesnik - Technical Gazette*, 22(5), pp. 1171-1178.

Faghihi, V., Nejat, A., Reinschmidt, K.F. and Kang, J.H., 2015, 'Automation in construction scheduling: a review of the literature', *The International Journal of Advanced Manufacturing Technology*, 81(9-12), pp. 1845-1856.

Hajdasz, M., 2014, 'Flexible management of repetitive construction processes by an intelligent support system', *Expert Systems with Applications*, 41(4), pp. 962-973.

Hamada, M.A., Abdallah, A., Kasem, M. and Abokhalil, M., 2021, 'Neural Network Estimation Model to Optimize Timing and Schedule of Software Projects', *2021 IEEE International Conference on Smart Information Systems and Technologies (SIST)*, Nur-Sultan, Kazakhstan: IEEE, pp. 1-7.

Han, W., Jiang, L., Lu, T. and Zhang, X., 2015, 'Comparison of Machine Learning Algorithms for Software Project Time Prediction', *International Journal of Multimedia and Ubiquitous Engineering*, 10(9), pp. 1-8.

Savio, R.D. and Ali, J.M., 2023, 'Artificial Intelligence in Project Management & Its Future', *Saudi Journal of Engineering and Technology*, 8(10), pp. 244-248.

Sree, P.R. and Ramesh S.N.S.V.S.C., 2016, 'Improving Efficiency of Fuzzy Models for Effort Estimation by Cascading & Clustering Techniques', *Procedia Computer Science*, 85, pp. 278-285.

Taboada, I., Daneshpajouh, A., Toledo, N. and de Vass, T., 2023, 'Artificial Intelligence Enabled Project Management: A Systematic Literature Review', *Applied Sciences*, 13(8), p. 5014.

Wang, Y.R., Yu, C.Y. and Chan, H.H., 2012, 'Predicting construction cost and schedule success using artificial neural networks ensemble and support vector machines classification models', *International Journal of Project Management*, 30(4), pp. 470-478.

Weng, J., 2023, 'Putting Intellectual Robots to Work: Implementing Generative AI Tools in Project Management'. *New York University.*

Minimizing makespan and number of preemptions in resource-constrained project scheduling problems with time-varying resources

Ilaria Salvadori[1] and Alessandro Agnetis[1]

University of Siena, Italy
`ilaria.salvadori@student.unisi.it, agnetis@diism.unisi.it`

Keywords: RCPSP, preemption, ILP formulations, bicriteria problems, time-varying resources, case study.

1 Problem definition

In project scheduling, allowing the possibility of interrupting the execution of activities can play an important role. This leads us to consider the preemptive version of the resource constrained project scheduling problem (preemptive RCPSP). Our purpose is to investigate the tradeoff between the total number of interruptions across a project and the makespan. In particular, we want to investigate the problem in a context in which the resource profile varies over time, which typically may create the need for activity preemption.

Preemptions may bring benefits in terms of project makespan, but in some cases they also entail undesired effects, such as an increased risk of operational mistakes, or difficulties in realigning resources with project execution. In the preemptive RCPSP literature, most studies either do not consider any restriction on the number of activity preemptions (Moukrim *et. al.* 2015), or consider that the number of preemptions of each activity is limited, and each part of an activity must have a minimum duration (Quintanilla *et. al.* 2015). In this study we consider that activities can be preempted at any time and we want to determine the set of Pareto optimal schedules from the viewpoint of makespan and total number of preemptions.

We address the scenario in which resource availability is time-varying. This feature is inspired by a real case study in which the resources correspond to personnel from distinct company departments. These resources are not constantly available over time because the same unit may be temporarily involved in other projects. Time intervals during which each resource is available for the current project are known in advance.

2 ILP formulation

In order to solve this bicriteria problem, namely minimizing makespan and number of preemptions, new ILP formulations are proposed. We consider the problem in which activities can be interrupted and resumed later with no losses. Each activity requires a definite number of resources (of one or more types) throughout its execution. In each time slot, there is a limit on the available resources of each type, which cannot be exceeded by the total resource requirement for that time slot.

A project is represented by an acyclic graph $G(V, E)$, in which the set of $|V| = n$ nodes corresponds to activities and each arc $(i, j) \in E$ indicates that activity j can only start after activity i is completed. Time is discretized in time slots, i.e., time slot t starts at time $t - 1$ and ends at time t. There is a set K of renewable resources. Each activity i requires an integer number d_i of time slots (*duration*) to be performed and a_{ki} units of resource k ($k \in K$) during each time slot throughout its execution. During time slot t, there are n_{kt}

units of resource k available, so the total usage of resource k in each time slot t must not exceed n_{kt} ($t = 1, \ldots, T$, $k \in K$).

We present a formulation (MIN_PRMP) in which the objective is to minimize the number of preemptions with the constraint that project makespan cannot exceed a value Q. We let $x_{it} = 1$ if activity i is being executed during time slot t.

$$\min \sum_{i=1}^{n} \sum_{t=1}^{T-1} z_{it} \tag{1}$$

$$\sum_{t=1}^{T} t x_{nt} \leq Q \quad \forall t = 1, \ldots, T \tag{2}$$

$$z_{it} \geq x_{it} - x_{i,t+1} \quad \forall i \in V, \quad t = 1, \ldots, T-1 \tag{3}$$

$$\sum_{t=1}^{T} x_{it} = d_i \quad \forall i \in V \tag{4}$$

$$d_i x_{j,t+1} \leq \sum_{q=1}^{t} x_{iq} \quad \forall (i,j) \in E, \quad t = 1, \ldots, T \tag{5}$$

$$\sum_{i=1}^{n} a_{ki} x_{it} \leq n_{kt} \quad \forall k \in K, \quad t = 1, \ldots, T \tag{6}$$

$$x_{it}, z_{it} \in \{0, 1\} \qquad \forall i \in V, \quad t = 1, \ldots, T \tag{7}$$

The makespan constraint is modeled by (2). Constraints (3) set z_{it} variables to 1 exactly when $x_{it} = 1$ and $x_{i,t+1} = 0$, i.e., when the activity is either completed or preempted at time t. Activity duration is enforced by constraints (4). Precedence relations between activities are modeled by (5). In fact, if i must precede j ($(i,j) \in E$), i must have been in execution for d_i periods before j can start. In (6) we impose that the total amount of each resource in use in each time slot t does not exceed its availability in that time slot. Variables z_{it} allow counting the number of preemptions. In fact, for a given feasible schedule, the actual number of preemptions is $\sum_{i=1}^{n} \sum_{t=0}^{T-1} z_{it} - n$, which explains the objective function (1). These variables can be discarded when computing the two extreme PO schedules, respectively the minimum-makespan nonpreemptive schedule and the minimum-makespan schedule with an unlimited number of preemptions.

If preemptions are beneficial, each Pareto front has two extreme values of Q: the minimum nonpreemptive makespan and the minimum makespan with an unlimited number of preemptions. To compute these two extreme values two formulations are used (NON_PRMP and INF_PRMP), derived from Kaplan (1988).

3 Computational experiments

The experiments were conducted using a 12th Gen Intel(R) Core(TM) i9-12900 processor running at 2.40 GHz, with 128 GB of RAM. The operating system utilized was Windows Server 2022 Standard version 21H2. The formulations were implemented in Python 3.11 within PyCharm 2022.3.2 environment. The Gurobi solver (version 10.0.0) was employed for optimization.

We conducted two distinct experiments. First, we applied the ILP formulations to a set of benchmark instances, namely the 480 instances of the J30 set of PSPLIB. These instances (for all of which $n = 30$) were appropriately modified to simulate various resource availability scenarios. The second experiment concerns a practical application within an IT company, involving a real-life project.

In the first experiment, we consider seven different scenarios concerning how resources vary over time. Scenarios are characterized by parameters such as resource variation intensity (i.e., the percentage decrease of resource availability) with respect to the baseline scenario, variation lag (i.e., the minimum time distance between two variation intervals) and variation length.

By applying the ILP formulations to the benchmark we were able to obtain the whole set of Pareto optimal solutions. This allows to assess how many Pareto optimal solutions arise and how much the makespan benefits from activity preemption.

Various insights can be obtained from the computational experiments, including:

– The utility of preempting some activity is strictly related to intensity variation. In 26% of the instances in the baseline scenario preemption is useful, and this figure grows to roughly 50% for 20% of intensity variation, and 70% when intensity variations are higher.
– The benefit of preemption increases with intensity and frequency variation. In the baseline scenario, the average makespan gain of a preemptive solution (with respect to an optimal nonpreemptive schedule) is 2.9%. The impact of preemption is higher as both variation frequency and intensity grow. Considering a variation intensity of 20%, the gain is between 4 and 5%, while for greater intensity variation (40% decrease with respect to the baseline) it is higher than 6%.
– Similarly, the count of PO solutions increases with variation intensity. As long as variation intensity is 20%, in more than 70% of the instances the Pareto set consists solely of two PO solutions, while this occurs only in 50% of the instances in scenarios with higher intensity variation. While in the baseline scenario the average size of the Pareto front is 1.3, it is 1.8 and 2.3 in the other two scenarios.
– With respect to CPU times, for 365 out of 480 instances the whole Pareto set was computed in the baseline scenario, using a time limit of 1000 seconds for each ILP. The greater the intensity variation, the higher the runtimes. In fact, the above figure decreases to 325/480 and 296/480 for medium and large intensity variation respectively. However, when resource variation follows an irregular pattern, the problems are solved faster.
– Most often, the *first* preemption leads to the most significant marginal improvement on makespan, compared to subsequent preemptions (if any). This outcome remains consistent across all scenarios, although the gain is more apparent as variation intensity increases. Apart from very few exceptions, subsequent preemptions display decreasing marginal returns, as typical of limited resource settings.
– Considering all the PO schedules computed in all instances of each scenario, the number of PO schedules in which there is at least one activity which is preempted *more than once* ranges between 5% (variation intensity of 40%) and 13% (in the baseline scenario). This suggests that if a constraint is enforced on the maximum number of preemptions *for each single activity* (as in Ballestín *et. al.* (2009), Ma *et. al.* (2022), Zhu *et. al.* (2011)), for most of the instances derived from J30, setting this bound to 1 is equivalent to allowing unlimited preemptions.

For the second experiment, we applied the proposed approach to a real-life project arising in a pharmaceutical machinery manufacturer based in Siena, Italy, specialized in the design of machinery for the fill-finishing process of injectable drugs. The machines produced have a modular design, which offers the advantage of easy customization. Correspondingly, the production project follows a standard template, in which some activities depend on the specific customization of the machine being produced. As such activities typically take place before project starts, here we consider a *standard project* including all the activities which are always present in any actual project. The benefits achieved in terms of makespan reduction on the standard project are typically applicable to any customer order. Due to the long time span required by the project, the time unit adopted for planning purposes is the working week, i.e., 5 days, and hence this is also the minimum duration of a task. In this case study there are four resource types, corresponding to the company business units. Resources are renewable and one unit corresponds to one employee of the respective department. Resource availability is determined by staff commitments to other pre-scheduled projects, hence the availability of resources varies over time and it is known in advance. The project instance consists of 78 activities, each having up to three predecessors. A limited number of activities cannot be interrupted for technical reasons. We account for this by adding to the model a non-preemption constraint for each of them.

The planning time horizon that the project is expected not to exceed is $T = 100$ working weeks. This limit is based on previous experience on similar projects. The experiments show that in this instance the Pareto front consists of four solutions, corresponding to 0, 1, 2 and 3 preemptions respectively. Despite the relatively large number of activities, the model has been solved in few seconds of computation.

The results show that without preemptions the project makespan is 99 weeks, and that the possibility to interrupt activities presents a significant opportunity to decrease the overall project duration. The largest *marginal* gain (6 weeks) is achieved by the second preemption. Allowing three preemptions achieves a total gain of 11 weeks on makespan, i.e., 11%. In this example, preemptions occur due to variations in resource availability.

Finally, since many project activities have unit duration, they are treated as non-interruptible by the model. Adopting a shorter time unit (e.g., day), the duration of some activities might be expressed in greater detail, and hence further gains might be obtained, but the resulting schedule would be inherently less robust, especially on a long time span.

Acknowledgments

This research was supported by the Italian Ministry of University and Research, grant PNRR - Next Generation EU - THE - Spoke 10 - CUP: B63C22000680007.

References

Ballestín F., V. Valls and S. Quintanilla, 2009, “Scheduling projects with limited number of preemptions", *Computers and Operations Research*, Vol. 36, pp. 2913-2925.

Kaplan L.A., 1988, “Resource-constrained project scheduling with preemption of jobs", *Doctoral dissertation, University of Michigan*, University of Michigan Library.

Kolisch R., A. Sprecher, 1997, “PSPLIB - A project scheduling problem library", *European Journal of Operational Research*, Vol. 96, pp. 205-216.

Ma Z., M. Ning and Y. Wang, 2022, “Proactive Project Scheduling With Activity Splitting and Resource Transfer Times Under Uncertain Environments", *IEEE Access*, Vol. 10, pp. 87490-87499.

Moukrim A., A. Quilliot and H. Toussaint, 2015, “An effective branch-and-price algorithm for the Preemptive Resource Constrained Project Scheduling Problem based on minimal Interval Order Enumeration", *European Journal of Operational Research*, Vol. 244, pp. 360-368.

Quintanilla S., P. Lino, Á. Pérez, F. Ballestín and V. Valls, 2015, “Integer preemption problems", *Handbook on project management and scheduling*, Vol. 01, pp. 231-250, Springer, Berlin.

Zhu J., X. Li and W. Shen, 2011, “Effective genetic algorithm for resource-constrained project scheduling with limited preemptions", *International Journal of Machine Learning and Cybernetics*, Vol. 2, pp.55-65.

Mixed-Integer Programming Models for Mid-Term Production Planning in Integrated Steel Production

Jonas Saupe, Philipp Fath and David Sayah

FZI Research Center for Information Technology, Karlsruhe, Germany
{saupe,fath,sayah}@fzi.de

Keywords: Mixed-Integer Linear Programming, Due-Date Management, Applications, Make-to-Order, Steel Industry.

1 Introduction and application context

This work presents a case study concerned with automating a recurring planning task at a German steel manufacturing company. The planning task is situated in a make-to-order environment. Operations managers face this task attempting to align limited production capacities of multiple production facilities with booked customer orders over a mid-term planning horizon. Similar tactical planning tasks appear frequently in various operations and supply chain management contexts, and they are often referred to as *master planning* (Mönch *et al.* 2017) or *supply network planning* (Neumann *et al.* 2002).

As a process industry (Schwindt and Trautmann 2000), integrated steel production can be seen as a sequence of successive transformation processes (*operations*), e.g., iron making, continuous casting, hot rolling, or cutting. It is further characterized by a variety of products such as raw materials (iron ore, coal, etc.), semi-finished products (pig iron, molten steel, slabs, etc.), and finished products (plates, sheets, long products, etc.) that are being routed through a network of processing units (blast furnace, continuous caster, hot rolling mill, etc.) and storage facilities (a slab yard, for instance) (Dutta and Fourer 2001). The production capabilities of a plant often allow for multiple sequences of operations (*process plans*) to manufacture a given customer order (*job*).

Specifically, we address the planning problem which consists in selecting a process plan mix (set of selected process plans) and a production schedule (set of selected start times) that fulfills the current order book (set of jobs). Under a regular objective function (e.g., makespan minimization), this problem is known as the *process planning and scheduling problem* (PPSP). A feasible process plan mix contains a unique process plan for each job. A feasible schedule has to respect the limited capacities of all processing units (*machines*) at any time. We tackle a variant of this problem with no-wait requirements and a non-regular objective function. Our contribution consists of two compact *mixed-integer programming* (MIP) models. The first one is a special case of an existing PPSP formulation. The second one is a knapsack-type model. We conducted experiments with an MIP solver based on instances derived from a real-world data set.

The remainder of this paper is organized as follows. We formally define the planning problem in Section 2 and sketch our MIP models in Section 3. Section 4 and 5 provide, respectively, preliminary computational results and a conclusive outlook on future work.

2 Problem statement

2.1 General assumptions

Due to the tactical nature of the problem at hand, we consider a planning horizon of one year that is discretized into time buckets (*periods*) each representing, e.g., a week of the year. In accordance with the current planning process implemented by the company, resource capacities are aggregated on a weekly basis. Further, it is assumed that the full capacity requirement of an operation applies to the period in which its execution ends.

Furthermore, operations are executed *without preemption* and delays between successive operations are forbidden (*no-wait requirement*). Some machines might be temporarily unavailable for production due to, e.g., maintenance work, according to a given *calendar*. In this case, production is halted and resumed as soon as the maintenance work is completed.

Finally, there are pre-defined *due window requirements* associated with so-called milestone operations. Milestone operations of the selected process plan should finish within their due window. Otherwise, a milestone operation incurs earliness or tardiness cost. For instance, relevant milestones in steel manufacturing are the production start, milling of semi-finished products, and shipping of finished products to customers. The planning objective is to determine a process plan mix and schedule matching the respective due windows as closely as possible.

2.2 A conceptual model

Let $\mathcal{T}$, $\mathcal{J}$, and $\mathcal{M}$ denote the set of planning periods, jobs, and machines, respectively. The set of process plans, or, (production) *routes*, is denoted by Π. The subset $\Pi_j \subseteq \Pi$ contains only routes eligible for job $j \in \mathcal{J}$ so that $\Pi = \cup_{j\in\mathcal{J}}\Pi_j$. A route $\pi \in \Pi$ is a sequence $(o_{\pi,1}, \ldots, o_{\pi,n_\pi})$ of n_π operations. Let $\mathcal{O}_\pi$ denote the set of operations in route $\pi \in \Pi$ and let $\mathcal{O} = \cup_{\pi\in\Pi}\mathcal{O}_\pi$. Processing an operation $o \in \mathcal{O}_\pi$ requires a (net) duration of $p_o \geq 0$ and r_o capacity units of machine $m_o \in \mathcal{M}$. The capacity of machine $m \in \mathcal{M}$ in period $t \in \mathcal{T}$ is given by R_{mt}. We assume any two of the index sets in $\{\Pi_j : j \in \mathcal{J}\}$ to be pairwise disjoint; the same holds for $\{\mathcal{O}_\pi : \pi \in \Pi\}$. Hence, any parameters associated with route π or operation o can be uniquely identified by the respective index.

For each route $\pi \in \Pi$, the subset $\Omega_\pi \subseteq \mathcal{O}_\pi$ represents its milestone operations, indexed by k. A due window $[e_k, l_k]$ is defined for each milestone operation $k \in \Omega_\pi$. We refer to the start e_k and end l_k of a due window as the earliest and latest due date for completing milestone k, respectively.

A start time S_j and a route π_j have to be selected for each job $j \in \mathcal{J}$. To incorporate the given machine calendars, we compute a cumulative duration $\tilde{p}_{ot}$ for completing route $\pi \in \Pi$ up to (including) operation $o \in \mathcal{O}_\pi$ if production starts in t. Hence, the cumulative duration $\tilde{p}_{ot}$ contains waiting times, e.g., due to machine shutdowns. (Adjusting the durations in this way is known as *calendarization* in project scheduling.) Thus, the completion time $C_o(S_j, \pi_j)$ of an operation $o \in \mathcal{O}_{\pi_j}$ can be determined purely based on the start period S_j and route π_j selected for job j using the cumulative durations $\tilde{p}_{ot}$. In particular, this holds for milestones $k \in \Omega_{\pi_j}$.

By comparing the completion time of a milestone $k \in \Omega_{\pi_j}$ to its due window $[e_k, l_k]$, earliness and tardiness values are obtained, respectively, as $E_k(S_j, \pi_j) = [e_k - C_k(S_j, \pi_j)]^+$ and $T_k(S_j, \pi_j) = [C_k(S_j, \pi_j) - l_k]^+$, where $[x]^+ := \max\{0, x\}$ for any $x \in \mathbb{R}$. For given start times $\mathbf{S} = (S_j)_{j\in\mathcal{J}}$ and selected routes $\boldsymbol{\pi} = (\pi_j)_{j\in\mathcal{J}}$, the overall earliness-tardiness is defined as $ET(\mathbf{S}, \boldsymbol{\pi}) := \sum_{j\in\mathcal{J}} \sum_{k\in\Omega_{\pi_j}} (E_k(S_j, \pi_j) + T_k(S_j, \pi_j))$. Furthermore, the utilization $r_{mt}(\mathbf{S}, \boldsymbol{\pi})$ of machine m in period t is uniquely determined. Altogether, we can state the following conceptual model for the *no-wait earliness-tardiness process planning and scheduling problem* (nw-ET-PPSP):

$$\min_{\mathbf{S}\in\mathcal{T}^{|\mathcal{J}|}, \boldsymbol{\pi}\in \bigtimes_{j\in\mathcal{J}} \Pi_j} \left\{ ET(\mathbf{S}, \boldsymbol{\pi}) : r_{mt}(\mathbf{S}, \boldsymbol{\pi}) \leq R_{mt}, m \in \mathcal{M}, t \in \mathcal{T} \right\}. \tag{1}$$

Problem (1) is about selecting start times and routes so as to minimize overall earliness-tardiness while respecting time-dependent capacity limits and calendars of all machines.

3 Modeling approach

We formulate the problem described in Section 2 as an MIP model. There are two reasons for this. First, the model presented in this paper can be generalized to account for additional requirements. These requirements can be flexibly incorporated in an MIP-based approach. Some examples are addressed in the outlook (Section 5). Second, in tactical planning, the time required to determine a solution is usually not critical. This makes exact MIP-based approaches favorable over approximate methods. Next, we outline two formulations for the nw-ET-PPSP stated in (1).

Job shop model (JS). This model contains binary variables $s_{\pi t}$ and $x_{j\pi}$ expressing, respectively, whether route $\pi \in \Pi$ starts in period $t \in \mathcal{T}$ and whether route $\pi \in \Pi_j$ is selected to produce job $j \in \mathcal{J}$. Two sets of continuous variables are further used to measure earliness and tardiness with respect to milestones. There are five sets of constraints. First, it is imposed that one route is selected for each job. Second, a single start period must be chosen for a route if and only if it is selected. A third set of constraints implements the capacity limits for all machines and periods. The last two sets of constraints define the earliness and tardiness variables. Similar formulations have been used in the job shop scheduling literature (Kim and Egbelu 1999, Baptiste *et al.* 2017).

Knapsack model (KP). To further exploit the inherent combinatorial structure of the problem at hand, we cast nw-ET-PPSP as a multi-dimensional multiple-choice knapsack problem based on the following observations: First, the item set N is partitioned into subsets $N_1, \ldots, N_{|\mathcal{J}|}$ of route-start combinations for each job, i.e., $N_j := \{(\pi, t) : \pi \in \Pi_j, t \in \mathcal{T}\}$ $(j \in \mathcal{J})$, and it is required that exactly one such combination (item) per subset is selected. Second, selecting an item implies a certain amount of additional capacity utilization for each machine and period (possibly zero). Moreover, one can pre-compute for each job the earliness/tardiness implied by a route-start combination. Indeed, the knapsack formulation contains only one set of binary variables representing item selection. This allows us to eliminate redundant binary and continuous variables used in the JS formulation.

4 Computational results

To conduct preliminary numerical tests, we generated ten problem instances based on real-world data from the steel manufacturer. These instances contain 250, 500, or 1000 jobs each to be scheduled within a time horizon of one year. For each job, one or two routes are available. In contrast to most benchmark instances from the literature, the durations of operations in the given data are not necessarily integer. We used a computer with 32 GB RAM and AMD Ryzen 7 Pro 2.70 GHz CPU. The models were implemented in Java v20.0.1 and solved using SCIP v8.0.4. A time limit was set to 600 seconds per instance.

The results of our computational evaluation are summarized in the table below. The problem instances are grouped by the number of jobs. Each instance group is described by its size (#), the mean number of routes per job ($|\Pi|/|\mathcal{J}|$) and the mean number of operations per route ($|\mathcal{O}|/|\Pi|$). Furthermore, we provide the average number of binary variables (*#BinVars*) and constraints (*#Cons*) (rounded) for either of the model formulations. To assess the performance of the model formulations, we report the following three measures: First, column *Opt(%)* contains the percentage of instances for which an optimal solution was determined within the time limit. Second, column *Gap(%)* reports the average MIP gap (in percent), computed as $(UB - LB)/UB$, where UB and LB, respectively, denote the best upper and lower bound determined by the solver. Last, in column *TimeOpt(s)*, we report the average time (in seconds, rounded) required to determine an optimal solution, accounting only for instances for which the time limit was not reached.

$\lvert\mathcal{J}\rvert$	#	$\frac{\lvert\Pi\rvert}{\lvert\mathcal{J}\rvert}$	$\frac{\lvert\mathcal{O}\rvert}{\lvert\Pi\rvert}$	**#BinVars**		**#Cons**		**Opt(%)**		**Gap(%)**		**TimeOpt(s)**	
				JS	KP	JS	KP	JS	KP	JS	KP	JS	KP
250	6	1.36	10.88	17,561	17,223	5,598	3,907	83	100	0	0	30	1
500	3	1.35	10.84	35,122	34,446	7,539	4,157	33	67	190	1	4	2
1000	1	1.33	10.85	69,244	67,910	11,327	4,657	0	0	7	3	-	-

In the table above, it can be observed that the KP model, which is more compact in terms of the number of constraints, outperforms the JS model both with respect to the average gap and with respect to the average solution time. The average gap is about 88 times smaller when using the KP model, compared to the JS model. Furthermore, two additional instances are solved to optimality within the time limit.

5 Conclusions and future work

In this paper, we considered a master planning case study in integrated steel production. We described two MIP formulations of the problem and evaluated them using instances derived from real-world data. Our preliminary evaluation indicates that both models can be applied to problems of practically relevant size. For medium-sized and large instances, the knapsack formulation outperforms the job shop formulation in terms of the percentage of instances solved to optimality and average optimality gaps.

As indicated by the company, future work should focus on a more general process plan structure and on integration with related planning tasks such as transportation planning. For example, one has to account for concurrent material testing procedures that could potentially delay the processing of a job. Moreover, during production planning, operations managers at the company already have in mind how many transportation devices (e.g., ships or trucks) need to be dispatched in order to deliver the scheduled customer orders.

References

P. Baptiste, M. Flamini and F. Sourd, 2008, “Lagrangian bounds for just-in-time job-shop scheduling", *Computers & Operations Research,* Vol. 35(3), pp. 906–915.

G. Dutta and R. Fourer, 2001, “A survey of mathematical programming applications in integrated steel plants", *Manufacturing & Service Operations Management,* Vol. 3(4), pp. 387–400.

K.-H. Kim and P. Egbelu, 1999, “Scheduling in a production environment with multiple process plans per job", *International Journal of Production Research,* Vol. 37(12), pp. 2725–2753.

L. Mönch, R. Uzsoy and J. W. Fowler, 2017, “A survey of semiconductor supply chain models part III: master planning, production planning, and demand fulfilment", *International Journal of Production Research,* Vol. 56(13), pp. 4565–4584.

K. Neumann, C. Schwindt, and N. Trautmann, 2002, “Advanced production scheduling for batch plants in process industries", *OR Spectrum,* Vol. 24(3), pp. 251–279.

C. Schwindt and N. Trautmann, 2000, “Batch scheduling in process industries: an application of resource-constrained project scheduling", *OR Spectrum,* Vol. 22(4), pp. 501–524.

Bilevel scheduling of uniform parallel machines in the context of coupling maintenance and scheduling decisions

Schau Q.[1],Ploton O.[1],T'kindt V.[1] and Della Croce F.[2]

[1] Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée, Tours, France,
{quentin.schau,olivier.ploton,tkindt}@univ-tours.fr
[2] DIGEP, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, CNR, IEIIT, Torino, Italy.
federico.dellacroce@polito.it

Keywords: Scheduling, bilevel optimization, parallel machines

1 Introduction

Recent advancements in technology, such as cloud computing and the Internet of Things (IoT), have enabled the efficient management of data and information sharing. These technological developments have given rise to a new industrial era known as *Industry 4.0* (I4.0)(Chen et al. 2018). I4.0 introduces Cyber-Physical Systems (CPS) that monitor physical processes and enable the development of smart factories, including automated scheduling (Serranoruiz et al. 2021). In this paper, we focus on *Zero-Defect Manufacturing* (ZDM), a quality improvement approach grounded in data-driven methods. Specifically, we delve into a process-oriented variant of ZDM that incorporates the health status of production machines into scheduling decisions. For instance, ZDM can alleviate a machine's workload if it predicts an impending failure or, if maintenance is required, it can reschedule tasks accordingly. In essence, this approach involves the intertwining of maintenance issues with scheduling.

The ZDM framework can be represented as a bilevel optimization problem. In this setup, the leader manages system health and global scheduling, while the followers are responsible for multiple CPSs. The leader makes decisions first and hands over a subset of jobs to the followers, each one scheduling the tasks it has been assigned. Notably, both the leader and followers can only evaluate their objective function when both have taken their decisions. This bilevel optimization structure introduces complexities as the leader and followers may have distinct conflicting objectives (Dempe et al. 2015). Notably, the followers' problem can have multiple optimal solutions, leading to two categories of bilevel problems: *optimistic* and *pessimistic* (Dempe 2003). In the optimistic scenario, the follower selects, among its optimal solutions the one that leads to the smallest value of the leader's objective function, whereas in the pessimistic case, the follower chooses the one that is the worst for the leader.

Lately, to the best of our knowledge, the literature on bilevel scheduling is relatively limited (T'kindt et al. 2023).

This study focuses on a scenario with one leader and one follower (CPS) composed of parallel uniform machines where scheduling decisions occur periodically. The problem is introduced in Section 2, while complexity results and solution approaches are sketched in Section 3.

2 Optimistic bilevel parallel uniform machines scheduling

Assume we are given a set $\mathcal{J}$ of N jobs, each job j being characterized by its processing time p_j, weight w_j and due date d_j. We have a leader that manages the system health and global scheduling and a CPS, composed of parallel uniform machines, which schedules the jobs given by the leader. The leader aims to minimize the number of weighted tardy jobs while the follower aims to minimize the total completion time. We consider the optimistic case, i.e. the follower returns the schedule that leads to the minimum number of weighted tardy jobs among schedules optimal for the total completion time. So, the bilevel problem can be defined as:

$$\begin{aligned} &\min_{\substack{\mathcal{I}\subset\mathcal{J}\\ |\mathcal{I}|=n}} \sum_{j\in\sigma^*} w_j U_j^L \\ &\text{st.} \\ &\qquad \sigma^* = \operatorname*{argmin}_{\sigma\in\mathfrak{S}_{\mathcal{I}}} \left(Lex\left(\sum_{j\in\sigma} C_j^F, \sum_{j\in\sigma} w_j U_j^L \right)\right) \end{aligned} \tag{1}$$

where:

- $\mathfrak{S}_{\mathcal{I}}$ is the set of schedules form of all jobs in $\mathcal{I}$.
- C_j^F is the completion times of the job j.
- $U_j^L = \begin{cases} 1 \text{ if } C_j^F > d_j \\ 0 \text{ otherwise} \end{cases}$
- $Lex\left(\sum_{j\in\sigma} C_j^F, \sum_{j\in\sigma} w_j U_j^L\right)$ stands for minimizing the first criterion $\sum_{j\in\sigma} C_j^F$ first, then the second criterion $\sum_{j\in\sigma} w_j U_j^L$ if there is a tie in the first criterion.

We assume that the leader periodically makes scheduling decisions every T units of time. Prior to each period, the leader receives data from the shop floor. Using this data, the leader faces three cases and makes related decisions:

- If one or more machines are not defective, the leader sets the machine speed to the maximum, i.e. V_{max}.
- If one or more machines are predicted to be defective, the leader reduces the machine speeds to the minimal speed V_0 to minimize the risk of breakdowns.
- If one or more machines are confirmed to be defective, the leader removes those machines from the follower for maintenance. The follower can no longer assign jobs to these machines for the scheduling period.

Next, the leader selects a subset $\mathcal{I} \subset \mathcal{J}$ of n jobs and assigns them to the follower. In the follower's problem, there is a lexicographical objective function. Initially, the follower minimizes the total completion time and subsequently minimizes the number of weighted tardy jobs.

Correspondingly, following the three-field classification (Graham et al. 1979), we denote this bilevel problem as $Q|V_i \in \{V_0, V_{max}\}, OPT - n| \sum_j C_j^F, \sum_j w_j U_j^L$, while the related follower problem is denoted as $Q|V_i \in \{V_0, V_{max}\}|Lex\left(\sum_j C_j^F, \sum_j w_j U_j^L\right)$.

3 Complexity and solution approaches

We show in the following section several complexity results (proofs are omitted and will be presented at the time of the conference). With respect to the complexity of problem $Q|V_i \in \{V_0, V_{max}\}|Lex\left(\sum_j C_j^F, \sum_j w_j U_j^L\right)$, the following theorem holds.

Theorem 1. *The problem $Q|V_i \in \{V_0, V_{max}\}|Lex\left(\sum_j C_j^F, \sum_j w_j U_j^L\right)$ (and correspondingly of problem $Q|V_i \in \{V_0, V_{max}\}, OPT - n|\sum_j C_j^F, \sum_j w_j U_j^L$) is $\mathcal{NP}$-hard in strong sense by reduction from the numerical 3-dimensional matching ($NUM - 3DM$).*

The answer to the $NUM - 3DM$ problem is true iff all jobs can be scheduled on time ($\sum_j w_j U_j = 0$), as a consequence, the problem is not approximable in polynomial or pseudo-polynomial time unless $\mathcal{P} = \mathcal{NP}$.

When the number of machines m is not part of the input, we can already show that problem $P2||Lex\left(\sum_j C_j^F, \sum_j U_j^L\right)$ is $\mathcal{NP}$-hard by reduction from the well known even-odd partition problem. To complete the complexity analysis, we can even prove the following theorem by providing a dynamic programming pseudo-polynomial algorithm.

Theorem 2. *$Qm|V_i \in \{V_0, V_{max}\}|Lex\left(\sum_j C_j^F, \sum_j w_j U_j^L\right)$ with m constant are $\mathcal{NP}$-hard in the ordinary sense.*

We remark, first, that problem $Q||\sum_j C_j$ is polynomial (Conway, Maxwell, Miller 1967) and that the algorithm we propose is based on a characterization of the set of solutions of the problem $Q|V_i \in \{V_{max}, V_0\}|\sum_j C_j$.

Assume we have a set $\mathcal{J}$ of n jobs, let N_{max} (respectively N_0) be the number of machines with high-speed V_{max} (respectively, with low speed V_0). Assume that V_0 divide V_{max}. We can show that an optimal schedule of the problem $Q|V_i \in \{V_{max}, V_0\}|\sum_j C_j$ is given by the repetition of patterns. In Figure 3, an illustration depicts a pattern consisting of $\alpha - 1$ blocks on high-speed machines and one block on low-speed and high-speed machines. Jobs within the blue-dashed group signify their membership to a block and can subsequently undergo permutation without change the value of $\sum_j C_j$. Let us define $\mathcal{B}_h$ a block that can be assigned to high-speed machines or low-speed machines.

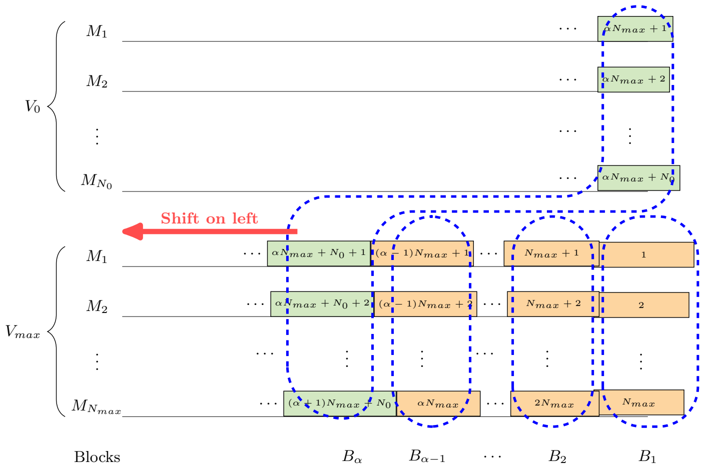


Fig. 1. Scheduling pattern with block structure for the optimal solution of $Q|V_i \in \{V_{max}, V_0\}|\sum_j C_j$

Hence, we know for all jobs j on which block $\mathcal{B}_h$ it is scheduled. Using this characterization, by extending the approach on parallel machines scheduling proposed in (Hall et al. 2001) to the considered problem, a dynamic programming recursion can be formulated to solve $Q|V_i \in \{V_0, V_{max}\}, OPT-n|\sum_j C_j^F, \sum_j w_j U_j^L$ by means of function OPT.

$$OPT(\vec{\mathcal{A}}^0, \mathcal{A}^{\vec{max}}, \vec{C}^0, C^{\vec{max}}, j, n') \tag{2}$$

where:

- j is the job to add to the schedule under construction
- n' the number of jobs that already scheduled
- $\vec{C^\alpha}$ is the vector of completion times for the machines with the speed α
- $\vec{\mathcal{A}}^\alpha$ is the vector indicating at the current recursion the available machines with speed α.

For each machine i we have $\vec{\mathcal{A}}_i^\alpha = \begin{cases} \text{True if the job } j \text{ can be schedule on the machine } i \\ \qquad\qquad \text{with the speed } V_\alpha \\ \text{False otherwise} \end{cases}$

Here, OPT returns the minimal value of $\sum_{k=1}^n w_k U_k$ when jobs $\{1, \ldots, j\}$ have been scheduled and machines complete at times $\vec{C^0}$ and $C^{\vec{max}}$. By computing $OPT(T\vec{rue}, T\vec{rue}, \vec{C^0}, C^{\vec{max}}, n)$ for all relevant $\vec{C^0}$ and $C^{\vec{max}}$, the following lemma holds.

Lemma 1. *A dynamic programing algorithm find an optimal solution for the problem $Q|V_i \in \{V_0, V_{max}\}, OPT-n|\sum_j C_j^F, \sum_j w_j U_j^L$ with the time complexity $\mathcal{O}\left(4^m \left(\sum_j p_j\right)^m \times N \times n^2 log(n)\right)$*

In addition, a MIP can be formulated for the bilevel problem that will be presented in details at the time of the conference.

References

Conway, R. W., Maxwell, W. L., Miller, L. W. 1967, Theory of Scheduling, Addison-Wesley Publishing Company

Graham, R. L., Lawler, E. L., Lenstra, J. K., Kan, A. R. , 1979, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Journal Title*, Vol. 5, pp. 287–326.

Hall, N. G. , Lesaoana, M., Potts, C. N. : Scheduling with fixed delivery dates. Operations Research. Vol 49, pp134–144 .

Dempe, S.: Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. Optimization. 52, 333–359 (2003).

Dempe, S., Kalashnikov, V. Perez-Valdes, G.A., Kalashnikova, N., 2015, "Bilevel programming problems", *Springer*.

Chen, B., Wan, J., Shu, L., Li, P., Mukherjee, M., Yin, B. 2017, Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges,IEEE Access, Vol. 6, pp. 6505-6519

Serrano-Ruiz, J. C., Mula, J., Poler, R., 2021, Smart manufacturing scheduling: A literature review, Journal of Manufacturing Systems, Vol. 61, pp. 265-287

T'kindt, V., Della Croce, F., Agnetis, A.: Adversarial bilevel scheduling on a single machine. European Journal of Operational Research, articles in press (available on line november 2023).

New procedure for the resource-constrained project scheduling problem with alternative subgraphs using satisfiability solvers

Tom Servranckx[1], José Coelho[1,2,3] and Mario Vanhoucke[1,4,5]

[1] Ghent University, Belgium
`tom.servranckx@ugent.be, mario.vanhoucke@ugent.be`
[2] INESC - Technology and Science, Portugal
[3] Universidade Aberta, Portugal
`jose.coelho@uab.pt`
[4] Operations and Technology Management Centre, Vlerick Business School, Belgium
[5] UCL School of Management, University College London, UK

Keywords: Project scheduling, Alternative subgraphs, Satisfiability, Genetic algorithm.

1 Introduction

Many complex scheduling problems are currently solved by academics worldwide in order to increase the body of theoretical knowledge and improve their practical applicability in the field of project management. However, the theoretical and practical focus of research often conflict and academics decide to direct their research efforts into one of the two directions. On the one hand, complex extensions of project scheduling problems are investigated by relaxing unrealistic assumptions. Frequently, these extensions are derived from company- or industry-specific cases and the collaboration with these companies need to result in solution methodologies that are capable of solving large-scale instances fast. As a result, many problem-specific (meta)heuristics are presented in the literature to create high-quality (but suboptimal) solutions. On the other hand, state-of-the-art exact approaches are continuously developed as the shared knowledge on effective solution approaches in the literature grows constantly and improved computational resources are made available to researchers at an increasing speed. Unfortunately, these methods are typically only capable of solving small-scale instances using long computational runs. A challenge of academic research remains the development of better, yet practically applicable, methods for complex and relevant scheduling problems.

In this study, a new application of an innovative solution approach is presented that has a lot of potential for obtaining high-quality solutions for complex project scheduling problems. The resource-constrained project scheduling problem (RCPSP) is a standard problem in the project scheduling literature that has been extensively studied in the last decades. The aim is to provide a timetable with starting times for activities in the project subject to limited resource availabilities and precedence relations between the activities such that the project duration or makespan is minimised. A restrictive assumption in the RCPSP is that the project network structure (i.e. the number of activities and their relations) and the activity characteristics (i.e. the activity durations and resource requirements) should be pre-defined and known prior to the project start. In practice, this might be impossible because of the multi-year scope of many projects and the uncertain project environment. Therefore, we will focus on an extension of the basic RCPSP, called the RCPSP with alternative subgraphs (RCPSP-AS) in which subsets of activities, labelled work packages, can be solved in alternative ways. A possible application of this problem is the existence of alternative technologies that can be used to execute a certain subpart (or work package) of the project in different ways, yet realising the same scope. As a result, the scheduling problem now consists of two subproblem: a scheduling and a selection subproblem. In the selection subproblem, a specific (alternative) technology needs to be selected for each work package, while the resulting selected activities need to be scheduled in the scheduling subproblem (which corresponds with the traditional RCPSP). The problem is introduced and formulated by Servranckx and Vanhoucke (2019) and these authors present a tabu search (TS) for solving the problem using problem-specific operators and an intelligent search process guided by the project properties. Subsequently, Nekoueian *et. al.* (2023) present a priority-rule based approach supported by an extensive search of different rules-of-thumb and several constructive heuristics.

We aim to investigate the potential of a new solution approach, namely a metaheuristic extended with a satisfiability (SAT) solver, for the RCPSP-AS. A boolean SAT solver is an algorithm for establishing satisfiability of binary decision variables that are connected by logical AND and OR relations. In order to solve the SAT problem, clauses need to be constructed that combines variables and their complements using a series of logical OR relations and these clauses are then combined through AND relations. To allow algorithmic search processes checking the satisfiability, this boolean logic should be converted to a standard form, known as the conjunctive normal form. The logic of the SAT solver is perfectly suited for solving the selection subproblem of the RCPSP-AS since binary decisions on the inclusion (=1) or exclusion (=0) of activities need to be made. The scheduling subproblem of the RCPSP-AS could also be solved using this logic since there already exist studies in the literature that solve the traditional RCPSP with a SAT solver (Horbach 2010) given that the time-indexed problem formulation uses binary decision variables to model the start time of each activity. Due to the explosive growth in the number of resulting clauses, however, these methods are typically only capable of solving small-scale instances. In this study, we therefore follow the logic proposed by Coelho and Vanhoucke (2011) to integrate the SAT solver into a genetic algorithm (GA) framework. More precisely, the GA is responsible for scheduling the activities, while the selection subproblem is assigned to the SAT solver.

2 Problem description

In the RCPSP-AS, we distinguish between *fixed activities* that should always be selected and scheduled, and *alternative activities* that could potentially be selected and scheduled. The set of alternative activities that are interchangeable and thus belong to the same work package is referred to as an *alternative subgraph*. Each subset of activities that together form a specific alternative in the alternative subgraph is referred to as an *alternative branch*. The objective of the RCPSP-AS is to select for each alternative subgraph exactly one alternative branch such that the resulting precedence, resource and logical feasible schedule has a minimal project makespan. However, two types of complex dependencies between alternatives can be modelled.

- *Linked alternative branches* indicate that (part of) the activities in a single alternative branch should be selected when another alternative branch is selected.
- *Nested alternative subgraphs* indicate that an entire alternative subgraph is triggered (and the corresponding choice) by the selection of an alternative branch.

In order to illustrate the different concepts and terminology, we show an illustrative example in Figure 1 where each curved line ')' marks a choice between alternative branches in an alternative subgraph. We observe two alternative subgraphs with two alternative branches each, of which one alternative subgraph is nested and one alternative branch is linked.

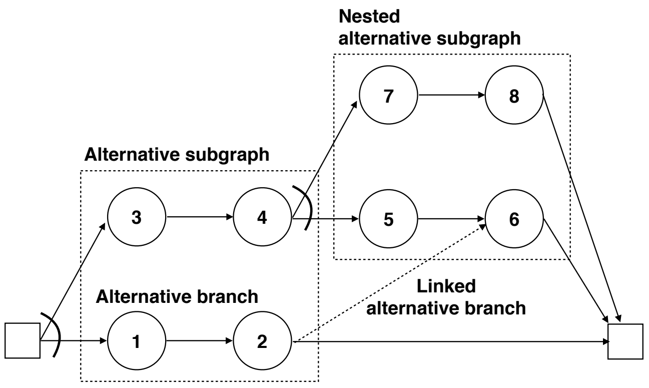


Fig. 1. Illustrative example of the RCPSP-AS

3 Methodology

Due to the interaction between the scheduling and selection subproblem in the complex RCPSP-AS, the search for a (near-)optimal solutions is hard. As already discussed before, metaheuristic frameworks are explicitly developed in order to solve such complex problems fast. However, the search for better schedules (in terms of minimising the project makespan) can be improved by slightly increasing the computational runtime in case that the metaheuristic framework is extended with a SAT solver. The search process will be decomposed, and a GA will be used for solving the scheduling subproblem and a SAT solver for solving the selection subproblem. Both methods are iteratively called to solve the RCPSP-AS and are briefly introduced below:

SAT: Several clauses will deal with the various combinations of alternative branches and the complex relations between them. First, the choices in the previous iteration of the method are reset by changing all variables to true. Then, a different subset of activities can be selected in order to create a satisfiable solution. Clause learning ensures that information obtained during earlier iterations is retained in order to improve the efficiency of future iterations. It represents an integral part of any SAT solver since this can avoid the occurrence of similar conflicts (i.e. unsatisfiability) throughout the search process.

GA: The solution of the selection subproblem in the previous iteration provides activities with a positive (or zero) duration in case the activity has (not) been selected. The selection of activities has been logically checked such that such the successors of non-selected activities also have a zero duration. Then, solutions are assigned to one of two separate populations: the left population containing left-justified schedules and the right population containing right-justified schedules (Valls *et. al.* 2005). The scheduling process aims to improve the elements of both populations by iteratively combining them using parent selection, crossover and mutation operators derived from Debels and Vanhoucke (2007).

Finally, we also investigate the option to eliminate certain alternative branches in the search procss by means of *preprocessing*. The resulting limited subset of alternative branches (i.e. output of the preprocessing) will be used as input for the solution method. The aim is to reduce the solution space and thus limit the computational complexity.

4 Computational results and conclusions

In order to test this integrated GA with SAT solvers, we use a large-scale artificial dataset (ASLIB-0) for the RCPSP-AS developed by Servranckx and Vanhoucke (2019) using a controlled design based on flexibility parameters such as the degree of linked alternative branches (%L) and the degree of nested alternative subgraphs (%N). The %L and %N indicate the actual number of linked alternative branches and nested alternative subgraphs relative to, respectively, the total number of alternative branches and subgraphs.

Based on the preliminary results, we can validate the good performance of the proposed approach for the complex RCPSP-AS as it is competitive with and even outperforms existing metaheuristics developed for the problem. In Table 1, we observe that the performance of the SAT solver performs better as more complex relations (links and nesting) are observed to a certain degree, as its performance again lowers for high levels of complexity. For low to medium levels of %N and %L, the added value of the clause learning is observed, while the large number of these clauses hinders the search process in case of very high levels of %N and %L. As expected, preprocessing has a negative impact on the overall solution quality since part of the solution space is ignored. However, it is interesting from a practical perspective that the increase in project makespan is rather limited. The exclusion of some (potentially good) alternative branches implies that more computational effort can be invested in finding good solutions for the remaining (smaller) set of alternatives. The theoretical knowledge derived from this study will also contribute to further academic developments in the field of (project) scheduling.

Table 1. Improvement (% Avg. makespan decrease compared to basic GA) for different levels of %N and %L (with/without preprocessing)

%N	Low			Medium			High		
%L	Low	Medium	High	Low	Medium	High	Low	Medium	High
Without preprocessing	1.95	5.70	3.35	1.85	5.97	4.02	1.90	4.08	2.95
With preprocessing	1.45	4.95	3.03	1.64	5.22	3.62	1.77	3.54	2.25

References

Coelho J. and M. Vanhoucke, 2011, “Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers”, *European Journal of Operational Research*, Vol. 55, pp. 73-82.

Debels D. and M. Vanhoucke, 2007, “A decomposition-based genetic algorithm for the resource-constrained project scheduling problems”, *Operations Research*, Vol. 55, pp. 457-469.

Horbach A., 2023, “A boolean satisfiability approach to the resource-constrained project scheduling problem", *Annals of Operations Research*, Vol. 181, pp. 89-107.

Nekoueian R., T. Servranckx and M. Vanhoucke, 2023, “Constructive heuristics for selecting and scheduling alternative subgraphs in resource-constrained projects", *Computers & Industrial Engineering*, Vol. 182, pp. 109399.

Servranckx T. and M. Vanhoucke, 2019, “A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs”, *European Journal of Operational Research*, Vol. 273, pp. 841-860.

Valls V., F. Ballestin and S. Quintanilla, 2005, “ Justification and RCPSP: A technique that pays", *European Journal of Operational Research*, Vol. 165, pp. 375-386.

Vector Bin Packing for Resource Management in Distributed Systems: A Case Study

Shakhlevich N.V.[1], Erlebach T.[2], Mommessin C.[3], Yang R.[4], Sun X.[1], Kumar S.[5], Xiao J.[6] and Xu J.[1]

[1] University of Leeds, UK
`nshakhlevich, scxs, j.xu@leeds.ac.uk`
[2] University of Durham, U.K.
`thomas.erlebach@durham.ac.uk`
[3] IMT Atlantique, Nantes Université, École Centrale Nantes, France
`clement.mommessin@imt-atlantique.fr`
[4] Beihang University, China
`renyu.yang@buaa.edu.cn`
[5] Leeds Beckett University, UK
`s.kumar@leedsbeckett.ac.uk`
[6] Alibaba Group, China
`junqing.xjq@alibaba-inc.com`

Keywords: applications, distributed computing, vector bin packing, heuristics.

1 Introduction

Many important resource provisioning problems in distributed computing are modelled as vector bin packing, often with additional constraints. The case study we consider deals with *long-running applications* (LRAs), which may occupy substantial capacity of cloud data centres, see, e.g., (Garefalakis *et al.* 2018), (Liu and Yu 2018), (Verma *et al.* 2015). In the underlying optimization models it is usually assumed that within a planning horizon LRAs are allocated permanently, consuming a certain number of resources of different types: CPU cores, RAM, GPUs, etc. Planning resource allocation can be performed before LRA deployment and it is aimed at minimizing the number of nodes to which LRAs are allocated. This scenario gives rise to the *vector bin packing* problem (VBP), a generalization of the bin packing problem which has been under study since the 70s.

In the VBP problem, there are n items $\mathcal{I} = \{1, 2, \dots, n\}$ and a set of bins. Each item $i \in \mathcal{I}$ is characterized by a d-tuple of sizes s_{ih} in dimensions $h = 1, \dots, d$. The bins are identical, with a given size c_h for each dimension h. In the context of distributing computing, bins represent compute nodes of a data centre, and dimensions h correspond to specific resource types of a node. Items $i \in \mathcal{I}$ represent LRAs with their demands s_{ih} in resources of type h. For example, s_{i1} and s_{i2} may represent the number of CPU cores and the number of units of memory needed for LRA i. The demands of any LRA cannot exceed the capacity of a node, i.e., $s_{ih} \le c_h$ for every LRA i and every resource type h.

In VBP, allocating a subset of items $\mathcal{I}' \subseteq \mathcal{I}$ to one bin is feasible if

$$\sum_{i \in \mathcal{I}'} s_{ih} \le c_h \quad \text{for each } h = 1, \dots, d. \tag{1}$$

In the context of allocating LRAs to compute nodes, the total demand of all LRAs assigned to the same node cannot exceed the node capacity in each dimension.

In our case-study, we explore construction heuristics for VBP (Section 2) and adjust them for solving an enhanced version of the problem, typical for allocation LRAs to cloud nodes (Section 3). In that version, there are multiple replicas of each LRA $i \in \mathcal{I}$, which need to be deployed. Affinity restrictions are defined for pairs of LRAs which replicas can be jointly co-located to the same node, but with some limits, or for pairs of incompatible LRAs, which cannot be co-located. If LRA i is restrictive to LRA j, then there is an integer affinity value a_{ij} which sets up an upper bound on the maximum number of replicas of j that can be co-located on a node where at least one replica of i is allocated.

The most recent surveys on approximation algorithms for VBP are presented in (Christensen *et al.* 2017), (Epstein and van Stee 2018), while the main survey on heuristics is (Panigrahy *et al.* 2011). In our work, we extend the algorithmic toolkit by grouping heuristics of similar nature in a systematic way, compiling a comprehensive list of heuristic parameters and developing new approaches. The VBP toolkit, which we call *VectorPack*, is additionally adjusted and tuned for solving the affinity-aware version of the VBP problem. The talk is based on our papers (Mommessin *et al.* 2023a) and (Mommessin *et al.* 2023b).

2 Heuristics for VBP

There are two traditional classes of the VBP algorithms: *item-centric* and *bin-centric*, which we discuss in Sections 2.1 and 2.2. Both approaches activate a new bin only if an item cannot be allocated using bins, which have been already activated. An alternative set of approaches outlined in Section 2.3 perform *multi-bin activation* and then attempts to allocate the items exploring the whole pool of activated bins. Considering a series of problems with different target number of bins m, the aim is to find the smallest m for which all items can be allocated.

The toolkit *VectorPack* is evaluated via computational experiments, providing guidelines for practitioners on selecting most appropriate algorithms depending on the features of datasets.

2.1 Item-centric approach

The item-centric approach stems from the classical algorithms originally proposed in the context of one-dimensional BP. Items are considered one-by-one and a current item is allocated to the "best" bin selected according to specially defined rules.

The performance of an algorithm essentially depends on ordering of items and bins. Since VBP is multidimensional, d-tuples of item sizes s_{ih}, $1 \leq h \leq d$, are converted into scalars $v(i)$ which represent for every item $i \in \mathcal{I}$ its *combined size*. Similarly, d-tuples of residual capacities of activated bins $B_k \in \mathcal{B}$ are converted into scalars $v(B_k)$ representing *combined residual capacity* $v(B_k)$ for every bin B_k.

Using combined size measures for items and residual capacities of the bins, the items and bins can be ordered. A current item i from the item list is allocated to the first bin, which fits that item, considering the ordered list of bins $\mathcal{B}$. If no bin can accommodate i, then a new bin is activated and added at the end of list $\mathcal{B}$. Since every allocation changes the residual capacity of the bin used, the ordering of list $\mathcal{B}$ requires updating after each packing of an item.

Typical priority rules for ordering $\mathcal{I}$ and $\mathcal{B}$ were originally formulated for one dimensional bin packing, and later on adopted for the multidimensional case of VBP, resulting in the VBP versions of the famous algorithms First Fit, First Fit Decreasing, Best Fit, Best Fit Decreasing, Worst Fit, Worst Fit Decreasing. Each of the six algorithms gives rise to a family of the VBP algorithms, which depend on the way the size measures $v(i)$ and $v(B_k)$ are computed. The expressions are based on ℓ_1, ℓ_2 and ℓ_∞ norms of h-tuples of item sizes $(s_{i1}, s_{i2}, \ldots, s_{id})$ and h-tuples of bins' residual capacities, with specially defined weights w_h computed for dimensions h, $1 \leq h \leq d$.

2.2 Bin-centric approach

Bin-centric algorithms consider the bins one-by-one and find (heuristically) the most promising items to be allocated to a current bin. The items for a current bin B_k are selected according to *item-bin scores* ξ_{ik}, which measure how well item i fits into a current bin B_k. The initial list of item-bin scores was proposed in (Panigrahy *et al.* 2011). We extend that list by adding the expressions from most recent works and introducing two new item-bin scores. We also incorporate weights w_h, $1 \leq h \leq d$, which can be kept static during the execution of an algorithm, or updated dynamically.

2.3 Multi-bin activation approach

Multi-bin activation approach deals with a series of problems VBP(m), each with a fixed value of available bins, all activated at the start. The trial values of m can be increased in steps or enumerated via binary search, although the latter method has its pitfalls, associated with non-monotone behavior of the algorithms.

For solving an individual problem VBP(m), we propose special adaptations of item-centric algorithms, namely Worst Fit and Worst Fit Decreasing, and a new family of *pairing* algorithms. The latter considers all feasible item-bin pairs when making allocation decisions. Dealing with a large pool of bins is beneficial for making better-informed decisions compared to bin-centric and item-centric algorithms, which in every stage are limited to one bin or to the current set of activated bins. This, however, incurs higher computation cost.

2.4 Computational Experiments

The talk will discuss the outcomes of extensive computational experiments performed on famous benchmark instances from (Caprara and Toth 2001) and (Panigrahy *et al.* 2011), as well as on a new set of benchmarks. The experiments identify the most successful approaches for each family of algorithms, depending on an instance type, and on the choice of tuning parameters: combined size measures for items and bins, computation of item-bin scores, and on the choice of the weights for dimensions. Further work can consider randomized versions of the heuristics and analyze instance features for automated tuning of heuristics and their parameters.

3 Adjusting VBP heuristics for affinity-aware resource allocation in Clouds

The affinity-aware version of VBP deals with replicas of LRAs, which are considered as individual items, and with compute nodes, considered as bins. All algorithms outlined in Section 2 require feasibility checks, which include verifying conditions (1) and conditions incurred by the restrictions on co-location of LRA replicas. Whenever an algorithm considers a possibility of allocating an additional replica of LRA i to a partially packed node, it needs to verify for every LRA j, which replicas have been already allocated to the node, whether the affinity parameters a_{ij} and a_{ji} are not exceeded.

The expressions for combined size measures $s(i)$ of replicas, combined size measured of nodes' residual capacities $v(B_k)$ and replica-node scores ξ_{ik} are computed using the range of expressions elaborated for the classical VBP, with changes mostly related to the computation of weights w_h for the dimensions. They take into account the demand of all replicas of all LRAs in each dimension, as well the characteristics of LRAs that measure their level of incompatibility. The latter is formalized as a vertex degree in a graph with verticies corresponding to LRAs and arcs corresponding to affinities.

With a number of sophisticated tuning parameters, we obtain dozens of algorithms and their versions for the affinity-aware VBP problem. They are evaluated via extensive computational experiments based on the Alibaba Tianchi dataset `https://tianchi.aliyun.com/dataset/6287` and on additional instances of similar nature, which differ in densities of affinity graphs, the numbers of LRAs and their replicas. The LRAs, which require resources of two types, CPU and memory, have time varying resource demands. The planning horizon is split into 98 time slots, which, together with two types of resources, result in the VBP problem with 98×2 dimensions. The number of LRAs varies in the range 10,000 - 100,000. Solutions of the highest accuracy are obtained by the multi-bin activation algorithms. Those algorithms ensure spreading of replicas over compute nodes, which is particularly important for the problem with affinity restrictions. The second successful algorithm is Worst Fit Decreasing with specially computed weights for dimensions: it is less accurate than multi-bin activation, but it has shorter running time. The known state-of-the-art algorithms, LRASched (Cai *et al.* 2022) and (Garefalakis *et al.* 2018), are slightly faster, but they produce solution of lower quality.

To summarize, the toolkit for VBP provides a solid foundation for developing new enhanced methods for problems with advanced features arising in applications.

Acknowledgments

This work is supported by the project EP/T01461X/1 "Algorithmic Support for Massive Scale Distributed Systems" funded by the UK Engineering and Physical Sciences Research Council (EPSRC). All algorithms codes and data are publicly available at
`https://github.com/Vectorpack/Vectorpack_cpp`,
`https://github.com/Vectorpack/experiments_vector_paper` and
`https://github.com /DSSGroup-Leeds/LRA-binpacking-expe`.

References

Cai B., Guo Q., Yu J., 2022, "LRASched: admitting more long-running applications via auto-estimating container size and affinity", *Comput. J.* 65, 2377–2391.

Caprara A., Toth P., 2001, "Lower bounds and algorithms for the 2-dimensional vector packing problem". *Discrete Appl. Math* 111, 231–262.

Christensen H.I., Khan A., Pokutta S., Tetali P., 2017, "Approximation and online algorithms for multidimensional bin packing: A survey". *Comput. Sci. Rev.* 24, 63–79.

Epstein L., van Stee R., 2018, "Multidimensional packing problems". In: Gonzalez, T. (eds) *Handbook of Approximation Algorithms and Metaheuristics*, pp. 553–570. Chapman and Hall/CRC, New York.

Garefalakis P., Karanasos K., Pietzuch P., Suresh A., Rao S., 2018, "Medea: scheduling of long running applications in shared production clusters", In: *Proc. of EuroSys*, pp.1–13.

Liu Q., Yu Z., 2018, "The elasticity and plasticity in semi-containerized co-locating cloud workload: a view from Alibaba trace", In: Proc. of *ACM SoCC*, pp.347–360.

Mommessin C., Erlebach T., Shakhlevich N.V., 2023a, "Classification and evaluation of the algorithms for vector bin packing", Available at SSRN 4450606.

Mommessin C., Yang R., Shakhlevich N.V., Sun X., Kumar S., Xiao J., Xu J., 2023b, "Affinity-aware resource provisioning for long-running applications in shared clusters", *J Parallel Distrib. Comput.* 177, 1–16.

Panigrahy R., Talwar K., Uyeda L., Wieder U., 2011, "Heuristics for vector bin packing". Microsoft Research.

Verma A., Pedrosa L., Korupolu M., Oppenheimer D., Tune E., Wilkes J., 2015, "Large-scale cluster management at Google with Borg", In: *Proc. of ACM Eurosys*, pp.1–17.

The agile project management scheduling problem—definition and discussion

Avraham Shtub[1] and Claudio Szwarcfiter[2]

[1] Faculty of Data and Decision Sciences, Technion—Israel Institute of Technology, Haifa, Israel
`shtub@technion.ac.il`
[2] Faculty of Industrial Engineering and Technology Management, Holon Institute of Technology, Israel
`szwarcfiterc@hit.ac.il`

Keywords: Agile project management, project scheduling, value points, story points, backlog scheduling, sprint scheduling.

1 Introduction

Agile project management (PM) is widely used (Özkan and Mishra 2019) but sprint scheduling is challenging. We address sprint scheduling to maximize value under constraints. Prior work examines agile broadly (Raharjo and Purwandari 2020), but sprint scheduling has received little focus. We define the problem and provide a simulator and model.

We outline an agile simulator for interactive modeling. A mixed-integer linear program (MILP) minimizes duration and cost. The model has iterative sprints, adaptive planning, integration, and prioritization.

To our knowledge, this represents the first attempts to assist agile practitioners with a decision support tool capturing:

- Iterative sprints incrementally delivering shippable product
- Embracing adaptive planning
- Frequent code integration
- Prioritizing backlog items by value

The paper is organized as follows. Section 2 describes the agile simulator for interactive planning and learning. Section 3 formally defines the sprint scheduling problem and outlines the MILP model.

2 The modeling tool—the agile simulator

The agile simulator has three major components: Scenario editing, sprint planning, and sprint execution.

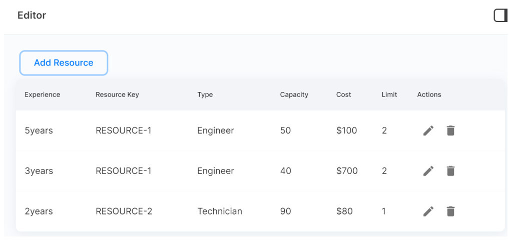
2.1 The scenario editor

The scenario editor is the tool by which new scenarios are created and existing scenarios are modified. A scenario has several components:

Resources: Resources can be defined by type and instance. For example, an engineer resource type may have experienced and new engineer instances. While both instances can perform the same issues, the experiences engineer does so faster with higher capacity. The resource editor screen (Fig. 1a) allows defining distinct resource types and instances with differing capacities and costs. This supports modeling teams with members of varying skill sets and levels.

The example defines Engineer and Technician resource types. Engineer instances include a 5-year and 3-year experienced engineer with differing capacities measured in story points - an abstract measure of complexity. Each instance also has an associated cost per sprint. Sprints are regular iterative cycles to deliver working functionality. The resources are limited, for instance 2 senior engineers vs 1 junior technician.

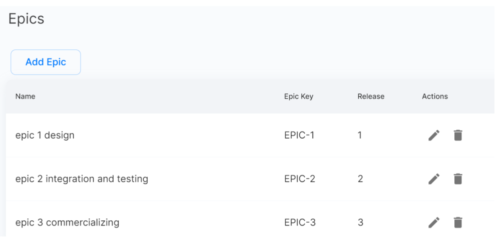
Epics: Epics represent large bodies of work broken into issues - concise user stories to capture requirements. The example has 3 epics (Fig. 1b) comprising issues to be done by the resources. Epics are comparable to work packages in traditional PM.


Editor

Add Resource

Experience	Resource Key	Type	Capacity	Cost	Limit	Actions
5years	RESOURCE-1	Engineer	50	$100	2	
3years	RESOURCE-1	Engineer	40	$700	2	
2years	RESOURCE-2	Technician	90	$80	1	

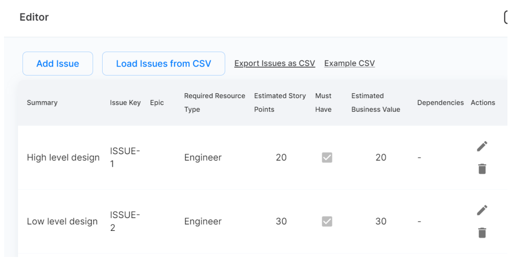
(a) Resource editor


Epics

Add Epic

Name	Epic Key	Release	Actions
epic 1 design	EPIC-1	1	
epic 2 integration and testing	EPIC-2	2	
epic 3 commercializing	EPIC-3	3	

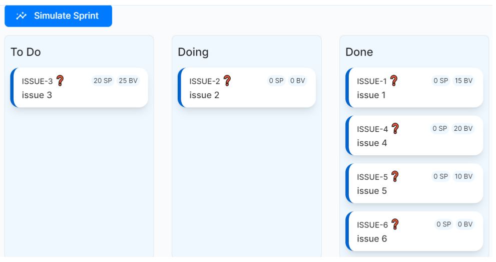
(b) Epics editor


Editor

Add Issue | Load Issues from CSV | Export Issues as CSV | Example CSV

Summary	Issue Key	Epic	Required Resource Type	Estimated Story Points	Must Have	Estimated Business Value	Dependencies	Actions
High level design	ISSUE-1		Engineer	20	✓	20	-	
Low level design	ISSUE-2		Engineer	30	✓	30	-	

(c) Issues editor


Simulate Sprint

To Do
- ISSUE-3 20 SP 25 BV issue 3

Doing
- ISSUE-2 0 SP 0 BV issue 2

Done
- ISSUE-1 0 SP 15 BV issue 1
- ISSUE-4 0 SP 20 BV issue 4
- ISSUE-5 0 SP 10 BV issue 5
- ISSUE-6 0 SP 0 BV issue 6

(d) Results of a sprint simulation

Fig. 1. Resources, epics issues and sprints in the simulator

Issues, user stories or tasks: Issues link to required resources and have story point size estimates reflecting the relative complexity. Issues may represent critical functionality for the minimum viable product (MVP) or nice-to-have items. Issues concisely describe desired features from an end-user perspective to capture requirements. The issues editor (Fig. 1c) allows defining these user stories as connected across epics, along with the resources and estimates needed to realize them.

2.2 Planning a sprint

Sprint planning marks the start of each sprint, requiring collaborative planning to commit to the upcoming work. The team decides which issues to include based on story point estimates reflecting complexity. Proper assignment avoids overload or idle time. It prioritizes high value and must-have issues first. Capacity planning ensures a realistic workload so the team can deliver on their commitments. Too many issues pressures completion while too few causes unused capacity. The goal is to maximize value delivered each sprint within resource constraints.

2.3 Executing a sprint

Sprint execution involves actively developing user stories to deliver functionality within the fixed sprint timeframe. The simulator introduces uncertainty by randomizing resource capacity and issue effort based on distributions. It tracks progress - completed, started, or not started issues. Outcomes include value generated, cost incurred, and over/under utilized resources. This allows experimentation and insights into different plans. Fig. 1d displays sample sprint simulation results on key metrics like cost, value, and issues status.

3 The scheduling problem

The planning goal is a two-stage process: (1) Deliver a minimum viable product (MVP) as early as possible at minimum cost, (2) then maximize overall product value under budget, deadline, and Stage 1 schedule constraints. Key decisions involve assigning resources and issues to sprints. We formulate a mixed integer linear programming model (MILP) with Stage 1 minimizing weighted duration and cost. Table 1 defines the parameters, variables, and sets for the model including issues, resources, story points, costs, sprints, and more.

The optimization model, shown below, minimizes (1) a weighted combination of the number of sprints (project duration) and total resource cost, subject to: (2) matching resources and sprints to issue requirements, (3) limiting issue assignment to once, (4) enforcing resource capacity limits per sprint, (5) limiting resource usage per availability, (6) linking assigned issues to sprints executed, (7) precedence constraints forcing predecessor completion before an issue's assignment, (8)-(10) constraints to define the auxiliary variables z_{srk} connecting resource assignments to sprints.

Our current effort is focusing on modeling Stage 2 and to develop a reinforcement learning algorithm to solve it based on our previous work (Szwarcfiter et al. 2022, Szwarcfiter et al. 2023a, Szwarcfiter et al. 2023b).

Table 1. Sets, parameters, and decision variables

Sets	
M	Set of must-have issues (part of the MVP)
P_i	Set of predecessor issues that must precede issue i
Parameters	
I	Number of issues
S	Upper bound for number of sprints
R	Number of resource types
K_r	Number of resource instances for resource type r
sp_i	Story points for issue i
cap_{rk}	Capacity (story points) of instance k, resource r per sprint
c_{rk}	Cost of instance k, resource r per sprint
u_{rk}	Number of units of instance k, resource r per sprint
req_{ir}	1 If issue i requires resource type r, 0 otherwise
w_1, w_2	Weights to balance duration and cost, respectively
Decision variables	
x_{sirk}	1 if issue i assigned to instance k, resource r in sprint s, 0 otherwise
y_s	1 if sprint s executed, 0 otherwise
z_{srk}	1 if instance k, resource r is assigned to sprint s, 0 otherwise

$$\text{Min}\left(w_1\sum_{s=1}^{S}y_s+w_2\sum_{s=1}^{S}\sum_{r=1}^{R}\sum_{k=1}^{K_r}c_{rk}\cdot z_{srk}\right), \tag{1}$$

subject to:

$$\sum_{r=1}^{R}\sum_{k=1}^{K_r}x_{sirk}=req_{ir}\cdot y_s,\quad \forall s=1,\cdots,S,\forall i=1,\cdots,I, \tag{2}$$

$$\sum_{s=1}^{S}\sum_{r=1}^{R}\sum_{k=1}^{K_r}x_{sirk}\le 1,\quad \forall i=1,\cdots,I, \tag{3}$$

$$\sum_{i=1}^{I}sp_i\cdot x_{sirk}\le cap_{rk}\cdot y_s,\quad \forall s=1,\cdots,S,\forall r=1,\cdots,R,\forall k=1,\cdots,K_r, \tag{4}$$

$$\sum_{i=1}^{I}x_{sirk}\le u_{rk}\cdot y_s,\quad \forall s=1,\cdots,S,\forall r=1,\cdots,R,\forall k=1,\cdots,K_r, \tag{5}$$

$$x_{sirk}\le y_s,\quad \forall s=1,\cdots,S,\forall i=1,\cdots,I,\forall r=1,\cdots,R,\forall k=1,\cdots,K_r, \tag{6}$$

$$\sum_{\varsigma=1}^{s}\sum_{j\in P_i}\sum_{r=1}^{R}\sum_{k=1}^{K_r}x_{sjrk}\ge\sum_{r=1}^{R}\sum_{k=1}^{K_r}x_{sirk}\cdot|P_i|,\quad \forall i:|P_i|>0,\forall s=1,\cdots,S, \tag{7}$$

$$z_{srk}\le y_s,\quad \forall s=1,\cdots,S,\forall r=1,\cdots,R,\forall k=1,\cdots,K_r, \tag{8}$$

$$x_{sirk}\le z_{srk},\quad \forall s=1,\cdots,S,\forall i=1,\cdots,I,\forall r=1,\cdots,R,\forall k=1,\cdots,K_r, \tag{9}$$

$$z_{srk}\le\sum_{i=1}^{I}x_{sirk},\quad \forall s=1,\cdots,S,\forall r=1,\cdots,R,\forall k=1,\cdots,K_r. \tag{10}$$

References

Özkan, D. and Mishra, A., 2019. "Agile Project Management Tools: A Brief Comparative View". *Cybernetics and Information Technologies* 19.4. ISSN: 13144081. DOI: `10.2478/cait-2019-0033`.

Raharjo, T. and Purwandari, B., 2020. "Agile project management challenges and mapping solutions: A systematic literature review". *ACM International Conference Proceeding Series*. Association for Computing Machinery, pp. 123–129. ISBN: 9781450376907. DOI: `10.1145/3378936.3378949`.

Szwarcfiter, C., Herer, Y. T., and Shtub, A., 2022. "Project scheduling in a lean environment to maximize value and minimize overruns". *Journal of Scheduling* 25.2. ISSN: 10991425. DOI: `10.1007/s10951-022-00727-9`.

Szwarcfiter, C., Herer, Y. T., and Shtub, A., 2023a. "Balancing Project Schedule, Cost, and Value under Uncertainty: A Reinforcement Learning Approach". *Algorithms* 16.8, p. 395. ISSN: 1999-4893. DOI: `10.3390/A16080395`.

Szwarcfiter, C., Herer, Y. T., and Shtub, A., 2023b. "Shortening the project schedule: solving multimode chance-constrained critical chain buffer management using reinforcement learning". *Annals of Operations Research*, pp. 1–28. ISSN: 1572-9338. DOI: `10.1007/S10479-023-05597-8`.

An interaction-oriented schedule risk analysis to reduce the project duration

Yuxuan Song[1] and Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
yuxuan.song@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 Risk Management and Control, Network Modeling, Risk Interaction.

1 Introduction

Completing a project on time and within costs is one of the challenges of project management. The management of projects at risk has been explored in the literature from various angles and has led to many methodologies to better predict, manage and even reduce risks. Many of these studies approach project risk from an activity point-of-view, where variability in their durations (or costs) can impact the overall duration of the project. The risks in the durations are usually modeled by probability distributions, and the risk is imitated through simulation studies and examined how it affects the project performance during project execution, and ultimately the final project status.

The current study will build on the insights from previous studies, but the risk will be defined much more specifically than arbitrary probability distributions. In addition, the interactions between risks (i.e., the occurrence of one risk is likely to trigger another risk) are taken into account. More specifically, the *project network* (PN) will be linked to a so-called *risk interaction network* (RIN) in which the risk is defined as a connected set of possible disruptions that could impact parts of the project. This link between a risk network and the activity network will be analyzed in detail using existing and new sensitivity metrics that measure the sensitivity of activity variability to the total project duration. An extended computation experiment is established to analyze whether these metrics can facilitate the corrective actions that project managers need to take during project execution.

2 Risk Model

2.1 Model

A double-layer network (PN-RIN) is constructed by linking a project network (PN) and a risk interaction network (RIN) to model the interaction between the risks and the activities of a project, as visualized in Figure 1. More precisely, a project is initially represented by an activity-on-the-node network consisting of activities between finish-to-start precedence relations, namely project network. Then, the project risks and the interactions between these risks (cause/effect relations) are identified and constructed as a risk interaction network, which is similar to the project network. Since each risk affects different activities, the final step involves connecting the relationships between activities in the project network and risks in the risk interaction network.

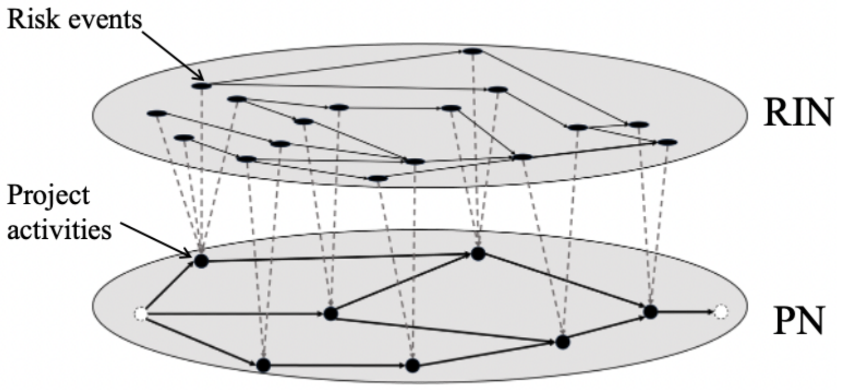


Fig. 1. The PN-RIN Model

2.2 Schedule risk analysis

Schedule risk analysis (SRA) comprises three steps, namely constructing a baseline schedule as a reference, defining uncertainty for activity, and calculating the sensitivity metrics of each activity. An earliest start schedule is initially built based on the project network using the well-known critical path calculation. Subsequently, the risks in the RIN are served as the source of uncertainty for activity, and modeled by three different risk concepts. The risk event can occur for no reason other than the random occurrence of the event itself (risk spontaneous occurrence), but can also be triggered by the occurrence of another risk event that is linked to the current risk event (risk propagation). Both events can have an impact on the associated activities, causing delays and need for corrective actions (activity delay). The dynamic process of risk spontaneous occurrence, risk propagation, and activity delay are formulated and utilized as the basis for conducting the simulation in the SRA metrics calculations. Finally, Monte Carlo simulation is employed to generate real activity duration that is changed by the various simulated risk disruptions. After sufficient simulations, the simulated data can be used to calculate various SRA metrics that, depending on the calculation, reflect the sensitivity of the activities in different ways. In this paper, five existing activity sensitivity metrics and a novel proposed metric are calculated and compared. More precisely, the existing metrics include the criticality index (CI), the significance index (SI), the schedule sensitivity index (SSI), the cruciality index (CRI, three versions), and the management oriented index (MOI) (Van Slyke 1963, Williams 1992, Vanhoucke 2010, Madadi M. and Iranmanesh H. 2012). A novel SRA metric, the risk contribution index (RCI), is designed based on the PN-RIN model to measure activity sensitivity from a risk interaction point of view.

3 Control model

This section describes the general methodology to simulate project progress, where the sensitivity metrics will be used to take corrective actions when the project is in danger of slowing down resulting in delays and project failure.

3.1 Data generation

To compare the different sensitivity metrics, we tested our risk and control models on a set of various artificial projects from the literature, which were linked to different types of risk networks. A series of parameters are selected to maximally vary the structure of the project and risk networks, as shown in Table 1.

We have selected an existing dataset of project instances from Martens and Vanhoucke (2019) which consists of projects with $|N| = 30$ activities, and the serial/parallel (SP) values ranging from 10% to 100%, in steps of 10%. On the basis of the project network, a network generator is designed and programmed to obtain the PN-RIN model with different structures. Specifically, the size and complexity of the risk interaction network are determined by two indicators, namely risk quantity (RQ) and network density (ND). The quantity of risks (RQ), equivalent to the number of nodes in the RIN, is set as several times the number of activities. The network density (ND) is defined as the ratio of the actual edges in RIN to the maximum number of potential edges in the network. With a given risk quantity, the higher the network density, the more risk interactions exist in the RIN. Considering the difference between activities in the number of risks impacting each individual activity, the risk uniformity (RU) is designed to measure the evenness of risk amounts among activities. In the experiment, each combination of parameters, as shown in Table 1, contains 100 projects. As a consequence, 19,200 ($= 4 \times 3 \times 4 \times 4 \times 100$) PN-RIN models are generated and tested in the experiments.

Table 1. Parameters level

Network structure	Parameter	Values
Project network	Serial/ Parallel network (SP)	0.2, 0.4, 0.6, 0.8
Risk interaction network	Risk quantity (RQ)	$2\times\|N\|, 4\times\|N\|, 8\times\|N\|$
	Network density (ND)	0.2, 0.4, 0.6, 0.8
Links between networks	Risk uniformity (RU)	0.2, 0.4, 0.6, 0.8

3.2 Simulation with corrective actions

During the project execution, the project manager should determine the SRA metric for activity sensitivity assessment, select the activity for corrective actions and take corrective actions on the selected activity. The efficiency of the SRA metric is tested in measuring the impact of the risks of the activity on project duration. Subsequently, three activity selection strategies are examined, namely interventive strategy, preventive strategy, and hybrid strategy. More precisely, the *interventive strategy* takes actions on the ongoing activities to mitigate the impact of the risks occurring in the ongoing activities. The *preventive strategy* focuses on future activities that not start yet, aiming to prevent their potential risks. In the *hybrid strategy*, both ongoing and future activities are considered. In addition to selecting the activity that should be taken actions, decisions still need to be made about when and how to perform the corrective actions. Accordingly, the periodicity and the control intensity of the actions are considered and tested.

3.3 Project output performance

As the goal is to reduce the project duration, time effectiveness is employed to evaluate the performance of corrective actions, which has been introduced by Martens and Vanhoucke (2019). The time effectiveness calculates the ratio between average delay reduction due to actions ($\overline{Delay^{no}} - \overline{Delay^{yes}}$) and the average delay without actions ($\overline{Delay^{no}}$), as shown in Eq. 1.

$$Time\ effectiveness = \frac{\overline{Delay^{no}} - \overline{Delay^{yes}}}{\overline{Delay^{no}}} \tag{1}$$

4 Results

First, the overall results show that the proposed RCI metric outperforms the existing sensitivity metrics in steering project control. The time effectiveness of corrective actions using the novel RCI metric is highest, followed by the SSI and MOI metrics. In addition, the sensitivity metrics are more beneficial for projects with more parallel activities. With the increasing SP value, the RCI exhibits a marginal decrease, in contrast to the steep declines observed in other metrics. It indicates that the RCI is more reliable than other metrics in determining sensitive activities to take corrective actions for different project network structures under risk interaction. Moreover, at the same size of risk interaction network, the superiority of RCI becomes more significant as the density of risk interactions increases (referred as to ND). This illustrates how important it is to also incorporate the indirect impact of risk (risk events triggered by other risk events) on the project duration into measuring activity sensitivity under risk interaction, something which is completely ignored by the other sensitivity metrics (like SSI and MOI).

Subsequently, the overall results show that the use of the hybrid strategy results in better time effectiveness regardless of the project network structures. Moreover, on average, the time effectiveness of actions increases with the growing control intensity of action, which conforms to the intuition. However, in some projects characterized by low or medium complexity in the RIN, the effect of the actions with low control intensity can rival that of the actions with high control intensity. This can be explained as follows. In these projects, the corrective actions with low intensity can mitigate the major risks of sensitive activities that have both direct impact and indirect impact on project delay, cutting the key paths of risk propagation in RIN, and thus, significantly decreasing the occurrence probability of project risks. The slack of activity can counteract the impact of risks remaining in the activities.

References

Van Slyke, R.M., 1963, "Letter to the Editor-Monte Carlo Methods and the PERT Problem", *Operation Research*, Vol. 11, pp. 839-860.

Williams.T. M., 1992, "Criticality in Stochastic Networks", *Journal of the Operational Research Society*, Vol. 43, pp. 353-357.

Vanhoucke M., 2010, "Using activity sensitivity and network topology information to monitor project time performance", *Omega*, Vol. 38, pp. 359-370.

Madadi M., Iranmanesh H., 2012, "A management oriented approach to reduce a project duration and its risk (variability)", *European Journal of Operational Research*, Vol. 219, pp. 751-761.

Martens A., Vanhoucke M., 2019, "The impact of applying effort to reduce activity variability on the project time and cost performance", *European Journal of Operational Research*, Vol. 277, pp. 442-453.

Song, J., Martens, A. and Vanhoucke, M., 2020, "The impact of a limited budget on the corrective action taking process", *European Journal of Operational Research*, Vol. 286, pp. 1070-1086.

An integrated project and personnel scheduling problem using an adaptive large neighbourhood search procedure

Brede Sørøy[1], Benjamin S. Buan[1], Henrik Andersson[1] and Anders N. Gullhav[1]

[1]Norwegian University of Science and Technology, Norway
brede.soroy@ntnu.no, bensbuan@gmail.com, henrik.andersson@ntnu.no, anders.gullhav@ntnu.no

Keywords: project scheduling, personnel scheduling, metaheuristic, adaptive large neighbourhood search, warm start.

1 Introduction

In project-based industries, such as the construction industry, having cost-efficient project schedules is essential to allocate resources for the lowest cost possible. Conventionally, project and personnel scheduling are performed separately in sequence. However, in previous research, we show that scheduling these two sequentially may not be the most cost-efficient way. While integrated project and personnel scheduling using mixed-integer programming (MIP) models has shown promising results, many of the instances considered are too complex for commercial MIP solvers. In fact, in several cases, the solver is not even able to find feasible solutions before it is terminated after three hours. Therefore, we present an adaptive large neighbourhood search (ALNS) procedure to address these challenges.

Our work aims to build upon these findings by comparing solutions obtained from the MIP solver and our ALNS procedure. In addition, we want to explore whether there is any improvement of fixing project schedules and initialising the solver with the best ALNS solutions. This is of particular interest for the instances in which the MIP solver finds no feasible MIP solutions within a three-hour time limit.

Although the integrated project and personnel scheduling has not received the same attention in the literature compared to the project scheduling and personnel scheduling problems (Maenhout and Vanhoucke 2017), there are some papers that apply heuristic solution methods to solve the integrated problem. For instance, Kolisch and Heimerl (2012) introduce a hybrid heuristic approach that utilises a genetic algorithm followed by a tabu search. Moreover, Van Den Eeckhout *et al.* (2019) propose an iterated local search (ILS) procedure. Their perturbation procedure is based on both randomisation and the solution quality of the previous solutions, and their acceptance criterion is based on hill climbing. Furthermore, Cordeau *et al.* (2010) introduce an ALNS procedure to schedule technicians and tasks in a telecommunications company.

2 Problem description

This paper addresses the integrated project and personnel scheduling problem for multiple projects at different locations. Each project involves scheduling various preemptive activities, each with multiple modes and potential precedence constraints. These activities require personnel with specific skills, and equipment to be completed. The activities are subject to earliest- and latest-start time windows. Additionally, projects must be finished by a given deadline.

To satisfy the activities' personnel demands, we allocate personnel that are available from both a limited internal and an unlimited external source. Internal employees are multi-skilled, while external employees possess one skill only. Both internal and external employees are allocated to work in non-overlapping stints that respect labour regulations. A stint is a set of working days and off days that an employee is assigned to. For instance, if we have a 12–9 stint, the employees work 12 days before they have 9 days off. Each stint is associated with one project. The costs of scheduling an employee for a specific stint are influenced by travel and accommodation considerations, which come from the varying distances between employees' home locations and the respective project sites. Furthermore, equipment, either owned or rented, must be allocated to activities. Also, internal equipment needs to be transported between the activities.

The objective is to minimise overall costs, including salaries to internal employees, costs related to hiring external personnel, preference violation costs, equipment transportation, and costs for renting external equipment. Decisions include activity scheduling, personnel allocation to stints, equipment transportation, and renting of external equipment.

3 Solution methods

The adaptive large neighbourhood search (ALNS) procedure is first presented in Røpke and Pisinger (2006). The ALNS is an extension of the large neighbourhood search (LNS) procedure proposed in Shaw (1997). In contrast to many other neighbourhood-based metaheuristic procedures, the neighbourhoods of the LNS and ALNS are not defined explicitly. Rather, they use destroy and repair operators. The LNS uses only one of each, while the ALNS can include several destroy and repair operators. What makes the ALNS an adaptive procedure is that it chooses the destroy and repair operators based on all the operators' performance in previous iterations. The better the operators perform in previous iterations, the more likely they are to be chosen in the next iteration. Our ALNS implementation consists of 28 destroy operators and 15 repair operators designed to improve various parts and structures of the current solution. The destroy operators are divided into four categories, which consists of destroying the activity schedules, the employee allocations, changing the modes of one or more activities, and removing the equipment allocations. To repair the destroyed solution, we have three types of repair operators that repair the activity schedules, allocate employees to meet the personnel demands, and allocate equipment to fulfil the equipment demands. The initial solutions used by the ALNS are found using a greedy construction heuristic.

In addition, we provide the solutions generated by our ALNS to the MIP solver using two different solution methods that we call the *fixed-projects* and *warm-start* methods. The fixed-projects approach is a method where the project schedules determined by ALNS are fixed, while the MIP solver optimises the allocation of personnel and equipment. This strategy uses the scheduling capabilities of our ALNS to find a good initial schedule, upon which the MIP solver efficiently allocates the resources. By narrowing the MIP solver's focus, this approach can significantly enhance the overall efficiency and effectiveness of the problem-solving process.

Warm-starting the MIP solver with an initial solution from ALNS is a technique designed to begin with an already promising solution, rather than starting the search from scratch. This approach enables the MIP solver to search for improved solutions more effectively since the ALNS solution is more likely to be closer to the optimal solution than the first feasible solution found by the MIP solver.

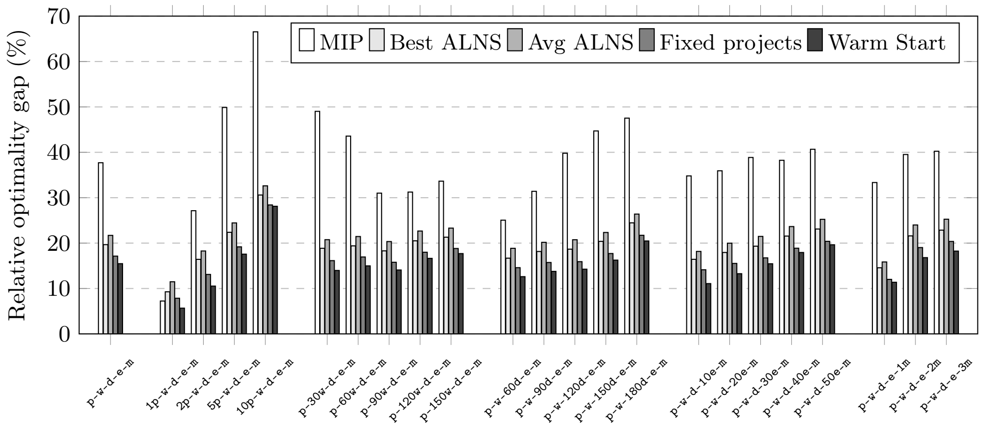


Fig. 1. Gap comparisons by aggregation pattern

4 Computational study and results

Both a mixed-integer programming (MIP) model and an adaptive large neighbourhood search (ALNS) procedure have been formulated and implemented, respectively, to solve this problem. The implementation of the MIP is written in Xpress Mosel and solved using the Xpress Optimizer, while the ALNS is implemented using C++.

The MIP solver is executed for a maximum duration of three hours for each instance, while the ALNS is terminated after five minutes have passed. Moreover, the ALNS is run twelve times for each instance in parallel threads. Also, we check whether fixing the best project schedules from the ALNS and warm-starting the MIP solver with the best ALNS solutions yield better results. As the ALNS is run for five minutes before we start the fixed-projects and warm-start approaches, the maximum computation time for the MIP solver is set to two hours and 55 minutes.

Both the MIP solver and the ALNS have been tested on 1,500 instances of various sizes based on the dimensions of the number of projects (`p`), employees (`w`), days (`d`), pieces of equipment (`e`), and modes (`m`). For the results, we use aggregation patterns to indicate which instances we are aggregating. For example, "`p-w-d-e-m`" gives the average over all instances, while "`1p-w-d-e-m`" is aggregating all the instances with one project. We measure the performance by using the gap[1] between the primal and dual bounds. In cases a solution method does not find any feasible solutions, the gap is set to 100%.

In Figure 1, we see that the MIP on average yields the highest gaps in all aggregations apart from the one-project aggregation, mostly because the MIP solver mainly finds optimal solutions for the instances with one project. Furthermore, for the majority of instances involving five and ten projects, the MIP solver fails to find any feasible MIP solutions, negatively impacting the aggregated averages. Apart from the one-project aggregation, the ALNS yields lower optimality gaps on average than the MIP solver for both the best and average objective value aggregations. On the whole, the best ALNS solution for each instance roughly halves the optimality gap when compared to the MIP solution from the solver This shows that the ALNS is able to find relatively good solutions in a short time. Also, presenting the best ALNS solution for each instance to the MIP solver improves the solution further by either fixing the projects from the solution or warm-starting the solver using the solution. As we can see from the aggregations in Figure 1, the warm-start approach has consistently the lowest average gaps based on these specific aggregations.

[1] We define the gap to be $Gap = 1 - \frac{\textit{Best dual bound from MIP}}{\textit{Best solution}}$.

Looking at the overall distribution of the best solutions across the different methods, the warm-start approach accounts for approximately 48%, the fixed-projects approach for 30%, and the MIP formulation for the remaining 22%. Consequently, even though it finds relatively good solutions in only five minutes of execution, there are no instances in which the ALNS finds the best solution. This can be explained by the fact that both the fixed-projects and warm-start approaches are improving the solutions provided by the ALNS. Although the MIP formulation finds the best solutions in 22% of the instances, notably in the instances with one project, as shown in Figure 1, and is able to prove optimality in 25 of the instances, it remains the only method that does not find any feasible solutions in 361 of the instances. The warm-start method does not find more optimal solutions beyond the same 25 instances as for the MIP, while the fixed-projects approach identifies ten optimal solutions. As the ALNS does not find any optimal solutions, albeit some are very close, this indicates that ten of the ALNS solutions have the optimal project schedule but not the most cost-efficient allocation of personnel and equipment. When it comes to execution times, both the MIP formulation and the warm-start approach on average spend mostly all the three allocated hours. In comparison, the longest time for the fixed-projects approach is one hour and forty minutes, having an average time of eight minutes, which includes five minutes of running the ALNS, and the ALNS always terminates after five minutes.

5 Concluding remarks

This study presents a problem for the integrated project and personnel scheduling in project-based industries, such as the construction industry. As the previously formulated MIP struggles to find feasible solutions for 24% of the instances, an ALNS procedure is proposed. This approach not only provides feasible solutions to all the 1,500 instances under study but also allows for further comparisons with the computed MIP solutions.

We find that our ALNS is a promising tool for finding good solutions to the integrated project and personnel scheduling problem efficiently. Introducing the ALNS solutions to the MIP solver, either by fixing the project schedules from the ALNS before re-optimising the resource allocation or to warm-start the solver, further improves the solutions. While both strategies offer ways to improve the initial ALNS solutions, they require additional time after the ALNS is terminated. Particularly for the fixed-projects approach, while it generally does not add much more time, there is a possibility the scheduler might have to wait more than an hour to find the optimal fixed-projects solution.

References

Cordeau J.-F., G. Laporte, F. Pasin and S. Røpke, 2010, “Scheduling technicians and tasks in a telecommunications company”, *Journal of Scheduling*, Vol. 13(4), pp. 393–409.

Kolisch R., C. Heimerl, 2012, “An efficient metaheuristic for integrated scheduling and staffing IT projects based on a generalized minimum cost flow network”, *Naval Research Logistics*, Vol. 59(2), pp. 111–127.

Maenhout B., M. Vanhoucke, 2017, “A resource type analysis of the integrated project scheduling and personnel staffing problem.”, *Annals of Operations Research*, Vol. 252(2), pp. 407–433.

Røpke S., D. Pisinger, 2006, “An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows.”, *Transportation Science*, Vol. 40(4), pp. 455–472.

Shaw P., 1997, *A new local search algorithm providing high quality solutions to vehicle routing problems* [Technical report], Department of Computer Science, University of Strathclyde, Scotland.

Van Den Eeckhout M., B. Maenhout and M. Vanhoucke, 2019, “A heuristic procedure to solve the project staffing problem with discrete time/resource trade-offs and personnel scheduling constraints”, *Computers & Operations Research*, Vol. 101, pp. 144–161.

Scheduling identical jobs with durational uncertainty and maximum time lags

Evelin Szögi[1,2], Péter Györgyi[1] and Tamás Kis[1,2]

[1] HUN-REN Institute for Computer Science and Control, Budapest, Hungary
[2] Department of Operations Research, Eötvös Loránd University, Budapest, Hungary
{szogi.evelin,kis.tamas,gyorgyi.peter}@sztaki.hu

Keywords: Project scheduling, durational uncertainty, maximum time lags.

1 The scheduling problem

In this paper we consider a resource-constrained project scheduling problem in which there are uncertain task durations as well as minimum and maximum time lags between pairs of tasks. We do not assume any distribution of task durations. We focus on a special case with the following key features. There is a finite set of resources $R_1, \ldots, R_m$, and the capacity of R_k is $c_k > 0$. There are n identical jobs $J_1, \ldots, J_n$, where each job J_i is a sequence of ℓ tasks, $T_{i,1}, \ldots, T_{i,\ell}$. The resource requirements of $T_{i,j}$ is given by the vector $\rho_j \in \mathbb{R}^m_{\geq 0}$ for $j = 1, \ldots, \ell$, and it does not depend on the job, since the jobs are identical. Each task has a fixed duration, except for the tasks $T_{i,\ell-1}$, which have an interval of possible durations. That is, for $j \in [\![\ell]\!] \setminus \{\ell - 1\}$, the common duration of all the tasks $\{T_{ij} \; : \; i = 1, \ldots, n\}$ is d_j, a positive number, whereas the duration of the $T_{i,\ell-1}$ is specified by an interval $[d^{\min}_{\ell-1}, d^{\max}_{\ell-1}]$, meaning that the duration of $T_{i,\ell-1}$ can be any number between the given lower and upper bounds, and may be different for each i. The tasks of each job J_i are connected by end-to-start precedence constraints, that is, $T_{i,j+1}$ can only start if $T_{i,j}$ is completed. The minimum time lag between $T_{i,j}$ and $T_{i,j+1}$ is 0, and the maximum time lag is also 0, except for the last pair, i.e., between $T_{i,\ell-1}$ and $T_{i,\ell}$, where it is a common non-negative number $\varDelta$. Finally, each job J_i has a job-dependent release date $e_i \geq 0$.

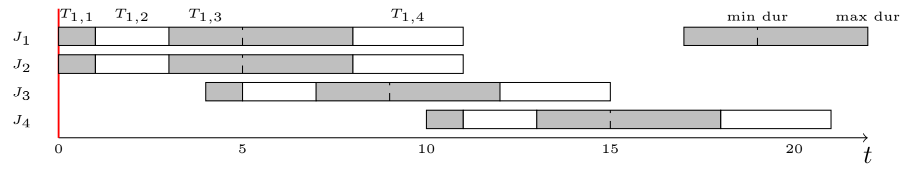
Since the tasks $T_{i,\ell-1}$ have uncertain durations, and we also have maximum time lags, it is not possible to determine the starting times of the tasks in advance, in general. Instead, the starting times will be determined incrementally by a scheduling policy, which takes into account the realized durations $\tilde{d}_{i,\ell-1} \in [d^{\min}_{\ell-1}, d^{\max}_{\ell-1}]$ of the uncertain-duration tasks. Notice that $\tilde{d}_{i,\ell-1}$ becomes known only upon completing $T_{i,\ell-1}$. Let $S_{i,j}$ be the starting time of task $T_{i,j}$ determined by the policy. Then the $S_{i,j}$ have to satisfy the following constraints:

- $S_{i,1} \geq e_i$ for each job J_i,
- $S_{i,\ell-1} + \tilde{d}_{i,\ell-1} \leq S_{i,\ell} \leq C_{i,\ell-1} + \varDelta$ for each J_i, where $C_{i,\ell-1} = S_{i,\ell-1} + \tilde{d}_{i,\ell-1}$ is the completion time of $T_{i,\ell-1}$,
- $C_{i,j} = S_{i,j+1}$ for $j = 1, \ldots, \ell - 2$ for each J_i, where $C_{i,j} = S_{i,j} + d_j$ is the completion time of $T_{i,j}$, and
- For any time point $t \geq 0$, let $\mathcal{A}_t$ be the set of tasks being executed at time t, i.e., $\mathcal{A}_t = \{T_{i,j} \mid S_{i,j} \leq t < C_{i,j}\}$. The capacity constraints $\sum_{T_{i,j} \in \mathcal{A}_t} \rho_j \leq c$ are satisfied for any time $t \geq 0$.

To define the objective function, we need one more notion. The *waiting time* of a job J_i is the total time between e_i and $C_{i,\ell}$ where no task of the job is processed, i.e., $W_i = (S_{i,1} - e_i) + (S_{i,\ell} - C_{i,\ell-1})$. The objective is to minimize $\sum_i W_i$

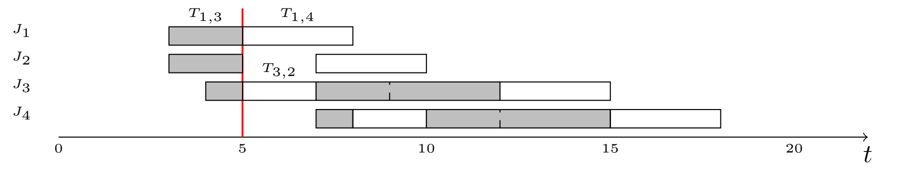
Our main goal is to give a policy that determines the start times of the tasks according to the incrementally changing schedule. Taking the jobs one-by-one in release date order,

we determine the earliest time point t for the current job with the property that starting the job at t, no matter what the realized durations of the previously scheduled uncertain tasks are, the resource constraints and time lag constraints are always met. At each decision point (i.e. when a task with uncertain duration finishes), we update the schedule according to the realized task durations and update the start time of the not-yet started jobs.
Motivation from practice. A possible domain of applications is bio-manufacturing, where some processes have large uncertainty in their duration. For instance, the production of CAR T cells for personalized gene therapies of patients with a serious disease. In such a therapy, white blood cells of the patient are genetically modified and returned to the patient. The production process consists of a number of steps, where the penultimate one is cell expansion, where millions of CAR T cells are grown in a cell culture (June *et. al.* 2018), and the duration of cell expansion may vary from patient-to-patient. In addition, the variance of cell expansion time may be several days, while the other production steps are deterministic and take a couple of days (Bäckel *et. al.* 2023).
Numerical example. Suppose we have four identical jobs $J_1, \ldots, J_4$. Each job J_i is a sequence of 4 tasks $(T_{i,1}, T_{i,2}, T_{i,3}, T_{i,4})$. $T_{i,j}$ must be started without any delay after the completion of $T_{i,j-1}$ for $j = 2, 3$, and $T_{i,4}$ must be started after the end of $T_{i,3}$ at most $\Delta = 2$ time units later. The tasks $T_{i,1}, T_{i,2}$ and $T_{i,4}$ have a duration of 1,2 and 3, respectively, while $T_{i,3}$ has uncertain duration in the interval $[2, 5]$. We have two resources, R_1 with capacity 3 and R_2 with capacity 2. For each job J_i, $T_{i,1}$ and $T_{i,3}$ require one unit from R_1, and $T_{i,2}$ and $T_{i,4}$ require one unit of R_2. Without loss of generality, the jobs are scheduled in increasing index order. Our policy selects the earliest starting time for each job such that under any realizations of the uncertain task durations, the schedule remains feasible. The selected earliest starting times are 0, 0, 4, and 10. The Gantt chart below depicts the solution constructed at time 0.



Both of J_1 and J_2 can be started at 0, since each resource has a capacity of at least 2. Job J_3 cannot be started earlier than 2, since $T_{1,2}$, $T_{2,2}$ and $T_{3,2}$ cannot be executed in parallel because the capacity of R_2 is 2. It cannot be started at 2 or 3, otherwise if the realized duration of both $T_{1,3}$ and $T_{2,3}$ is 5 and that of $T_{3,3}$ is 2, then, since $T_{1,4}, T_{2,4}$ and $T_{3,4}$ cannot be all executed in parallel, at least one of J_1, J_2 and J_3 would violate the maximum time lag constraint. It is not difficult to check that any realizations of the uncertain task duration permit starting J_3 at 4, therefore its earliest starting time is 4. Similar reasoning shows that the earliest starting time of J_4 is 10.

Now suppose the uncertain tasks $T_{1,3}$ and $T_{2,3}$ both complete at time 5. Then, the schedule is updated as shown in the Gantt chart below. As we can see, job J_4 can be started earlier.



Related work. The literature on project scheduling with minimum and maximum time lags combined with uncertain task durations is rather scarce. Fu *et. al.* (2012) consider stochastic local search to minimize the expected project duration and they note that the failure rate

of their method increases as the maximum time lags between the tasks diminish. Research on stochastic project scheduling, or project scheduling with minimum and maximum time lags is abundant. For instance, Rostami *et. al.* (2018) propose a new stochastic scheduling policy which after some preprocessing, makes the final decisions online as the project progresses. However, no maximum time lags between pairs of activities are permitted. Davari and Demeulemeester (2019) considered a variant where the possible realizations of task durations are vectors that constitute a finite set of scenarios, and each scenario has a probability. The authors propose four models for solving the problem. There are many approaches for solving RCPSP/max either exactly, or heuristically Kreter *et. al.* (2016).
Main results. (i) We describe an iterative algorithm for constructing a robust schedule. The schedule is feasible for any realization of the uncertain task durations within the given limits, and at the same time each job starts as early as possible relative to the schedule of previous jobs; (ii) We prove that our method is of polynomial time complexity. This is a striking result as robust scheduling problems may well be beyond the class NP, the class of decision problems which admit polynomial size certificates that can be checked in polynomial time. (iii) To measure the efficiency of our method, one can compare its waiting time to the offline optimum, i.e. the total waiting time of the best schedule for the problem where the duration of every task is known in advance. We have shown that our problem does not have an approximation algorithm of constant approximation ratio.

2 Main results

In this section we give an overview of our scheduling policy and its properties. We have a set of jobs $J_1, \ldots, J_n$, with release dates $e_1 \leq \cdots \leq e_n$. The *initial part* of any job consists of the first $\ell - 1$ tasks. Our method for completing all the jobs goes as follows:

Proactive-Reactive Method

1. Compute an initial schedule S using **Robust Scheduling Algorithm**.
2. Execute the tasks in S until an uncertain task or all the job get completed.
3. **If** all the jobs are completed **then** STOP.
4. Recompute the schedule S using **Robust Scheduling Algorithm**, and proceed with Step 2.

The crux of the method is the **Robust Scheduling Algorithm** which takes as input the current state of the system, and outputs a new schedule. The state of the system is described by the set of completed tasks, their completion times, and the starting times of those tasks that are being executed at the moment the scheduling algorithm is invoked. The schedule specifies a starting time for each not-yet-started task except for those last tasks, where their preceding uncertain task (of the job) is not completed. The algorithm processes the jobs which are not started yet in non-decreasing release date order, and determines the earliest starting time of each job in turn in polynomial time. Finally, it provides the starting times for the last tasks of those jobs which admit a completed uncertain task.

Robust Scheduling Algorithm
input: status of the jobs, and current schedule S.
output: updated schedule S'.

1. Let $\mathcal{J}_u$ be the subset of those jobs not started yet, and $\mathcal{J}_c$ the subset of those jobs where the uncertain task is completed, but the last task is not started yet.
2. Process the jobs of $\mathcal{J}_u$ in non-decreasing release date order. Let J_i be the next job to be processed.

3. Determine a set Λ_i of possible starting times for J_i.
4. Process the time points in Λ_i in increasing order. Let t be the next time point to be verified.
5. Check if J_i can be started in t such that no resource and maximum time lag constraints are violated for any choice of the duration of the uncertain tasks which belong to some job in S and also that of J_i. If the answer is YES, then fix the starting time of J_i at t, and add to S the initial part of J_i. Otherwise proceed with the next time point in Λ_i.
6. Extend S by the last tasks of the jobs in $\mathcal{J}_c$.

In Step 5, the procedure for checking if J_i can be started at time point t is of polynomial time complexity, and boils down to scheduling the last tasks greedily after setting the durations of the uncertain tasks. The main point is that the set of task durations to be verified is of polynomial size in the input, and depends only on the schedule of the fixed duration tasks in S, and determining this set takes polynomial time. Finally, the computation of the set Λ_i also takes polynomial time. Hence, we have the following result:

Theorem 1. *The* Robust Scheduling Algorithm *constructs a schedule in polynomial time which is feasible for any realization of the uncertain task durations within the given limits. Moreover, each job starts as early as possible relative to the previously scheduled jobs.*

3 Final remarks

Our Proactive-Reactive method can also be used in an online scenario, where the jobs arrive one-by-one at their release dates. This is indeed our main application area. However, the method is suitable to handle this problem, since it schedules the jobs in non-decreasing release date order. In future work we aim to compare our method to stochastic ones, where some distribution of task durations are known, and which can take, at least in principle, maximum time lag constraints into account.

Acknowledgments

This work was supported by the EU project AIDPATH (grant no. 101016909).

References

Bäckel, N., S. Hort, T. Kis, D.F. Nettleton, J.R. Egan, J.J. Jacobs, et al., 2023, "Elaborating the potential of Artificial Intelligence in automated CAR-T cell manufacturing". *Frontiers in Molecular Medicine*, Vol. 3, 1250508.

Davari M., E. Demeulemeester, 2019, "The proactive and reactive resource-constrained project scheduling problem", *Journal of Scheduling*, Vol. 22, pp. 211-237.

Fu N., H.C. Lau, P. Varakantham, F. Xiao, 2012, "Robust local search for solving rcpsp/max with durational uncertainty". *Journal of Artificial Intelligence Research*, Vol. 43, pp. 43-86.

June, C.H., R.S. O'Connor, O.U. Kawalekar, S. Ghassemi, M.C. Milone, 2018, "CAR T cell immunotherapy for human cancer". *Science*, Vol. 359(6382), pp. 1361-1365.

Kreter, S., J. Rieck, J. Zimmermann, 2016, "Models and solution procedures for the resource-constrained project scheduling problem with general temporal constraints and calendars". *European Journal of Operational Research*, Vol. 251(2), pp. 387-403.

Rostami S., S. Creemers, R. Leus, 2018, "New strategies for stochastic resource-constrained project scheduling", *Journal of Scheduling*, Vol. 21, pp. 349-365.

Reinforcement Learning for Lean and Robust Project Scheduling under Duration Uncertainty

Claudio Szwarcfiter[1], Yale Herer[2] and Avraham Shtub[2]

[1] Faculty of Industrial Engineering and Technology Management, Holon Institute of Technology, Israel
`szwarcfiterc@hit.ac.il`

[2] Faculty of Data and Decision Sciences, Technion—Israel Institute of Technology, Haifa, Israel
`yale@technion.ac.il, shtub@technion.ac.il`

Keywords: project scheduling; uncertainty; chance constraints; reinforcement learning; lean project management; schedule stability; project value.

1 Introduction

Industrial projects often face cost and time overruns due to uncertainties (De Meyer et al. 2002). Furthermore, almost half of all projects fail to satisfy stakeholders by not delivering the expected value (The Standish Group 2015). To address these challenges, this research puts forth an innovative reinforcement learning (RL) based approach to balance project schedule, cost, and value under uncertain activity durations.

Project value is determined by multiple attributes that represent the extent to which the project meets stakeholders' needs and expectations. These attributes may include scope, quality, reliability, aesthetics, features, functions, size, availability, and more. Value parameters associated with each project activity capture its contribution to overall value. For example, in a radar system project, value attributes can be range, quality, and reliability. The project value is a weighted sum of the value attributes. The selection of modes for each activity directly impacts the value parameters and thus the overall project value.

The research introduces models and algorithms to tackle three distinct project management problems that aim to create stable and robust project baseline schedules:

1. Lean project management (LPM) to maximize value while meeting chance constraints for schedule and budget.
2. Chance-constrained critical chain buffer management (CCBM) to minimize project delivery time.[1]
3. Tradeoff between project value and net present value (TVNPV) to optimize weighted achievement of both objectives.

Together, these problems cover the interrelated aspects of overruns, value shortfalls, and scheduling delays plaguing projects.

2 Novel Contributions

The research offers several key contributions:

- New LPM, CCBM, and TVNPV models with chance constraints and stochastic activity durations.
- An LPM model maximizing value under chance constraints (first such formulation, Szwarcfiter et al. 2022).
- A direct chance-constrained approach for CCBM (new to literature, Szwarcfiter et al. 2023b).

- Joint optimization model for project value and robust NPV (new concept, Szwarcfiter et al. 2023a).
- RL heuristics to solve the models (innovative application).
- Efficient frontiers showing tradeoffs between objectives for decision-making.

3 Solution Methodology

The study utilizes a Monte Carlo control RL method leveraging simulation (refer to Sutton and Barto 2018 for the RL terms employed in this paper). Chance constraints are handled via a scenario approach (introduced in Calafiore and Campi 2005) to tackle uncertainty. This involves:

1. Generating N samples or scenarios to represent possible outcomes for the random activity durations in the chance constraints.
2. Replacing the deterministic constraints with their scenario-based counterparts. For example, the due date constraint becomes a set of N constraints, one for each scenario.
3. Ensuring the proportion of scenarios that satisfy the probabilistic constraints meets the desired levels. For example, if the desired on-time probability is 95%, at least 95% of the N sampled scenarios must finish within the due date.
4. Solving the resulting mixed-integer linear program with the scenario-based chance constraints.

In this way, the SA method transforms a stochastic optimization model with chance constraints into a deterministic equivalent that can be readily solved by standard solvers.

The RL framework begins by placing an agent in a state S, which represents a project activity. The agent takes an action A, which involves selecting a mode $\hat{m}_j$ and additionally, in the CCBM and TVNPV cases, a start time $\hat{t}_j$ for activity j. The agent then transitions to the next state S' and receives a reward R'. This pattern continues as the agent goes through each activity. The objective is to learn a policy $\pi(S, A)$ to maximize cumulative reward. An action-value function $q(S, A)$ estimates the expected reward for taking action A in state S.

For the LPM problem, the reward R is the project value if the solution meets the chance constraints; otherwise, $R = 0$. For CCBM, R is proportional to $\frac{1}{\text{project delivery time}}$. For TVNPV, R is the weighted objective function value.

The RL algorithm has an initialization phase where optimistic action values encourage exploration. Then in each iteration:

1. Calculate an ε-greedy policy π using the action values q
2. Follow policy π to select modes $\hat{m}_j$ and times $\hat{t}_j$
3. Compute reward R for the selected actions
4. Update action values q using the reward R
5. Repeat until convergence

Two action value update rules are used. The first averages all rewards obtained when selecting a mode. The second uses constant step size to give more weight to recent rewards.

4 Experiments

The models and algorithms were tested on standardized PSPLIB (Kolisch and Sprecher 1997) and MMLIB (Vanhoucke and Coelho 2018) datasets. Duration uncertainty was modeled via 3-point estimates. Comparative analysis used:

- LPM: Genetic algorithm and mixed integer linear program (MILP).
- CCBM: Priority rules and MILP.
- TVNPV: Tabu search and MILP.

5 Key Results

The RL methods proved effective, outperforming benchmarks in multiple experiments. The main findings include:

- RL generates higher project value than genetic algorithms.
- Solving chance-constrained CCBM yields shorter schedules.
- RL delivers competitive CCBM solutions vs. heuristics and MILP.
- RL determines efficient project value vs. NPV tradeoffs.

6 Implications and Future Research

The models and methods equip managers to balance objectives under uncertainty. Plotting efficient frontiers facilitates focused decision-making. Fig. 1 depicts an example efficient frontier for LPM. Fig. 2 shows an efficient frontier for TVNPV, where the robust

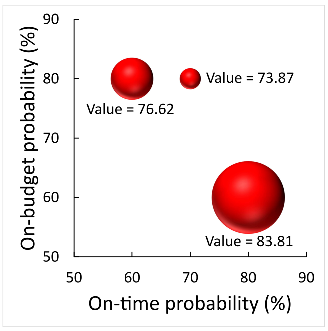


Fig. 1. LPM efficient frontier

NPV (rNPV) refers to the project NPV that is achieved with a likelihood of at least the minimum probability threshold set by the decision makers. Proposed RL enhancements

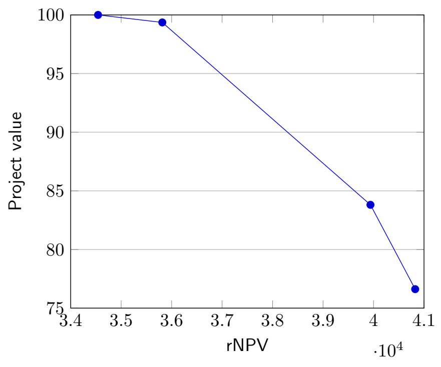


Fig. 2. TVNPV efficient frontier

include scaling with approximation methods.

7 Conclusion

The study introduces new models advancing project value, scheduling, and risk management. RL demonstrates applicability for these problems, despite limited use thus far. The findings offer promising directions to support planning and optimization in project-centric operations.

Notes

[1]The project delivery time refers to the overall time-buffered project duration that includes the baseline schedule, activity durations, and project buffer. By minimizing this duration subject to the chance constraint of meeting the desired on-time probability, we directly search for the shortest robust schedule.

References

Calafiore, G. and Campi, M. C., 2005. "Uncertain convex programs: Randomized solutions and confidence levels". *Mathematical Programming* 102.1, pp. 25–46. ISSN: 00255610. DOI: 10.1007/s10107-003-0499-y.

De Meyer, A., Loch, C. H., and Pich, M. T., 2002. "Managing project uncertainty: from variation to chaos". *MIT Sloan Management Review* 43.2, pp. 60–67.

Kolisch, R. and Sprecher, A., 1997. "PSPLIB – A project scheduling problem library". *European Journal of Operational Research* 96.1, pp. 205–216. ISSN: 0377-2217. DOI: 10.1016/S0377-2217(96)00170-1.

Sutton, R. S. and Barto, A. G., 2018. *Reinforcement Learning: An Introduction.* Second edi. Cambridge, Massachusetts: The MIT Press. ISBN: 9780262039246. DOI: 10.1108/k.1998.27.9.1093.3.

Szwarcfiter, C., Herer, Y. T., and Shtub, A., 2022. "Project scheduling in a lean environment to maximize value and minimize overruns". *Journal of Scheduling* 25.2. ISSN: 10991425. DOI: 10.1007/s10951-022-00727-9.

Szwarcfiter, C., Herer, Y. T., and Shtub, A., 2023a. "Balancing project schedule, cost, and value under uncertainty: A reinforcement learning approach". *Algorithms* 16.8, p. 395. ISSN: 1999-4893. DOI: 10.3390/A16080395.

Szwarcfiter, C., Herer, Y. T., and Shtub, A., 2023b. "Shortening the project schedule: solving multimode chance-constrained critical chain buffer management using reinforcement learning". *Annals of Operations Research*, pp. 1–28. ISSN: 1572-9338. DOI: 10.1007/S10479-023-05597-8.

The Standish Group, 2015. *CHAOS Report 2015 Edition.* Tech. rep. URL: https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf.

Vanhoucke, M. and Coelho, J., 2018. "A tool to test and validate algorithms for the resource-constrained project scheduling problem". *Computers and Industrial Engineering* 118, pp. 251–265. ISSN: 03608352. DOI: 10.1016/j.cie.2018.02.001.

A heuristic approach for a multi-line hybrid flow shop scheduling problem with energy considerations

Sara Taguemount , Damien Lamy and Xavier Delorme

Mines Saint-Etienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158
LIMOS, F-42023, France
`sara.taguemount, damien.lamy, delorme@emse.fr`

Keywords: Energy-efficiency, multi-line hybrid flow shop, scheduling, heuristic.

1 Introduction and related work

High-energy prices are driving industries to consider energy efficiency as an important metric to evaluate their performance. Researchers on the other hand, have developed methods to treat energy efficient scheduling problems. The relevant literature for our work lays in the intersection between two main elements: scheduling problems and energy efficiency. Hybrid and flexible flow shop scheduling problems are very common in the industry. When combined with a multi-line system, they may offer more flexibility and improve the production capacity of the manufacturing system. Energy efficient hybrid flow shop scheduling problems have been addressed in the literature using exact methods and meta-heuristics (Schulz *et. al.* (2019)). Li *et. al.* (2018) proposed a heuristic with the goal to minimize total energy consumption and the makespan for a hybrid flow shop scheduling problem. Gong *et. al.* (2020) suggested a meta-heuristic to tackle human and energy efficiency indicators for a flexible flow shop problem. In this paper, we treat a new scheduling problem which is an extension of a hybrid flow shop scheduling problem with energy considerations. To the best of our knowledge, there is a very limited available literature regarding energy efficient multi-line hybrid flow shop scheduling problem and was treated for the first time in Taguemount *et. al.* (2023). Another scheduling problem that may be assimilated to a multi-line hybrid flow shop is the distributed hybrid flow shop. Meta-heuristic algorithms have been widely proposed to solve distributed hybrid flow shop scheduling problems for their performance in solving industrial instances and in generating quality solutions in a reasonable computing time. Lu *et. al.* (2021) developed a heuristic for the optimization of total energy consumption and the makespan for a distributed flow shop scheduling problem. Wang and Wang (2022) tackled energy efficiency within a distributed hybrid flow shop system by optimizing total energy consumption and makespan.

2 Problem Definition

The problem is formulated as a multi-line hybrid flow shop scheduling problem (MHFSSP) with a time of use pricing structure. In this problem, a job $j \in J$ has to be assigned to a line $l \in L_j$ and to be operated on consecutive stages consisting of one or more parallel heterogeneous machines $m \in M_{jlo}$. Every operation $o \in O_{jl}$ has a processing time P_{jlom} and can be divided into P_{jlom} sub-operations. The problem considers a total energy consumption E_{jlom} for every operation and a peak power W_{ijlom} for its i^{th} sub-operation.W_{max} and TEC_{max} are respectively maximum power consumption and maximum total energy consumption over the makespan. The goal is to determine the starting dates of all operations $o \in O_{jl}$ in order to minimize the total energy cost C_T. In Taguemount *et. al.* (2023), a time-indexed approach has been investigated to solve the problem.

2.1 An example of a multi-line hybrid flow shop architecture

We consider a system of three jobs, two lines and five machines (Fig.1). Every line has a number of stages. j_0 can be processed both on line 1 and 2, j_1 can only be processed on line 1, j_2 can only be processed on line 2. The processing times and energy consumptions are predefined. Energy prices are defined per consecutive time slot durations.

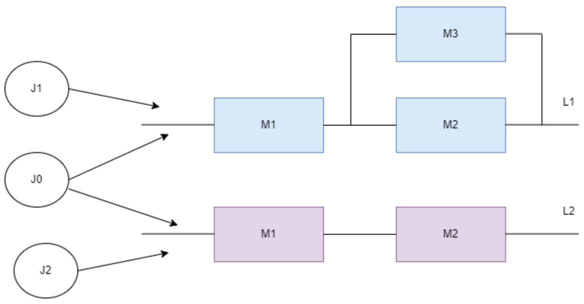


Fig. 1: Multi-line hybrid flow shop

3 Algorithm

In order to obtain solutions to a MHFSSP, we propose a multi-start heuristic approach consisting of: (i) a randomized solutions generation, (ii) an evaluation procedure, (iii) a stochastic local search algorithm.

3.1 Encoding approach

Amongst all solution representations in the scientific literature, we have selected the repetition vector as this representation permits to consider the order between jobs according to their starting date on machines. Therefore, solutions represent a feasible order between operations. Considering the characteristics of a MHFSSP, a solution is viewed as a triplet vector (γ,σ,π), where γ denotes the assignment of each job to a production line, σ is the assignment vector stating which of the parallel machines at a given stage should perform the operation (i.e. Suppose that job j_0 is assigned to line 1 in Fig.1. At stage 2 of line 1, an operation could be performed either on M2 or M3 if both are available. If operation 2 of job j_0 is performed on M2 then $\sigma[2]<0.5$, if M3 is chosen then $\sigma[2]>0.5$) and π determines the sequencing order of jobs and is built as a repetition vector.

3.2 Decoding approach

The decoding approach is based on an earliest starting date approach. All jobs j are assigned to a line l. For every job, a machine m is selected amongst all parallel machines at every stage. To determine the earliest start date of each job j on machine m, we shall consider the order between the operations of a job and positions of jobs on machines. We note C_{jk-1} as the completion time of job j at stage $k-1$ and $C_{j'k}$ is the completion time of job j' at stage k. A job j is operated on machine m at stage k once job j at stage $k-1$ and job j' on machine m at stage k have been completed. For each job j on machine m, the start date equals maximum $(C_{j'k},\, C_{jk-1})$. The start date of a job j at stage k equals the end date of job j at stage $k-1$ with respect to the processing order of operations of job j. Moreover, it equals the end date of job j' at stage k when there is only one machine at stage k. The job j on machine m starts at the earliest date considering the precedence constraints and the availability of energy resource. The computed objective function is equal to the energy cost while adding up a penalty if the makespan, the energy consumption and the peak power upper-bounds have been exceeded.

3.3 Local Search Algorithm

Launching a stochastic local search algorithm permits to generate potential vectors (γ,σ,π) minimizing energy cost with respect to makespan, energy and peak limits by exploring a set of neighbors (γ',σ',π') of the initially generated solution. A number p is randomly selected from $]0,1[$. If $p < 0.33$ then we generate the neighboring vector π' of π. Furthermore, when $p < 0.66$, the neighboring vector σ' of σ is constructed and if $p < 1$ then the neighboring vector γ' of γ is built. The neighboring vectors are generated through randomly applying disturbances on the initial solution's triplet vectors. π' is generated by switching the position of two randomly selected jobs' occurrences. A new vector σ' consists in randomly changing the value of a machine's assignment. Moreover, A different line is selected for a randomly chosen job to generate the neighboring vector of γ. The objective function of the neighboring solution S' obtained from S is then computed. If $objectivefunction(S') < objectivefunction(S)$ then S' is kept as the current solution, and the local search continues until reaching a predefined number of iterations. Finally, $S*$ is kept as the best found solution.

4 Results

The implementation of the algorithm to solve the example in section 2 has given the solution in Fig.2 below. This solution can be represented by the triplet $\pi[i] = [1\ 2\ 2\ 0\ 1\ 0]$, $\sigma[i] = [1\ 0.6\ 1\ 0.2\ 1\ 1]$ and $\gamma[j] = [1\ 1\ 2]$. The algorithm is implemented considering the following parameters: Maximum total energy consumption $TEC_{max} = 500$; maximum power peak $W^T_{max} = 30$; Maximum makespan $C_{max} = 30$. The result is displayed as a Gantt chart in Fig.2 which illustrates the peak power per step time and the periods of energy price variations. The returned cost equals 4962. As can be viewed in this figure, maximum power consumption is reached at periods P1 and P5, the heuristic places operations at the earliest starting date considering the order between operations and the positions of jobs on machines (i.e. the second operation of j_0 is assigned to M3 at stage 2 of line 1 starts right after the end of operation 1 of job j_0).

5 Conclusion

In this abstract, a heuristic approach is investigated for a multi-line hybrid flow shop problem with energy considerations. First, further tests and analysis of the heuristic's performance will be conducted by implementing the algorithm on a set of instances. We will create instances that vary in terms of instance size, flexibility level, upper-bounds on parameters (i.e. makespan, total energy consumption and peak power). Since hybrid flow shop and distributed hybrid flow shop scheduling problems have been treated in the literature using heuristics and meta-heuristics, it would be also interesting to compare the performance of the proposed algorithm to other algorithms by applying it to a set of instances from the literature while adapting them to this particular scheduling problem structure and energy efficiency context.

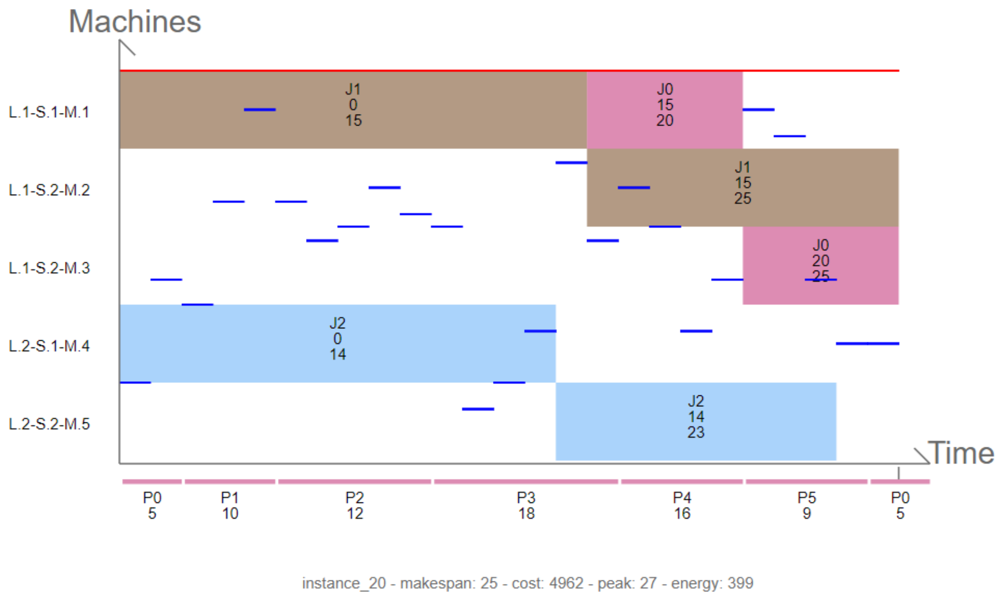


Fig. 2: Obtained solution using the meta-heuristic on the numerical example: Gantt chart

Acknowledgments

This work is supported by the Corenstock Industrial Chair ANR-20-CHIN-0004-01, co-founded by elm.leblanc, IMT and its schools IMT Nord Europe and Mines Saint-Étienne. Funded by the French National Research Agency, the project is also endorsed by the Institut Carnot Télécom & Société Numérique and accredited by the EMC2 cluster.

References

Li, JQ., Sang, H Y., Han, YY., Wang, CG., Gao, KZ., 2018, “Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions", *Journal Of Cleaner Production*, Vol. 181, pp. 584-598.

Schulz, S., Neufeld, JS., Buscher, U., 2019, “A multi-objective iterated local search algorithm for comprehensive energy-aware hybrid flow shop scheduling", *Journal Cleaner Production*, Vol. 224, pp. 421-434.

Gong, G., Chiong, R., Deng, Q., Han, W., Zhang, L., Lin, W., Li, K., 2020, “Energy-efficient flexible flow shop scheduling with worker flexibility", *Expert Systems With Applications*, Vol. 141, pp. 112902.

Lu, C., Gao, L., Yi, J., Li, X., 2021, “ Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in China", *IEEE Transactions on Industrial Informatics*, Vol. 17, pp. 6687-6696.

Wang, J., Wang, L., 2022, “A Cooperative Memetic Algorithm with Learning-Based Agent for Energy-Aware Distributed Hybrid Flow-Shop Scheduling", *IEEE Transactions on Evolutionary Computation*, Vol. 26, pp. 461-475.

Taguemount, S., Lamy, D., Delorme, X., 2023, “Multi-line hybrid flow shop scheduling problem with energy considerations", *20th IEEE International Conference on Networking, Sensing and Control*, Vol. 01, pp. 1-5.

The trade-off between stability and expected makespan in stochastic project scheduling problem

Ziyang Tang[1], Mario Vanhoucke[1,2,3] and Tom Servranckx[1]

[1] Faculty of Economics and Business Administration, Ghent University,Gent 9000, Belgium
`Ziyang.Tang@ugent.be, Tom.Servranckx@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: Resource-constrained project scheduling, Stochastic, Overlap, Tabu search.

1 Introduction

As projects become more and more complex and the external environment increasingly uncertain, projects become susceptible to various risks that could potentially disrupt their execution. Such interruptions can have a profound impact on the availability of project resources and the realized project duration, and project managers should therefore evaluate the possibility and severity of uncertainty before establishing a baseline schedule (Deblaere *et. al.* 2011). By building sufficient buffers against time and resource conflicts into the project schedule, they can be better prepared for possible disruptions and better mitigate the impact on the planned schedule when unforeseen emergencies arise (Lambrechts *et. al.* 2008).

Of all the sources of uncertainty that exist in a project, disruptions in the duration of activities are a central concern for project managers (Chen *et. al.* 2018). Since the duration estimates in the project schedule always contain a certain degree of uncertainty, deviations from these estimates must be taken into account (Van de Vonder *et. al.* 2008). Therefore, significant research efforts have been devoted to the domain of uncertainty in project scheduling, with an emphasis on creating schedules that show a high degree of resilience in the face of unforeseen disruptions.

This study will extend the deterministic *resource-constrained project scheduling problem* (RCPSP) to a stochastic version focusing on the uncertainty of the duration of activities. More specifically, the study aims to construct a project schedule that changes as little as possible when disruptions occur during project execution by using activity buffers. The project schedule will be generated by an adaptive Tabu Search method to optimize the trade-off between stability and the expected project duration. This method makes use of three definitions of possible conflicts that may arise in the project schedule to obtain the best possible schedule.

2 Problem Statement

2.1 Stochastic activity duration

In the stochastic project scheduling problem, a project can be represented by an acyclic activity-on-the-node network $G = (N, A)$, where the non-dummy activities are characterized by the node set $N = \{1, ..., n\}$ and the precedence relations between activities i and j by the arc set A, i.e. $(i, j) \in A$. The project also consists of a dummy start activity 0 and a dummy end activity $n + 1$, which both have a zero duration and resource requirement. Each activity $i \in N$ has a non-zero duration d_i and a non-zero requirement $r_{i,k}$ of resource type k per unit time. There are K types of renewable resources and each resource type k has a fixed resource availability R_k. TS_i denotes the set of all the transitive successor activities of activity i and S_i^* denotes the set of all its immediate successors. A schedule $\mathbf{s}$ contains the start times s_i of all activities $i \in N$ and the project should be completed prior to the deadline δ. The weight w_i represents the instability cost caused by activity i and has a crucial part in the trade-off between the stability and the expected project duration of a schedule.

This paper assumes that every activity i has both a certain part and an uncertain part of the duration, and the uncertain part follows a lognormal distribution, i.e. $d_i \sim Ln(\mu_i, \sigma_i^2)$ with the mean duration μ_i and standard deviation σ_i. This uncertain part of the duration is then limited to a range of the activity duration between two percentiles, a lower percentile and an upper percentile. As a result, the activity execution period can be formulated as a combination of the certain and uncertain part.

Therefore, the cumulative distribution function $cdf_i(t)$ of d_i will represent the possibility of activity i being completed before time t given the uncertain part of the activity execution period and $1 - cdf_i(t)$ represents the possibility that activity i is not completed before time t. This information will subsequently be used to determine the probability and impact of conflicts between activity i and its successors, and thus the maximum risk caused by the uncertainty of activity i. This degree of conflict risk among related activities impacts the instability of the project execution, and in the following section the types of potential sources of conflicts are briefly outlined.

2.2 Three types of conflict overlap

Three sources of conflicts will be analyzed and a measure of the possibility of conflicts will be proposed along the following lines:

Precedence conflict Conflicts arise when, for precedence related activities, the succeeding activity starts before the certain completion of the preceding activity and the total risk of the conflict can be computed based on $cdf_i(t)$.

Resource conflict Besides the precedence relations, the resource availability also constrains the activities in the project. When creating the baseline schedule based on the mean value of the activity durations μ_i, the overlap of the uncertain parts of the duration from various activities could potentially result in a resource conflict and should therefore be considered.

Deadline conflict Project deadlines limit the execution of activities near the end of the project and thus restricts the certain execution interval of activities to exceed the project deadline. However, there is a possibility that the uncertain execution interval of these activities exceeds the project deadline, resulting in a deadline conflict.

Given these three types of conflicts, the overall conflicts in the project are determined by multiplying the individual activity conflicts together with the activity weights. The more conflicts in the project, the greater the project risk during actual implementation. Therefore, this paper aims to minimize the project conflicts by reasonably arranging the start time of activities by using an efficient solution method discussed in the following section.

3 Solution Method

Our solution method is derived from the Earliest-Start Policy by Stork F. and Uetz M. (2005) in order to reduce the number of *temporary minimal forbidden sets (TMFS)* in a baseline schedule using a Tabu Search procedure. Below, we briefly outline the different steps of the algorithm.

Step 1 Initiate a schedule $\mathbf{s}_0$ using the serial schedule generating scheme (SSGS) with a random activity list AL and a time buffer list BL presented in Lambrechts *et. al.* (2008). The AL and BL will be added to the activity tabu list ATL and buffer tabu list BTL respectively.
Step 2 Find out all the *TMFS*, in which the resource requirement of all the activities is over-usage but the resource conflict will be eliminated when any one of the activities is removed.
Step 3 Adjust the start time of activities in *TMFS* forward or backward in order to mitigate the conflicts and add all the local improved solutions into the tabu lists ATL and BTL.
Step 4 Generate a new AL after a certain number of iterations without improvement in BL, by swapping the activity pair following the greedy strategy to reduce the most weighted overlap respecting the precedence relationship.
Step 5 Go back to step 2 until any of the stop criteria is reached.

4 Experiments

4.1 Generation of test instances

We use the 480 instances with 30 activities from the well-known PSPLIB for the RCPSP and assume that the activity durations follow a lognormal distribution, as shown in Table 1. The weights of all the non-dummy activities are generated from a discrete, triangular shaped distribution between 1 and 10 with respect to $P(w_i = x) = 0.21 - 0.02x$ Van de Vonder *et. al.* (2008). The weight of the dummy end activity denotes the marginal cost of not meeting the project deadline and is important for dealing with the deadline conflicts. It is set equal to w_{n+1} will be fixed at $\lfloor 10 \times 3.85 \rfloor$. In our experiment, the project deadline δ of each instance is set equal to the optimal makespan in the deterministic environment and a deadline factor α. This factor models the trade-off between the project stability and project duration, and is set to three levels in our study (10%, 20%, 30%). For the evaluation of a schedule, the weighted deviation of realized and scheduled start times represents the <u>stability measure</u> *(M1)* and the total number of times that the deadline is surpassed represents the <u>protection measure</u> *(M2).*

Table 1. The parameters of the lognormal distribution

	μ	σ^2	**variance**
Lgn1	$ln(\frac{3d_i^*}{\sqrt{10}})$	$ln(\frac{10}{9})$	$\frac{d_i^*}{9}$
Lgn2	$ln(\frac{\sqrt{3}d_i^*}{2})$	$ln(\frac{4}{3})$	$\frac{d_i^*}{3}$
Lgn3	$ln(\frac{d_i^*}{\sqrt{2}})$	$ln(2)$	d_i^*

4.2 Benchmarks schedules

The schedules generated using our will be benchmarked against three types of schedules:

Benchmark Schedule 1 (BS1) This benchmark solution assumes a static environment as we use a state-of-the-art heuristic method to solve the basic RCPSP under deterministic activity duration.

Benchmark Schedule 2 (BS2) This benchmark solution is generated using the robustness measure proposed in Lambrechts *et. al.* (2008), i.e. $MaxFS = \sum_{j=1}^{N} CIW_j \sum_{i=1}^{FS_j} e^{-i}$.

Benchmark Schedule 3 (BS3) is generated by the STC method proposed in Van de Vonder *et. al.* (2008).

The results of the computational experiments provide the following insights. First, a increased attention to stochastic activity durations improves the schedule stability and decreases the expected project makespan compared to the deterministic assumption. Second, the method adopted in this study proves to be more effective than existing methods in improving both the stability and reducing the project makespan. Finally, decision makers can achieve a trade-off between the schedule stability and the protection of the project makespan by adjusting the relative activity weights.

5 Conclusion

In uncertain and complex projects, the assumption of deterministic activity duration has become obsolete and project managers need to consider stochastic activity durations. In order to deal with the trade-off between the schedule stability and the project makespan, we study three types of conflicts (precedence, resource and deadline). We conclude that our adaptive Tabu search procedure can generate better schedules compared to three benchmarks in terms of both schedule stability and expected makespan objectives.

References

Chen Z. , Demeulemeester E. and Bai S. , 2018, "Efficient priority rules for the stochastic resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 270, pp. 957-967.

Deblaere F. ,Demeulemeester E. and Herroelen W. , 2011, "Reactive scheduling in the multi-mode RCPSP", *Computers and Operations Research*, Vol. 38, pp. 63-74.

Lambrechts O. , Demeulemeester E. and Herroelen W. , 2008, "A tabu search procedure for developing robust predictive project schedules", *International Journal of Production Economics*, Vol. 2, pp.493-508.

Ma Z. , Demeulemeester E. and He Z. , 2019, "A computational experiment to explore better robustness measures for project scheduling under two types of uncertain environments", *Computers & Industrial Engineering*, Vol. 131, pp. 382-390.

Stork F.,Uetz M., 2005, "On the generation of circuits and minimal forbidden sets", *Mathematical programming*, Vol. 120, pp. 185-203.

Van de Vonder S, Demeulemeester E, Herroelen W. , 2008, "Proactive heuristic procedures for robust project scheduling: An experimental analysis", *European Journal of Operational Research*, Vol. 189, pp. 723-733.

Sallam K M, Chakrabortty R K, Ryan M J. , 2021, "A reinforcement learning based multi-method approach for stochastic resource constrained project scheduling problems", *Expert Systems with Applications*, Vol. 169, pp. 114479.

Ramos A S, Miranda-Gonzalez P A, Nucamendi-Guillén S , 2023, "A Formulation for the Stochastic Multi-Mode Resource-Constrained Project Scheduling Problem Solved with a Multi-Start Iterated Local Search Metaheuristic", *Mathematics*, Vol. 11, pp. 337.

Bilevel adversarial single machine scheduling problems with job selection

T'kindt V.[1], Della Croce F.[2] and Agnetis A.[3]

[1] Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée, Tours, France,
`tkindt@univ-tours.fr`
[2] DIGEP, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy, CNR, IEIIT, Torino, Italy.
`federico.dellacroce@polito.it`
[3] Università degli Studi di Siena, Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, via Roma 56, 53100 Siena, Italy.
`agnetis@diism.unisi.it`

Keywords: Scheduling, bilevel optimization.

1 Introduction

In this contribution we focus on a particular setting in which two agents are concerned by the scheduling of a set of n jobs. The first agent, called the *leader*, can take some decisions before providing the jobset to the second agent, called the *follower*, who then takes the remaining decisions to solve the problem. As an example, the leader could select a subset of $n' \leq n$ jobs that the follower has to schedule. Notice that the decisions the agents can take are exclusive: in this example, the follower cannot decide the jobs to schedule and the leader cannot schedule the jobs. This setting falls into the category of *bilevel optimization* (Dempe et al. 2015). In such problems it is assumed that the leader and the follower pursue their own objectives which can be contradictory, so leading to very hard optimization problems. The existence of potential multiple optimal solutions to the follower's problem leads to two classes of problems (Dempe 2003): optimistic and pessimistic bilevel problems. In an *optimistic* bilevel problem, it is assumed that the follower returns, among all its optimal solutions, the one that leads to the smallest value of the leader's objective function. On the contrary, in a *pessimistic* bilevel problem, it is assumed that the follower returns its optimal solution that is the worst for the leader's objective function. Another class of bilevel optimization problems can be met in the literature, that are called *adversarial* bilevel problems. In this setting, the leader takes decision so that the optimal solution of the follower's problem is the worst possible.

Recently, many papers on bilevel combinatorial optimization appeared, here we refer to (Caprara et al. 2016, Della Croce and Scatamacchia 2019, Fischetti et al. 2017, Fischetti et al. 2019, Fischetti et al. 2018, Woeginger 2021) just to mention a few. On the other hand, to the authors knowledge, the literature on bilevel scheduling is much more limited. (Karlof and Wang 1996), and next (Abass 2005), first considered a flowshop scheduling problem with operators where the leader determines the schedule of operators to minimize the sum of job completion times while the follower determines the schedule of jobs to minimize the makespan. (Kovacs and Kis 2011) consider an optimistic bilevel single machine problem in which the leader selects the set of jobs the follower next schedules. Following the standard scheduling notation, this problem is denoted by $1|OPT{-}n, r_j, \tilde{d}_j|\sum_j w_j^L x_j, \sum_j w_j^F C_j^F$, with $OPT{-}n$ meaning that the optimistic setting is considered and $|\sum_j w_j^L x_j$ is the total weight of selected jobs. (Kis and Kovacs 2012) considered both the $P|OPT-A_k|\sum_j w_j^L C_j^L, \sum_j w_j^F C_j^F$ and $P|PES-A_k|\sum_j w_j^L C_j^L, \sum_j w_j^F C_j^F$

problems: they correspond to the optimistic (OPT) and pessimistic (PES) settings of the problem where the leader defines the set of jobs A_k assigned to any machine k while the follower sequence them on each machine. The problem is shown to be strongly $\mathcal{NP}$-hard. (Kis and Kovacs 2012) considered again the $1|OPT-n, r_j, \tilde{d}_j|\sum_j w_j^L x_j, \sum_j w_j^F C_j^F$ problem and showed that it is weakly $\mathcal{NP}$-hard.

In this work we focus on single machine scheduling problems in the bilevel adversarial setting. It is a follow up of (T'kindt et al. 2024). We define in section 2 the different scenario that can be met and we provide a synthesis of the proposed results. Section 3 provides details about some of these results. Section 4 consider extensions of these problems to optimistic problems.

2 Adversarial bilevel single machine scheduling

It is assumed that n jobs are to be scheduled on a single disjunctive machine. Each job j is defined by a processing time p_j and, depending on the problem, a weight w_j or a due date d_j. The follower is scheduling jobs so that its objective function $f^F \in \{\sum_j C_j^F, \sum_j w_j^F C_j^F, L_{max}^F, \sum_j U_j^F\}$ is minimized. Beforehand, the leader can take some decisions that impact the instance solved by the follower. In this contribution we consider the scenario when a set of N jobs is given to the leader. Next, he selects a subset of $n \leq N$ jobs, for any given n, that the follower schedules.

The leader takes decisions so that the optimal solution computed by the follower is as bad as possible. In this contribution, we focus on scenario (S1). Considering the three-field notation for scheduling problems, the corresponding problems are denoted by $ADV-n$. The proposed results, discussed during the conference, are summarized in Table 1. Notice that we focused on problem that are polynomially solvable in a classical setting: those that are already known to be $\mathcal{NP}$-hard in the classical setting, e.g. $1||\sum_j T_j$, cannot be polynomially solvable in the adversarial bilevel setting. Table 1 shows that, surprisingly, the minimization of $\sum_j w_j C_j^F$ is an open problem. Also, minimizing the number of tardy jobs $\sum_j U_j^F$ can be achieved in polynomial time but the problem of deciding whether or not there exists a schedule with jobs in $\mathcal{L}$ tardy, with $\mathcal{L}$ a given set of jobs, is an $\mathcal{NP}$-hard decision problem.

Polynomially solvable problems	
$1\|ADV-n\|\sum_j C_j^F$	$1\|ADV-n\|L_{max}^F$
$1\|ADV-n\|\sum_j U_j^F$	
$\mathcal{NP}$-hard problem	
$1\|ADV-n, \mathcal{L}\|-$	
Open problem	
$1\|ADV-n\|\sum_j w_j C_j^F$	

Table 1. Complexity status of some bilevel single machine scheduling problems

3 Solving problem $1|ADV-n|\sum_j w_jC_j^F$ and $1|ADV-n|\sum_j U_j^F$

First, consider the unweighted case, i.e. the minimization of $\sum_j C_j^F$. The bilevel problem can solved in polynomial time as follows: (1) the leader selects the n jobs with the largest processing times, (2) the follower schedules these n jobs in increasing order of their processing time (SPT order).
When generalizing to the weighted case, one could think that the following algorithm is optimal: (1) the leader selects the n jobs with the largest ratio $\frac{p_j}{w_j}$, (2) the follower schedules these n jobs in increasing order of their ratio (WSPT order). It is a natural intuitive extension of the algorithm for the unweighted case but that is, unfortunately, not correct. It is enough to consider the following counter example with $N = 4$ jobs, $p = [10; 1; 3; 1]$ and $w = [1000; 2; 4; 1]$. Suppose $n = 3$ and let $s = (2, 3, 4)$ be the solution obtained if the leader selects the 3 jobs with largest ratio $\frac{p_j}{w_j}$. We have $\sum_j w_jC_j^F(s) = 23$. But if the leader selects jobs $\{1, 2, 3\}$ then WSPT gives $s' = (1, 2, 3)$ and $\sum_j w_jC_j^F(s') = 10078$.

A thorough analysis of the problem enables to establish the following result.

Lemma 1. *Let be two jobs k and ℓ such that:*

1. $\frac{p_k}{w_k} < \frac{p_\ell}{w_\ell}$, *and*
2. *There are at least n jobs j such that* $\frac{p_k}{w_k} < \frac{p_j}{w_j}$, *and*
3. *There are strictly less than n jobs j such that* $\frac{p_\ell}{w_\ell} < \frac{p_j}{w_j}$.

Then, the two following conditions hold:

(C1) if $w_\ell \geq w_k$ then there does not exist an optimal solution to the bilevel problem in which k is selected and not ℓ.
(C2) if $w_\ell < w_k$ and $p_k \geq p_\ell$ then there does not exist an optimal solution to the bilevel problem in which ℓ is selected and not k.

However, the complexity of the $1|ADV-n|\sum_j w_jC_j^F$ problem remains open: this problem seems to be at the border between easy and hard problems.

Now, assume that each job j has a due date d_j and the follower minimizes the number of tardy jobs $\sum_j U_j^F$ with $U_j^F = 1$ if $C_j^F > d_j$ and 0 otherwise. We assume that jobs are indexed following the non-decreasing order of their due date (EDD order), i.e. $d_1 \leq ... \leq d_N$. To solve this problem, we propose a dynamic programming algorithm.

Let $\mathcal{C}(j, k, \epsilon)$ be the value of the smallest makespan when k jobs among the j first ones are selected and ϵ of them are tardy. We have:

$$\mathcal{C}(j,k,\epsilon) =$$
$$\max\left(\underbrace{\mathcal{C}(j-1,k,\epsilon)}_{j \text{ is not selected}}\ ; \underbrace{\min\Big(\underbrace{\mathcal{C}(j-1,k-1,\epsilon-1)}_{j \text{ is tardy}};\quad \underbrace{\mathcal{C}(j-1,k-1,\epsilon)+p_j}_{\text{if } \mathcal{C}(j-1,k-1,\epsilon)+p_j \leq d_j;\ +\infty \text{ otherwise}}\quad\Big)}_{j \text{ is selected}}\right)$$

with $\mathcal{C}(j, k, \epsilon) = +\infty$ whenever $\epsilon < 0$ or $\epsilon > j$, $\mathcal{C}(j, k, \epsilon) = -\infty$ whenever $j < k$, and $\mathcal{C}(0, 0, 0) = 0$. Notice that the two terms inside the min() correspond to j being scheduled tardy and, respectively, being scheduled early. Besides, whenever the min() returns $+\infty$, this value must be transformed into $-\infty$ during the recursions. This is due to the fact that when the min() returns $+\infty$ there is no feasible solution when j is selected and this term must be transformed to $-\infty$ to be taken into account in the max() as an infeasible decision. Solving the bilevel problem requests to determine the greatest value ϵ_U such that $\mathcal{C}(N, n, \epsilon_U) \neq -\infty$. This dynamic programming algorithm requires $O(Nn^2)$ time and space which implies that the $1|ADV - n|\sum_j U_j^F$ problem can be solved in polynomial time.

To conclude this section, and in order to further highlight how challenging bilevel scheduling problems are, we consider a problem closely related to the above $1|ADV - n|\sum_j U_j^F$ problem. Assume that the list $\mathcal{L}$ of tardy jobs is imposed, that is the problem turns to a decision problem where the leader has to select $(n - |\mathcal{L}|)$ jobs so that when the follower schedules the n jobs, only those in $\mathcal{L}$ are tardy. This problem is denoted by $1|ADV - n, \mathcal{L}|-$ and is proved to be $\mathcal{NP}$-complete by reduction from Equal-size Partition.

4 Extension to optimistic problems

At the conference we will also provide results about some optimistic problems. We will notable focus to the case when the follower minimizes the sum of completion times $\sum_j C_j^F$. We will show that when the leader minimizes the number of tardy jobs $\sum_j U_j^L$ the bilevel problem can be solved in polynomial time. However, the problem turns to be $\mathcal{NP}$-hard when the leader minimizes the weighted number of tardy jobs $\sum_j w_j U_j^L$ or the total tardiness $\sum_j T_j^L$.

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

Abass S.A.: Bilevel programming approach applied to the flow shop scheduling problem under fuzziness. Computational Management Science. 2: 279–293 (2005)

Caprara, A., Carvalho, M., Lodi, A., Woeginger, G.: Bilevel Knapsack with Interdiction Constraints. INFORMS Journal on Computing. 28, 319–333 (2016)

Della Croce, F., Scatamacchia, R.: An exact approach for the bilevel knapsack problem with interdiction constraints and extensions. Mathematical Programming, 183, 249–281 (2020).

Dempe, S.: Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. Optimization. 52, 333–359 (2003).

Dempe, S., Kalashnikov, V. Perez-Valdes, G.A., Kalashnikova, N., 2015, "Bilevel programming problems", *Springer*.

Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: Interdiction Games and Monotonicity, with Application to Knapsack Problems. INFORMS Journal on Computing. 31, 390–410 (2019).

Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: A New General-Purpose Algorithm for Mixed-Integer Bilevel Linear Programs. Operations Research. 65, 1615–1637 (2017).

Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: On the use of intersection cuts for bilevel optimization. Mathematical Programming. 172, 77–103 (2018).

Karlof, J.K., Wang, W.: Bilevel programming applied to the flow shop scheduling problem. Computers and Operations Research. 23:5, 443–451 (1996).

Kovacs, A., Kis, T.: Constraint programming approach to a bilevel scheduling problem. Constraints. 16, 317–340 (2011).

Kis, T., Kovacs, A.: On bilevel machine scheduling problems. OR Spectrum. 34, 43–68 (2012).

T'kindt, V., Della Croce, F., Agnetis, A.: Single machine adversarial bilevel scheduling problems. European Journal of Operational Research. 315(1), 63–72 (2024).

Woeginger, G.: The trouble with the second quantifier. 4OR. 19, 157–181 (2021).

Solving a robust multi-skill resource-constrained multi-project scheduling problem

Rahman TORBA[1,2], Stéphane DAUZÈRE-PÉRÈS[1], Claude YUGMA[1], Cédric GALLAIS[2] and Juliette POUZET[2]

[1] Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, Centre CMP, Departement SFL, F - 13541 Gardanne, France
`r.torba, yugma, dauzere-peres@emse.fr`
[2] SNCF, 93210 Saint-Denis, France
`rahman.torba, cedric.gallais, juliette.pouzet@sncf.fr`

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

1 Introduction

In this work, a robust real-life multi-skill resource-constrained multi-project scheduling problem is modeled and solved. The problem is motivated by the scheduling problem faced by the ten heavy maintenance centers of SNCF, the French national railway company, where the most heavy maintenance operations, and thus longest ones (several weeks to several months), are carried out. Several rolling stock units are maintained simultaneously, and each unit is considered as a project. To complete each project, a set of activities that require resources with different skills must be performed. Different types of multi-skilled resources (maintenance operators and machines) with different characteristics and constraints are taken into account (Torba et al. 2024).

In the context of heavy maintenance, many operations are performed by human operators and estimating the required workload is not easy. Moreover, uncertain tasks, with a known probability to be required or not, should also be considered. This paper extends our previous work in (Torba et al. 2024) and integrates the different uncertainties in the decision process to generate robust schedules i.e., schedules that remain efficient in case of disturbances. To build robust schedules, involving hundreds of projects and thousands of operations, scenarios with a limited budget of uncertainty (Bertsimas & Sim 2004) are embedded into a memetic algorithm (MA). The objective is to maximize the probability of meeting the customer deadlines.

2 Literature review

The resource-constrained project scheduling problem (RCPSP) is a NP-hard optimization problem that involves scheduling a set of activities subject to precedence constraints and resource availability (Deblaere et al. 2011). Since its formulation by (Pritsker et al. 1969), many extensions and solution methods have been proposed. A generalization of the RCPSP is to consider the resource-constrained multi-project scheduling problem (RCMPSP) which better addresses the complexity of some real-world applications (Lova et al. 2000). Solving the RCMPSP is more challenging due to the simultaneous management of multiple projects, leading to larger problem sizes and other factors to consider such as project tightness and delay penalties. The Multi-Skill Resource-Constrained Project Scheduling Problem (MSRCPSP) extends the RCPSP by considering resources that have multiple skills (Bellenguez & Néron 2004). Over the past decade, researchers are showing a growing interest in the the Stochastic Resource-Constrained Project Scheduling Problem (S-RCPSP). In fact, in many real-world problems, ignoring uncertainties may lead to schedules of poor quality under real conditions (Bruni et al. 2017). The main approaches proposed in the literature to deal with uncertainties are:

– **Reactive Scheduling**, which involves finding the best scheduling policies when disruptions affect the initial schedule. This approach is relevant when disruptions are unpredictable, lacking any prior information about uncertainties (Deblaere et al. 2011).
– **Predictive Scheduling**, which requires a precise estimation of uncertainties to compute a predictive baseline schedule. However, accurate data on uncertainties being often not available, the disruptions are tackled by the reactive approach, recognized as predictive-reactive scheduling (Herroelen & Leus 2005) .
– **Stochastic Scheduling**, when the probability distributions of uncertainties are known. However, two major problems arise: the availability of probability data and the practical difficulty of solving stochastic models (Deblaere et al. 2011).
– **Robust (or Proactive) Scheduling**, where robust schedules are determined that are less sensitive to disturbances. Unlike predictive and stochastic scheduling, this approach can be used even if uncertainties are known only approximately (Bruni et al. 2017).

A robust scheduling approach is the most reasonable approach to deal with real-world problems where data is not (always) accurate, in particular in the considered problem where activities are performed by maintenance operators. Hence, a novel robust optimization method for the RCPSP with multiple projects and multi-skilled resources is presented.

3 Generation of scenarios and solution approach

3.1 Generation of scenarios

To generate robust schedules, we first start by defining scenarios based on the analysis of historical data. For uncertain tasks, we compute the probability of having to perform the task or not. In each scenario, an uncertain task has either a workload equal to zero with a probability of p_1, or a non-zero workload with a probability of p_2 (where $p_1 + p_2 = 1$). Following the same reasoning, to handle poor workload estimation, we define several realization modes for a given task. Each mode has a different workload and an associated probability of occurrence. As many uncertainties are observed, the generation of all possible scenarios is computationally intractable. Using the worst scenarios to determine a robust schedule is (in most cases) very pessimistic and over-conservative (Bruni et al. 2017). Furthermore, the chances that all activities take their worst modes are nearly zero. Generating a subset Ω of random scenarios using the probabilities associated to each mode and each task could be a potential solution. However, this leads to the same issue as when generating all scenarios. In fact, a huge number of scenarios must be generated to effectively capture uncertainties, and computing a robust solution with a lot of scenarios induces significant computational challenges. Considering that our primary objective is to tackle large industrial instances, we limit ourselves to a subset of scenarios with a specified budget of uncertainty (Bertsimas & Sim 2004). To better represent reality, distinct budgets of uncertainty $\Gamma_{r,k,t}$ are defined and computed using historical data for each resource r, skill k and period t. By increasing the budget of uncertainty, the approach tends to be more conservative and, for large enough $\Gamma_{r,k,t}$, it is similar to considering worst-case scenarios. More precisely, $\Gamma_{r,k,t}$ defines the following uncertainty polytope: $\Theta_{r,k,t} = \{\phi_{a,r,k}, a \in A / ES_a = t, \sum_{a \in A}(\phi_{a,r,k} - \bar{\phi}_{a,r,k}) <= \Gamma_{r,k,t}\}$, where $\phi_{a,r,k}$ (respectively $\bar{\phi}_{a,r,k}$) is the realized (respectively deterministic) workload for activity $a \in A$, resource r and skill k, and ES_a is the earliest starting time of activity a. To generate scenarios, the scheduling horizon H is divided into months. The general idea is to model the variation of the task workload, transitioning from the planned values to those observed in historical data. A scenario w represents a set of realized workloads necessary to process the activities: $w = \{\phi_{a,r,k}, a \in A, r \in R, k \in K\}$. Note that $\phi_{a,r,k}$ has a direct impact on the resource requirement and, consequently, on the starting time of the activities for a given sequence. Furthermore, if for an activity $a \in A$, $\phi_{a,r,k} = 0$ for all $r \in R$ and $k \in K$, then the activity is not required, and the processing time p_a is set to 0.

3.2 Solution approach

To address this large industrial problem, an efficient memetic algorithm (MA) was implemented, and validated in both real instances and benchmark instances with multiple projects, for the deterministic problem (Torba et al. 2024). The same algorithm was adapted to compute a robust schedule that maximizes the probability of meeting the rolling stock due dates. Given a set of scenarios Ω and a sequence of activities S (representing an assignment and a precedence feasible activity list), this probability is defined as the tardiness service level (following (Dauzère-Pérès et al. 2008) and (Flores-Gómez et al. 2023)) and can be written: $\alpha(S, \Omega, e) = \mathbb{P}(T_e(S, \Omega) = 0)$ for rolling stock unit e. As we deal with several projects (i.e. rolling stock units), and since the projects do not have the same priority, the weighted sum of the tardiness service level of the rolling stock units is maximized: $\alpha(S, \Omega, \mathcal{E}) = \sum_{e \in \mathcal{E}} w_e \mathbb{P}(T_e(S, \Omega) = 0)$, where w_e is the weight of project $e \in \mathcal{E}$. The initial solutions are computed using the deterministic data and the original memetic algorithm. Then, the service level of each initial solution is evaluated and further improved using the MA. The computation of the service level of a sequence S, given a set of scenarios Ω, is formalized in Algorithm 1.1.

Algorithm 1.1 Evaluation of $\alpha(S, \Omega, \mathcal{E})$ of a given sequence S

```
1: Initialize $\mathbb{P}(T_e(S,\Omega) = 0)$ to zero for each $e \in E$;
2: for $\omega \in \Omega$ do
3:     $\{T_e, e \in E\} \leftarrow Evaluate(S_i(\omega))$;▷ Returns the tardiness of each project given scenario $w$;
4:     for $e \in \mathcal{E}$ do
5:         if $T_e$==0 then
6:             $\mathbb{P}(T_e(S,\Omega) = 0) \leftarrow \mathbb{P}(T_e(S,\Omega) = 0) + 1/|\Omega|$;
7:         end if
8:     end for
9: end for
10: $\alpha(S_i, \Omega, \mathcal{E}) \leftarrow \sum_{e \in \mathcal{E}} w_e \mathbb{P}(T_e(S,\Omega) = 0)$;
11: return $\alpha(S_i, \Omega, \mathcal{E})$;
```

4 Numerical results

Preliminary results show that schedules with a very good service level can be computed in a few iterations, even if the service level of the initial schedules is very poor. An example is illustrated in Figure 1, where the service level of the best initial solution is 20%, and after 1,200 computed schedules the MA found a solution with a service level of 100%. Note that this is a real instance of a SNCF maintenance center that includes 174 projects and 4,553 activities. To evaluate the service level, 500 scenarios with a budget of uncertainty were generated.

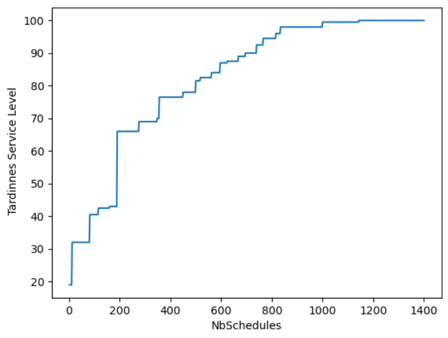


Fig. 1. Convergence of the service level (in %) over the number of computed schedules

5 Conclusions

An original robust multi-skill resource-constrained multi-project scheduling problem is addressed. Different uncertainties are considered and modelled using scenarios defined with a budget of uncertainty and real data. An adaptation of a memetic algorithm proposed for the deterministic problem is proposed to determine robust schedules. Computational experiments, with different budgets of uncertainty and numbers of scenarios are being conducted, the results of which will be presented and discussed at the conference.

Bibliography

Bellenguez, O. & Néron, E. (2004), Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills, *in* 'International conference on the practice and theory of automated timetabling', Springer, pp. 229–243.

Bertsimas, D. & Sim, M. (2004), 'The price of robustness', *Operations research* **52**(1), 3553.

Bruni, M. E., Pugliese, L. D. P., Beraldi, P. & Guerriero, F. (2017), 'An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations', *Omega* **71**, 66–84.

Dauzère-Pérès, S., Castagliola, P. & Lahlou, C. (2008), 'Service level in scheduling'.

Deblaere, F., Demeulemeester, E. & Herroelen, W. (2011), 'Proactive policies for the stochastic resource-constrained project scheduling problem', *European Journal of Operational Research* **214**(2), 308–316.

Flores-Gómez, M., Borodin, V. & Dauzère-Pérès, S. (2023), 'Maximizing the service level on the makespan in the stochastic flexible job-shop scheduling problem', *Computers & Operations Research* **157**, 106237.

Herroelen, W. & Leus, R. (2005), 'Project scheduling under uncertainty: Survey and research potentials', *European journal of operational research* **165**(2), 289–306.

Lova, A., Maroto, C. & Tormos, P. (2000), 'A multicriteria heuristic method to improve resource allocation in multiproject scheduling', *European journal of operational research* **127**(2), 408–424.

Pritsker, A. A. B., Waiters, L. J. & Wolfe, P. M. (1969), 'Multiproject scheduling with limited resources: A zero-one programming approach', *Management science* **16**(1).

Torba, R., Dauzère-Pérès, S., Yugma, C., Gallais, C. & Pouzet, J. (2024), 'Solving a real-life multi-skill resource-constrained multi-project scheduling problem', *Annals of Operations Research* pp. 1–46.

Generation, application and empirical evaluation of a hybrid risk model for forecasting project duration and cost

Izel Unsal Altuncan[1], Mario Vanhoucke[1,2,3]

[1] Ghent University, Belgium
`izel.unsalaltuncan@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

Recent studies acknowledge that project risks are not independent but rather dependent through cause-effect relationships. Tools to facilitate these relationships are referred to as risk models serving in sensitivity analysis, risk response planning, and forecasting. This research, however, specifically emphasizes on forecasting.

In literature two methods are used for forecasting with risk models. The first method is the Bayesian Networks (BN) for forecasting project duration and cost relying on probabilistic causation between risks. The second method is the Structural Equation Modeling (SEM) founded on cause-effect reasoning. BN exhibits more effectively in forecasting compared to SEM due to the ability to update forecasts when the model is introduced with new data. SEM is less optimal because of the potentially changing model structure when new data is introduced. Based on this limitation, also noted in Gupta and Kim (2008), this research utilizes these two methods within a hybrid procedure. First, SEM is used to validate theoretical relationships derived from the literature. Then a BN relying on the validated SEM as an input is subsequently used for predicting the final performance of unseen projects. The accuracy results will be compared against forecasts generated by various benchmark methods. The general methodology is performed in four phases, each detailed in the following sections.

2 Phase 1. Theoretical risk model

The theoretical model in Figure 1 depicts proposed cause-effect relationships between risk variables and the final project performance relying on the previous research Vanhoucke (2012). The model sonsists of two types of variables, *Latent variables* illustrated with ellipses and *observable variables* illustrated with rectangles. *Latent variables* cannot be directly measured and must therefore be calculated using statistical methods over the *observable variables*. Latent variables consist of Network Topology (NT), mean and standard deviation of Time Sensitivity (TSm and TSsd) and Cost Sensitivity (CSm and CSsd) which are assumed to causally affect two dependent risk variables, Time (T) and the Cost (C) of the project. Observable indicators consists of given project network indicators (for NT) and risk metrics for time and cost sensitivity (for TS and CS) and performance data for T (Actual Duration / Planned Duration) and C (Actual Cost / Budget at Completion).

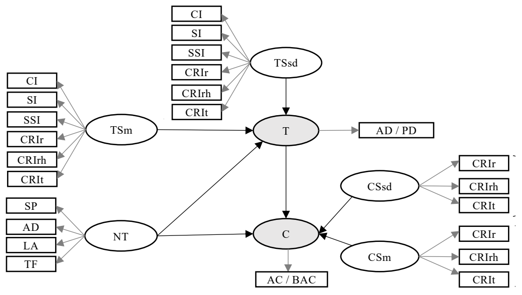


Fig. 1. Theoretical model

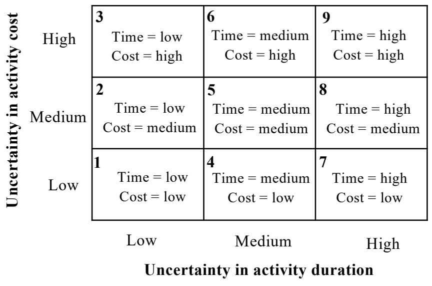

Uncertainty in activity cost
High
Medium
Low
3 Time = low Cost = high
6 Time = medium Cost = high
9 Time = high Cost = high
2 Time = low Cost = medium
5 Time = medium Cost = medium
8 Time = high Cost = medium
1 Time = low Cost = low
4 Time = medium Cost = low
7 Time = high Cost = low
Low
Medium
High
Uncertainty in activity duration

Fig. 2. 9 simulation scenarios

3 Phase 2. Data generation and simulation scenarios

This phase generates the project data for validating the theoretical model utilizing 900 artificial project networks from Vanhoucke et al. (2008). During simulation, uncertainty in the activity durations and costs are modelled across nine different simulation scenarios given in Figure 2, controlled by the coefficient of variation at different levels. Duration variation is modelled using triangular distributions with minimum, most likely and maximum values where the most likely value is assumed to be the baseline duration. Cost variation is modelled by employing combinations of fixed cost and variable cost sampled from uniform distributions, each having different ranges.

4 Phase 3. Model construction and training

This phase is designed in two steps. In the first step, SEM is employed to assess the fit between the theoretical model and data, using the Maximum Likelihood Estimation algorithm resulting in two different model structures. The model on the left side of Figure 3 is resulted from scenarios 1, 2, 3, 4, and 7, excluding CS variables since activity costs are higly correlated with the activity durations due to low variation either in duration or cost. Conversely, the model on the right side emerges from scenarios 5, 6, 8, and 9, retaining the cost sensitivity variable (CSsd) in the theoretical model. This highlights the potential impact of CSsd on project performance, in addition to the TSsd and TSm when variation in activity duration and cost are either medium or high. The first step of this phase results in two model structures accompanied by SEM parameter sets including factor loadings to quantify the importance of each observable indicator in measuring the respective risk variable.

In the second step of this phase, validated observable indicators along with the factor loadings are employed to calculate scores for the risk variables which cannot be directly measured. These scores are then utilized to train BN models with structures determined in the initial step by the SEM. Finally, Bayesian Maximum Likelihood Estimation algorithms are employed to estimate BN parameter sets, consisting of conditional probability distributions for each risk variable.

5 Phase 4. Forecasting and results analysis

During this phase, the scores for NT, TSm, TSsd and CSsd variables are utilized as input to simulate the BN models for 1,000 runs. Following this, the weighted average of the T and C values over 1,000 runs are determined, yielding a single point estimate for

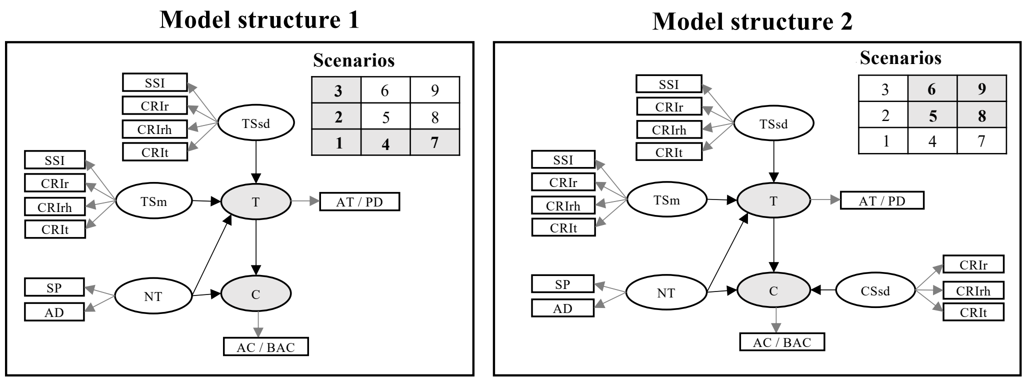


Fig. 3. Validated risk models

the predicted time and cost performance. In the remaining of this section, we present the results of the forecasting phase using five key concepts.

1. Data selection: 33 projects are selected from a database of 181 empirical projects sourced from Batselier and Vanhoucke (2015) considering the availability of observable indicators and the even distribution across various sectors.

2. Model identification: Using the k-nearest neighbour technique, the best-performing risk model is selected from nine candidates, which will be referred to as RM in the remainder of this paper. This selection is based on how closely the model matches the characteristics of a test project, assessed by comparing the estimated percentage differences in duration and cost.

3. Benchmark selection: The accuracy of the RM is compared with various benchmark methods. These include baseline estimates (BS), Monte Carlo simulations (MCS), machine learning algorithms (decision trees (DT) and random forest (RF)). In this study, we also explore a total of nine versions of Earned Value Management (EVM) and three versions of Earned Duration Management (EDM) for time forecasting. Additionally, we evaluate eight versions of EVM for cost forecasting. However, due to space constraints, we only discuss the best-performing versions in comparison to RM.

4. Time accuracy: Table 1 compares the accuracy of methods in terms of Mean Absolute Percentage Error (MAPE) on average and for different percentiles of the empirical projects. As shown, RM outperforms the BS and MCS, while slightly falls behind the machine learning algorithms DT and RF on average. RM performs worse than DT and RF for only 33% of projects that with higher but less frequent deviations from the baseline schedule (75th percentile). Table 1 also shows that the best performing EVM method (ES1) exhibits slightly better performance compared to the RM, *on average over all tracking periods*, while the best performing EDM method (EDM1) falls behind the RM. The average accuracy over all periods does not completely elucidate the overall accuracy since ES1 and EDM1 only begin to outperform RM in the later stages, around 50% completion, as illustrated in Figure 4.

5. Cost accuracy: Table 1 shows that RM performs equal or slightly better cost forecasting than BS, MCS, DT and RF for all percentiles. Similar to the time accuracy, the advantage of the best performing EVM version (EAC2) over RM typically starts only at around 30% completion, depicted in Figure 4. Therefore, it can be concluded that RM outperforms EVM in cost forecasting during the early stages of the project, as observed in time forecasting.

Table 1. Accuracy comparison in MAPE (%)

	Time accuracy							**Cost accuracy**					
MAPE	BS	MCS	DT	RF	RM	ES1	EDM1	BS	MCS	DT	RF	RM	EAC2
25th percentile	4.2	6.3	3.0	4.5	**2.4**	-	-	1.5	1.9	1.1	1.0	**0.9**	-
50th percentile	12.6	24.7	7.0	7.1	**5.4**	-	-	4.0	4.5	3.6	**3.1**	**3.1**	-
75th percentile	20.7	31.0	**10.3**	11.9	14.0	-	-	13.1	17.2	9.1	8.9	**8.7**	-
Average	14.6	22.1	**8.8**	9.7	10.1	9.8	11.2	8.7	10.3	5.9	5.6	5.4	**4.7**

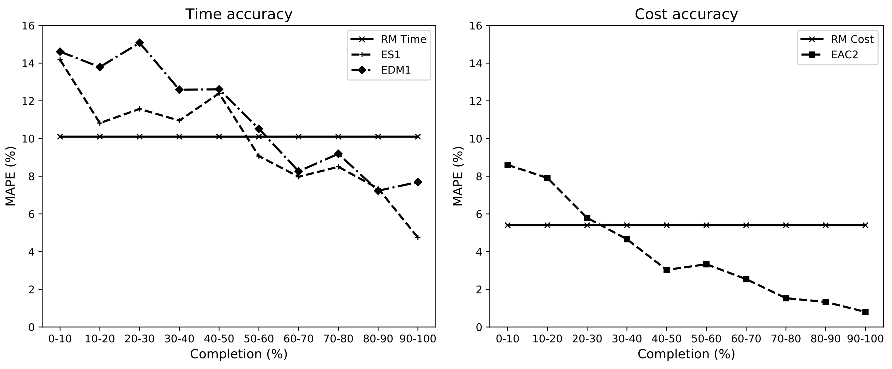


Fig. 4. Comparison of risk model accuracy to the EVM methods at different stages

This research has presented a hybrid forecasting model for predicting the total duration and cost of projects combining SEM and BN. 33 empirical projects were used for testing the model, and the accuracy of those were compared with various methods from the literature. The results have shown that an accurately selected risk model potentially outperforms not only the forecasts of the existing pre-project methods but also the EVM/EDM in the early stages of the project. This highlights the value of applying RM in the initial project phases, followed by the EVM/EDM, leveraging performance data in the latter stages. Future research aims at extending the analysis by the benchmark methods and the risk variables under consideration.

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., Coelho, J., Debels, D., Maenhout, B., & Tavares, L., 2008, An evaluation of the adequacy of project network generators with systematically sampled networks, *European Journal of Operational Research*, 187:511-524.

Vanhoucke M., 2012, *Project Management with Dynamic Scheduling*, Springer.

A comparison of activity ranking methods for taking corrective actions during project control

Forough Vaseghi[1], Mario Vanhoucke[1,2,3]

[1] Faculty of Economics and Business Administration, Ghent University, Gent 9000, Belgium
`Forough.Vaseghi@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: Activity ranking, Activity sensitivity, Corrective actions, Project control.

1 Introduction

Effective project management involves meticulous planning, execution, and control to ensure that project objectives are achieved within specified timeframes and budgets. The ongoing monitoring and control of projects play a pivotal role in facilitating corrective actions in case of deviations and, ultimately, delivering projects promptly. As projects unfold, project managers are confronted with the unavoidable realities of uncertainty and variability. These factors can lead to delays or cost overruns, requiring timely corrective actions to realign the project with its baseline schedule. Conversely, projects may present opportunities for early completion or cost savings, which necessitate proactive actions.

A multitude of project control methodologies have been proposed in the literature, often incorporating schedule risk analysis, which considers the sensitivity of activities and their expected impacts on the overall project duration. However, despite the extensive research in the realms of risk analysis and project control, a few key questions remain unaddressed. Our study concentrates on two fundamental aspects. Firstly, project managers operate within resource constraints and must make judicious decisions about which actions to take during a project's lifecycle. Although warning systems can flag the need for action, they often fall short in providing specific guidance on the selection of activities. In this study, we employ simulation and analytical sensitivity metrics to develop an "activity ranking" system that assists in the judicious selection of actions. We compare two classes of activity ranking to enhance the process of corrective actions under uncertainty. Secondly, the effectiveness of corrective actions not only depends on the chosen activities but also on the nature of the action itself. Limited research has explored how different types of actions interact with activity uncertainty. Our study delves into two classes of actions: one aimed at altering the average duration of selected activities and the other at reducing variability in activity durations. This research contributes to a deeper comprehension of the relationship between activity uncertainty and the type of action deployed.

Our findings, based on an array of artificial projects, demonstrate that specific simulation-based ranking and the analytical ranking method outperform other approaches. They not only excel in predicting the impact of actions on expected project duration and variability but also enhance the efficiency of project managers in controlling project outcomes.

2 Literature study

In the realm of schedule risk analysis, prior research has primarily centered on the development of various sensitivity metrics for project activities. These metrics, often generated through Monte Carlo simulations, offer a quantitative measure of activity criticality or sensitivity, typically expressed as a percentage. Higher percentages correspond to more sensitive activities, as seen in the works of Van Slyke (1963) and Williams (1992). A recent study by Vaseghi *et. al.* (2022) has introduced an innovative *analytical approach* as an alternative to Monte Carlo simulations. This approach assesses activity sensitivity by gauging the influence of changes in activity durations on the overall project duration. Both simulation-based and analytical methodologies are designed to enhance our comprehension of activities that exert a substantial impact on project outcomes. These insights serve as valuable guides for project managers, enabling them to make informed decisions regarding corrective actions during project execution. Taking actions on highly sensitive activities are expected to yield greater effects compared to those involving activities with low sensitivity values.

Furthermore, extensive exploration in the domain of project control has delved into the implementation of *corrective actions* during project execution. These studies encompass a variety of monitoring methods aimed at evaluating project performance at specific junctures, particularly in terms of detecting delays or cost overruns. Earned value management is a frequently adopted approach across diverse project scenarios, including various project network structures and levels of activity variability. The literature has considered three key types of corrective actions: activity crashing, variability reduction, and fast tracking. Activity crashing entails reducing activity durations by increasing effort to curtail the project timeline. This approach has been explored in multiple studies (Hegazy and Petzold (2003); Vanhoucke (2010); Vanhoucke (2011); Hu *et. al.* (2016); Song *et. al.* (2020)). Variability reduction focuses on diminishing project variability and activity uncertainty (Madadi and Iranmanesh (2012); Martens and Vanhoucke (2019)). Lastly, fast tracking seeks to expedite project completion by concurrently executing partially precedence-related activities, circumventing the typical project network structure. While previous studies have predominantly merged the use of simulation-based sensitivity metrics into the corrective action process, this study seeks to incorporate both simulation-based sensitivity metrics and analytical ranking procedures. A comparative analysis of their performance within the framework of *activity-based bottom-up project control* is a central objective of this research.

3 Computational experiments

This paper conducts a comparative analysis of two distinct *activity ranking approaches* aimed at enhancing the corrective action process within uncertain project environments. In each of these methods, activities are ranked based on specific criteria, and the highest-ranked activities are grouped into what is termed an *action set*. This action set guides the selection of activities for particular corrective actions, as it consists of the most sensitive activities where corrective actions are likely to have a significant impact. The first class of ranking measures are the *analytical-based ranking measures*, which relies on exact or approximate analytical computations to establish activity rankings. This ranking measure comprises two analytical metrics, AR_m and AR_s, introduced by Vaseghi*et. al.* (2022), designed to assess the anticipated impact of activities on the distribution of project duration following corrective action. AR_m ranks activities based on their impact on the mean project duration, while AR_s ranks activities based on their impact on the standard deviation of project duration when subjected to a particular type of action. This analytical method is comparad with a *simulation-based ranking method*, which employs Monte Carlo simulations to assess activity sensitivity. The simulation-based ranking measures represent the main sensitivity metrics derived from schedule risk analysis (SRA) methodologies, as proposed by various researchers in the literature. These metrics include CI, SI, SSI, CRI, MOI, and CSS.

For computational experiments, the simulation-based and analytical ranking approaches are evaluated within *preventive* and *protective* control strategies. The preventive strategy takes all actions before project initiation, representing a pure control approach, while the protective control strategy entails periodic monitoring and the execution of corrective actions on activities in the action set during project progress. Additionally, we explore the influence of network topology and action set size on the cumulative impact of corrective actions. The performance measures used in the computational experiments, taken from previous research study by Vaseghi*et. al.* (2022), measure the relative impact of the actions on the mean and standard deviation of the project duration distribution, represented by TC_m and TC_s, respectively.

Preliminary results. To demonstrate the relevance of activity ranking, three distinct methods of activity selection were employed to form the action set up to six activities. First, the *good ranking* includes the highest-ranked activities, assuming they are the most sensitive. The *random ranking* randomly selects activities, disregarding the activity ranking, while the *bad ranking* selects the lowest-ranked activities. Figure 1 presents a comparison of these activity selection methods, employing the analytical ranking measure AR_m and the simulation-based sensitivity metric SSI for implementing Action 1, which is a specific type of action to reduce the activity and project durations. The results indicate that the good ranking method outperforms the other two approaches, underscoring the importance of ranking activities before project initiation. Furthermore, as the number of actions in the action set increases, both the total contribution and the significance of a robust ranking system grow. The findings also suggest that the simulation-based SSI metric can be competitive with analytical indicators.

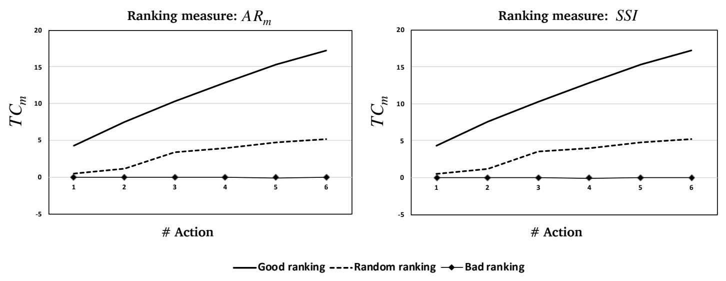


Fig. 1. Comparison of good, bad and random activity ranking

Figure 2 illustrates the performance of analytical and simulation-based ranking measures, classified in three groups for taking Action 1 in protective control strategy. The results demonstrate a notable dependency on the serial/parallel indicator (SP), with performance decreasing as SP values increase. This trend aligns with existing literature, which consistently indicates that bottom-up project control, particularly when employing sensitivity metrics, is notably effective for parallel projects but significantly less so for serial network structures. The first group consists of the best performing methods, which consists of the analytical ranking methods and the SSI. The second group consists of the MOI and the three versions of the CRI metric, which shows a relatively stable deviation from Group 1, which means that they perform never as good as the best performing metrics, but the difference from them is independent of the network structure. The last group consists of three sensitivity-based metrics, SI, CI and CSS, which perform relatively well for parallel networks but their good performance drops quickly when projects are more serial in their structure. At the very serial network side, these sensitivity metrics have a very weak performance compared to the other ranking methods, and should therefore not be used at all for activity ranking.

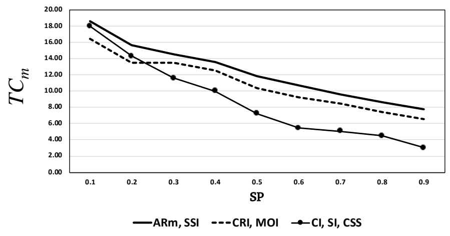


Fig. 2. Performance of all ranking methods split in 3 groups vs. SP

Acknowledgments

This work was supported by the Bijzonder Onderzoeksfonds (BOF) under Grant BOF24Y207001101, and by the Wetenschappelijk Onderzoek (FWO) under Grant FWOOPR2017001301.

References

Hegazy T., K. Petzold, 2003, “Genetic optimization for dynamic project control", *Journal of Construction Engineering and Management*, Vol. 129, pp. 396-404.

Hu X., N. Cui., E. Demeulemeester and Bie., L , 2016, “Incorporation of activity sensitivity measures into buffer management to manage project schedule risk", *European Journal of Operational Research*, Vol. 249, pp. 717-727.

Madadi M., H. Iranmanesh, 2012, “A management oriented approach to reduce a project duration and its risk (variability)", *European Journal of Operational Research*, Vol. 219, pp. 751-761.

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.

Song J., A. Martens and M. Vanhoucke, 2020, “The impact of a limited budget on the corrective action taking process", *European Journal of Operational Research*, Vol. 286, pp. 1070-1086.

Vanhoucke M., 2010b, “Using activity sensitivity and network topology information to monitor project time performance", *Omega The International Journal of Management Science*, Vol. 38, pp. 359-370.

Vanhoucke M., 2011b, “On the dynamic use of project performance and schedule risk information during project tracking", *Omega The International Journal of Management Science*, Vol. 39, pp. 416-426.

Van Slyke R.M., 1963, “Monte Carlo methods and the PERT problem", *Operations Research*, Vol. 11, pp. 839-860.

Vaseghi F., A. Martens and M. Vanhoucke, 2022, “Analysis of the impact of corrective actions for stochastic project networks", *Manuscript submitted to the journal,*.

Williams T.M., 1992, “Criticality in stochastic networks", *Journal of the Operational Research Society*, Vol. 43, pp. 353-357.

Priority heuristic for the multi-skilled resource-constrained project scheduling problem

Vermeire Guillaume[1], Vanhoucke Mario[2]

[1] University of Ghent, Belgium
guillaume.vermeire@ugent.be
[2] University of Ghent, Belgium
mario.vanhoucke@ugent.be

Keywords: Project-scheduling, Multi-skilled, Priority rule heuristic.

1 Introduction

The resource-constrained project scheduling problem (RCPSP) stands out as one of the most investigated problems in the context of project scheduling. The core of this problem revolves around finding start times of project activities that are subject to precedence and resource constraints while minimizing the project makespan. Many extensions to the RCPSP have been proposed to make this problem more applicable to real-life situations. One of these extensions describes resources as human workers that are able to master one or multiple skills. In contrast to the RCPSP, activities in the multi-skilled resource-constrained project scheduling problem (MSRCPSP) have a requirement for skills instead of a direct requirement for resources. Therefore, it is necessary to not only assign resources to activities but also to decide on which specific requirement these resources will be deployed. In literature, several solution methodologies are described to solve project instances for the MSRCPSP. These methodologies range from simple and fast single-pass heuristics to more advanced but slower multi-pass meta-heuristics. Given the NP-hard nature of the problem (Correia *et al.* 2012), the possibility of obtaining optimal solutions through exact methods is inherently constrained by instance size, and therefore the development of efficient and fast heuristics that produce high-quality solutions is of relevance.

In this research, a new heuristic procedure is developed which is based on a parallel schedule generation scheme that uses various types of priority rules (PR). Well performing priority rule combinations are selected based on their solution quality and employed to generate solutions for a benchmark dataset. Additionally, preliminary results are provided on existing and newly developed priority rules and their ability in generating solutions of high quality for the MSRCPSP. The procedure is able to generate new best-known solutions (BK) for the considered dataset.

2 Problem description

In the MSRCPSP, a project network can be represented by a topologically ordered acyclic activity-on-the-node network $G = (N, A)$. In this network, N $= \{0, ..., n+1\}$ is the set of nodes that represent activities in the project. In total, $|N|$ activities are considered, including a dummy start- and end-node, and n project activities. Each project activity i has a standard processing time p_i, which is predefined and can not be pre-empted. A is the set of arcs that represent a finish-start precedence relation with a time-lag of zero, between the activities. The multi-skilled workforce is specified by the resource-set $R = \{1, ..., |R|\}$. All resources master one or multiple skills which are defined by the skill-set $S = \{1, ..., |S|\}$. To allow resources to master skills at a specific level, the skill level set $L = \{1, ..., |L|\}$ and the discrete skill level distribution bl_{jk} are included in the problem definition. bl_{jk} equals

$l \in L$ if resource k masters skill j at level l and zero otherwise. In the MSRCPSP, a resource k can be assigned to an activity when it masters the required skill.

The function of skill levels in the problem description is dependent on which MSRCPSP version is used. In this research, four different MSRCPSP versions are considered, one of which is discussed here. For a detailed overview of the different versions, the reader is referred to Snauwaert and Vanhoucke (2022).

3 Solution methodology

To solve project instances for the described problem, a procedure is required that is able to give each activity a start-time while assigning the required resources to these activities. To do this, a parallel schedule generation scheme (PSGS) is developed which is able to give activities start times and includes an assignment procedure for the resources. This PSGS is based on the heuristic provided by Almeida *et al.* (2016) although a different assignment procedure is applied. A single run of the PSGS takes three types of priority rules (PR) and the instance data as input and generates the makespan for the selected rule combination. The three types of PR are classified as follows. An **activity PR** determines which eligible activity receives the highest priority to be scheduled with the available resources. The choice of an *activity PR* has an impact on the project makespan. When the selected activity has multiple requirements, the **skill PR** determines the order in which the skill level requirements are considered for resources to be assigned to. When a specific skill requirement is selected, the **resource PR** determines the order in which unassigned resources are considered for assignment to that requirement. Both the *skill PR* and *resource PR* have an impact on which resources are assigned to which requirement of a given activity. By using the three types of PR's the PSGS becomes deterministic. This means that it will always generate the same makespan and resource assignment for the selected PR combination and project-instance. However it is possible that the used PR's result in an infeasible solution for a given project instance.

For this research, 12 activity PR's, 12 resource PR's and 4 skill PR's are considered, represented in Table 1. The newly developed PR's are indicated in bold text. These rules allow for a total of 576 rule combinations to solve a single project instance.

Table 1. Priority rules used in heuristic

	Rule	Explanation		Rule	Explanation		Rule	Explanation
	SPT	Shortest Processing Time		LB	Lowest Breadth		**HSS**	Highest Skill Strength
	LPT	Longest Processing Time		HB	Highest Breadth		**LSS**	Lowest Skill Strength
	MIS	Most immediate successors		LGB	Lowest Grouped Breadth		**HSC**	Highest Skill Criticality
	EST	Earliest Start Time		HGB	Highest Grouped Breadth		**LSC**	Lowest Skill Criticality
	EFT	Earliest Finish Time		LAD	Lowest Average Depth			
Activity	LST	Latest Start Time	Resource	HAD	Highest Average Depth	Skill		
	LFT	Latest Finish Time		LTND	Lowest Total Negative Depth			
	MSLK	Minimum Slack		HTND	Highest Total Negative Depht			
	LSF	Lowest Skill Factor		LBRC	Lowest Basic Resource Criticality			
	HSF	Highest Skill Factor		HBRC	Highest Basic Resource Criticality			
	LRSC	Lowest Ranked Skill Criticality		LARC	Lowest Advanced Resource Criticality			
	HRSC	Highest Ranked Skill Criticality		HARC	Highest Advanced Resource Criticality			

4 Experiments

To test the solution quality of the procedure, the MSLIB4 dataset is considered. This is one of the subsets of the MSLIB dataset, created by Snauwaert and Vanhoucke (2023).

It is composed of 5,000 project instances which includes projects of 30 activities, 4 skills and a variable amount of resources. The procedure is run on these instances using all 576 rule combinations. To gain insight in the solution-space quality of the PR's, the average of the best makespans for each project instance can be compared with the average makespan of the best-known solutions (BK). These BK's are obtained by the genetic algorithm (GA) developed by Snauwaert and Vanhoucke (2021). The average makespan of the BK equals 83.43, while the average of the best makespans of all 576 rule combinations equals 84.46.

The instances are categorized in four classes,"Better", "Equal", "Worse", or "Infeasible". These imply that the heuristic obtains better, equal, worse or infeasible results compared to the BK. The number of instances for each of these classes is represented in Table 2 with their average $\%GAP$. Note that the priority heuristic is able to generate 16.20% solutions which are better than the solutions generated by the GA. However, it is still required for the heuristic to generate 576 solutions per project instance. In a next experiment we select rules in a smart way to reduce this amount significantly.

Table 2. Solution space quality of the considered PR's for the MSRCPSP.

Quality	#instances	$\%GAP$
Better	16.20%	−3,69%
Equal	36.10%	0.00%
Worse	47.66%	4.80%
Infeasible	0.00%	-

A subset of MSLIB4 is created, which includes all the instances for which all rule combinations lead to a feasible solution. This minimum feasibility set (MF) is used to see how the cumulative performance changes when the solutions of more rule combinations are added. The rule combinations are ranked on basis of their average performance for the MF set. The cumulative performance of these ranked rule combinations is represented in Figure 1 (dotted line) for the 100 best performing rule combinations. However, if each rule combination is iterative checked on their improvement when they are added to the set of considered rule combinations, rule combinations can be selected in a smart manner. All the rule combinations are cross compared and the rule combination which increases the performance of the set the most, is added to the set. Notice that, by selecting 10 rule combinations in this manner, the solution quality (full line) is equal to selecting the first 48 ranked rule combinations. This allows us to reduce the amount of schedules generated per project instance significantly.

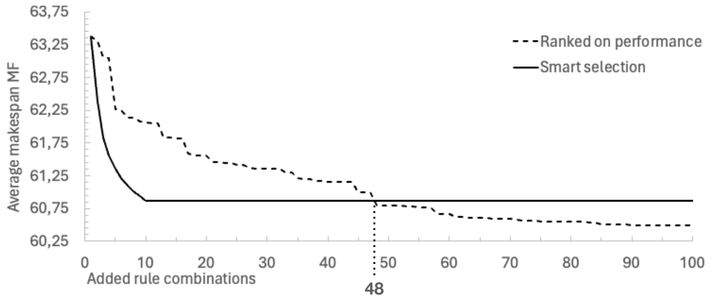


Fig. 1. Evolution of the cumulative performance in relation to the number of rule combinations considered for the MF subset.

In a next step, the previous 10 selected rule combinationS are tested on their solution quality. The cumulative performance of the selected rule combinations, compared to the BK is represented in Table 3. The heuristic obtains 10.96% better solutions compared to the BK's. Additionally, it is able to generate equal solutions in 30.12% of the instances. These results confirm that using a simple heuristic procedure with a reasonable amount of priority rule combinations to solve MSRCPSP instances provides, on average, promising results. Especially because the heuristic can find these equal (and better) solutions much faster. The presented priority rule heuristic only generates 10 schedules for $5,000$ instances which requires an average of only 0.0064 seconds computation time per schedule.

Table 3. Comparison of the cumulative performance of the selected rules and BK

Quality	#instances	$\%GAP$
Better	10.96%	-3.56%
Equal	30.12%	0.00%
Worse	58.92%	6.06%
Infeasible	0.00%	-

5 Conclusion

The MSRCPSP is an NP-hard problem that requires activity-scheduling and the assignment of resources to these activities. In literature, advanced meta-heuristic procedures are often employed to obtain near optimal solutions. However, this paper shows that it is possible to obtain reasonably good solutions with the use of simple heuristic priority rules. A parallel schedule generation scheme is developed which combines three types of priority rules to solve project-instances for the MSRCPSP. New PR's are developed that consider skill-specific priority values. Preliminary results for the solution quality of existing and new PR's is provided. A heuristic procedure is developed which selects well performing PR combinations in order to minimise the project makespan. The heuristic obtains new best-known solutions for the MSLIB4-dataset.

References

Almeida B. F., Correia I., F. Saldanha-da-Gama, 2016, "Priority-based heuristics for the multi-skill resource constrained project scheduling problem", *Expert Systems With Applications*, Vol. 57, pp. 91-103.

Correia I., L.L. Lourenco, F. Saldanha-da-Gama, 2012, "Project scheduling with flexible resources: formulation and inequalities", *OR Spectrum*, Vol. 34, pp. 635-663.

Snauwaert J., M. Vanhoucke, 2021, "A new algorithm for resource-constrained project scheduling with breadth and depth of skills", *European Journal of Operational Research*, Vol. 292, pp. 43-59.

Snauwaert J., M. Vanhoucke, 2022, "Mathematical formulations for project scheduling problems with categorical and hierarchical skills", *Computers & Industrial Engineering*, Vol. 169, pp. 108147

Snauwaert J., M. Vanhoucke, 2021, "A solution framework for multi-skilled project scheduling problems with hierarchical skills", *Working paper (under submission)*

Snauwaert J., M. Vanhoucke, 2023, "A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 307, pp. 1-19.

Vanhoucke M., J. Coelho, D. Debels, B. Maenhout, L. V. 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.

Improvement method to solve large scale instances of the resource overload problem with tardiness penalty

Lena Sophie Wohlert[1] and Jürgen Zimmermann[1]

Clausthal University of Technology, Germany
lena.sophie.wohlert@tu-clausthal.de, juergen.zimmermann@tu-clausthal.de

Keywords: Project Scheduling, Resource Leveling, Tardiness Penalty, Heuristics.

1 Introduction

We address the resource overload problem with general temporal constraints, where a tardiness penalty is added to the objective function when a predefined project makespan is exceeded. This penalty allows for a trade-off between a balanced resource utilization and a minimization of the project makespan. For small and medium-sized instances, genetic algorithms obtain good solutions in reasonable time e.g. described in Schnabel *et. al.* (2018) and Wohlert and Zimmermann (2023). For large instances with up to 500 activities, we propose a population-based greedy method. First, a diverse population of solutions is generated with a serial generation scheme. Afterwards, the solutions are subjected to a greedy improvement procedure which aims to create a more balanced resource utilization for selected project makespans.

2 Problem description and structural properties

A given project is composed of a set of activities V consisting of $1, \ldots, n$ real activities and fictitious activities 0 and $n+1$ which represent the project start and the project completion. Each activity $i \in V$ is assigned a processing time $p_i \in \mathbb{N}$, which must be carried out without interruption. The start time of each activity $i \in V$ is specified as $S_i \geq 0$, and the sequence of all activity start times, ordered by their index, is given in a schedule $S = (S_0, S_1, \ldots, S_{n+1})$. The start times are restricted by general temporal constraints of the form $S_j - S_i \geq \delta_{ij}$, where $\delta_{ij} \geq 0$ indicates a minimum time lag and $\delta_{ij} < 0$ a maximum time lag. The maximum project duration $\overline{d} \in \mathbb{N}$ is given as a maximum time lag between the project completion S_{n+1} and the project start S_0. Using a longest path algorithm, the longest distance d_{ij} from activity i to activity j can be determined. With that, the earliest start time $ES_i = d_{0i}$ and latest start time $LS_i = -d_{i0}$ for each activity $i \in V$ can be obtained. Besides the maximum project duration $\bar{d}$, we assume a given prescribed project makespan $\overline{T} \in \{ES_{n+1}, \ldots, \overline{d}\}$, which can be exceeded, but then a constant delay cost factor $\rho > 0$ is added to the objective for each time unit of delay. During their execution $[S_i, S_i + p_i)$, the activities have a demand for a set of renewable resources $\mathcal{R}$. The resource utilization of each activity i and resource k is assumed to be constant and is given by $r_{ik} \geq 0$. The cumulative resource utilization of each activity in execution at a given point in time t and schedule S for a resource k is given by $r_k(S,t)$. This resource utilization $r_k(S,t)$ is not constrained by a resource capacity, but positive deviations from a resource threshold Y_k are penalized with a cost factor c_k.

The objective of the given problem is to obtain a schedule, which satisfies all temporal constraints, that minimizes the combined cost of resource overload and a project delay. The search space of this problem can be reduced by dividing the problem into two subproblems for which there is always a quasistable schedule among the optima. The first subproblem has a maximum project duration of $\overline{T}$, whereas the second subproblem has a minimum project duration of $\overline{T}$ (Wohlert and Zimmermann 2023). A quasistable schedule S has useful structural properties as for the start time of each activity $i \in V$, there is always an activity $j \in V$ with either a binding precedence relationship $S_j = S_i + p_i$ or $S_j = S_i - p_j$ or a binding temporal constraint $S_j = S_i + \delta_{ij}$ or $S_j = S_i - \delta_{ji}$. Assuming that the project starts at $S_0 = 0$ and all $\delta_{ij} \in \mathbb{Z}$, a quasistable schedule contains only integer start times. The problem with discretized start times can be described as follows.

$$\begin{array}{lll}
\text{Minimize} & f(S) = \sum\limits_{k \in \mathcal{R}} c_k \sum\limits_{t \in \{0,\ldots,\overline{d}-1\}} (r_k(S,t) - Y_k)^+ + \rho \cdot (S_{n+1} - \overline{T})^+ & \\
\text{subject to} & \sum\limits_{i \in V : S_i \leq t < S_i + p_i} r_{ik} \leq r_k(S,t) & (k \in \mathcal{R}, t \in \{0,\ldots,\overline{d}-1\}) \\
& S_j - S_i \geq \delta_{ij} & (\langle i,j \rangle \in E) \\
& S_0 = 0 & \\
& S_i \geq 0 & (i \in V)
\end{array}$$

3 New solution approach

The solution approach starts with the generation of a diverse initial population. Then, an improvement method is applied to the solutions in the population which is based on unscheduling and rescheduling subsets of activities.

The initial set of schedules is generated based on a decoding scheme for a genetic algorithm described by Wohlert and Zimmermann (2023) which exploits the already presented search space reduction. It is shortly described in the following. At initialization, the project start is scheduled at $S_0 = 0$ and added to set $C = \{0\}$. All of the other activities remain in the set of to be scheduled activities $\overline{C}$. In each scheduling step, it is assessed for each $h \in \overline{C}$ whether a binding temporal or precedence constraint with any $j \in C$ results in a feasible start time S_h. For the start time of the project completion, $S_{n+1} = \overline{T}$ has to be considered additionally. If at least one feasible S_h can be found, h is added to a set of eligible activities. Of those, one activity i is chosen and a complete set of eligible start times is determined which consists of all start times that lead to at least one binding temporal or precedence constraint with an activity in C. In a greedy variant, the start time with the smallest increase in the resource overload objective is selected as S_i. In case of tied start times, the earlier start time is chosen. If $\overline{C} = \emptyset$, the scheme terminates and a feasible schedule S results. To increase the diversity of the population, we have adapted the generation scheme by also considering the later start time to break a tie. When generating a solution, the same tie-breaker is used for scheduling all activities and is saved alongside the resulting schedule.

The generated schedules are subjected to the improvement method in ascending order of their objective value. In the following, the originally generated schedule is called baseline schedule S^B. In the first improvement step, the resource overload of S^B is tried to be reduced with a fixed project makespan $m = S^B_{n+1}$. For that, each real activity i is unscheduled as a primary activity in order of non-increasing resource demands $p_i \sum_{k \in \mathcal{R}} r_{ik}$. Together with i further activities are unscheduled with regard to a selected strategy. Rescheduling is done as described previously with a greedy start time selection in regards to the resource overload objective. When every real activity has been considered as the primary activity and no schedule with a better objective value can be found, the improvement method terminates for the given project makespan. Afterwards, the current project makespan m is adapted by $\Delta_m = 1$ time units. If the project makespan of the baseline schedule is closer or equally as close to the project deadline than to the prescribed makespan ($S^B_{n+1} \geq (\overline{d} + \overline{T})/2$), the project makespan is reduced by Δ_m. To obtain a feasible schedule

S^m with the given project makespan, S^B is copied and the start times for real activities where $S_i^B + d_{i,n+1} > (S_{n+1}^B - \Delta_m)$ are adjusted by the difference between the left and right sides of the inequality. In case of $S_{n+1}^B < (\overline{d} + \overline{T})/2$, the project makespan m is increased by Δ_m. If a better schedule is found with the adjusted project makespan, Δ_m is doubled. If that is not the case, after at least one more iteration, the direction is changed to re-approach the last makespan where an improvement could be found, Δ_m is halved.

Algorithm 1 Improvement method

```
Require: Schedule $S^B$, $U_1$-strategy and tie-breaker
  Sort activities $1, \dots, n$ according to non-increasing resource demands resulting in order $O$
  $S^* \leftarrow S^B$
  while Makespan-adaption is feasible do
      Adapt $m$ and adjust $S^B$ to feasible $S^{*,m}$
      while Improvement is found do
          for $i \in O$ do
              $S \leftarrow S^{*,m}, U_1 = \{i\}, U_2 = \{\}$
              Extend $U_1$ and $U_2$ according to given strategy and schedule $S$
              Sort $U_1, U_2$ according to non-increasing resource demands
              Unschedule activities $\in U_1 \cap U_2$ from $S$
              Schedule activities in $U_1$ into $S$ regarding the given tie breaker rule
              Schedule activities in $U_2$ into $S$ regarding the given tie breaker rule
              if $f(S) < f(S^{*,m})$ then
                  $S^{*,m} \leftarrow S$
      if $f(S^{*,m}) < f(S^*)$ then
          $S^* \leftarrow f(S^{*,m})$
```

The activities which are unscheduled never include the project start or the project completion and are classified into two sets U_1 and U_2, where U_1 always contains i. Activity set U_1 can be enlarged utilizing two strategies to select further activities related to i. The first strategy (`vertical`) focuses on activities that are executed in parallel with activity i. A similar strategy is utilized in Ballestin *et. al.* (2007). In Harris (1990), a heuristic for a resource leveling problem with precedence constraints is presented, where an activity i is scheduled considering the best start times of its predecessor and successors. Since we assume general temporal constraints, we exploit this idea by a second strategy (`horizontal`) that includes all activities that have a binding precedence constraint with activity i in the current schedule. This enables activities with binding precedence constraints to be shifted together or their order to be changed.

To remove all possible start time restrictions for S_i, activity set U_2 includes all real activities (not already part of U_1) which have a binding temporal constraint with activity i and recursively all other real activities which have a binding temporal constraints with the activities already included in U_2. The activities in both sets are sorted by non-increasing resource demands. To make use of the lack of start time restrictions for S_i, the activities in U_2 are scheduled after the activities in U_1. In each rescheduling step, the next activity j is taken from the respective set and a set of eligible start times is determined. Firstly, this set includes all start times which result in a binding precedence constraint with an already scheduled activity. As it cannot be guaranteed that this set always contains at least one start time, the updated ES_j and LS_j are added to the set. This is done instead of assessing binding temporal constraints, as the predecessors and successors might be in set U_2. Out of the set of eligible start times, the start time S_j is chosen which results in the lowest increase in resource overload. In case of a tie, the tie-breaker used to create the baseline schedule is utilized. When the scheduling is complete and a better solution has been found, the schedule is saved and used for the following improvement attempts.

4 Experiments

In order to evaluate the performance of our new solution approach, we conduct experiments with the presented method and the genetic algorithm described in Wohlert and Zimmermann (2023), which are both implemented in C++. The problem instances are based on the benachmark test sets UBO for the RCPSP/max (Schwindt 1998), where the maximum project duration is limited with $\overline{d} := \lceil \alpha \cdot ES_{n+1} \rceil$ and the prescribed project makespan is chosen as $\overline{T} := \lfloor \beta \cdot ES_{n+1} \rfloor$. The resource thresholds Y_k are set according to the rounded up ratio of the respective total resource utilization and the prescribed project makespan. The cost factors c_k are assumed to be 1 and the delay cost factor ρ is set by $\rho := \gamma \sum_{k \in \mathcal{R}} Y_k$. The mutation probability for each gene within the genetic algorithm is set to 2.5%, the scaling factor to 16 and the population size to 200. For the new solution approach, a population of the same size is created, where in contrast half of the solutions are generated with either tie-breaker. Three different variants to obtain set U_1 are tested. Firstly, U_1 only consists of the primary activity (strategy `i`). The other strategies are `horizontal` and `vertical`. Table 1 shows the preliminary results for all U_1 strategies with a runtime of 60s, displaying the average gap ($\varnothing gap_{ga}$) to the best solution obtained by the genetic algorithm and the number of solutions ($\#n_{best}$), which are the best alongside all found for a specific instance. In total, there are 90 instances within each test set.

Table 1. Preliminary results

Instances				U_1 strategy					
				`i`		`horizontal`		`vertical`	
α	β	γ	n	$\varnothing gap_{ga}[\%]$	$\#n_{best}$	$\varnothing gap_{ga}[\%]$	$\#n_{best}$	$\varnothing gap_{ga}[\%]$	$\#n_{best}$
1.25	1.0	0.3	100	4.86	5	2.52	20	6.37	1
			200	–7.19	9	–9.61	79	–5.52	0
			500	–22.42	4	–24.79	86	–21.02	0

The results indicate that the best strategy to form U_1 is `horizontal` and that our new approach is able to obtain considerably better solutions for $n \in \{200, 500\}$ than the genetic algorithm. The results could be further improved by parallelization, since the improvement of a solution is independent of the other solutions. For instances with 100 activities, the improvement method might be more interesting when applied to a later population in the genetic algorithm.

References

Ballestin, F., C. Schwindt and J. Zimmermann, 2007, “Resource leveling in make-to-order production: modeling and heuristic solution method", *International Journal of Operations Research*, Vol. 4, No. 1, pp. 50-62.

Harris, R. B., 1990, “Packing Method for Resource Leveling (Pack)", *Journal of Construction Engineering and Management*, Vol. 116, No. 2, pp. 331-350.

Schnabel, A., C. Kellenbrink, and S. Helber., 2007, “Profit-oriented scheduling of resource-constrained projects with flexible capacity constraints", *Business Research*, Vol. 11, No. 2, pp. 329-356.

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

Wohlert, L. S., Zimmermann, J., 2023, “Resource overload problems with tardiness penalty: structural properties and solution approaches", *Annals of Operations Research*, (forthcoming).