Modelling and solving approaches for scheduling problems in reconfigurable manufacturing systems

Xavier Delorme, Gérard Fleury, Philippe Lacomme, Damien Lamy

Modelling and solving approaches for scheduling problems in Reconfigurable Manufacturing Systems[1]

Xavier Delorme [a], Gérard Fleury [b], Philippe Lacomme [b], Damien Lamy [a*]

[a] Mines Saint-Etienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F-42023 Saint-Etienne, France.

delorme@emse.fr, damien.lamy@emse.fr

[b] Université Clermont Auvergne, Clermont Auvergne INP, Mines Saint-Etienne, CNRS, LIMOS, F-63000 Clermont–Ferrand, France.

gerard.fleury@isima.fr, philippe.lacomme@isima.fr

Reconfigurable manufacturing systems (RMS) intend to bridge the gap between dedicated and flexible manufacturing systems. If the literature is mainly focused on the design step and tactical planning of such systems, few research projects have addressed scheduling at the operational level. While setup times may occur in flexible manufacturing systems, reconfiguration times considered in RMS may affect several resources at once, and hence require specific modelling and solving approaches to be considered. This paper first formalises the problem at hand through linear programming. An iterative search method is then provided to obtain solutions to larger scale instances. Results obtained on generated instances show that managing even few possible configurations can yield significant improvements on solutions' quality. Meanwhile, the extended search space implied by the increase in available configurations hinders the convergence to a good solution in a reasonable computation time, which suggests further investigations.

Keywords*:* Reconfigurable manufacturing systems; linear programming; metaheuristics; scheduling

1 Introduction

Production systems are submitted to several external constraints including but not limited to unpredictable events, and high-frequency market changes due to mass customisation (Yoram Koren, Gu, and Guo 2018). To address such changes, the production systems must be adaptive and able to evolve in order to integrate (i) modifications in parts of existing products, (ii) fluctuations in demand, (iii) evolution in legal regulations and (iv) changes in process technology. Initially introduced in Y Koren et al. (1999) Reconfigurable Manufacturing Systems (RMS) have received a lot of attention during past years as an appropriate solution to aforementioned issues. RMS are built around six key features, namely: modularity, scalability, diagnosability, integrability, convertibility and customisation (Y. Koren 2006) that allow to adjust their production functionalities and production capacities in accordance with market demand (Bortolini, Galizia, and Mora 2018). Moreover, these systems intend to bridge the gap between Dedicated Lines (DL), and Flexible Manufacturing Systems (FMS) (Y. Koren 2006). In DLs, high production rates can be observed due to several tools processing operations simultaneously implying lower costs per part. In FMS, variety of products can be achieved thanks to multi-axis CNC machines, which are designed to use many different tools but only one at a time, implying sequential processing as well as setup times and thus lower production

[1] This is an Accepted Manuscript version of the following article, accepted for publication in International Journal of Production Research: "Xavier Delorme, Gérard Fleury, Philippe Lacomme & Damien Lamy (2023) Modelling and solving approaches for scheduling problems in reconfigurable manufacturing systems, International Journal of Production Research, DOI: 10.1080/00207543.2023.2224446". It is deposited under the terms of the Creative Commons Attribution-Non Commercial License ©2023 CC-BY-NC 4.0 (http://creativecommons.org/licenses/by-nc/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited.

rates and expensive cost for parts. RMSs aim at reaching high throughput while preserving flexibility, which is made possible by the concept of Reconfigurability.

Reconfigurability is the capacity of a set of resources to be reconfigured in a period of time and both reconfigurable machine tools (RMT) and computer numerically controlled (CNC) machines are the core components of any reconfigurable manufacturing system. Machine switching delay from one configuration to another can include but is not limited to cleaning the working zone, loading, positioning and unloading the parts and can imply extra costs coming from energy expenditures, equipment maintenance and labour (Borgia, Matta, and Tolio 2013). Several designs of RMSs can be found in the literature, such as Reconfigurable Flow Lines (Yang et al. 2023), Reconfigurable Cellular Manufacturing Systems (RCMS) (Bortolini, Galizia, and Mora 2018), or Rotary Machining Systems (Battaïa, Dolgui, and Guschinsky 2017). These last production systems include routing of parts, which could be seen as Reconfigurable Job-shops (Yelles-Chaouche et al. 2021). If some features outlined above refer to CNC machines and RMT, reconfigurations are not only about physical resources but can also be related to assignment of operators to specific operations depending on their skills and expertise. Indeed, production systems still require human resources in order to operate and to match the competences and/or functionalities required by the operations to be performed (Ferjani et al. 2017; Grosse et al. 2015). Considering reconfigurable workforce can make the production system also considered as reconfigurable manufacturing system (Hashemi-Petroodi et al. 2020). Actually, a specificity of configurations lies in the number of resources that can be affected during a reconfiguration, such as hardware (machines, tools), software or operators.

If different research streams on RMS have emerged during past decades, it can also be explained by the spread of industry 4.0 concepts, which act as enablers for features required by RMS such as modularity. However, applied research is mainly directed towards product family definition (Galan et al. 2007), design and line balancing (Essafi, Delorme, and Dolgui 2012), layout problems (Yamada, Ookoudo, and Komura 2003) and configuration selections (Youssef and ElMaraghy 2007). In these problems, objectives mainly concern the minimisation of a number of machines or the cost minimisation of the designed production system including costs related to reconfigurations. As stressed in the recent literature survey provided by Bortolini, Galizia, and Mora (2018), production planning and scheduling in RMS environments still are difficult tasks. If some research projects have addressed process planning at the tactical level (Bensmaine, Dahane, and Benyoucef 2014; Touzout and Benyoucef 2019), very few consider scheduling problems at the operational level whereas it is known to be a necessity since the early 90s (Liles and Huff 1990). However, as integrated problems are among the hardest problems, decisions taken at the operational level, such as schedules, are complementary to decisions taken at the strategic level (such as line balancing, line design or configuration selections at the conception phase), or tactical level such as process planning.

Hence, given a set of possible configurations, selected in prior stages under which the production system can operate, scheduling operations in RMS requires to define both the schedule of applied configurations and of operations that are processed according to these configurations. It is expected that processing time of operations will depend on the chosen configurations, and that reconfiguration delays may occur between two configurations. Once configurations are selected, routing of product orders will highly depend on given processes and on the shop floor's structure. This structure can be very different from one RMS to another (Yoram Koren, Gu, and Guo 2018). The objective of the problem consists in minimising the completion time of the last operation (referred to as makespan). However, it should be mentioned that reconfigurations are not just setups and may involve several resources at once and they can have a larger impact on the production system. These observations, i.e. having variable processing times, and reconfigurations that can affect several resources, may lead to unpace production systems (Hillier 2013). Furthermore, due to the presence of operators

mentioned before, such systems are better suited as human variables should be taken into consideration (Ostermeier 2020).
As a step towards further integration of scheduling issues in RMS, this work focuses on the mathematical formalisation of the problem and introduces results obtained using a linear solver and a metaheuristic-based approach. This paper addresses the following contributions:

- We introduce a new Job-Shop Scheduling Problem with multiple configurations, named Reconfigurable Job-Shop Scheduling Problem (RJSSP), which addresses the practical case of unpace Reconfigurable Cellular Manufacturing Systems.
- We generalise the concept of setup times to consider reconfigurations which may affect any subset of resources at once.
- The reconfigurations are a decision on their own, and not a consequence of the tasks to be performed.
- We propose a MILP and a metaheuristic based on a Multi-Start Evolutionary Local Search to deal with this problem.
- Test instances have been generated and computational experiments are reported.

The rest of the paper is as follows: in the next section, literature review is provided. Section 3 introduces the mathematical formalisation. Section 4 introduces the heuristic approach, and section 5 presents first results. The paper ends on concluding remarks and a presentation of future research directions.

2 Related works on Planning and scheduling for RMS

Moghaddam, Houshmand, and Fatahi Valilai (2018) observed that papers on RMS mainly focus on analysing performances of various configurations and they concentrate on developing approaches and mathematical models for design and configuration selections. According to Bortolini, Galizia, and Mora (2018), planning and scheduling problems in RMS refer mostly to production planning, process planning or integrated approaches considering both configuration designs and sequencing problems. In the following, we review some of the works related to planning, sequencing and scheduling of RMS.

Several papers of the literature are addressing both design and control of RMS. For instance, in Moghaddam, Houshmand, and Fatahi Valilai (2018), the authors introduce a two phase method in order to build the primary system configuration design and handle its necessary reconfigurations by considering demand changes. A Mixed Integer Linear Programming approach is used to rearrange the system design by selecting the best possible transformation meeting demand level while minimising the additional cost due to purchasing new RMTs and replacing modules. However, durations of reconfigurations are not considered in their work. In Haddou Benderbal, Dahane, and Benyoucef (2017), machine selection for design of RMS and process planning are addressed. In order to evaluate the performance of a process plan, the authors introduce a new indicator, referred to as the flexibility index. Given a selection of machines and a process plan, the flexibility index is the number of alternative solutions available in case of machine unavailability. These alternative solutions can be obtained by using another selected machine for processing operations, or by postponing the operations with respect to precedencies. A bi-objective optimisation approach is provided where objectives are the total completion time and the total flexibility.

Other research projects are focusing on production planning, as in Hees et al. (2017) who provides a method using RMS which is validated in an application scenario. Reconfigurations allow to adjust functionalities or capacities of the system, or both. The objective is to minimise the total production cost, which consists in reconfigurations, processing, and inventory costs as well as idle time related costs. In Touzout and Benyoucef (2019), different objectives are addressed in process planning optimisation, namely the total cost, the total completion time of all products and the maximum machine exploitation time, in order to avoid reliability issues of components. Delorme et al. (2023) investigate a scenario-based bi-level optimization problem consisting of balancing the operations and planning the configurations of RMS. Three objectives are considered: the number of stations, the expected energy cost per produced unit and the expected service level facing uncertain demands.

As stressed by Azab and Naderi (2015), very few papers deal with scheduling of RMS. In their research work, they addressed reconfigurations in the context of Flow-shop production systems. When a change in configuration has to be considered, the whole production system is stopped. In Yang et al. (2023), the authors address a real time reconfigurable permutation Flow-shop scheduling problem. The objective is to minimise the total tardiness cost of jobs, that arrive dynamically at the shop floor. The system operates under different production modes, each one capable of processing a set of jobs. Such a specific situation can be seen as synchronicity in Flow-shop problems, where a job can start its process on the following machine only after completion of machine operations (Waldherr and Knust 2015). However, if several machines may be inactive in order to operate a switch from a configuration to another, some of them may not be affected, especially in multiple pathways production systems. Furthermore, the concept of reconfiguration can be seen at the machine and production line level, but also at the factory level, which requires adapted models, as in Renna (2013) where production departments are considered. The manufacturing system is modelled as a Job-shop based on reconfigurable machines. Machines can be reassigned to other departments according to workload. A reconfiguration control policy is designed to cope with production fluctuations, mix change and dynamicity level of the environment. Its performance is evaluated with a simulation-based approach, through different indicators such as throughput, or tardiness.

In Doh et al. (2016), several policies are investigated through simulation in the context of a flexible Job-shop production system with one reconfigurable manufacturing cell (RMC). The production system can be divided into a Job-shop and the RMC. Parts can be processed by either one or the other. The problem consists in determining the process routes of the different parts, the sequence of parts that are processed in the RMC, and the sequence of parts assigned to each machine. The objectives consist in minimising the makespan, the mean flow time and the mean tardiness. However, no reconfiguration times, neither setup times are considered. In Borisovsky, Delorme, and Dolgui (2014), balancing of reconfigurable machining lines is addressed. Cycle times of workstations consider the processing of operations and sequence dependent setup times. Hence, a sequencing problem is solved in addition to the assignment of operations to stations. Nevertheless, the problem concerns the design of the initial configuration under which the system is operating and the setup times concern operations within a station, and not the reconfigurations applied at the operational level.

Prasad and Jayswal (2017) presented an approach for reconfiguration of a multi-products line based on two consecutive phases: design and sequencing of products. In the design phase, the number of machines is computed and all resources are arranged in the best possible way. In the second phase, selection of the required reconfigurations is achieved in order to sequence products efficiently. Selection of a transformation is based on the effort for switching from the current configuration to the other one but no reconfiguration times are considered. In Mahmoodjanloo et al. (2020), the authors investigate the use of reconfigurable machine tools in the context of a flexible Job-shop. The specificity of the problem is to have setup times between machine reconfigurations rather than classical sequence dependent setup times between operations. Three decision levels are considered, namely assignments of operations to machines, sequence of operations and configuration selections. In Dou et al. (2020), an integrated approach for configuration design and scheduling of operations is investigated. Two conflicting objectives are considered: Total cost and Total tardiness. Two consecutive periods are considered in this research project, and switching to multiple periods would require future improvements. Once the number of machines is defined, the scheduling problem consists in ordering multiple jobs using a priority rule, and the problem could be referred to a permutation Flow-shop.

In Lamy, Schulz, and Zaeh (2020), a mathematical formalisation is provided for scheduling in reconfigurable multiple path shop floors with consideration of energy efficient machine tools. If setup times are considered for mounting machine tools, it only affects the receiving machine, and not several machines at a time. Similarly, in Fan et al. (2022) the author address a Flexible Job-shop Problem, with reconfigurable machines where modules can be mounted on the machines while considering setup times. In a recent work, Vahedi-Nouri et al. (2022) address a production scheduling and workforce planning problem in the context of a parallel machines shop floor based on heterogeneous RMTs. The problem includes health consideration in the assignment of workers to the machines. However, reconfigurations only affect one machine at a time but not the whole system, and no routing of jobs is considered. In another work, Vahedi-Nouri et al. (2023) investigate a production scheduling and workforce planning in a RMS modelled as a Job-shop with RMT and cobots, with the objective of minimizing the makespan. Operations can be handled at a machine by either an operator, a cobot alone, or both collaboratively. If reconfigurations are operated at the machine level, the authors mention system-level reconfigurations as perspectives. In Tang, Haddad, and Salonitis (2022), a grid-shaped RMS is considered, where reconfigurable machine tools are given cells, and parts are moved using Automated Guided Vehicles. Discrete event simulation and machine learning are used to improve the performance of the system considering jobs with due dates.

As stressed by this short literature review, condensed in Table 1, some papers are considering both design of RMS and sequencing of products at the station or at the system level. However, reconfigurations require time because of addition or removal of resources, or modifications of modules on workstations and this characteristic of RMS seems not to be largely addressed in the reviewed literature. If these reconfigurations can be considered as setup-times, reconfigurations may impact several machines or resources, when generally one machine is affected at a time by a setup time in the scheduling literature. Also, the setup times are generally sequence dependent and not machine dependent, which is closer to reconfiguration times.

In their review of setup times in Job-shop scheduling problems Sharma and Jain (2016) identify two types of setup times: sequence dependent and sequence independent setup times, both in the context of batch and non-batch (job) shop environments. If the authors identify several perspectives in research among which sequence dependent setup times in the context of batch scheduling problems, all the mentioned papers consider setups at one machine at a time, not on a set of machines. Also, the setup times to which their research work refer to depend on the job sequences on the machine, and not on machines consecutive states. Machine dependent setup times have been addressed in Shen, Dauzère-Pérès, and Neufeld (2018), where a Flexible Job-shop is addressed considering also sequence dependent setup times of operations. However, these machine setup times do not require several machines to be stopped.

Considering the above literature, the current paper aims at addressing scheduling at the operational level in multiple-path reconfigurable manufacturing environments where setup-times (i.e. reconfigurations) can affect several machines simultaneously.

Table 1. Synthesis of literature review

Reference	Problem Type	Shop Floor Type	Reconfiguration Type	Criteria	Objectives	Solving Approach
Renna (2013)	Scheduling	Job-shop	One Resource at a Time (setup)	Throughput, Tardiness	-	Simulation
Borisovsky, Delorme, and Dolgui (2014)	Design & Sequencing	Flexible Flow-shop	n/a	Cost	Mono-Obj.	CGA, MIP
Azab and Naderi (2015)	Scheduling	Flow-shop	All Resources at Once	Makespan	Mono-Obj.	MILP
Doh et al. (2016)	Scheduling	Flexible Job-shop	n/a	Makespan, Mean Flow Time, Mean Tardiness	-	Simulation
Haddou Benderbal, Dahane, and Benyoucef (2017)	Design & Process Planning	Flexible Group-shop (A)	One Resource at a Time (setup)	Makespan (A), Total Flexibility	Multi-Obj.	NSGA-II
Hees et al. (2017)	Process Planning & Sequencing	Single Machine (A)	All Resources at once	Total Production Costs	Mono-Obj.	MILP
Prasad and Jayswal (2017)	Design & Sequencing	Flow-shop (A)	One Resource at a Time (setup)	Cycle Time, Reconfiguration Effort, Profit Over Cost, Tardiness	Mono-Obj.	Shannon Entropy, RIM
Moghaddam, Houshmand, and Fatahi Valilai (2018)	Design & Reconfiguration	Flow-shop	One Resource at a Time (setup)	Total Cost	Mono-Obj.	MILP
Touzout and Benyoucef (2019)	Process Planning	Job-shop (A)	One Resource at a Time (setup)	Total Production Costs, Makespan, Exploitation Time	Multi-Obj.	MOILP, NSGA-II, RSUPP, ILSSUPP
Dou et al. (2020)	Design & Scheduling	Permutation Hybrid Flow-shop (A)	One Resource at a Time (setup)	Total Cost, Total Weighted Tardiness	Multi-Obj.	MoPSO
Lamy, Schulz, and Zaeh (2020)	Scheduling	Job-shop	One Resource at a Time (setup)	Makespan	Mono-Obj.	ILP
Mahmoodjanloo et al. (2020)	Scheduling	Flexible Job-shop	One Resource at a Time (setup)	Makespan	Mono-Obj.	MILP, SADE-NMMS
Fan et al. (2022)	Scheduling	Flexible Job-shop	One Resource at a Time (setup)	Total Weighted Tardiness	Mono-Obj.	IGA/MILP
Tang, Haddad, and Salonitis (2022)	Scheduling	Parallel Machines (A)	One Resource at a Time (setup)	Tardiness	-	Simulation, DDQN
Vahedi-Nouri et al. (2022)	Scheduling & Workforce Planning	Parallel Machines	One Resource at a Time (setup)	Makespan, Workers' Preferences, Vulnerability Risks	Agreg.	MILP/CPP
Delorme et al. (2023)	Design & Planning	Flexible Flow-shop (A)	All Resources at Once	Stations' number, Electricity Cost, Service Level	Multi-Obj.	Matheuristic (MOSA+LP)
Vahedi-Nouri et al. (2023)	Scheduling & Workforce Planning	Job-shop	One Resource at a Time (setup)	Makespan	Mono-Obj.	MILP/CP
Yang et al. (2023)	Scheduling	Flow-shop	All Resources at Once	Total Tardiness Cost	Mono-Obj.	EDQN
This paper	Scheduling	Job-shop	Any Subset of Resources at Once	Makespan	Mono-Obj.	MILP, MS-ELS

(A) is for "Assimilated" when the problem is not referred to a shop-floor architecture.

3 Formalisation

The problem under study considers reconfigurable manufacturing shop floors modelled as a Job-shop, and is named Reconfigurable Job-Shop Scheduling Problem (RJSSP). In this problem, a set J of n jobs has to be scheduled $J = \{J_1, J_2 \dots J_n\}$ on a set M of v machines. Each job $j \in J$ consists in a set of l_j ordered operations, noted $O_j = \{O_{1,j}, \dots, O_{l_j,j}\}$. The whole system operates under configurations $k \in K$, which are similar to setup times in scheduling and impact processing times. However, switching from a configuration to another can affect several machines at once, and not necessarily all of them. Let us note M_{k_1,k_2} the set of machines that should be stopped when changing from $k_1 \in K$ to $k_2 \in K$. When a reconfiguration from k_1 to k_2 is triggered, each machine $m \in M_{k_1,k_2}$ is made unavailable for a specific time windows $RT^m_{k_1,k_2}$, and this configuration switch can only be operated when the concerned machines are idled. Processing times of operations also depend on the running configuration and each operation $O_{i,j}$ has a processing time $P^k_{i,j}$ where $k \in K$. This processing time varies from one configuration to another, because a configuration may be defined by different assignments of resources (operators, materials, tools, etc.) to the machines. Therefore, the problem considers modularity and convertibility of RMS. A reconfiguration time window is required when switching from a configuration k_1 to k_2, modelling the operators, materials and tools management. The objective is to schedule efficiently operations and configuration switches in order to minimise the completion time of the last operation on the last machine (makespan). If only one machine is concerned by a reconfiguration, then the problem lies in the field of setup times, meanwhile, if all machines are concerned by a reconfiguration then the problem would be close to the one of Azab and Naderi (2015), however it generalises it as we are focusing on Job-shop floors. Other assumptions consider no pre-emption and availability of all resources and jobs from beginning to end of scheduling time horizon.

These concepts are illustrated in Figure 1, where operations $O_{1,1}$, $O_{2,1}$ and $O_{1,3}$ are operated under configuration K_1. Then a reconfiguration occurs from K_1 to K_2 (noted $RT_{1,2}$), requiring both Machines M_1 and M_2 to be stopped. Next, operations $O_{2,3}$ and $O_{1,2}$ are processed under configuration K_2, until another reconfiguration occurs, switching to configuration K_3. As can be stressed from the Gantt chart, the operation $O_{2,3}$ is still in process when switching from K_2 to K_3 and finishes at time 36, as the reconfiguration does not affect M_1. Thanks to this specificity, we could postpone both operations $O_{2,3}$ and $O_{3,3}$ by 3 time units without being affected by the reconfiguration. In the following, a time indexed mathematical model is introduced to formalise the different constraints of the problem.

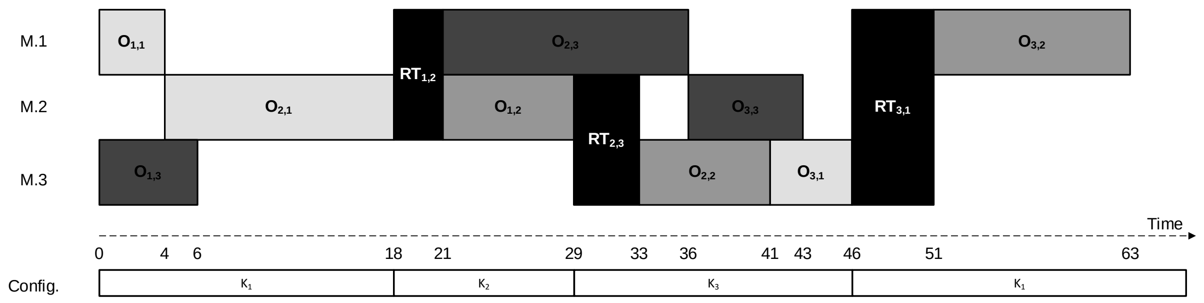

Figure 1. Example of a solution, displaying several reconfigurations ($\boldsymbol{RT}$) that affect different machines

Figure 1. alt. text: Gantt diagram of a schedule that illustrates reconfigurations that affect only specific machines, allowing unaffected ones to continue processing their operations.

Parameters:

M	The set of machines;
J	The set of jobs to schedule;
K	Number of configurations;
j, m, k, t	indices for, respectively, jobs, machines, configuration and time index;
l_j	number of operations of job j;
$O_{i,j}$	Operation number i of job j;
$M_{i,j}$	Machine required for the operation $O_{i,j}$;
$R^m_{k_1,k_2}$	Parameter equal to 1 if the machine m must be switched off during a reconfiguration from configuration k_1 to $k_2$and 0 elsewhere;
RT_{k_1,k_2}	Reconfiguration time for going from k_1 to k_2, i.e. reconfiguration time is considered equal for each affected machine when switching from k_1 to k_2
$RT^m_{k_1,k_2}$	Reconfiguration time required on machine m to switch from configuration k_1 to k_2 ($RT^m_{k_1,k_2}$ = 0 if $R^m_{k_1,k_2} = 0$ and $RT^m_{k_1,k_2} = RT_{k_1,k_2}$ if $R^m_{k_1,k_2} = 1$);
$P^k_{i,j}$	Processing time of the i^{th} operation of job j in the configuration k.
$\overline{P_{\iota J}}$	Maximum value of all $P^k_{i,j}$
P_{max}	Largest processing time of all operations;
T	Time horizon;

Variables:

$bs^t_{i,j}$	Binary variable equal to 1 if operation $\mathrm{O}_{i,j}$ starts at date t and 0 otherwise;
$be^t_{i,j}$	Binary variable equal to 1 if operation $\mathrm{O}_{i,j}$is under process at time t, 0 otherwise;
bc^t_k	Binary variable equal to 1 if the configuration k is used at time t and 0 otherwise;
$br^t_{k_1,k_2}$	Binary variable equal to 1 if at time t configuration is switched from configuration k_1 to k_2;
c_{max}	integer variable corresponding to finishing time of the last operation on the last machine (makespan).

The linear formalisation is a time based indexed formulation that avoids binary variables for disjunctions and that has been proven to be efficient for numerous disjunctive problems including Job-shop (Masmoudi, Delorme, and Gianessi 2019). This model is based on 14 constraints and assumes that the makespan is upper bounded by T, an upper bound obtained by any constructive heuristic or by the sum of all processing time of operations considering the worst case of configurations.

The objective is the minimisation of the makespan.

$$Min\ c_{max} \tag{0}$$

Constraints (1) ensure that one and only one configuration k is used at any time t i.e. at time t only one variable bc^t_k is valued 1.

$$\sum_{k=1..K} bc^t_k = 1 \qquad \forall t = 1..T \tag{1}$$

Constraints (2) ensure that operation $O_{i,j}$ has one starting time only meaning that $bs^t_{i,j}$ is set to 1 at time t if and only if the operation starts at t.

$$\sum_{t=1..T} bs_{i,j}^t = 1 \qquad \forall j = 1..|J|, \forall i = 1..l_j \qquad (2)$$

Constraints (3) and (4) define the makespan i.e. the finishing time of the last operation on the last machine and ensure that this makespan is upper bounded by T.

$$c_{max} \geq \sum_{t=1..T}(t \times bs_{i,j}^t + be_{i,j}^t) \qquad \forall j = 1..|J|, \forall i = 1..l_j \qquad (3)$$

$$c_{max} \leq T \qquad (4)$$

Given a time t, constraints (5) set variables $be_{i,j}^t$ to 1 if the operation starts its processing at time t (based on variable $bs_{i,j}^t$).

$$bs_{i,j}^t \leq be_{i,j}^t \qquad \forall j = 1..|J|, \forall i = 1..l_j, \forall t = 1..T \qquad (5)$$

Considering the starting time of an operation, and the active configuration at time t, constraints (6) ensure that the operation is under process at time $t + P_{i,j}^k - 1$ (i.e. ending date of the operation) as stressed on the Figure 2 (A).

$$bs_{ij}^t + \sum_{k'=1..K:P_{i,j}^{k'} \geq P_{i,j}^k} bc_{k'}^t - 1 \leq be_{ij}^{t+P_{ij}^k-1} \qquad \begin{array}{l}\forall j = 1..|J|, \forall i = 1..l_j, \forall k \in K, \\ \forall t = 1..T - P_{ij}^k\end{array} \qquad (6)$$

Constraints (7) ensure that a configuration is not selected for an operation if its required processing time, starting from t, exceeds the remaining available time (i.e. $T - t$).

$$bs_{i,j}^t + bc_k^t \leq 1 \qquad \begin{array}{l}\forall j = 1..|J|, \forall i = 1..l_j, \forall k \in K, \\ \forall t = 1..T, P[i,j,k] > T - t\end{array} \qquad (7)$$

Constraints (8) ensure that, given a duration d, if the operation is under process at time t and $t + d$, then all variables $be_{i,j}^t$ between these two dates are set to 1. This allows to have all $be_{i,j}^t$ set to 1 during the whole process of an operation, from start to finish.

$$(d-1).\left(be_{ij}^t + be_{ij}^{t+d} - 1\right) \leq \sum_{t'=t+1}^{t+d-1} be_{ij}^{t'} \qquad \begin{array}{l}\forall j = 1..|J|, \forall i = 1..l_j, \\ \forall t \in T - 2, \forall d \in 2..\min(\overline{P_{ij}}, T - t)\end{array} \qquad (8)$$

Constraints (9) define the relative order of successive operations of jobs, respecting Flow-shop and Job-shop conjunctive constraints. For all operations $O_{i-1,j}$ and $O_{i,j}$ the starting time of $O_{i,j}$ is greater than the finishing time of operation $O_{i-1,j}$.

$$\sum_{t=1..T}(t \times bs_{i-1,j}^t + be_{i-1,j}^t) \leq \sum_{t=1..T} t \times bs_{ij}^t \qquad \forall j = 1..|J|, \forall i = 2..l_j \qquad (9)$$

Constraints (10) to (14) deal with reconfigurations and processing dates of operations (Figure 2 (B)). More specifically, constraints (10) ensure that (i) only one operation is processed on each machine at any time t (disjunctive constraints) and (ii) that $O_{i,j}$ cannot be processed at time t if a change from configuration k_1 to k_2 is operated at this moment and if this change implies machine m to be idle during RT_{k_1,k_2}^m time units.

$$\sum_{j \in J} \sum_{i=1..l_j:M_{i,j}=m} be_{i,j}^t + \sum_{\substack{k_1=1..K,k_2=1..K: \\ R_{k_1,k_2}^m=1}} \sum_{\substack{t'=t..T: \\ t'<t+RT_{k_1,k_2}^m}} br_{k_1,k_2}^{t'} \leq 1 \qquad \forall m \in M, \forall t = 1..T \qquad (10)$$

Constraints (11) ensure that a reconfiguration must be finished, before another reconfiguration starts. Each step time of a reconfiguration is modelled through $br_{k_2,k_2}^{t'}$.

$$\sum_{t'=t+1}^{t+RT^m_{k_1,k_2}} br^{t'}_{k_2,k_2} \geq (RT^m_{k_1,k_2}-1) * br^t_{k_1,k_2} \qquad \begin{array}{l}\forall t = 1..T, \forall m \in M, \forall k_1 = 1..K,\\ \forall k_2 = 1..K, k_1 \neq k_2\end{array} \qquad (11)$$

Constraints (12) ensure that binary variables $br^t_{k_1,k_2}$ are set to 1 if a switch from configuration k_1 to k_2 is processed at time t.

$$(bc^{t-1}_{k_1} + bc^t_{k_2} - 1) \leq br^t_{k_1,k_2} \qquad \forall t = 2..T, \forall k_1 = 1..K, \forall k_2 = 1..K \qquad (12)$$

Constraints (13) ensure that if the system does not operate under configuration k_2 ($bc^t_{k_2} = 0$) at time t, then no change in configuration from any other configuration to k_2 is possible. If $bc^t_{k_2} = 1$, then $br^t_{k_1,k_2}$ can be equal to 1 if the switch into configuration k_2 has been achieved, or 0. If $k_2 = k_1$ the configuration time from k_1 to $k_1$is null.

$$bc^t_{k_2} \geq \sum_{k_1=1..K} br^t_{k_1,k_2} \qquad \forall t = 2..T, \forall k_2 = 1..K \qquad (13)$$

Constraints (14) state that if the system is not operating under configuration k_1 ($bc^t_{k_1} = 0$) at time $t-1$, then no change in configuration from k_1 to any other configuration remains possible. If $bc^{t-1}_{k_1} = 1$, then $br^t_{k_1,k_2}$ can be equal to 1 if the switch in configuration k_2 has just been processed, or 0.

$$bc^{t-1}_{k_1} \geq \sum_{k_2=1..K} br^t_{k_1,k_2} \qquad \forall t = 2..T, \forall k_1 = 1..K \qquad (14)$$

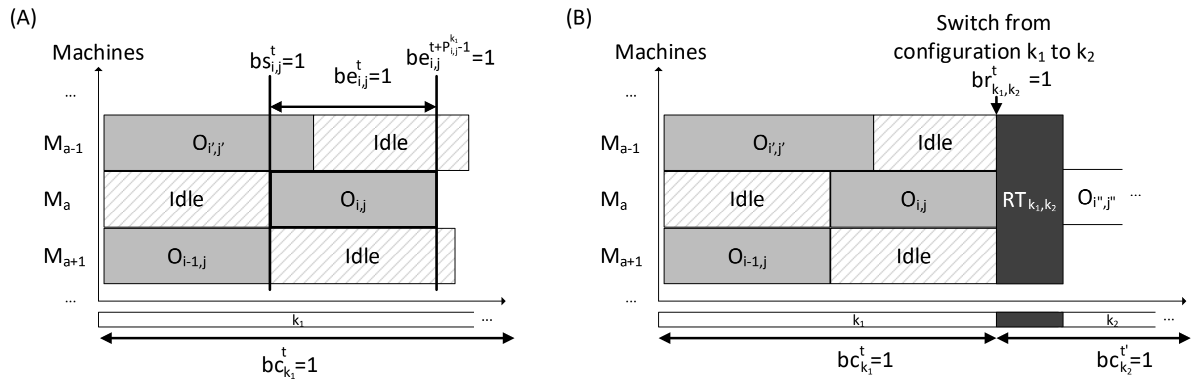

Figure 2. Illustration of values for variables and parameters related to an operation $O_{i,j}$ (A) and reconfigurations (B)

Figure 2. alt. text: two figures displaying Gantt diagrams with illustrations of variables used in mathematical formulation.

Considering all these constraints, and because only one configuration is active at time t, the time horizon, for good solutions, will be divided into periods. Each one of these periods consists in a time window where the same configuration is applied for all the machines. Once all periods are defined, the problem consists in scheduling efficiently operations within the time windows of the different periods.

4 Heuristic based approach

Because the problem is NP-Hard, finding an optimal solution in a reasonable computational time is intractable using the linear formulation. To address this issue, other optimisation methods such as heuristic and metaheuristic approaches, can provide solutions in acceptable computational time. In this paper a multi-start evolutionary local search is designed to address the problem. Evolutionary local search (ELS) has been proposed first by Wolf and Merz (2007) for the super peer selection problem, and applied successfully in several scheduling problems (Chassaing et al. 2014; Palacio and Rivera 2020; Ben-Said, El-Hajj, and Moukrim 2019). It is based on several key elements including a representation of solutions that considers the schedule and the configuration assignment, an evaluation method of these schedules, mutation operators, and a local search procedure.

To illustrate these elements, let us consider a scheduling problem with three jobs, each one having three operations, and let us assume that one operation has to be processed on a machine, with a predetermined duration that depends on the configuration the system operates under. Assume that three configurations are available and let us consider that processing times of operations depend on these configurations as stressed in Table 2, where each cell contains information $M_m(k_1, k_2, k_3)$, where M_m refers to the required machine to perform the operation, and the following triplet corresponds to P_{ij}^k (i.e. the processing time of each operation $O_{i,j}$ depending on the selected configuration). Table 3 reports reconfiguration durations on machines while switching from a reconfiguration to another, and $(x;\ y;\ z)$ is a vector where $x > 0$ enforces that machine $m = M_1$ is concerned by the reconfiguration (M_1 has to be idle during x units of time) and $x = 0$ means that the machine is not concerned and could remain busy until processing another job. Similar remarks hold for y that refers to machine M_2 and z that refers to machine M_3. For example, $(3; 3; 0)$ at the line 1 and column 2 in the Table 3, enforces that machine M_3 is not concerned by a reconfiguration from configuration 1 to 2, whereas machine M_1 and M_2 have to be idle for 3 time units.

Table 2. Jobs sequence, and configuration dependent processing times

Job \ Operations	1	2	3
J_1	M_1(4;12;12)	M_2(14;13;3)	M_3(30;30;5)
J_2	M_2(5;8;2)	M_3(4;4;8)	M_1(12;11;11)
J_3	M_3(6;6;26)	M_1(7;15;15)	M_2(4;2;7)

Table 3. Definition of reconfiguration times ($\boldsymbol{RT_{k_1,k_2}^m}$) that affect specific machines

	1	2	3
1		(3;3;0)	(5;5;5)
2	(3;3;0)		(0;4;4)
3	(5;5;5)	(0;4;4)	

A problem as introduced in Tables 2-3 can be modelled as a conjunctive-disjunctive graph $G = (V, A, E)$, where V refers to the operations, A refers to the conjunctive arcs (i.e. arcs that correspond to the technological order of operations inside a job sequence) and E refers to the disjunctive arcs that should be oriented. Following this representation, the aforementioned problem can be represented as the graph in Figure 3, based on the disjunctive graph model introduced first in 1964 (Roy and Sussmann 1964). For instance, at the top of the Figure 3, $O_{1,1}$, $O_{2,1}$, and $O_{3,1}$ are represented consecutively (conjunctive arcs), modelling the sequence of operations in Job J_1.

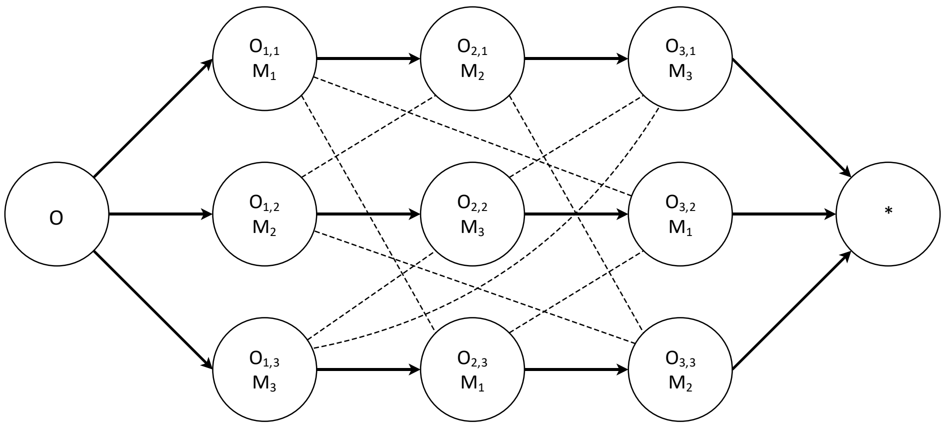

Figure 3. Modelling of a problem as a graph, where no configurations are considered.

Figure 3. alt. text: initial graph of a scheduling problem, displaying sequences of operations for jobs, and edges related to machine disjunctions.

In the problem at hand, each operation should be processed considering a configuration, and processing times depend on selected configurations and are not yet included in the graph of Figure 3.

As stipulated in Table 3, reconfiguration times will affect machines M_1 and M_2 when changing from configuration K_1 to K_2; all machines are affected by a switch from K_1 to K_3; and only machines M_2 and M_3 are concerned with a switch from K_2 to K_3. All these constraints can be modelled with edges in the graph, that are not included in the figure to avoid non-readability. However, when a configuration is assigned to one operation, the graph can be updated with this new information as stated in Figure 4, where only edges related to configurations are revealed.

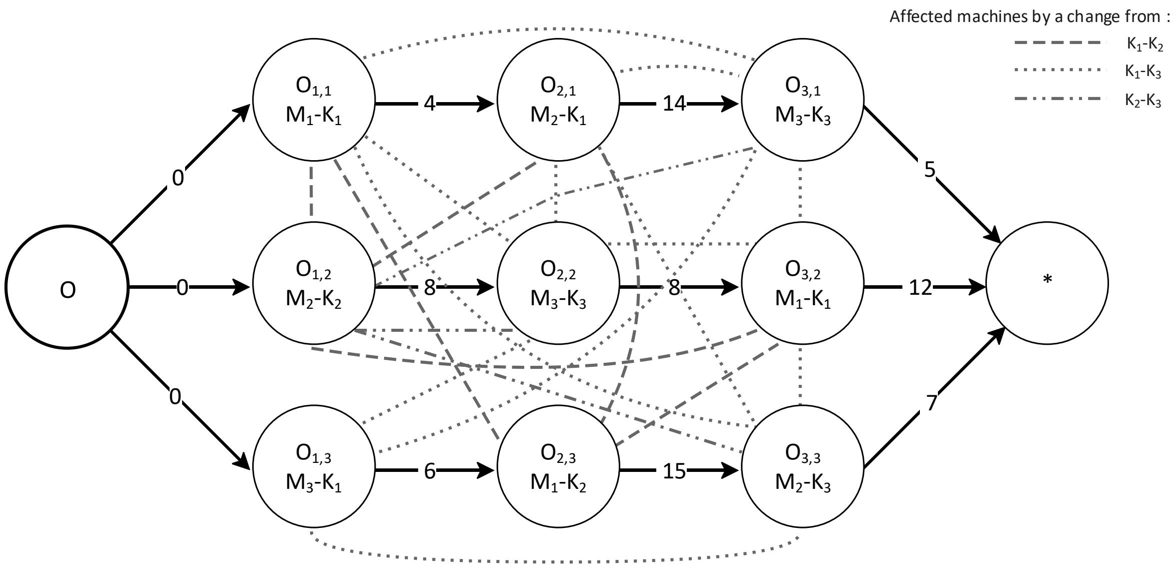

Figure 4. Modelling of a problem that includes configurations assigned to each operation. (edges related to machine disjunctions are withdrawn for readability).

Figure 4. alt. text: Extension of Figure 3, with the graph now displaying selected configurations for operations and edges related to configurations' switches.

In Figure 4, conjunctive arcs are now weighted with the processing times that depend on configurations. Non-solid edges connect operations that are processed with different configurations, if their machines could require a reconfiguration. For instance, dashed arrows connect operations processed under configurations K_1 or K_2, and performed with machines

M_1 or M_2 since M_3 is the only machine that is not affected by a change from the first to the second configuration, or reversely. Finally, an orientation of these edges should be made, to define the starting dates of operations. Such an orientation could be as in Figure 5, where plain grey arcs are modelling precedencies on machines, and dotted arcs are representing remaining reconfigurations that should be respected. Arcs are weighted with the duration of operations plus the reconfiguration time to be considered.

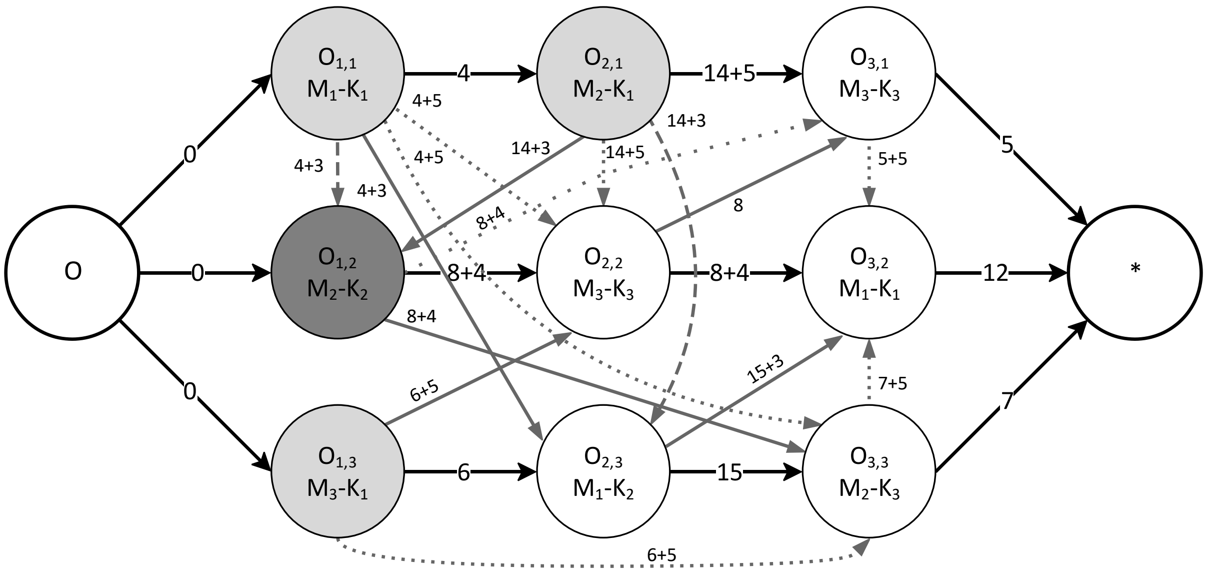

Figure 5 graph with oriented edges and weighted with duration of operations and reconfiguration times

Figure 5. alt. text: Extension of Figure 4, with the graph now displaying oriented edges of operations on machines. An operation is coloured dark grey to illustrates the necessary waiting time due to the reconfiguration.

To move from Figure 3 to Figure 5, a representation of solutions is used and further defined in the following subsection.

4.1 Solution representations

In scheduling problems, a classical representation of solution consists in a list of operations, or jobs, referred to as repetition vector (π) (Bierwirth 1995). In the studied problem, this vector must be extended with another vector (σ) for assignment of configurations to the machines. If such a vector is similar to an assignment vector, as in Flexible Job-shop, it is not assigning a machine to an operation but a configuration, which can affect several machines when a switch is performed. A solution of such a problem can be written with two vectors as in Figure 6. Reading vector π from left to right, the first operation scheduled is the first operation of job 1, in configuration K_1 (read from σ). Next operations to schedule are $O_{2,1}$ and $O_{1,3}$ still under configuration K_1. All these operations are represented with a node in Figure 5 (same light grey) The next job in the sequence π is the job $2 = \pi[4]$ and as it is its first occurrence, it leads to operation $O_{1,2}$ being scheduled in configuration K_2 (darker grey in Fig.5), and so on.

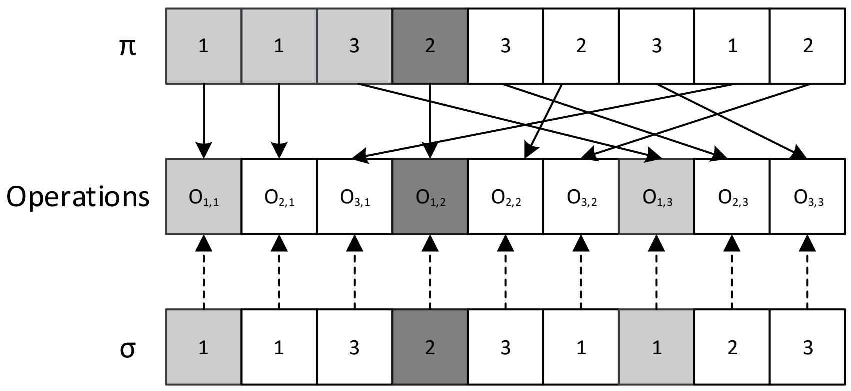

Figure 6. Matching between values in π and σ with operations.

Figure 6. alt. text: illustration of vectors π and σ used in metaheuristic to represent a solution of a problem. Arrows show the correspondence of vector values with operation numbers, which are listed in a third vector.

Once a couple (π, σ) is selected, the transformation of this solution into an evaluated graph (having a starting date to all operations) is achieved using an evaluation function.

4.2 Evaluation function

The evaluation function consists in scanning the π vector from the left to the right and assigning a starting date to each operation according to an iterative scheme. Considering the vector π in Figure 6, the first operation read ($O_{1,1}$) starts at 0 on machine M_1. Next operation to schedule is $O_{2,1}$, which starts at 4 on machine M_2 as it must wait until the end of $O_{1,1}$. The next operation, $O_{1,3}$, that is processed on machine M_3 starts at 0 in configuration 1. Next operation is $O_{1,2}$ that has to be scheduled on machine M_2 in configuration K_2. As there is a switch in configuration from K_1 to K_2, and as this switch affects machines M_1 and M_2, a delay is added on these machines to model the reconfiguration, and the starting time of $O_{1,2}$ must be greater than the finishing time of all operations previously scheduled in configuration 1 (i.e. operation $O_{1,1}$ and operation $O_{2,1}$ in this specific situation). Consequently, a disjunctive arc is added from $O_{1,1} \rightarrow O_{1,2}$ and $O_{2,1} \rightarrow O_{1,2}$ respectively weighted with the sum of processing time of operations $O_{1,1}$ and reconfiguration time from K_1 to K_2 (i.e. $4 + 3 = 7$), and the processing time of $O_{2,1}$ and reconfiguration time (i.e. $14 + 3 = 17$). The starting time of operation $O_{1,2}$ is henceforth fixed at 21, which corresponds to the end of $O_{2,1}$ plus the reconfiguration time. Finally, the complete evaluation of the solution leads to the Gantt diagram presented in Figure 1.

The evaluation procedure is given in Algorithm 1 and is composed of one main loop to scan all the positions in π and at each iteration one operation is scheduled. At step 9, the conjunctive constraints are addressed and the starting time of the operation is updated depending on the finishing time of the previous operation of the job (ftj). At step 13, the available time for the machine (ftm) is updated with the finishing time of the previously scheduled operation. Steps 17 to 26 are dedicated to the consideration of delays for machines that are concerned by a configuration switch. If the condition holds at step 17, it means that a switch is applied from configuration $conf[m]$ and the new configuration c, and two information are computed: the finishing time of the last operation for the machine that must now operate in configuration c - if concerned by a reconfiguration: $ftc = \max(st[job][n[job]])$. Steps 23-25 delay the starting time of all operations scheduled at a machine concerned with the new configuration by $RT^{m}_{conf(m),c}$ units of time. $st[job][n[job]]$ at step 27 is updated with the largest value between $ftc,\ ftj$ (the end of the previous operation of the job i), and the end of the previous operation schedule at machine m (ftm).

Algorithm 1: evaluate_solution

```
Input:
    $(\pi, \sigma)$                 vectors representing a solution
Output:
    $st[i][j]$, $c[i][j]$      starting time and completion time of all operations $O_{i,j}$
    $c_{max}$                  makespan, initialised to 0
Local parameters:
    $n[job]$                   number of operations scheduled for the $job$
    $pm[m][]$                  stores last operation scheduled on the machine $m$
    $conf[m]$                  configuration under process for the machine $m$
    $ftc$                      finishing time of the previous configuration
    $ftj$                      finishing time of the previous operation of the $job$
    $ftm$                      finishing time of the previous operation on a machine
Begin
1.  $n[job] = 0$ for all $job = 1..n$;
2.  $pm[m] = O_{0,0}$ for all $m = 1..v$;
3.  For all $x$ in $\pi$ Do
4.     $job = \pi[x]$;
5.     $n[job] = n[job] + 1$;
6.     $O = O[job][n[job]]$; // the operation to schedule
7.     $m = M[job][n[job]]$; // the machine used for the operation
8.     $k = \sigma[O]$;     // selected configuration for the operation
9.     IF ($n[job] > 1$) THEN //Part 1: get conjunctive date
10.       // finishing time of the previous operation of the job
11.       $ftj = c[job][n[job] - 1]$;
12.    ELSE $ftj = 0$ END IF
13.    IF ($pm[m][0] \neq 0$) Then //Part 2: get disjunctive date
14.       // finishing time of the previous operation on the machine m
15.       $ftm = c[pm[m][0]][pm[m][1]]$;
16.    ELSE $ftm = 0$ END IF
17.    IF $conf[m] \neq c$ THEN //Part 3: get configuration date
18.       $prevC = conf[m]$; $E = \emptyset$; $ftc = 0$;
19.       FORALL $m$ concerned by a reconfiguration from $prevC$ to $c$ DO
20.          $ftc = \max(c[pm[m][0]][pm[m][1]])$
21.          $E += \{m\}$ // $m$ is concerned by the reconfiguration
22.       ENDFORALL
23.       FOR $m \in E$ DO
24.          $c[pm[m][0]][pm[m][1]] = ftc + RT^{m}_{conf(m),k}$
25.       END FOR
26.    ELSE $ftc = 0$ END IF
27.    $st[job][n[job]] = max(ftc, ftj, ftm)$;
28.    $c[job][n[job]] = st[job][n[job]] + P[job][n[job]][k]$ //the duration $P$ depends on the configuration $c$
29.    $pm[m][0] = i$; $pm[m][1] = n[job]$;
30.    IF $c[job][n[job]] > c_{max}$ THEN $c_{max} = c[job][n[job]]$
31. END FOR
End
```

4.3 Multi-start Evolutionary Local Search

The global scheme of the metaheuristic is given in Figure 7. Three procedures are required in this metaheuristic: (i) a random generation of solutions, (ii) diversification of solutions i.e. mutation operator that avoids premature convergence and favours search space investigation, and (iii) intensification through a local search scheme that takes advantages of the solution representation to avoid costly useless modifications and make intensive search into the promising direction. Connecting these components lead to the multi-start ELS presented in Figure 7. At each iteration, a new couple (π, σ) is generated (bloc A1) and evaluated to obtain a new solution S (S = Evaluate (π, σ)) that is next improved by a local search (A2). The following blocs implement the ELS part of the algorithm: the first blocs (B1 to B3) correspond to the neighbours generation. A neighbour S' is obtained by a mutation applied on the couple (π, σ) leading to a new couple (π', σ') ((π', σ') ≔Mutation(π, σ)) that is improved through the local search procedure, if S' has never been visited yet (clone detection). The exploration of neighbourhood during B1-B3 yields best found neighbour bS. The parameters of the metaheuristic are: (i) nb_{start}, which defines the number of initial solutions that will be explored, (ii) nb_n which is the number of neighbours that are obtained through mutation at each step, and

(iii) nb_{ELS} which corresponds to the number of layers of the evolutionary local search procedure. Finally, the algorithm returns the best-found solution S^*.

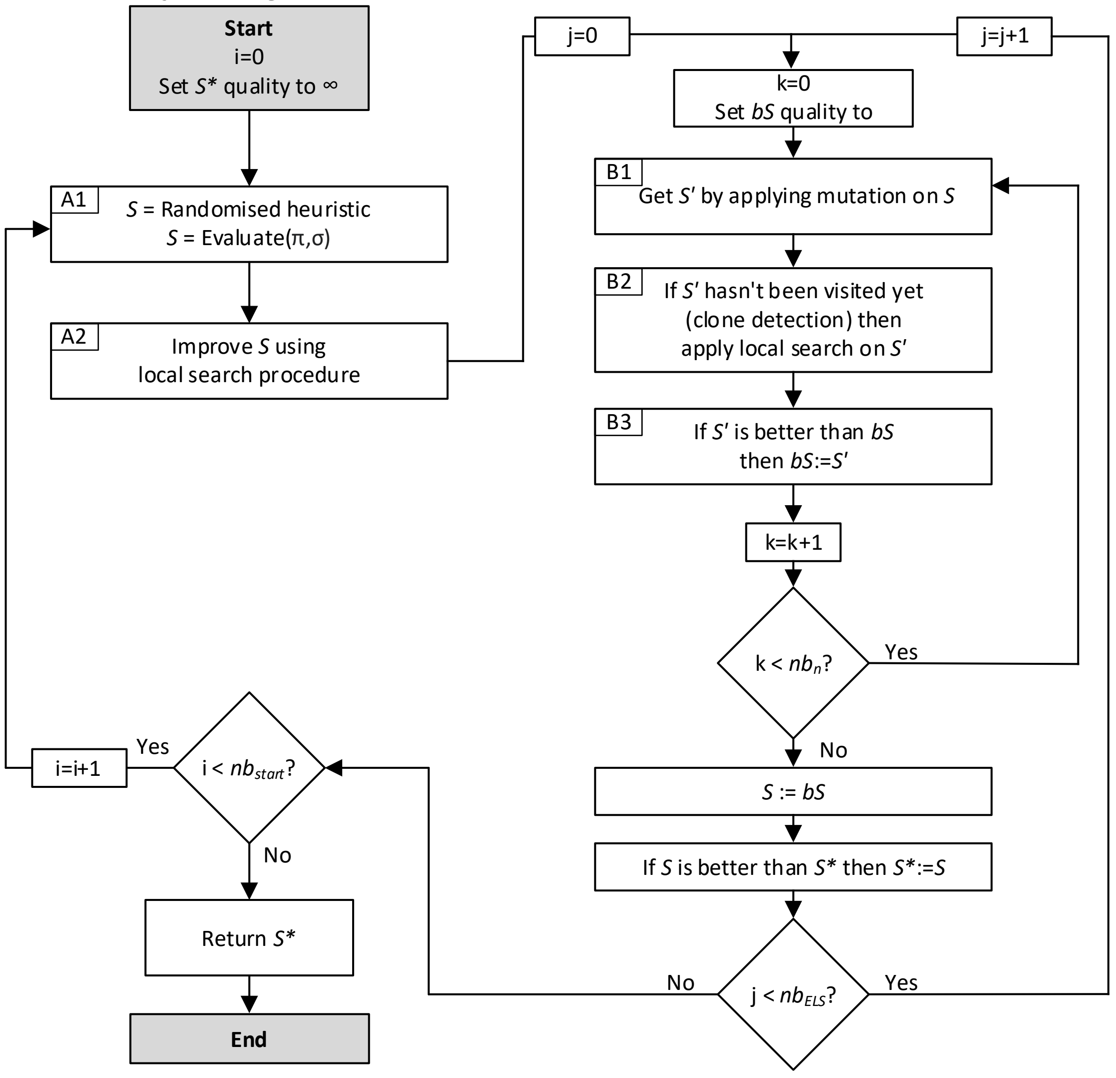
Start
i=0
Set S* quality to ∞
A1
S = Randomised heuristic
S = Evaluate(π,σ)
A2
Improve S using local search procedure
j=0
j=j+1
k=0
Set bS quality to
B1
Get S' by applying mutation on S
B2
If S' hasn't been visited yet (clone detection) then apply local search on S'
B3
If S' is better than bS then bS:=S'
k=k+1
k < nbn?
Yes
No
S := bS
If S is better than S* then S*:=S
j < nbELS?
No
Yes
i=i+1
Yes
i < nbstart?
No
Return S*
End

Figure 7 Flow chart of Multi-Start Evolutionary Local Search (MS-ELS)

Figure 7. alt. text: Flow chart of the designed metaheuristic, displaying sequence of applied procedures including solution generation, local search phase and mutation procedure.

4.3.1 Generating solutions

Initial solutions are built using a randomised heuristic based on the composite rule FDD/MWKR proposed by Sels, Gheysen, and Vanhoucke (2012), as it has shown good performances for Job-shop problems under Makespan objective. FDD stands for Flow Due Date, which corresponds to the sum of durations of already processed operations of a job, while MWKR corresponds to remaining work on this job. At each new generation step, the same configuration is assigned to all operations, and the composite rule is applied to determine a new solution, one operation at a time, favouring operations with lowest value of FDD/MWKR. Possible ties are addressed randomly. The best solution is kept as the starting point of the remaining steps.

4.3.2 Mutation

In order to diversify solutions during the ELS phase, a mutation procedure is used. This procedure is based on two different neighbourhoods and consists in modifying randomly sequence and configuration of operations, namely π and σ. The first neighbourhood is based on a swap operator that is used to move from a vector π to a vector π', meaning that two jobs in vector π are randomly exchanged in the sequence. For instance, $\pi' = [1\ 1\ 2\ 2\ 3\ 2\ 3\ 1\ 3]$ can be obtained from π in Figure 6 by exchanging third and last values of the vector. The second neighbourhood consists in changing σ by assigning randomly another configuration to an operation. To favour diversity of solutions obtained during the mutation phase, and increase chances of extracting from a local minimum, these two neighbourhoods are applied several times (e.g. x swaps and y reassignments) in order to move from a solution S to a mutated solution S',

4.3.3 Local Search

In this first study, a stochastic local search is used (Hoos and Stützle 2005). Starting from a given solution S, the local search mainspring is based on a main loop which randomly changes order of operations in π or assigns a new configuration to an operation in σ. Each modification in π and/or σ requires a new evaluation by the procedure introduced in section 4.2. If one of these modifications lead to a better solution S', then this new solution overwrites S (i.e. first improvement), and the local search continues from this solution. The local search stops when a maximal number of iterations (i_{max}) is achieved.
For instance, changing configuration of $O_{2,3}$ (in Figure 1) from K_1 to K_3, would make the makespan evolve from 63 to 49. Or, changing configuration of $O_{2,1}$ from K_2 to K_1, would improve the makespan to 62, and a following swap between O_2 and O_4 on M_2 would change makespan to 58. Hence, the local search relies on the same neighbourhoods as in the mutation phase, however, it is used differently, as only one change at a time, whether in π or in σ, is allowed to favour intensification.

5 Computational experiments

5.1 Data generation

In order to evaluate the proposed approaches, two datasets have been generated[2], considering 1, 3 or 5 configurations, and containing 120 medium (M) and 39 small (S) instances respectively. Instances are labelled *RJSSP_I_J_K* with I the data type (M or S), J the instance number, and K the number of possible configurations.

- The first dataset (*RJSSP_M*) relies on small and medium scale instances which are named *RJSSP_M_1_1* to *RJSSP_M_40_5*. The processing times of first configuration (p_i) are equal to the processing times of JSP instances taken from the literature (Lawrence, 1984), hence, each optimal solution of JSP's instances are upper bounds for the problem that considers configurations. Then, machines that will not be affected by reconfigurations are randomly defined. Machines are supposed to be affected by a reconfiguration with a probability of 2/3. Next step consists in generating processing times of operations running under second and third configurations. These durations belong to one of four different intervals: $[1.6p_i; 1.8p_i]$, $[1.4p_i; 1.6p_i]$, $[0.7p_i; 1.2p_i]$ or $[0.7p_i; 0.8p_i]$. Probabilities of an operation to have its processing time taken in one of these intervals are respectively $\{0.2; 0.2; 0.2; 0.4\}$. Once an interval is selected, the processing time of the operation is randomly generated according to the intervals' bounds. Similarly, the fourth and fifth configurations have processing times taken in $[1.8p_i; 2p_i]$, $[1.5p_i; 1.8p_i]$, $[0.3p_i; 1.5p_i]$, $[0.3p_i; 0.7p_i]$. If no reconfiguration time is to be considered between two configurations on a given machine, processing times are

[2] To favour fair future research comparisons, all these instances are available online at https://github.com/damienLamy/RMS-Sched

updated to be equal on these configurations. Finally, the reconfiguration times are randomly generated and ensure triangular inequality. Size of problems ranges from 50 to 300 operations.

- The second dataset (*RJSSP_S*) is composed by 39 small instances named *RJSSP_S_1* to *RJSSP_S_13*, ranging from 3 to 10 jobs, 3 to 5 machines (i.e. 9 to 50 operations), and a number of configurations taken in {1, 3, 5} in order to apply the exact approach on smaller instances. The jobs and processing routes are taken from second instance *RJSSP_M_2_5*. Processing times (p_i) are divided by 5 for the first 10 instances of the dataset (in order to avoid being penalised due to high time horizon) and equal to the original data in the last 3 instances.

5.2 Parameter Settings

All experiments have been carried out on a computer running Windows Server 2016, and embedding two E5-2667 Xeon processors (3.2GHz) and 256 Go RAM. For exact solving, a time limit is set to 7200 seconds. Considering the metaheuristic, 50 runs are applied for each instance of the datasets. Parameters of the metaheuristic have been empirically set as follows: $nb_{start} \coloneqq 2000$. To avoid large design experiments, nb_{els} and nb_n are randomly selected at each loop in [75; 100] and [5; 20] respectively. The total number of modifications allowed in the mutation phase varies from 1 to 3 ($x + y$ in section 4.3.2), in order to avoid a large modification of the solution. The mutation procedure favours configuration changings (probability of being chosen is equal to 0.6), while the stochastic local search favours schedule modifications (probability equal to 0.6). For the local search, i_{max} is set to $50 * size$, to have iterations depending on size of instances. A time limit criterion of 300 seconds is also considered in the metaheuristic to avoid large computation times in instances with hundreds of operations.

5.3 Results and discussion

5.3.1 Results computed with exact approach on RJSSP_S

At first, results were computed on *RJSSP_S* using CPLEX 12.8 solver and the results are presented in Table 4. In this table, **#J.**, **#M.** and **#op** consist in the number of jobs, machines and operations of the instance. **LB** and **UB** refer to the lower and upper bound (an asterisk denotes an optimal solution), and **Gap%** refers to the deviation between the two in percent. **CPU** displays the computational time in seconds.

Results synthesis: As can be stressed from this first table, optimal solutions can be obtained on small instances with 1 configuration (Job-shop scenario), with up to 24 operations, in less than twenty minutes. It appears that two reasons may hinders the solver to find solutions: (i) the size of instances, with large observed gaps, or no upper bound when the number of operations increases above 24 operations, or (ii) the duration of the operations. Indeed, whereas instances *RJSSP_S_*2 and *RJSSP_S_*3 were solved optimally, the solver is unable to yield solutions for instances *RJSSP_S_*12 and *RJSSP_S_*13, whose operation times are 5 times longer than those of *RJSSP_S_*2 and *RJSSP_S_*3. Obtaining optimal solutions seems to become even more difficult when the number of available configurations increases, while the maximum time horizon T is decreased. For example, when 3 configurations are considered, no solutions are found for *RJSSP_S_*7_3, to *RJSSP_S_*13_3 in the allowed computation time. Furthermore, as shown in these first results, improvements on UB can be achieved between instances with 3 configurations compared to 1 only (~2% on optimal solutions), and the gap is much larger when comparing 3 configurations and 5 (~22.5%). As can be stressed from LB columns, a potential for improvement exists for obtaining better bounds on problems with 3 and 5 configurations.

Drawbacks: While the time horizon T decreases when considering 3, or 5 configurations, which should help problem solving, the solver is unable to give optimal solutions when exceeding 12 operations. The CPU also increases, since it is 120 times more important when considering 5 configurations compared to 1. Visualisation of this CPU evolution over instance size is given in Figure 8. This figure shows the time used to reach the first solution, the best-known solution, the time to solve the problem to optimality and the time to end when optimality is not proven. Columns are grouped by three, in order to display number of available configurations. As can be stressed, above 20 operations (i.e. more than 5 jobs/4 machines), the solver usually spends all the available time searching for an initial solution without succeeding. *HD* refers to instances with higher durations of operations.

Table 4 results using an exact approach (CPLEX) on RJSSP_*S*

	#J.	#M.	#op	1 Configuration				3 Configurations				5 Configurations			
				LB	UB	Gap%	CPU	LB	UB	Gap%	CPU	LB	UB	Gap%	CPU
RJSSP_S_1	3	3	9	29	29*	0	1	28	28*	0	8	19	19*	0	1
RJSSP _S_2	3	4	12	56	56*	0	4	56	56*	0	340	45	45*	0	2568
RJSSP _S_3	4	3	12	45	45*	0	20	44	44*	0	88	37	37*	0	392
RJSSP _S_4	6	3	18	60	60*	0	190	43	60	28.33	/	23	/	∞	/
RJSSP _S_5	5	4	20	83	83*	0	191	24	/	∞	/	32	/	∞	/
RJSSP _S_6	6	4	24	88	88*	0	774	12	/	∞	/	11	/	∞	/
RJSSP _S_7	8	3	24	74	89	16.85	/	21	/	∞	/	10	/	∞	/
RJSSP _S_8	6	5	30	93	96	3.13	/	13	/	∞	/	14	/	∞	/
RJSSP _S_9	9	4	36	31	/	∞	/	12	/	∞	/	11	/	∞	/
RJSSP _S_10	10	5	50	30	/	∞	/	14	/	∞	/	13	/	∞	/
RJSSP _S_11	3	3	9	153	153*	0	1145	43	/	∞	/	15	/	∞	/
RJSSP _S_12	3	4	12	109	/	∞	/	14	/	∞	/	10	/	∞	/
RJSSP _S_13	4	3	12	52	/	∞	/	10	/	∞	/	9	/	∞	/

/: no solution found, *: optimal solutions

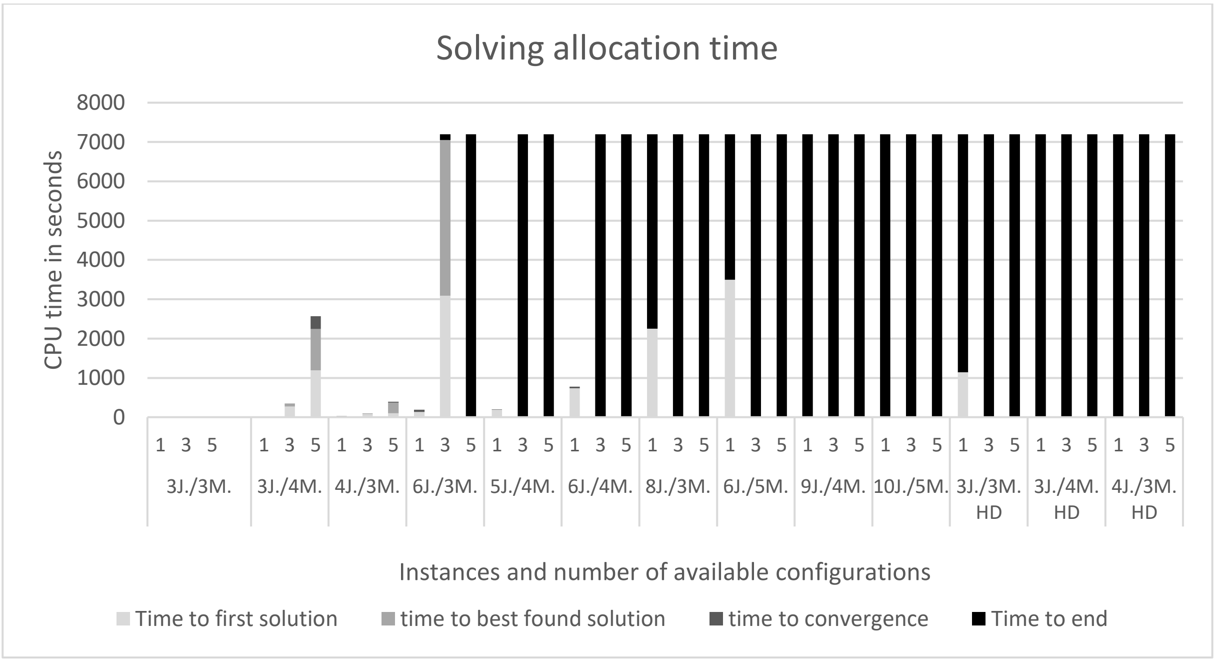

Figure 8. CPU repartition during exact solving

Figure 8. alt. text: Stacked diagram displaying CPU time allocated by the solver for reaching first solution, for reaching best-known solution, the time to convergence if optimal solution is proven, and remaining time to stop if not.

Obviously, these results should not be considered as granted, as they only consider few optimal solutions and other data could lead to different results if processing time of operations are very low on newly added configurations.

5.3.2 Results computed with metaheuristic approach on RJSSP_S

In Tables 5 and 6, results are obtained using the metaheuristic on *RJSSP_S* and *RJSSP_M* respectively. In these tables, S^* refers to the best-known solution (for future comparisons), which corresponds to the minimum value between the best-found solution over 50 runs for a given number of configurations and the best-known solution on instances with less available configurations. $\bar{S}$ corresponds to the average value of the makespan over the 50 runs, and σ corresponds to the average standard deviation. We also compute $S^{1\%}$, the percentage of solutions that are below 1% increase of S^*, representing success rate of reaching solutions close to S^*. Finally, $\bar{c}$ corresponds to the average configuration changes per solution, and $\bar{\mathrm{T}}$ is the average computation time to reach the best-found solution on each run (Computation times near 0 are noted "<1" in the table). Bold values represent results that are strictly better depending on number of reconfigurations (i.e. 3 configurations compared to 1, 5 configurations compared to 3).

Synthesis of obtained solutions for *RJSSP_S:*

- As can be stressed, the metaheuristic is capable of returning several previously found optimal solutions on this dataset. For instances with one configuration the metaheuristic reaches the 7 optimal solutions, improves the upper bound in one instance, and yields 4 upper bounds on instances where CPLEX did not returned solutions. When only one configuration is considered, the metaheuristic always converges towards the same solution ($\sigma = 0$ and $S^{1\%} = 100\%$). When the number of configurations is increasing (3 & 5 configurations) the metaheuristic is capable of finding rapidly solutions, and allows to define 18 new upper bounds on *RJSSP_S* (9 solutions for 3 configurations, 9 solutions for 5 configurations). However, it can be stressed that the criterion $S^{1\%}$ is slightly decreasing when the number of configurations increases, which can be explained by the size of the search space that also increases. Meanwhile, it can be seen that if the average number of reconfigurations increases, the ratio between $\bar{c}$ and the number of available configurations decreases. This may be explained by the achievable performance of operations running under configurations 4 and 5, allowing less reconfigurations, which is also shown by improvements of solutions between 1 configuration and 3 (2.8% improvements on makespan) and between 3 configurations and 5 (16.1% improvements).
- Standard deviation on this dataset remains low when the number of configuration increases (below 0.15% in average when 5 configurations are considered), yet it seems that it becomes challenging to reach the same best-found solution over the different runs on some instances (*RJSSP* _S_7 and *RJSSP* _S_10). This is also attested by $S^{1\%}$ on these instances, with only 34% of solutions below a 1% increase of S^* for *RJSSP* _S_7, and 4% for *RJSSP* _S_10 which suggests that the frequency of reaching a valuable solution decreases on these instances.
- As for the average computation times to best found solution, it remains low, below 15 seconds when considering 5 configurations. Considering the 39 instances of RJSSP_S, $\bar{T}$ remained below 75 seconds, and is below 1 second for 30 instances.

These results show that small size problems are efficiently addressed with the metaheuristic, having almost all optimal solutions reached at least one time, and a low deviation σ.

5.3.3 Results computed with metaheuristic approach on RJSSP_M

On larger instances (Table 6), results show that finding stable solutions seems to be harder when the number of configurations increases. In this table UB refers to optimal solutions of the literature, on instances with one configuration, which gives information on upper bound for problems with higher number of configurations. Cells in grey display solutions that equal the literature on these instances, which shows that the metaheuristic is capable of finding good solutions in this dataset (deviation to UB of 0.23% on average solutions $\bar{S}$) while it has been tuned for problems with higher number of configurations.

Synthesis of obtained solutions for *RJSSP_M:*

- Allowing three configurations, 22 instances display average results ($\bar{S}$) strictly better than best-known solutions with one configuration, and 8 instances display results equal to the best-known-solution over the 40 instances, while it is known by construction that solutions of problems with one configuration are feasible for problems with three configurations. This observation remains true for instances with five configurations: Only 12 instances display average results that are better that the best-known solution obtained with 3 configurations. For instance, it can be observed that average makespan on some problems (e.g. RJSSP_M_28, RJSSP_M_35) when considering five configurations, is worse than results having three configurations, whereas the first three configurations are similar between these datasets. However, average computed makespan ($\bar{S}$) decreases when considering five configurations, which suggests that better solutions can be achieved having two more configurations, even though processing times may have a wider dispersion (see section 5.1). When considering best-known solutions, 75% of instances could be improved considering 5 configurations, compared to instances with one configuration.
- If average solutions of 29 instances with 3 and 5 configurations are better or equal than the optimal solutions with one configuration, the global deviation of the whole dataset is larger than the one obtained with one configuration. For instance, average σ is equal to 0.94 with one configuration, while it attains 4.51 with three configurations. Also, it appears that three instances have deviations higher than 10%, while having better solutions in average than problems with 1 configuration only. These results can be explained by the extended search space implied by the increase in configurations number, while the allowed computation time remains the same, which shows the metaheuristic is capable of taking advantage of a larger space but at the cost of greater variability. This observation remains true for instances with five configurations. Considering σ, it increases on these instances (6.84), which underlines the difficulty for the metaheuristic to find close solutions, and manage the diversity of configurations in the allowed computation time.
- As can be stressed, the percentage of solutions below a 1% increase of S^* rapidly decreases when the number of configurations is rising. Actually, 10 instances have a $S^{1\%}$ value strictly below 10% when 5 configurations are considered, which shows that few obtained solutions are closed to the best achieved one over the different replications, and thus accessing valuable solutions while exploring the search space is harder with the number of configurations that increases.
- Another interesting point is the average number of applied reconfigurations. While only two configurations separate instances with three and five configurations, solutions from this last one display 4 more reconfigurations in average. It seems that several better solutions require lot of reconfigurations, as is the case with RJSSP_M_7, which requires more than 7 reconfigurations in average to reach the average makespan $\bar{S}$ valued 790, which is approximately 6.5% better than the average makespan $\bar{S}$ obtained with three configurations. However, the resources required to carry out these various

reconfigurations on a daily basis could become a challenge. On some instances, the obtained solution considers several reconfigurations without reaching same quality as the problem with lower available configurations (see for example RJSSP_M_28). On this instance, this is explained by the presence of several machines that are not affected by some reconfigurations, leading the solving approach to allow reconfigurations while some machines are idle, without interfering with machining operations. These reconfigurations could be easily discarded with a post processing. Hence, it is difficult to conclude at this stage on whether or not the number of reconfigurations would decrease while improving the solution quality.

- Finally, quartiles of solutions were computed in order to have better knowledge of dispersion of solutions. In Figure 9, average quartiles are presented as box-plots on normalised values of solutions yield by the metaheuristic approach ($HNS_{i,r,c}$), over the different replications. The normalised values are computed as follows:

$$\mathrm{S}^*_{i,c} = \min_{c' \leq c} \left\{ OPT_i, \min_{r=1..50} \left\{ S_{i,r,c'} \right\} \right\}, \forall i \in RJSSP\_M, \forall c \in \{1,3,5\}$$

$$HNS_{i,r,c} = 100. \frac{S_{i,r,c} - S^*_{i,c}}{S^*_{i,c}} \text{ }, \forall i \in RJSSP_M, \forall r \in [1;50], \forall c \in \{1,3,5\}$$

  Where $S_{i,r,c}$ corresponds to the r^{th} solution for instance i with c configurations, $\mathrm{S}^*_{i,c}$ is the best-found solution obtained by considering lower configuration numbers and the optimal solution for problems with one configuration (OPT_i). Then, quartiles are computed on $HNS_{i,r,c}$ and aggregated according to instances characteristics (i.e. number of jobs/machines). In Figure 9, it can be shown that for problems with 1 configuration (Job-shop scenario) worst solutions (higher whisker) are under 1.05% distance of the best-known normalised solutions (which is optimal). When observing instances with 3 configurations, this number rises to 2.13% (with 15 Jobs and 15 Machines), while 75% of solutions are below 1.2% increase and maximum value is below 1.5% increase, which shows that solutions are more scattered. When considering 5 configurations, 25% of solutions are below 1.41% increase of best-known normalised solutions which shows the difficulty of finding solutions close to S^* over the different runs. This is strengthened by the average value of $S^{1\%}$ on instances with 5 configurations (~34% of solutions are below a 1% increase of S^*). However, as the stopping criterion is based on both iteration numbers and a time limit, results could benefit from extending computation times, especially when large available configuration numbers are considered.

An interesting managerial insight can be derived from above results. Some decision makers may think they should avoid unnecessary reconfigurations since they generate non-productive time, even when using a reconfigurable manufacturing system. Such a situation might appear in a shop floor whose flexibility corridor allow to produce all required parts (Job-shop). However, even though the basis production system running only one configuration already embeds enough flexibility to process all jobs, our experiments show that using different configurations can further improve its performance. Actually, we achieved more than 4.5% gains on total completion time when comparing the optimal solution of a single configuration system with the solution obtained using up to five configurations. Furthermore, no additional investment is required here, as the system is already designed to be reconfigured. Moreover, as solutions with multiple configurations are not proven optima, the potential gains could be even greater.

Table 5 results with metaheuristic approach on RJSSP_S

		1 configuration						3 configurations							5 configurations						
	#op	LB	S^*	$\bar{S}$	σ	$S^{1\%}$	$\bar{T}$	LB	S^*	$\bar{S}$	σ	$S^{1\%}$	$\bar{c}$	$\bar{T}$	LB	S^*	$\bar{S}$	σ	$S^{1\%}$	$\bar{c}$	$\bar{T}$
RJSSP_S_1	9	29	29*	29	0	100	<1	28	**28***	28	0	100	1.58	<1	19	**19***	19	0	100	1.08	<1
RJSSP_S_2	12	56	56*	56	0	100	<1	56	56*	56	0	100	0	<1	45	**45***	45	0	100	2.14	<1
RJSSP_S_3	12	45	45*	45	0	100	<1	44	**44***	44	0	100	1.72	<1	37	**37***	37	0	100	0.34	<1
RJSSP_S_4	18	60	60*	60	0	100	<1	43	60	60	0	100	2.14	<1	23	**55**	55	0	100	3.34	<1
RJSSP_S_5	20	83	83*	83	0	100	<1	24	**80**	80	0	100	2	<1	32	**64**	64	0	100	2	21
RJSSP_S_6	24	88	88*	88	0	100	<1	12	**86**	86	0	100	2	6	11	**67**	67	0	100	1.2	<1
RJSSP_S_7	24	74	89	89	0	100	<1	21	**85**	85.7	0.76	46	1.96	42	10	**83**	83.6	0.57	44	4.6	71
RJSSP_S_8	30	93	95	95	0	100	<1	13	95	95	0	100	0	<1	14	**89**	89	0	100	3.04	9
RJSSP_S_9	36	31	114	114	0	100	<1	12	114	114	0	100	0.46	2	11	**106**	106.1	0.42	96	2.84	49
RJSSP_S_10	50	30	127	127	0	100	<1	14	**122**	122.06	0.31	98	1	17	13	**114**	116.74	0.60	4	1.28	23
RJSSP_S_11	9	153	153*	153	0	100	<1	43	**147**	147	0	100	3.64	<1	15	**106**	106	0	100	2.74	<1
RJSSP_S_12	12	109	287	287	0	100	<1	14	**257**	257	0	100	3.48	<1	10	**207**	207	0	100	2.46	<1
RJSSP_S_13	12	52	236	236	0	100	<1	10	**230**	230	0	100	2.74	<1	9	**175**	175	0	100	2.42	<1
Average:					0	100	<1				0.08	95.69	1.75	6				0.12	88.00	2.27	14

*:optimal solutions

Table 6 results with metaheuristic approach on RJSSP_M with 1, 3 and 5 configurations

			1 configuration					3 configurations						5 configurations					
	#op	UB	S^*	$\bar{S}$	σ	$S^{1\%}$	$\bar{T}$	S^*	$\bar{S}$	σ	$S^{1\%}$	$\bar{c}$	$\bar{T}$	S^*	$\bar{S}$	σ	$S^{1\%}$	$\bar{c}$	$\bar{T}$
RJSSP_M_1	50	666	666	666	0	100	<1	**609**	609	0.00	100	2.26	14	609	609	0	100	7.4	10
RJSSP_M_2	50	655	655	655	0	100	<1	**606**	612.38	3.29	30	3.58	59	**551**	563.82	6.30	16	6.16	128
RJSSP_M_3	50	597	597	597	0	100	5	**595**	595.2	0.61	100	2.26	50	**571**	579.82	4.94	26	12.18	92
RJSSP_M_4	50	590	590	590	0	100	<1	**578**	579.96	2.44	100	1	81	**563**	573.76	5.59	26	5.98	88
RJSSP_M_5	50	593	593	593	0	100	<1	**578**	578	0.00	100	1	2	**532**	532	0	100	9.32	<1
RJSSP_M_6	75	926	926	926	0	100	<1	926	926	0.00	100	8.8	<1	**783**	793.16	6.24	34	8.1	97
RJSSP_M_7	75	890	890	890	0	100	<1	**832**	846.14	6.79	24	3.14	97	**750**	790.04	13.90	2	7.52	91
RJSSP_M_8	75	863	863	863	0	100	<1	**809**	818.9	4.29	40	2.88	94	**750**	773.14	6.75	4	4.52	80
RJSSP_M_9	75	951	951	951	0	100	<1	**894**	911.68	7.00	10	3.84	77	**885**	905.14	9.08	12	5.56	58
RJSSP_M_10	75	958	958	958	0	100	<1	958	958	0.00	100	4.28	<1	958	958	0	100	5.42	<1
RJSSP_M_11	100	1222	1222	1222	0	100	<1	**1079**	1103.24	9.65	6	3.92	75	**1077**	1111.48	12.19	4	12.32	72
RJSSP_M_12	100	1039	1039	1039	0	100	<1	1039	1039	0.00	100	6.9	<1	1039	1039	0	100	7.72	<1
RJSSP_M_13	100	1150	1150	1150	0	100	<1	**1080**	1094.26	7.15	30	5.26	105	**1061**	1061.72	2.76	96	9.78	84
RJSSP_M_14	100	1292	1292	1292	0	100	<1	**1123**	1151.8	10.93	10	4.5	56	1123	1148.78	8.56	8	20.9	80
RJSSP_M_15	100	1207	1207	1207	0	100	<1	1207	1207	0.00	100	2.78	2	1207	1207	0	100	3.24	2
RJSSP_M_16	100	945	945	945	0	100	36	**922**	925.02	3.50	90	2.16	88	**918**	930.84	10.88	56	5.7	119
RJSSP_M_17	100	784	784	784	0	100	4	**761**	768.92	7.28	56	1.56	97	**690**	713.96	17.04	22	3.86	148
RJSSP_M_18	100	848	848	848	0	100	3	848	848	0.00	100	0	9	**793**	804.48	6.00	14	8.62	132
RJSSP_M_19	100	842	842	842	0	100	10	**835**	836	2.19	100	1.1	50	**809**	815.1	3.74	72	2.74	108
RJSSP_M_20	100	902	902	902.1	0.71	100	109	**886**	897.08	4.05	14	2.28	143	**861**	875.94	6.36	10	11.9	103
RJSSP_M_21	150	1046	1046	1050.18	3.03	100	112	**1036**	1054.06	4.46	4	0.76	82	**1019**	1037.94	9.05	30	2.46	115
RJSSP_M_22	150	927	927	928.24	2.20	100	119	**925**	933.3	3.49	38	1.12	94	925	939.88	4.98	2	3.46	98
RJSSP_M_23	150	1032	1032	1032	0	100	<1	**1005**	1022.68	6.97	22	1.24	146	**989**	1014.1	8.89	10	2.48	138
RJSSP_M_24	150	935	935	940.4	1.18	100	101	935	943.48	2.95	90	0.02	74	**931**	946.94	6.76	8	0.6	115
RJSSP_M_25	150	977	977	981.24	2.72	100	94	977	985.44	3.26	72	0.04	68	**962**	980.56	9.00	10	1.56	84
RJSSP_M_26	200	1218	1218	1218	0	100	4	**1204**	1215.82	3.87	30	0.38	56	1204	1217.46	1.76	10	0.38	27
RJSSP_M_27	200	1235	1235	1253.06	4.08	66	127	1235	1257.68	4.43	30	0.14	100	1235	1258.46	5.14	24	0.12	84
RJSSP_M_28	200	1216	1216	1217.26	1.93	100	111	1216	1226.58	8.23	68	1.32	124	1216	1234.48	10.25	30	38.42	162
RJSSP_M_29	200	1152	1152	1178.56	5.91	56	122	1152	1186.54	8.55	26	3.26	95	1152	1178.68	10.94	6	8.1	91
RJSSP_M_30	200	1355	1355	1355	0	100	<1	1355	1355	0.00	100	0.06	5	1355	1355	0	100	0.16	6
RJSSP_M_31	300	1784	1784	1784	0	100	<1	**1717**	1753.14	11.88	6	2.5	158	1717	1754.44	10.73	2	3.74	154
RJSSP_M_32	300	1850	1850	1850	0	100	<1	**1794**	1819.54	10.54	26	2.9	157	1794	1824.1	10.08	14	5.44	137
RJSSP_M_33	300	1719	1719	1719	0	100	<1	**1696**	1707.1	6.42	80	3.64	139	1696	1718.06	3.32	8	2.9	8
RJSSP_M_34	300	1721	1721	1721	0	100	<1	**1678**	1696.9	8.64	38	2.52	179	**1671**	1694.24	9.09	18	7.5	192
RJSSP_M_35	300	1888	1888	1888	0	100	<1	**1839**	1839.04	0.28	100	4.1	104	**1800**	1861.68	17.90	2	19.74	172
RJSSP_M_36	300	1268	1268	1278.14	2.36	98	150	1268	1288.48	7.83	40	1.96	145	1268	1292.7	10.36	28	3.1	140
RJSSP_M_37	225	1397	1397	1408.68	5.32	50	146	1397	1416.9	6.66	10	0.16	130	**1392**	1409.14	7.28	52	1.3	103
RJSSP_M_38	225	1196	1196	1205.72	3.34	70	128	1196	1212	5.67	16	0.1	138	1196	1210.92	8.07	40	1.16	128
RJSSP_M_39	225	1233	1233	1244.64	3.69	46	127	1233	1248.8	1.59	6	0.06	95	1233	1243.02	4.38	68	0.96	131
RJSSP_M_40	225	1222	1222	1228.3	1.30	100	112	**1212**	1230.94	5.63	6	0.16	144	1212	1230.46	5.45	10	0.3	114
Average:				1110.04	0.94	94.65	41.05		1092.48	4.51	53	2.25	83.85		1079.71	6.84	34.4	6.57	92.8

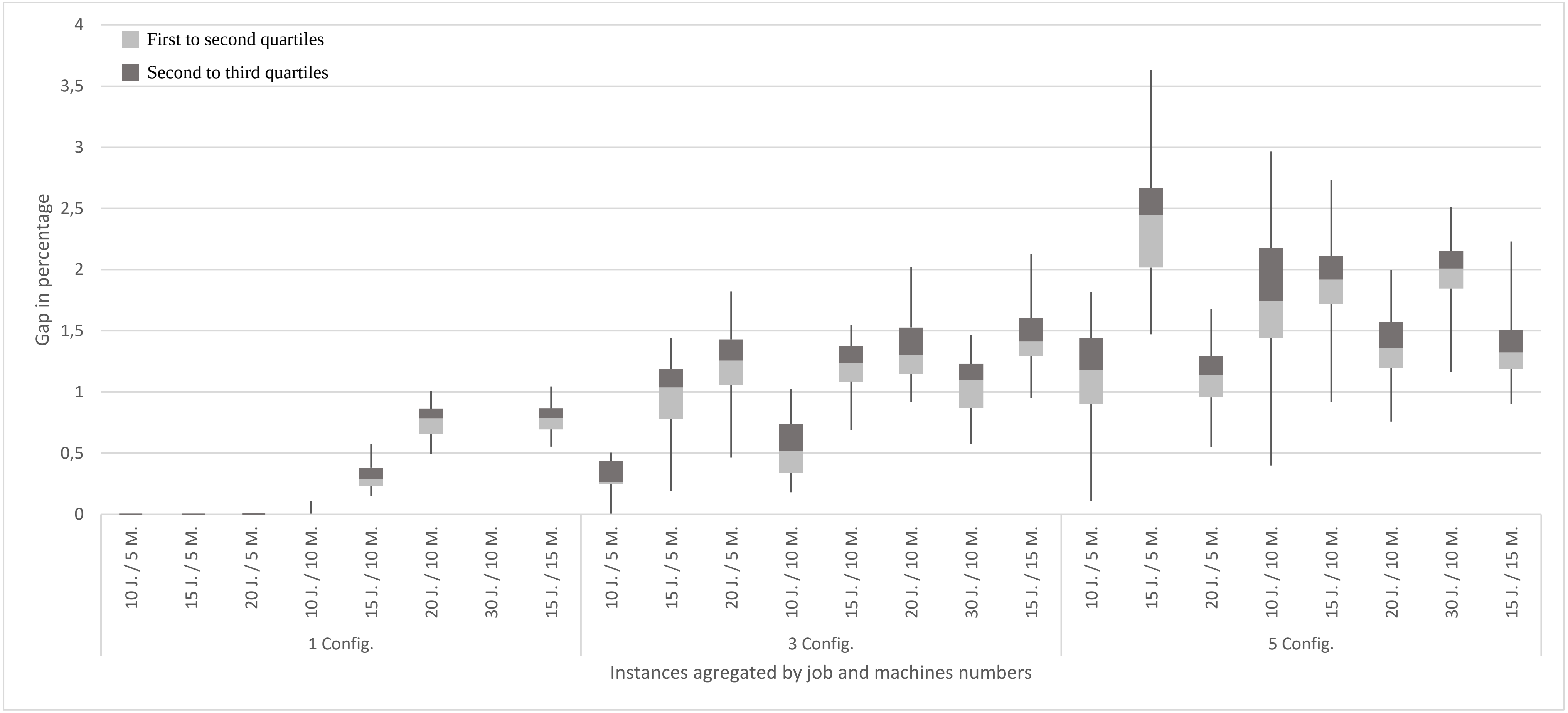

Figure 9. Box-plots of solutions (normalised) aggregated by jobs and machines numbers.

Figure 9. alt. text: The figure shows quartiles of solutions aggregated by instances' structures (i.e. job and machine numbers) and by number of configurations.

6 Conclusion

This work consists in the first formalisation of scheduling problems in Reconfigurable Cellular Manufacturing Systems. In this problem, reconfigurations can affect several machines at once, and are considered at the operational level. An exact method and a metaheuristic based one are provided to tackle different size instances. Results show that the problem as formulated is rapidly intractable using a linear solver (CPLEX). Other exact approaches or formulations and valid inequalities could be included to reduce computation time, as well as warm starts to improve the initial upper bound. Meanwhile, the metaheuristic provides good results on several instances, but it could be improved to have a more reliable behaviour. In order to strengthen quality of solutions from the perspective of average values and standard deviation it could be interesting to investigate other local search strategies, embedding specific features, such as critical path improvements strategies, or instance-based knowledge. The use of Constraint Programming in the local search phase could also be explored, and reducing the search space using machine learning approaches could be a promising direction (Laurent et al. 2021). Also, as optimal solutions of a dataset with few configurations are upper bounds of another dataset with more configurations, the search process could be derived in several steps increasing number of configurations from one step to another, and using the solution found at a prior step as a start point for the new one.

If the problem considers resources related setup times, it could also be interesting to address some other specific features of production systems including, for instance, two types of setup times (between operations on machines, or for reconfigurations) or transportation times because of the conveyors generally connecting different machines. The consideration of costs adjoined with configuration changes or usage could also be interesting, leading to bi-objective problems. In addition, and as stressed by computational experiments, the number of reconfigurations can become difficult to manage in some cases and it could be interesting to study the problem with limits on the number of reconfigurations within a schedule. As operators with variable skills may be present in RMS, it could also be interesting to address the problem with stochastic processing times, as reconfigurations may be postponed because of the constraint on inactivity of all machines concerned with reconfigurations. Finally, RMS are particularly suited to dynamic environments (new product orders to process or machine failures) that require to change configurations, and hence future designs of dynamic optimisation approaches are of great interest for practical industrial situations.

Data availability statement

The data that support the findings of this study are openly available at https://github.com/damienLamy/RMS-Sched and from authors, upon reasonable request.

References

Azab, Ahmed, and Bahman Naderi. 2015. “Modelling the Problem of Production Scheduling for Reconfigurable Manufacturing Systems.” *Procedia CIRP* 33: 76–80. doi:10.1016/j.procir.2015.06.015.

Battaïa, Olga, Alexandre Dolgui, and Nikolai Guschinsky. 2017. “Decision Support for Design of Reconfigurable Rotary Machining Systems for Family Part Production.” *International Journal of Production Research* 55 (5): 1368–1385. doi:10.1080/00207543.2016.1213451.

Ben-Said, Asma, Racha El-Hajj, and Aziz Moukrim. 2019. “A Variable Space Search Heuristic for the Capacitated Team Orienteering Problem.” *Journal of Heuristics* 25 (2): 273–303. doi:10.1007/s10732-018-9395-8.

Bensmaine, A., M. Dahane, and L. Benyoucef. 2014. “A New Heuristic for Integrated Process Planning and Scheduling in Reconfigurable Manufacturing Systems.” *International Journal of Production Research* 52 (12): 3583–3594. doi:10.1080/00207543.2013.878056.

Bierwirth, Christian. 1995. “A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms.” *Operations-Research-Spektrum* 17 (2–3): 87–92.

Borgia, Stefano, Andrea Matta, and Tullio Tolio. 2013. “STEP-NC Compliant Approach for Setup Planning Problem on Multiple Fixture Pallets.” *Journal of Manufacturing Systems* 32 (4): 781–791. doi:10.1016/j.jmsy.2013.09.002.

Borisovsky, Pavel A., Xavier Delorme, and Alexandre Dolgui. 2014. “Balancing Reconfigurable Machining Lines via a Set Partitioning Model.” *International Journal of Production Research* 52 (13): 4026–4036. doi:10.1080/00207543.2013.849857.

Bortolini, Marco, Francesco Gabriele Galizia, and Cristina Mora. 2018. “Reconfigurable Manufacturing Systems: Literature Review and Research Trend.” *Journal of Manufacturing Systems* 49 (October): 93–106. doi:10.1016/j.jmsy.2018.09.005.

Chassaing, Maxime, Jonathan Fontanel, Philippe Lacomme, Libo Ren, Nikolay Tchernev, and Pierre Villechenon. 2014. “A GRASP×ELS Approach for the Job-Shop with a Web Service Paradigm Packaging.” *Expert Systems with Applications* 41 (2): 544–562. doi:10.1016/j.eswa.2013.07.080.

Delorme, Xavier, Audrey Cerqueus, Paolo Gianessi, and Damien Lamy. 2023. “RMS Balancing and Planning under Uncertain Demand and Energy Cost Considerations.” *International Journal of Production Economics*, April, 108873. doi:10.1016/j.ijpe.2023.108873.

Doh, Hyoung-Ho, Jae-Min Yu, Yong-Ju Kwon, Dong-Ho Lee, and Min-Suk Suh. 2016. “Priority Scheduling for a Flexible Job Shop with a Reconfigurable Manufacturing Cell.” *Industrial Engineering and Management Systems* 15 (1): 11–18. doi:10.7232/iems.2016.15.1.011.

Dou, Jianping, Jun Li, Dan Xia, and Xia Zhao. 2020. “A Multi-Objective Particle Swarm Optimisation for Integrated Configuration Design and Scheduling in Reconfigurable Manufacturing System.” *International Journal of Production Research*, May, 1–21. doi:10.1080/00207543.2020.1756507.

Essafi, Mohamed, Xavier Delorme, and Alexandre Dolgui. 2012. “A Reactive GRASP and Path Relinking for Balancing Reconfigurable Transfer Lines.” *International Journal of Production Research* 50 (18): 5213–5238. doi:10.1080/00207543.2012.677864.

Fan, Jiaxin, Chunjiang Zhang, Qihao Liu, Weiming Shen, and Liang Gao. 2022. “An Improved Genetic Algorithm for Flexible Job Shop Scheduling Problem Considering Reconfigurable Machine Tools with Limited Auxiliary Modules.” *Journal of Manufacturing Systems* 62 (January): 650–667. doi:10.1016/j.jmsy.2022.01.014.

Ferjani, Aicha, Achraf Ammar, Henri Pierreval, and Sabeur Elkosantini. 2017. “A Simulation-Optimization Based Heuristic for the Online Assignment of Multi-Skilled Workers Subjected to Fatigue in Manufacturing Systems.” *Computers & Industrial Engineering* 112 (October): 663–674. doi:10.1016/j.cie.2017.02.008.

Galan, R., J. Racero, I. Eguia, and J.M. Garcia. 2007. “A Systematic Approach for Product Families Formation in Reconfigurable Manufacturing Systems.” *Robotics and Computer-Integrated Manufacturing* 23 (5): 489–502. doi:10.1016/j.rcim.2006.06.001.

Grosse, Eric H., Christoph H. Glock, Mohamad Y. Jaber, and W. Patrick Neumann. 2015. “Incorporating Human Factors in Order Picking Planning Models: Framework and

Research Opportunities." *International Journal of Production Research* 53 (3): 695–717. doi:10.1080/00207543.2014.919424.

Haddou Benderbal, Hichem, Mohammed Dahane, and Lyes Benyoucef. 2017. "Flexibility-Based Multi-Objective Approach for Machines Selection in Reconfigurable Manufacturing System (RMS) Design under Unavailability Constraints." *International Journal of Production Research* 55 (20): 6033–6051. doi:10.1080/00207543.2017.1321802.

Hashemi-Petroodi, S. Ehsan, Alexandre Dolgui, Sergey Kovalev, Mikhail Y. Kovalyov, and Simon Thevenin. 2020. "Workforce Reconfiguration Strategies in Manufacturing Systems: A State of the Art." *International Journal of Production Research*, October, 1–24. doi:10.1080/00207543.2020.1823028.

Hees, Andreas, Christina Bayerl, Brian Van Vuuren, Corné S.L. Schutte, Stefan Braunreuther, and Gunther Reinhart. 2017. "A Production Planning Method to Optimally Exploit the Potential of Reconfigurable Manufacturing Systems." *Procedia CIRP* 62: 181–186. doi:10.1016/j.procir.2016.06.001.

Hillier, Mark. 2013. "Designing Unpaced Production Lines to Optimize Throughput and Work-in-Process Inventory." *IIE Transactions* 45 (5): 516–527. doi:10.1080/0740817X.2012.706733.

Hoos, Holger H., and Thomas Stützle. 2005. *Stochastic Local Search: Foundations and Applications*. San Francisco, CA: Morgan Kaufmann Publishers.

Koren, Y. 2006. "General RMS Characteristics. Comparison with Dedicated and Flexible Systems." In *Reconfigurable Manufacturing Systems and Transformable Factories*, edited by Anatoli I. Dashchenko, 27–45. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/3-540-29397-3_3.

Koren, Y, U. Heisel, F. Jovan, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel. 1999. "Reconfigurable Manufacturing Systems." *CIRP Annals* 48 (2): 527–540.

Koren, Yoram, Xi Gu, and Weihong Guo. 2018. "Reconfigurable Manufacturing Systems: Principles, Design, and Future Trends." *Frontiers of Mechanical Engineering* 13 (2): 121–136. doi:10.1007/s11465-018-0483-0.

Lamy, Damien, Julia Schulz, and Michael F. Zaeh. 2020. "Energy-Aware Scheduling in Reconfigurable Multiple Path Shop Floors." *Procedia CIRP* 93: 1007–1012. doi:10.1016/j.procir.2020.04.020.

Laurent, Arnaud, Damien Lamy, Benjamin Dalmas, and Vincent Clerc. 2021. "Pattern Mining-based Pruning Strategies in Stochastic Local Searches for Scheduling Problems." *International Transactions in Operational Research*, April, itor.12984. doi:10.1111/itor.12984.

Lawrence, S. 1984. "Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)." Graduate School of Industrial Administration, Carnegie-Mellon University.

Liles, Donald H, and Brian L Huff. 1990. "A Computer Based Production Scheduling Architecture Suitable for Driving a Reconfigurable Manufacturing System." *Computers & Industrial Engineering* 19 (1–4): 1–5.

Mahmoodjanloo, Mehdi, Reza Tavakkoli-Moghaddam, Armand Baboli, and Ali Bozorgi-Amiri. 2020. "Flexible Job Shop Scheduling Problem with Reconfigurable Machine Tools: An Improved Differential Evolution Algorithm." *Applied Soft Computing* 94 (September): 106416. doi:10.1016/j.asoc.2020.106416.

Masmoudi, Oussama, Xavier Delorme, and Paolo Gianessi. 2019. "Job-Shop Scheduling Problem with Energy Consideration." *International Journal of Production Economics* 216 (October): 12–22. doi:10.1016/j.ijpe.2019.03.021.

Moghaddam, Shokraneh K., Mahmoud Houshmand, and Omid Fatahi Valilai. 2018. “Configuration Design in Scalable Reconfigurable Manufacturing Systems (RMS); a Case of Single-Product Flow Line (SPFL).” *International Journal of Production Research* 56 (11): 3932–3954. doi:10.1080/00207543.2017.1412531.

Ostermeier, Frederik Ferid. 2020. “The Impact of Human Consideration, Schedule Types and Product Mix on Scheduling Objectives for Unpaced Mixed-Model Assembly Lines.” *International Journal of Production Research* 58 (14): 4386–4405. doi:10.1080/00207543.2019.1652780.

Palacio, Juan D., and Juan Carlos Rivera. 2020. “A Multi-Start Evolutionary Local Search for the One-Commodity Pickup and Delivery Traveling Salesman Problem.” *Annals of Operations Research*, September. doi:10.1007/s10479-020-03789-0.

Prasad, Durga, and S. C. Jayswal. 2017. “Reconfigurability Consideration and Scheduling of Products in a Manufacturing Industry.” *International Journal of Production Research* 56 (19): 6430–6449. doi:10.1080/00207543.2017.1334979.

Renna, Paolo. 2013. “Virtual Job Shop Approach Based on Reconfigurable Machines.” *International Journal of Services and Operations Management* 14 (4): 445. doi:10.1504/IJSOM.2013.052838.

Roy, Bernard, and Bernard Sussmann. 1964. *Les Problemes d’ordonnancement Avec Contraintes Disjonctives*. SEMA, Rapport de recherche n°9.

Sels, Veronique, Nele Gheysen, and Mario Vanhoucke. 2012. “A Comparison of Priority Rules for the Job Shop Scheduling Problem under Different Flow Time- and Tardiness-Related Objective Functions.” *International Journal of Production Research* 50 (15): 4255–4270. doi:10.1080/00207543.2011.611539.

Sharma, Pankaj, and Ajai Jain. 2016. “A Review on Job Shop Scheduling with Setup Times.” *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 230 (3): 517–533. doi:10.1177/0954405414560617.

Shen, Liji, Stéphane Dauzère-Pérès, and Janis S. Neufeld. 2018. “Solving the Flexible Job Shop Scheduling Problem with Sequence-Dependent Setup Times.” *European Journal of Operational Research* 265 (2): 503–516. doi:10.1016/j.ejor.2017.08.021.

Tang, Jiecheng, Yousef Haddad, and Konstantinos Salonitis. 2022. “Reconfigurable Manufacturing System Scheduling: A Deep Reinforcement Learning Approach.” *Procedia CIRP* 107: 1198–1203. doi:10.1016/j.procir.2022.05.131.

Touzout, Faycal A., and Lyes Benyoucef. 2019. “Multi-Objective Multi-Unit Process Plan Generation in a Reconfigurable Manufacturing Environment: A Comparative Study of Three Hybrid Metaheuristics.” *International Journal of Production Research*, July, 1–16. doi:10.1080/00207543.2019.1635277.

Vahedi-Nouri, Behdin, Reza Tavakkoli-Moghaddam, Zdeněk Hanzálek, and Alexandre Dolgui. 2022. “Workforce Planning and Production Scheduling in a Reconfigurable Manufacturing System Facing the COVID-19 Pandemic.” *Journal of Manufacturing Systems* 63 (April): 563–574. doi:10.1016/j.jmsy.2022.04.018.

Vahedi-Nouri, Behdin, Reza Tavakkoli-Moghaddam, Zdeněk Hanzálek, and Alexandre Dolgui. 2023. “Production Scheduling in a Reconfigurable Manufacturing System Benefiting from Human-Robot Collaboration.” *International Journal of Production Research*, February, 1–17. doi:10.1080/00207543.2023.2173503.

Waldherr, Stefan, and Sigrid Knust. 2015. “Complexity Results for Flow Shop Problems with Synchronous Movement.” *European Journal of Operational Research* 242 (1): 34–44. doi:10.1016/j.ejor.2014.09.053.

Wolf, Steffen, and Peter Merz. 2007. “Evolutionary Local Search for the Super-Peer Selection Problem and the p-Hub Median Problem.” *Lecture Notes in Computer Science*, Lecture notes in computer science, 4771: 1–15.

Yamada, Y., K. Ookoudo, and Y. Komura. 2003. “Layout Optimization of Manufacturing Cells and Allocation Optimization of Transport Robots in Reconfigurable Manufacturing Systems Using Particle Swarm Optimization.” In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, 2:2049–2054. Las Vegas, NV, USA: IEEE. doi:10.1109/IROS.2003.1248968.

Yang, Shengluo, Junyi Wang, Liming Xin, and Zhigang Xu. 2023. “Real-Time and Concurrent Optimization of Scheduling and Reconfiguration for Dynamic Reconfigurable Flow Shop Using Deep Reinforcement Learning.” *CIRP Journal of Manufacturing Science and Technology* 40 (February): 243–252. doi:10.1016/j.cirpj.2022.12.001.

Yelles-Chaouche, Abdelkrim R., Evgeny Gurevsky, Nadjib Brahimi, and Alexandre Dolgui. 2021. “Reconfigurable Manufacturing Systems from an Optimisation Perspective: A Focused Review of Literature.” *International Journal of Production Research* 59 (21): 6400–6418. doi:10.1080/00207543.2020.1813913.

Youssef, Ayman M. A., and Hoda A. ElMaraghy. 2007. “Optimal Configuration Selection for Reconfigurable Manufacturing Systems.” *International Journal of Flexible Manufacturing Systems* 19 (2): 67–106. doi:10.1007/s10696-007-9020-x.