Computers & Industrial Engineering 173 (2022) 108660

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Minimizing task reassignments under balancing multi-product reconfigurable manufacturing lines

Abdelkrim R. Yelles-Chaouche [a,b,*], Evgeny Gurevsky [c], Nadjib Brahimi [d], Alexandre Dolgui [e]

[a] IRT Jules Verne, Bouguenais, France
[b] Mines Saint-Étienne, LIMOS, Gardanne, France
[c] LS2N, Université de Nantes, France
[d] Rennes School of Business, France
[e] IMT Atlantique, LS2N, Nantes, France

ARTICLE INFO

Keywords:
Manufacturing
Reconfigurability
Multi-product
Line balancing
MILP
Heuristic
Constraint generation

ABSTRACT

This paper deals with optimization aspects of designing reconfigurable production lines with fixed number of workstations. The studied lines are characterized by their ability to handle multiple products, where switching from one product configuration to another is performed by reassigning certain tasks between workstations. Thus, given a set of products and the order of their arrival, the considered optimization problem consists in designing an admissible line configuration for each product such as the total number of task reassignments is minimized. An admissible line configuration can be viewed as a distribution of a given set of tasks among workstations subject to the corresponding cycle time and precedence constraints. To solve this problem, a mixed-integer linear program (MILP) is proposed. Then, an appropriate MILP-based heuristic, named as Halt-and-Fix, is developed to efficiently tackle large scale instances. Computational results are reported on a collection of instances derived from the well-known benchmark data used in the literature for the simple assembly line balancing problem. These results show that the heuristic has demonstrated a good overall performance when dealing with large size instances.

1. Introduction

Nowadays, the current market is characterized by fluctuating demand, a high product mix and a frequent new product introduction driven by a global competition. In such an environment, manufacturing/assembly industries seek to achieve a certain level of flexibility that allows them to react quickly and effectively. Therefore, increasing the reconfigurability of their production systems through the principles of modularity, integrability, customization and convertibility can help them meet this need (see Koren, 2006; Mehrabi et al., 2000).

In this scope, Koren et al. (1999) introduced the concept of reconfigurable manufacturing systems (RMS) to cope with the limitations of flexible manufacturing systems (FMS) and dedicated manufacturing lines (DML), since DMLs are rigid and lack flexibility to adapt to changes and FMS are costly and offer relatively low production rates. RMS are product-oriented (Koren & Shpitalni, 2010), *i.e.*, product characteristics are used as input so as to design a system with the exact needed flexibility. As a result, RMS can be easily adapted to product evolution/changes and demand fluctuations. This is made possible due to the scalable RMS line structure, which allows to adjust its production capacity. Moreover, RMS is mainly composed of reconfigurable machine tools (RMT) or reconfigurable assembly machines (RAM), characterized by their modular design, which gives them the ability to be reconfigured by adding, moving or removing modules (Katz, 2006).

The concept of RMS has not gone unnoticed and has become the subject of growing interest from industry and researchers. This can be observed through recent literature reviews such as the ones of Bortolini et al. (2018) and Yelles-Chaouche et al. (2021) who identified the most discussed topics related to RMS. The design and operation of such systems are the most studied problematic in which the aim is to provide best line configurations that can handle one or several products. These issues are often formulated as optimization problems, where performance indicators such as modularity, scalability and reconfigurability indexes are optimized in order to allow a studied line to be reconfigured quickly and cost-effectively (Ashraf & Hasan, 2018; Benderbal et al., 2017; Borisovsky et al., 2013; Dou et al., 2016).

One of the challenges that companies face when they implement reconfigurability is the workload balancing between workstations. Compared to traditional systems, the design of a reconfigurable line should take into consideration different types of products, where each of which

* Corresponding author at: IRT Jules Verne, Bouguenais, France.
E-mail address: abdelkrim-ramzi.yelles-chaouche@univ-nantes.fr (A.R. Yelles-Chaouche).

https://doi.org/10.1016/j.cie.2022.108660
Received 16 November 2021; Received in revised form 30 July 2022; Accepted 9 September 2022
Available online 15 September 2022

is associated with its appropriate line configuration. This problem is similar to a multi-model line balancing problem (MuMLBP) in which several products are produced in batches according to a predefined order (Becker & Scholl, 2006; van Zante-de Fokkert & de Kok, 1997). Switching from one product to another requires setup operations where some tasks should be reassigned between existing workstations. Such problem is important and often raised in assembly and manufacturing companies (see, *e.g.*, Asl et al. (2019), Lapierre et al. (2000) and Tóth et al. (2018)). Most of the papers on this problematic attempt to solve MuMLBP by transforming it to the single model line balancing problem (SMLBP) as in van Zante-de Fokkert and de Kok (1997). The unique exception is the work of Asl et al. (2019), where the authors handle each product separately by considering multiple objectives. In the present paper, we only focus on one of that objectives, which is the minimization of task reassignments between workstations when shifting from one product line configuration to another. This allows us to get better insights into the problem by taking advantage of its characteristics and hence proposing appropriate approaches to solve it efficiently. Furthermore, Asl et al. (2019) validate their solution approach on a single medium size real case instance. That paper showed the importance of studying such problems in a real context, however, it makes it hard to assess the quality of the solution approach.

The objective of our paper is to contribute to research on the design of reconfigurable lines. Accordingly, we consider a paced line composed of a fixed number of reconfigurable machines, hereafter referred to as workstations, arranged in series and connected by a handling system such as conveyor belt. Each workstation is modular and able to perform different manufacturing tasks. One task is performed by one module. This structure allows to easily reconfigure the line by moving modules from one workstation to another. All modules are assumed to be compatible with all workstations. This is typically the case of RMT which consist of a basic structure where modular machining tools can be mounted. The studied line is able to handle multiple products separately (batch production). The products belong to a same family, meaning that they share some common characteristics. Each product requires a particular configuration of the line. Hence, if a new product has to be produced, the current configuration of the line has to be changed by moving modules through the different available workstations.

Hence, our major contributions in this work consist of:

- Presenting and analyzing a new optimization problem that arises for multi-product reconfigurable production lines which can be found in different industrial sectors,
- Proposing a MILP formulation with new strong pre-processing techniques, and
- Designing and implementing an efficient MILP-based heuristic to solve large scale instances of the problem.

The remainder of the paper is organized as follows. Section 2 presents a literature review. The problem is described and formulated in Section 3. A proposed MILP-based heuristic is presented in Section 4. Numerical experiments and results are detailed in Section 5. Section 6 presents conclusions and future research work.

2. Literature review

Among publications on the RMS, their design is one of the most discussed issues in the literature (see, *e.g.*, Yelles-Chaouche et al., 2021). It consists in assigning manufacturing tasks and allocating reconfigurable machines to workstations while providing a desired production capacity as well as a degree of flexibility. In this problem, the objective is not only to reduce the associated investment cost, as in traditional production systems, but also to take into account the reconfigurable nature of these systems to handle several products. For example, Goyal and Jain (2015) developed a multi-objective particle swarm algorithm to design a reconfigurable flow line while minimizing the capital cost and maximizing operational capability, machine utilization and configuration convertibility. Similarly, Dou et al. (2016) proposed an NSGA-II method to minimize the capital cost, reconfiguration cost and total tardiness of different products to be completed. The authors have considered both strategic (design) and operational (scheduling and reconfiguration) objectives, since the choice of machines has a direct impact on the line configuration and the reconfiguration process. As a result, these objectives need to be considered simultaneously. In Saxena and Jain (2012), the authors propose an artificial immune system (AIS) heuristic to minimize capital, reconfiguration, maintenance and operating costs. In our paper, we also consider the design of a reconfigurable manufacturing line that is able to handle several product types.

As with the reconfigurable flow line design, the process planning problem in a reconfigurable environment was also widely addressed. The aim of process planning consists in determining the detailed machining requirements, set of tasks to be executed and associated pieces of equipment for transforming a raw material into a finished part. Such problems often arise in mechanical part manufacturing, where a set of features, such as a geometric shape, holes, bosses, etc. are provided (Zhang et al., 1997). In such problems, the presence of reconfigurable machine tools (RMT) is considered. Thus, Bensmaine et al. (2013, 2014) developed multi-objective meta-heuristics in order to choose the best RMTs from a given set. The objectives are to minimize the total cost, the production completion time and the makespan. Later, Benderbal et al. (2017) introduced other performance indicators such as system flexibility and system modularity to ensure that the generated process plan is provided with different alternative configurations. Recently, Touzout and Benyoucef (2018) considered the minimization of energy consumption in the process planning problem, along with the total cost and completion time.

It can be observed in the literature that the most studied goal consists in designing/generating a configuration for one product or more. However, few of them tackle the reconfiguration process (*i.e.*, how to reconfigure when the product changes). In this scope, Battaïa et al. (2016) developed a decision support tool to design a reconfigurable rotary machining system able to produce different products (from a same family) in batches. The proposed heuristic is not only used to design a configuration for each batch corresponding to a product family, but also provides the reconfiguration process (which modules to move/remove or change) to switch from one configuration to another. The objective function minimizes the total cost. Recently, Battaïa et al. (2020) suggested a similar decision support tool for designing reconfigurable flow line (instead of rotary machining system) able to produce different products in batches. The authors proposed a mathematical formulation that assigns tasks to RMT while considering technical constraints such as precedence, exclusion and inclusion constraints and technological ones which consider the set of available tools. The objective function consists in minimizing the total cost. In this scope, our article also aims at studying reconfigurable line design and reconfiguration problems, but in a more fundamental and comprehensive manner, *i.e.*, without taking into account specific and sharp constraints related to a precise industrial sector.

The problem under consideration in this paper is referred to as multi-model assembly line balancing problem (MuMLBP). Few research has been done for MuMLBP despite its importance in the industry. For example, Lapierre et al. (2000) and Tóth et al. (2018) consider a multi-model printed circuit board (PCB) assembly line. The aim is to provide an appropriate line configuration in terms of production capacity that is able to produce several PCB-types in batches. The former proposed a MILP model along with a Lagrangian relaxation in order to minimize the cycle time of each batch, whereas the latter proposed an iterative algorithm to minimize the total production time. Both objectives aim to reduce the setup time when switching between batches. Another problem that arises in textile industry is tackled by Pereira (2018), where a line needs to be balanced for several product types. The

objective function is to minimize the resource and workstation associated costs. Chen et al. (2019) proposed a genetic algorithm to solve a MuMLBP in the thin film transistor-liquid crystal display (TFT-LCD) industry. The objective is to minimize both the number of workers and the number of workstations. Similarly, Berger et al. (1992) consider a flexible manufacturing line, where the objective is to optimize the number of workstations. That line is able to produce multiple products, which share common production tasks. A single combined graph represents task precedence relationships of all products. This allows the problem to be solved in the same way as the single product assembly line balancing problem. A flexible multi-model manufacturing line composed of flexible and non-flexible workstations was considered by Nazarian et al. (2010). The former is able to be reconfigured when the products change, whereas the latter only performs a limited number of fixed tasks. The authors proposed a MILP model and a heuristic based on a single combined precedence graph. The objective function minimizes the investment cost which includes the cost of flexible and non-flexible workstations. One of the closest problems to that studied in this paper is Asl et al. (2019), who addressed a MuMLBP. Unlike the previously-mentioned papers, which often combine the precedence graphs of each product into a single one, the authors consider each precedence graph individually. This is due to the fact that products may have different precedence graphs that cannot be merged. The authors study an engine assembly line, and proposed a multi-objective MILP model in order to minimize the cycle time for each product type, and maximize both the number of common tasks in same workstations and a level of workload smoothness between workstations.

Bautista et al. (2016) proposed a methodology that aims to provide a robust mixed-model line balancing under demand variations while considering ergonomic risk on workers. Akpinar et al. (2017) tackled simple and mixed-model assembly line balancing problems. The authors considered setup times that depend on a sequence of tasks assigned to each workstation. Typically, a setup represents a reconfiguration that has to be done within a workstation to perform different tasks on different products. The objective function seeks to minimize the number of workstations. To achieve this, the authors propose an exact algorithm based on Benders decomposition.

Other research work has been done to propose a re-balancing of an existing line in response to a product change or a demand variation. Such a problem is known in the literature as the assembly line re-balancing problem (ALRBP), which consists in reassigning some tasks between existing workstations to meet new requirements. This can lead to a relocation of available resources such as operators and machines. Naturally, one of the main goals of ALRBP is to minimize the number of reassigned tasks, or the cost associated with the re-balancing. Despite the importance of this problem in the industry, little research has been conducted on ALRBP.

Grangeon et al. (2011) addressed ALRBP by minimizing the number of reassigned tasks and the number of workstations. For the same problem, Makssoud et al. (2014) proposed a mixed-integer linear program to minimize the investment cost of new equipment considering the possibility of reusing the already available ones. Sancı and Azizoğlu (2017) considered a re-balancing problem in which a workstation failure or shutdown occurs. In this case, the concerned tasks have to be reassigned to available workstations. The authors sought to achieve a compromise between cycle time and the number of reassigned tasks. Yang et al. (2013) studied a mixed-model assembly line involving operators while considering seasonal demands. Such a line can be re-balanced (reconfigured) to adapt to changes in demand. The objective function aims to minimize the number of workstations and their load variations as well as the cost of re-balancing. In these contributions, while some authors have proposed tailored and general metaheuristics (Grangeon et al., 2011; Yang et al., 2013), others have developed exact procedures such as the branch-and-bound algorithm (Sancı & Azizoğlu, 2017), or have proposed a MILP formulation solved by a commercial solver (Makssoud et al., 2014).

Similar to ALRBP, the problem addressed in this paper seeks to minimize the number of task reassignments. However, the main difference is that we do not consider an already existing balanced line, but aim at finding a best line configuration for each product in terms of the number of potential task reassignments.

3. Problem description and formulation

In the section, a clear definition of the studied problem is given. Subsequently, a corresponding MILP formulation is proposed.

3.1. Problem description

Reconfigurable manufacturing lines are able to produce different products from the same family. Each product has to be produced according to a well-defined production rate. This is expressed by the cycle time, which indicates the time between the release of two successive products. For a product to be manufactured, it needs a given number of elementary tasks to be performed. Each task is associated with a predefined processing time. Moreover, the tasks have to be executed following a partial order that is usually represented through a directed acyclic precedence graph. The latter is composed of vertices representing the tasks and arcs connecting them, thus creating a precedence relationship between the tasks. Since the products belong to the same family, it can be considered that they share the same set of tasks to be performed. However, the precedence graph of each product and the processing times of the tasks can be different due to their level of complexity and process plans selected for each of them. An illustrative example is given in Fig. 1. The latter shows precedence graphs corresponding to three products requiring eight tasks. The processing time of each task is indicated over its corresponding vertex. The aim is to find a suitable line configuration for each product. This can be done by assigning all corresponding tasks to a fixed number of workstations. In each configuration, the cycle time and the precedence constraints have to be satisfied. Since the line is reconfigurable, it allows to switch from one configuration to another by reassigning some tasks. Thus, the objective is to generate appropriate configurations that minimize the total number of reassignments.

Accordingly, a MILP formulation is proposed where the demand for each product is known in a given time horizon, which makes it possible to determine the sequence of product arrival in advance. This is illustrated in Fig. 2, which shows optimal line configurations satisfying the respective precedence relations, represented by corresponding graphs in Fig. 1, constructed on three workstations and with an identical cycle time of 40 time units. Here, Configurations 1, 2 and 3 are related to precedence graphs (1), (2) and (3), respectively. Switching from Configuration 1 to 2 (resp. 2 to 3) requires 1 reassignment (resp. 4 reassignments) that consists of moving task 1 from Workstation 1 to Workstation 2. Hence, the total number of reassignments is 5.

It is not difficult to see that the studied problem is hard to solve, since the search of an admissible line configuration for each product corresponds to solving the feasibility version of the simple assembly line balancing problem (SALBP-F), known to be NP-hard in the strong sense (see Scholl, 1999).

3.2. Problem formulation

A MILP formulation is proposed to minimize the total number of task reassignments. Below, the used notations and variables are introduced.

Notations:

- V is a set of assembly tasks;
- W is a set of available workstations;
- P is a set of products;
- $R = \{(1,2),(2,3),\ldots,(|P|-1,|P|)\}$ is the set of all pairs of successive products;

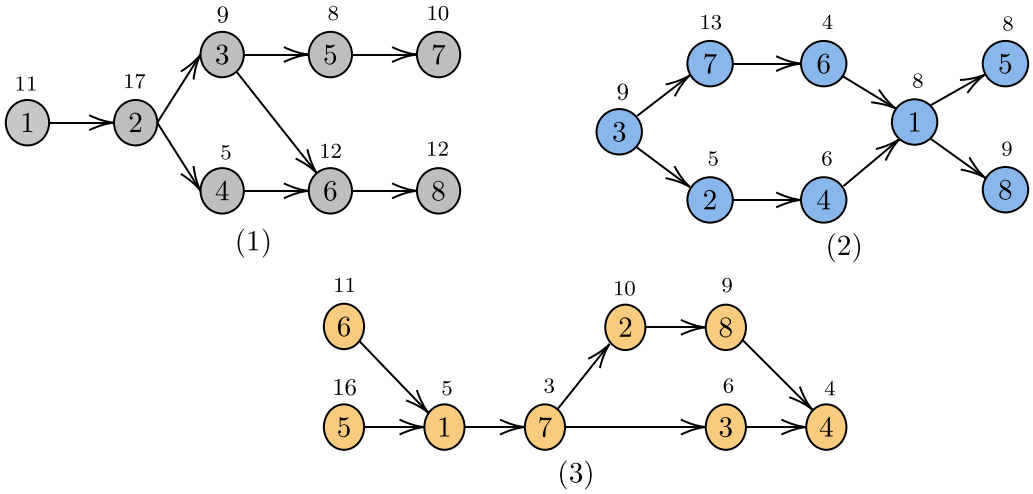

Fig. 1. (1), (2) and (3) are the precedence graphs that correspond to the products 1, 2 and 3.

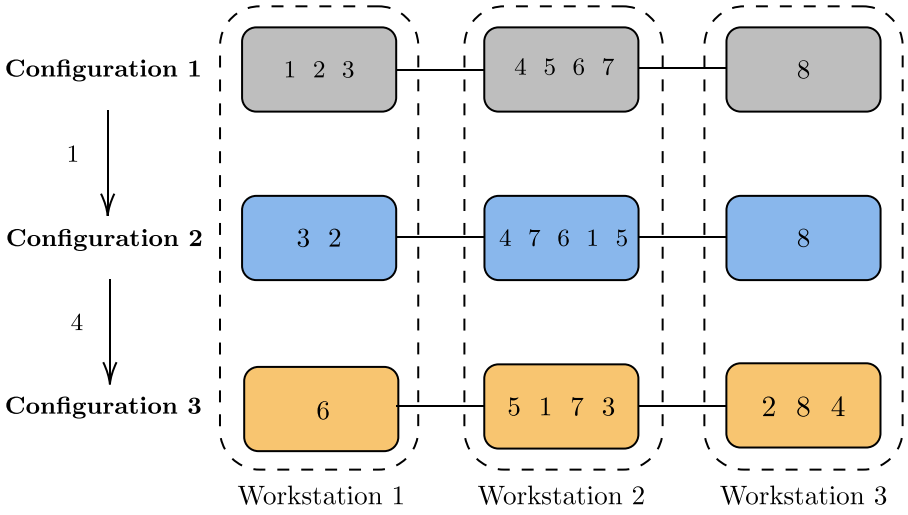

Fig. 2. Optimal product configurations that minimizes the total number of reassignments.

- $C^{(p)}$ is a cycle time corresponding to the product $p \in P$;
- $t_i^{(p)}$ is a processing time of the task $i \in V$ for the product $p \in P$;
- $G^{(p)} = (V, A^{(p)})$ is a directed acyclic graph representing the precedence constraints of the product $p \in P$. Here, $A^{(p)}$ is the set of arcs for $G^{(p)}$, where an arc $(i, j) \in A^{(p)}$ means that task j has to be executed after task i and, as a consequence, can be assigned either to the same workstation as the task i, or to succeeding ones.

Variables:

- $x_{ik}^{(p)}$ is equal to 1 if the task $i \in V$ is assigned to the workstation $k \in W$ in the configuration corresponding to the product $p \in P$, 0 otherwise;
- $z_{ik}^{(1,2)}$ (resp. $z_{ik}^{(2,3)}$, …, $z_{ik}^{(|P|-1,|P|)}$) is equal to 0 if the task $i \in V$ is either assigned or not assigned to the workstation $k \in W$ in both configurations corresponding to products 1 and 2 (resp. 2 and 3, …, $|P|-1$ and $|P|$), 1 otherwise.

Fig. 3 presents an example clarifying the definition of variable $z_{ik}^{(1,2)}$. In this example, task 1 is assigned to workstation 1 in both configurations 1 and 2, thus $z_{11}^{(1,2)} = 0$ meaning that there is no need to reassign task 1. This is also the case for task $|V|-1$. Conversely, tasks 2, 3, 4 and $|V|$ are not assigned to the same workstations in both configurations and have to be reassigned when the configuration changes (*i.e.*, $z_{21}^{(1,2)} = z_{32}^{(1,2)} = z_{42}^{(1,2)} = z_{|V|,|W|}^{(1,2)} = 1$).

$$\min \quad \frac{1}{2} \sum_{i \in V} \sum_{k \in W} \sum_{(r_1, r_2) \in R} z_{ik}^{(r_1, r_2)} \tag{1}$$

subject to:

$$-z_{ik}^{(r_1,r_2)} \leq x_{ik}^{(r_1)} - x_{ik}^{(r_2)} \leq z_{ik}^{(r_1,r_2)}, \quad \forall i \in V, \quad \forall k \in W, \quad \forall (r_1, r_2) \in R \tag{2}$$

$$\sum_{k \in W} x_{ik}^{(p)} = 1, \quad \forall i \in V, \quad \forall p \in P \tag{3}$$

$$\sum_{q=k}^{|W|} x_{iq}^{(p)} \leq \sum_{q=k}^{|W|} x_{jq}^{(p)}, \quad \forall k \in W, \ \forall (i, j) \in A^{(p)}, \quad \forall p \in P \tag{4}$$

$$\sum_{i \in V} t_i^{(p)} \cdot x_{ik}^{(p)} \leq C^{(p)}, \quad \forall k \in W, \quad \forall p \in P \tag{5}$$

$$x_{ik}^{(p)} \in \{0, 1\}, \quad \forall i \in V, \quad \forall k \in W, \quad \forall p \in P$$

$$z_{ik}^{(r_1,r_2)} \in [0, 1], \quad \forall i \in V, \quad \forall k \in W, \quad \forall (r_1, r_2) \in R$$

Objective function (1) minimizes the total number of task reassignments necessary to switch from configuration 1 to 2, 2 to 3, and up to configuration $|P|-1$ to $|P|$. Constraints (2) allow to check if

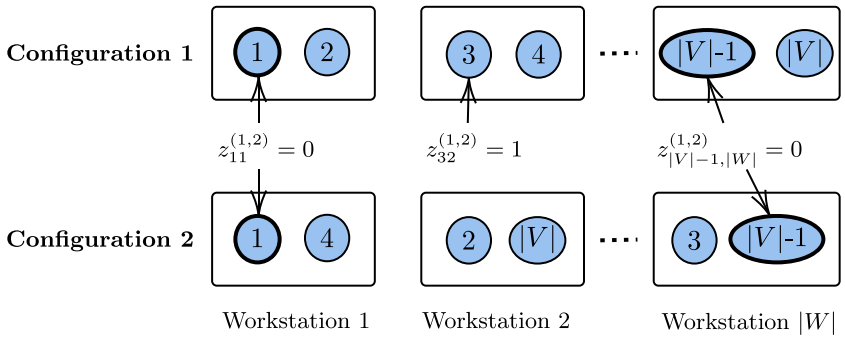

Fig. 3. Clarifying example for decision variable $z_{ik}^{(1,2)}$.

task i is assigned or not to the same workstation in two successive configurations. For example, $z_{ik}^{(1,2)} = 1$ means that task i is assigned to workstation k in only one of the two configurations. Thus, task i has to be reassigned when switching between configurations 1 and 2. Otherwise, $z_{ik}^{(1,2)} = 0$. Moreover, it is not difficult to see that $z_{ik}^{(\cdot,\cdot)}$ can be considered as continues on $[0, 1]$. Constraints (3) ensure that for each product p, task i has to be assigned to one workstation only. In order to satisfy the precedence graph of each product, constraints (4) are used. Constraints (5) ensure that for each product p, the sum of the processing time of the tasks assigned to the same workstation does not exceed the corresponding cycle time.

3.3. Enhanced MILP formulation

In order to strengthen the presented MILP model performance, preprocessing techniques are proposed in this section. These latter take advantage of the problem structure and input data in order to reduce the search space. They are mainly based on the precedence graphs and the task processing times. The use of this information allows to reduce the so-called task assignment intervals, noted by $Q_i^{(p)} = \left[l_i^{(p)}, u_i^{(p)}\right]$, where $l_i^{(p)}$ (resp. $u_i^{(p)}$) corresponds to the index of the earliest (resp. latest) workstation where task i of product p could be placed. Basically, the assignment interval of any task is set to $[1, |W|]$.

In order to reduce the assignment intervals as much as possible, we combine two distinct manners to compute them. The first interval, noted as $Q1_i^{(p)}$, is a well-known one developed by (Patterson & Albracht, 1975):

$$Q1_i^{(p)} = \left[\left\lceil \frac{t_i^{(p)} + \sum_{j\in\mathcal{P}_i^{(p)}} t_j^{(p)}}{C^{(p)}} \right\rceil, |W| + 1 - \left\lceil \frac{t_i^{(p)} + \sum_{j\in\mathcal{S}_i^{(p)}} t_j^{(p)}}{C^{(p)}} \right\rceil\right].$$

Here, $\mathcal{P}_i^{(p)}$ (resp. $\mathcal{S}_i^{(p)}$) represents the set of all predecessors (resp. all successors) of task i with respect to precedence graph $G^{(p)}$ of product p. In $Q1_i^{(p)}$, the index of the earliest workstation is given by the ratio of the sum of all processing times corresponding to the set of tasks $\mathcal{P}_i^{(p)} \bigcup \{i\}$ and the cycle time of product p. This ratio reflects a lower bound on the number of workstations necessary to assign i and all its predecessors. Similarly, the latest workstation where task i could be assigned is calculated based on all its successors.

The second way to compute assignment intervals is inspired by the approach initially developed by Johnson (1988) and presented in its advanced version in Scholl (1999, p. 44–47) for computing a lower bound, noted as LB_4, on the number of workstations for the SALBP-1 problem. It is based on representing the studied problem as a one-machine scheduling one. Namely, it is possible to associate to each task i of product p its earliest completion time $\theta_i^{(p)}$ and its latest starting time $\zeta_i^{(p)}$. Based on the cycle time of each product p, it is not difficult to see that these parameters can be computed in the following recursive manner:

$$\theta_i^{(p)} = t_i^{(p)} + \max_{q\in\overline{\mathcal{P}}_i^{(p)}} \max\left\{\theta_q^{(p)}, \left(\left\lceil \frac{\theta_q^{(p)} + t_i^{(p)}}{C^{(p)}} \right\rceil - 1\right)\cdot C^{(p)}\right\},$$

$$\zeta_i^{(p)} = \min_{q\in\overline{\mathcal{S}}_i^{(p)}} \min\left\{\zeta_q^{(p)}, \left(\left\lfloor \frac{\zeta_q^{(p)} - t_i^{(p)}}{C^{(p)}} \right\rfloor + 1\right)\cdot C^{(p)}\right\} - t_i^{(p)},$$

where $\overline{\mathcal{P}}_i^{(p)}$ (resp. $\overline{\mathcal{S}}_i^{(p)}$) is the set of direct predecessors (resp. direct successors) of task i corresponding to precedence graph $G^{(p)}$ of product p. $\theta_0^{(p)} = 0$ and $\zeta_{|V|+1}^{(p)} = C^{(p)} \times |W|$ are respectively the dummy start and end of the schedule. Thus, the second way of computing the assignment intervals, noted as $Q2_i^{(p)}$, can be expressed as follows:

$$Q2_i^{(p)} = \left[\left\lceil \frac{\theta_i^{(p)}}{C^{(p)}} \right\rceil, 1 + \left\lfloor \frac{\zeta_i^{(p)}}{C^{(p)}} \right\rfloor\right].$$

As a result, and in order to provide the tightest assignment intervals for each task, $Q_i^{(p)}$ is set as $Q1_i^{(p)} \bigcap Q2_i^{(p)}$ for each task i and product p.

Using $Q_i^{(p)}$, constraints (4) can be improved by (4'), and constraints (6) can strengthen the initial MILP model. Equalities (6) are important since they allow to set some decision variables $x_{ik}^{(p)}$ to 0 from the beginning. Thus, reducing considerably the search space.

$$\sum_{q=k}^{|W|} x_{iq}^{(p)} \le \sum_{q=k}^{|W|} x_{jq}^{(p)}, \quad \forall (i,j) \in A^{(p)}, \ \forall k \in Q_i^{(p)} \bigcap Q_j^{(p)}, \ \forall p \in P \tag{4'}$$

$$x_{ik}^{(p)} = 0, \quad \forall i \in V, \ \forall k \notin Q_i^{(p)}, \quad \forall p \in P \tag{6}$$

Adding assignment intervals improves the overall performance of the MILP model when dealing with small and medium size instances. However, as indicated in our experimental results, the commercial solver was still unable to solve large size problems. Accordingly, the next section presents an interesting appropriate heuristic, which is based on the enhanced MILP model.

4. MILP-based heuristic

In order to efficiently solve large size instances of the studied problem in a reasonable amount of time, an iterative MILP-based algorithm is proposed. Its principal idea consists in analyzing feasible solutions found by the MILP model within a short CPU time, then identifying certain characteristics linked to the problem and generating new constraints in order to reduce the search space. The developed approach is not an exact one, since it is based on near-optimal rules, described below.

To simplify the further presentation of this section, the following new notations are introduced:

- $K(r_1, r_2)$ is a set of tasks assigned to the same workstation for the products r_1 and r_2;
- T is the maximum CPU time allowed in each iteration.

The generated constraints have the following expression:

$$z_{ik}^{(r_1,r_2)} = 0, \quad \forall (r_1, r_2) \in R, \quad \forall i \in K(r_1, r_2), \ \forall k \in W. \tag{7}$$

At the beginning of the proposed approach, named hereafter as Halt-and-Fix heuristic, no feasible solution is known and the set of constraints (7) is empty, since $K(r_1, r_2) = \emptyset$ for each $(r_1, r_2) \in R$. Each iteration of this heuristic consists in solving, within the time limit T, the MILP model (1)–(7) by a MILP solver. The aim is to identify an admissible solution that is better (in terms of the objective function value) than the best one found in the previous iteration. If it is the case, then the following series of steps is carried out:

1. The MILP solver is interrupted
2. Based on this new better solution, the set $K(r_1, r_2)$ is updated for each $(r_1, r_2) \in R$
3. A novel iteration starts on the MILP model (1)–(7), enriched by new constraints (7) and using the current better solution as a warm-start one.

Two other specific cases are also possible within the time period T of each iteration. The first one occurs when an optimal solution is found before exceeding the time limit T, then that solution is returned as the final one and the algorithm stops. The second case happens when the goal of the current iteration is not realized (*i.e.*, no better admissible solution is found), then the MIP solver is not interrupted and continues to solve up to either improving the objective function value or reaching the global time limit, fixed for the Halt-and-Fix heuristic. For this last situation, the algorithm stops and the best found admissible solution is returned as the final one. The steps 1 and 2 above can be referred as Halt and Fix steps, respectively.

For two given configurations corresponding to the products r_1 and r_2, constraints (7) reduce the search space by forcing task i from $K(r_1, r_2)$ to be at a same workstation. However, constraints (7) do not exactly identify the workstation where the task i has to be assigned and thus offering some freedom of choice knowing that in most cases $|Q_i^{(r_1)} \bigcap Q_i^{(r_2)}| > 1$.

A more formal description of the proposed approach is given below by Algorithm 1.

In the next section, a detailed performance comparison between the MILP model employed alone with the developed Halt-and-Fix heuristic is done through computational experiments on well-known benchmark data instances.

5. Computational results

This section aims to present the experimental results of the exact MILP approach with CPLEX, noted hereafter as exact approach, and to assess the performance of the proposed heuristic. To do this, first the used benchmark data instances are introduced, designed and described. Then, an analysis of the results for the sole MILP model is provided and finally compared with those obtained by the heuristic. The above-mentioned numerical experiments were conducted on a 1.90 GHz Intel(R) Core(TM) i7-8650U computer with 32 GB RAM.

5.1. Generated instances and input data

For numerical experiments, the well-known benchmark instances from Otto et al. (2013) were used. They are mostly associated to assembly line balancing problems. Only the instances where precedence constraints are represented by a directed acyclic weakly connected graph were taken into account.

In each of these instances, the vertex enumeration of the precedence graph follows a topological order, which makes most of the instances inappropriate for our problem, as can be seen in the obtained computational results in Table 3. Therefore, we decided to alter the precedence graphs to obtain more *numerically challenging* instances. More precisely, each used precedence graph was modified with respect to a randomly generated permutation of vertices. Fig. 4 shows an example of such a modification for the permutation $(3, 7, 2, 6, 4, 1, 5, 8)$, where the original graph is on the left-hand side, and the modified one is on the right-hand side. Although some of these instances present extreme situations that might not exist in a real industrial context, they are more interesting for evaluating the performance of the MILP formulation and the heuristic.

Two sets of instances were used: medium size ($|V| = 50$) and large size ($|V| = 100$) ones. Each of them is classified into three series based on a density indicator of their corresponding precedence graphs, known as the order strength (OS). The OS is computed as $\frac{2 \cdot |E^{(p)}|}{|V|(|V|-1)}$, where $E^{(p)} = \left\{ (i, j) \mid i \in V, j \in S_i^{(p)} \right\}$ is the set of arcs in the transitive closure of $G^{(p)}$. first series (see Table 1) corresponds to the instances having OS around 0.2, noted $OS \approx 0.2$. The second (resp. third) one concerns the instances with $OS \approx 0.6$ (resp. $OS \approx 0.9$). For each series, the cases of $|P| = 2$ and $|P| = 3$ are considered. Thus, for example, to construct an instance for two (resp. three) products, it is needed to simply consider two (resp. three) successive instances of Otto et al. (2013) from the same series. In total, 640 instances were generated for $|P| = 2$ and 634 instances were generated for $|P| = 3$.

Moreover, for each instance, the cycle time $C^{(p)}$ of each product $p \in P$ is set to $\left\lceil 1.5 \cdot \max_{i \in V} t_i^{(p)} \right\rceil$ and the number of workstations $|W|$ is fixed to $\max_{p \in P} \left\{ \left\lceil 1.2 \cdot \sum_{i \in V} t_i^{(p)} / C^{(p)} \right\rceil \right\}$. This choice was made after preliminary experiments in order to obtain non-trivial instances.

Notice that the instances of Otto et al. (2013) have other characteristics such as west ratio, time interval and time variability ratio. However, in our numerical experiments, the only identified characteristics that have impacted the instance's hardness were instance size and order strength (OS). That is why only these two were analyzed in this paper.

5.2. Computational results for the MILP model

CPLEX 12.10 was used to solve the enhanced MILP formulation detailed in Section 3.3. The computational results are presented in Table 2. A global time limit of 1800 s was fixed for solving each instance. The first and second columns of Table 2 represent the number of products and the number of tasks, respectively. The third one illustrates the series of the corresponding set of instances, while the fourth shows the total number of instances in each series. The number of instances optimally solved within the global time limit and their corresponding average CPU time are shown in the fifth and sixth columns, respectively. The seventh column represents the average GAP for the remaining instances which were not solved to optimality. For each instance, the GAP is computed with respect to the objective function value of the best found solution (UB) and the lower bound value (LB), determined by CPLEX, using the following expression $100\% \cdot (UB - LB)/UB$. Finally, the last column shows the interval of the used number of workstations for each series.

As can be seen from Table 2, CPLEX fails to find optimal solutions in most cases. Indeed, the MILP model becomes more difficult to solve when the number of products and/or the number of tasks increase. 77.3% of the instances corresponding to $|V| = 50$ and $|P| = 2$ were optimally solved versus only 29.84% when $|V| = 50$ and $|P| = 3$. Also, for each set, instances with a larger OS seems to be easier to solve than those with a smaller one, where a decreasing trend of the average GAP and CPU time is noticed. Regarding the instances of $|V| = 100$, the solver is unable to find an optimal solution in most cases for both $|P| = 2$ and $|P| = 3$. This is due to the fact that CPLEX hardly improves the lower bound during the solving process.

As for Table 3, it shows the computational results obtained for the same instances used for Table 2, but without modifying their precedence graphs. From this table, it is clear that medium size instances ($|P| = 2$ or $|V| = 50$) are easy to solve for the majority of tests. Whereas, large size ones ($|P| = 3$ and $|V| = 100$) are relatively harder, but still easier to solve compared to modified instances (see Table 2). However, it can be noticed that the obtained average GAP values are not meaningful ($\geq$90%). Such a behavior is mainly due to the fact that the lower bound found by CPLEX is mostly equal to 0, because of

Algorithm 1: HALT-AND-FIX HEURISTIC

Step 1: **Initialization.**

- Set the iteration time limit T.
- Set $K(r_1, r_2) = \emptyset$ for each $(r_1, r_2) \in R$.
- Set the best current feasible solution $s^{(B)}$ as an empty solution.

Step 2: **Solving MILP model.**

- Start to solve the problem (1)–(7) with $s^{(B)}$ as a warm-start solution
- If an optimal solution is found within T, then go to Step 4.
- If T is reached and a solution s better than $s^{(B)}$ is found, then go to Step 3.
- Otherwise, continue to solve until a solution better than $s^{(B)}$ is found. Then, go to Step 3.

Step 3: **Analyze a new better solution.**

- Interrupt solving the problem (1)–(7).
- Update the set $K(r_1, r_2)$ for each $(r_1, r_2) \in R$ based on the new found solution s.
- Reset $s^{(B)} := s$ and repeat Step 2.

Step 4: **Stop, the final approximate solution is found.**

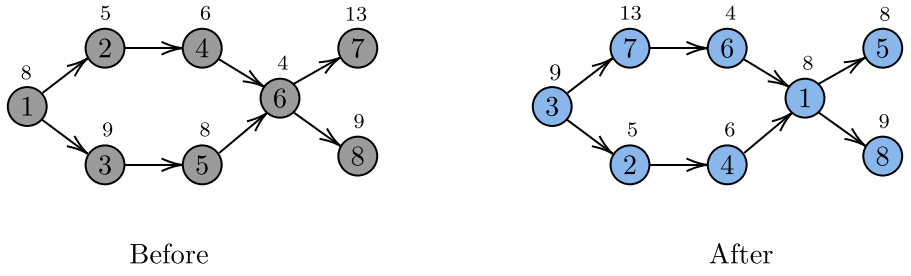

Fig. 4. Precedence graph modification.

Table 1
Enumeration of three series.

Series	1	2	3
OS	≈ 0.2	≈ 0.6	≈ 0.9

unmodified precedence graphs. As a result, this gives no insight on the complexity of the unsolved instances. This is not the case for the average GAP values provided in Table 2.

5.3. Performance evaluation of the heuristic approach

In this part, the performance of the HALT-AND-FIX heuristic is analyzed. The heuristic was coded in Python along with the DOcplex library, which is a Python API that allows to use the CPLEX solver. It was tested on the same described above two sets of instances corresponding to $|V| = 50$ and $|V| = 100$ with 2 and 3 products. A solving time limit of 1800 s per instance was set for the heuristic, and the iteration time limit T was fixed to 10 s. Tables 4 and 5 show the comparative results between the exact and the heuristic approaches for the modified and unmodified instances, respectively. For both tables, the fourth (resp. fifth) column represents the average gap from the best found upper bound and the upper bound found by the exact (resp. heuristic) approach. This specific gap is computed as $\mathrm{GAP}^{\mathrm{B}}_{\mathrm{E}} = 100\% \cdot (\mathrm{UB}_{\mathrm{E}} - \mathrm{UB}_{\mathrm{B}})/\mathrm{UB}_{\mathrm{B}}$ for the exact approach and $\mathrm{GAP}^{\mathrm{B}}_{\mathrm{H}} = 100\% \cdot (\mathrm{UB}_{\mathrm{H}} - \mathrm{UB}_{\mathrm{B}})/\mathrm{UB}_{\mathrm{B}}$ for the heuristic, where UB_{E} (resp. UB_{H}) is the upper bound found by the exact (resp. heuristic) approach and $\mathrm{UB}_{\mathrm{B}} = \min\{\mathrm{UB}_{\mathrm{E}}, \mathrm{UB}_{\mathrm{H}}\}$. The average CPU time (over all instances) of the exact approach ($\mathrm{CPU}_{\mathrm{E}}$) and the heuristic ($\mathrm{CPU}_{\mathrm{H}}$) are shown in the sixth and seventh columns, respectively. The last two columns show the average number of task reassignments for the exact (UB_{E}) and the heuristic (UB_{H}) approaches.

Table 4, for unmodified instances, shows that even if the MILP can solve many instances to optimality, the heuristic also manages to provide solutions close to optimum which can be particularly observed for $|P| = 2$ and $|V| = 50$, where the optimal solution was found in 99% of the cases. Moreover, it is clear that when the instance size increases ($|P| = 2$ and $|V| = 100$ or $|P| = 3$ and $|V| = 100$), the heuristic outperforms the exact approach in terms of solution quality and CPU time. For example, for $|P| = 3$ and $|V| = 50$, the HALT-AND-FIX heuristic found the best solutions in 77% of the cases. Also, when dealing with instances corresponding to $|V| = 100$, the same heuristic was able to find the best solutions in 95% of the cases for $|P| = 2$, and 82% for $|P| = 3$ in a significantly less CPU time than the exact approach. An exception where the heuristic is not efficient can be noticed for the instances corresponding to the third series ($OS \approx 0.9$). This is mainly due to the fact that these instances have dense precedence graphs which decreases the number of feasible solutions and conducts the heuristic to local optima of poor quality. Nonetheless, comparing the average UB_{E} with the average UB_{H} for these instances shows that the heuristic is still able to provide solutions for which the number of reassigned tasks is not considerably larger than those found by the exact approach. Namely, for $|P| = 2$, $|V| = 100$ and $OS = 3$ the difference between the average UB_{E} and the average UB_{H} is 0.57, and for $|P| = 3$, $|V| = 100$ and $OS = 3$ it is equal to 3.93.

Table 5 shows that, for modified instances, the heuristic approach is mostly better than the exact solution of the MILP model with CPLEX in most of cases. For the case of $|P| = 2$ and $|V| = 50$, where the exact approach was able to solve to optimality 77% of the instances, the heuristic showed promising results. Indeed, in 55% of the cases, the heuristic was able to find the same optimal solution in less than 30 s only. For the remaining instances, the heuristic provided solutions with an average $\mathrm{GAP}^{\mathrm{B}}_{\mathrm{H}}$ of 5.6%. When $|P| = 3$ and $|V| = 50$, the heuristic approach provided good quality solutions compared to the exact approach in a considerably shorter CPU time.

For better understanding, Figs. 5–7 respectively show how the average gap from the best found upper bound, noted as $\mathrm{GAP}^{\mathrm{B}}_{(.)}$, the

Table 2
Computational results for the MILP model on modified instances.

$\|P\|$	$\|V\|$	OS	#Instances	#OPT	Avg. CPU, (s.)	Avg. GAP, (%)	# Workstations
2	50	1	15	4	461.56	32.28	15 – 28
		2	219	161	274.76	9.34	13 – 32
		3	74	74	14.29	⊕	14 – 30
	100	1	39	0	⊖	85.66	29 – 54
		2	219	0	⊖	39.01	27 – 57
		3	74	6	824.97	12.77	31 – 56
Global			640	245 (38.28%)	212.60	34.55	13 – 57
3	50	1	14	0	⊖	72.02	18 – 28
		2	218	22	899.92	20.26	15 – 32
		3	73	70	209.58	9.42	15 – 30
	100	1	38	0	⊖	89.29	29 – 54
		2	218	0	⊖	53.54	29 – 57
		3	73	0	⊖	24.92	31 – 56
Global			634	92 (14.51%)	374.66	39.97	15 – 57

(⊕) All optimal solutions were found within the time limit of 1800 s.
(⊖) No optimal solution was found within the time limit of 1800 s.

Table 3
Computational results for the MILP model on unmodified instances.

$\|P\|$	$\|V\|$	OS	#Instances	#OPT	Avg. CPU, (s.)	Avg. GAP, (%)	# Workstations
2	50	1	15	15	1.28	⊕	15 – 28
		2	219	217	9.36	50.00	13 – 32
		3	74	74	5.57	⊕	14 – 30
	100	1	39	38	210.95	100.00	29 – 54
		2	219	218	121.17	97.96	27 – 57
		3	74	59	139.77	85.16	31 – 56
Global			640	621 (97.03%)	72.69	82.91	13 – 57
3	50	1	14	14	7.79	⊕	18 – 28
		2	218	198	125.59	63.46	15 – 32
		3	73	72	68.71	78.57	15 – 30
	100	1	38	19	786.27	96.14	29 – 54
		2	218	124	848.84	94.82	29 – 57
		3	73	2	613.80	93.27	31 – 56
Global			634	429 (67.66%)	352.78	91.27	15 – 57

(⊕) All optimal solutions were found within the time limit of 1800 s.

Table 4
MILP vs. Heuristic on unmodified instances.

$\|P\|$	$\|V\|$	OS	Avg. GAP^{B}_{E}, (%)	Avg. GAP^{B}_{H}, (%)	Avg. CPU_{E}, (s.)	Avg. CPU_{H}, (s.)	Avg. UB_{E}	Avg. UB_{H}
2	50	1	0.00	0.00	1.28	1.27	0.00	0.00
		2	0.00	0.57	25.71	3.56	0.05	0.07
		3	0.00	1.58	5.57	3.39	1.72	1.77
	100	1	2.56	0.00	251.70	24.16	0.15	0.00
		2	0.46	0.00	128.84	34.18	0.22	0.00
		3	0.00	15.84	476.67	60.51	1.08	1.65
3	50	1	0.00	0.00	7.79	4.33	0.00	0.00
		2	0.52	9.16	279.26	12.58	0.73	0.94
		3	0.68	21.08	68.13	9.88	6.22	7.90
	100	1	50.00	0.00	1293.13	206.38	8.95	0.00
		2	43.21	0.84	1259.03	318.33	10.13	0.46
		3	8.49	47.95	1769.78	192.19	10.97	14.90

average CPU time, and the average number of reassigned tasks vary with respect to the OS for both the exact approach (blue colored round markers) and the heuristic one (red colored square markers). In each figure, the graph on the left hand-side corresponds to the instances of 50 tasks and the graph on the right hand-side is for those with 100 tasks. Based on Fig. 5, one can confirm that the heuristic showed good performance in terms of average $GAP^{B}_{(.)}$. This is confirmed by Fig. 6, which shows that the average number of task reassignments is almost the same for both $|V| = 50$ and $|V| = 100$. This being said, it is worth mentioning based on Fig. 7 that the heuristic found these solutions in a much less CPU time compared to the exact approach.

6. Conclusion and perspectives

This paper studies a balancing problem for reconfigurable production lines with a fixed number of workstations and several different products. The aim is to find for each product the best feasible line configuration so as to minimize the total number of task reassignments between workstations when production changes. Each product line configuration is subject to a given cycle time and corresponding precedence constraints.

To solve this problem, we proposed a MILP model enhanced with two manners of computing task assignment intervals. An efficient MILP-based iterative heuristic, named as Halt-and-Fix, was elaborated as

Table 5
MILP vs. Heuristic on modified instances.

$\|P\|$	$\|V\|$	OS	Avg. GAP^{B}_{E}, (%)	Avg. GAP^{B}_{H}, (%)	Avg. CPU_{E}, (s.)	Avg. CPU_{H}, (s.)	Avg. UB_{E}	Avg. UB_{H}
2	50	1	0.44	7.47	1443.15	27.01	10.80	11.47
		2	0.08	2.22	671.13	14.42	22.71	23.20
		3	0.00	0.42	14.29	3.68	34.78	34.93
	100	1	18.11	0.20	1800	1698.05	44.59	38.38
		2	2.07	2.94	1800	545.00	56.31	56.71
		3	0.29	3.05	1721.31	342.47	76.55	78.72
3	50	1	8.10	0.97	1800	375.41	33.14	30.93
		2	1.50	2.52	1709.45	162.75	51.39	51.84
		3	0.02	1.66	275.03	21.54	71.36	72.52
	100	1	5.72	3.01	1800	1800	131.92	130.66
		2	6.29	1.41	1800	1756.91	141.40	135.85
		3	2.63	0.49	1800	1139.93	166.10	162.79

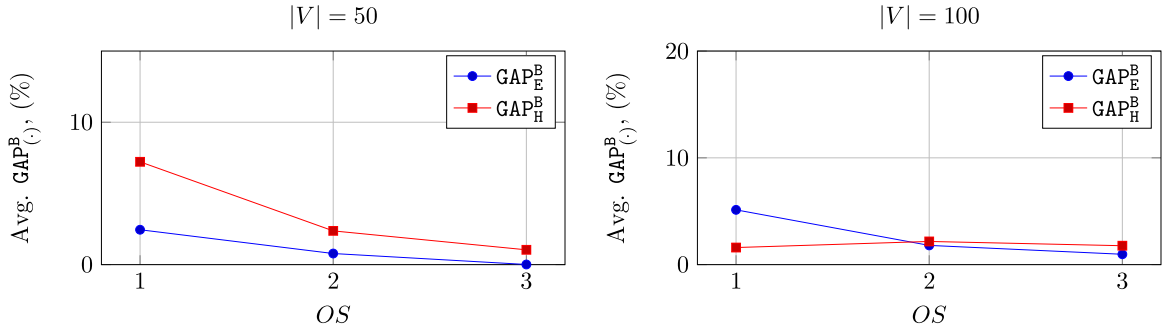

Fig. 5. Results comparison between GAP^{B}_{E} and GAP^{B}_{H} for modified instances.

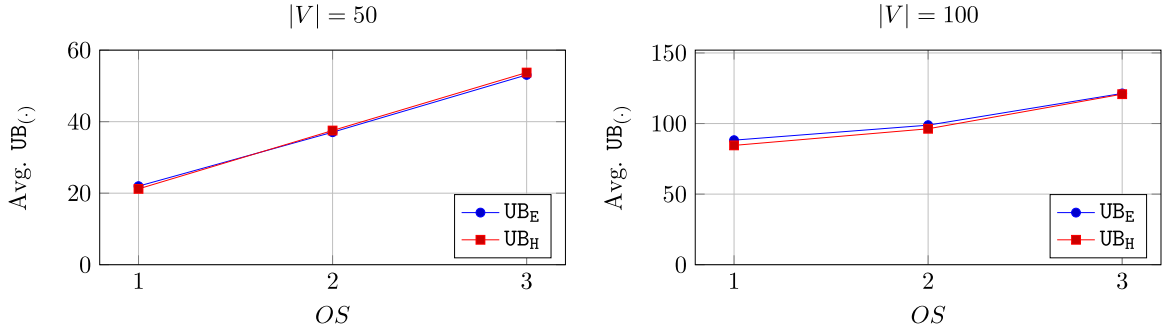

Fig. 6. Avg. number of task reassignments comparison between the MIP solver and the Halt-and-Fix heuristic for modified instances.

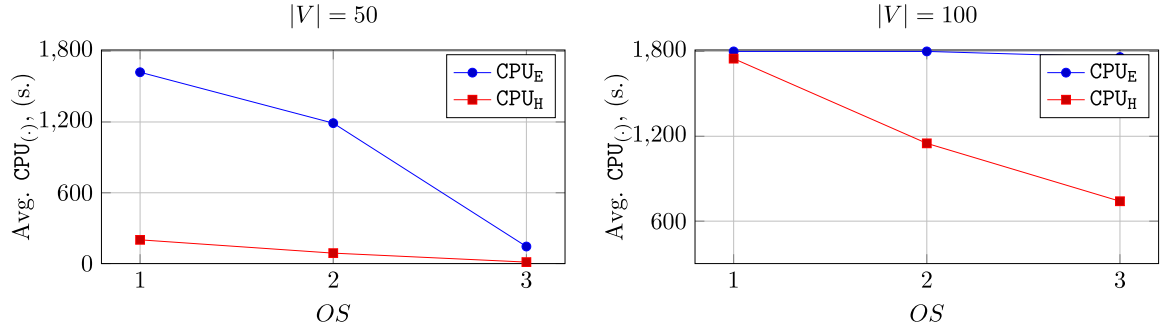

Fig. 7. Avg. CPU comparison between the MIP solver and the Halt-and-Fix heuristic for modified instances.

well. Numerical results showed that the developed heuristic approach provides very competitive results and outperforms the MILP model sole solved by CPLEX for large scale instances.

A first natural extension of our work can be to explore other possible ways to further improve the performance of the Halt-and-Fix heuristic. For example, a simple heuristic, such as the one developed by Hoffman (1963), can be adapted to generate a first feasible line configuration for each product. This solution can then be used as a warm-start by the Halt-and-Fix heuristic. Moreover, a solver-free heuristic or meta-heuristic can be developed to tackle large size instances with more

than 3 products. It would be also interesting to adapt it to situations where no assumptions are made on the sequencing of products. Another possible research direction consists of integrating the design of reconfigurable lines with production planning decisions such as product lot-sizing and scheduling. It was shown that such integration of decision can lead to considerable cost savings (*e.g.*, Chopra and Meindl (2016)).

CRediT authorship contribution statement

Abdelkrim R. Yelles-Chaouche: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Visualization. **Evgeny Gurevsky:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing – review & editing, Supervision. **Nadjib Brahimi:** Validation, Formal analysis, Writing – review & editing, Supervision. **Alexandre Dolgui:** Review & editing, Supervision, Project administration, Funding acquisition.

Data availability

Data will be made available on request.

References

Akpinar, S., Elmi, A., & Bektaş, T. (2017). Combinatorial benders cuts for assembly line balancing problems with setups. *European Journal of Operational Research*, *259*(2), 527–537.

Ashraf, M., & Hasan, F. (2018). Configuration selection for a reconfigurable manufacturing flow line involving part production with operation constraints. *International Journal of Advanced Manufacturing Technology*, *98*(5–8), 2137–2156.

Asl, A. J., Solimanpur, M., & Shankar, R. (2019). Multi-objective multi-model assembly line balancing problem: a quantitative study in engine manufacturing industry. *Opsearch*, *56*(3), 603–627.

Battaïa, O., Dolgui, A., & Guschinsky, N. (2016). Decision support for design of reconfigurable rotary machining systems for family part production. *International Journal of Productions Research*, *55*(5), 1368–1385.

Battaïa, O., Dolgui, A., & Guschinsky, N. (2020). Optimal cost design of flow lines with reconfigurable machines for batch production. *International Journal of Productions Research*, *58*(10), 2937–2952.

Bautista, J., Batalla-García, C., & Alfaro-Pozo, R. (2016). Models for assembly line balancing by temporal, spatial and ergonomic risk attributes. *European Journal of Operational Research*, *251*(3), 814–829.

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, *168*(3), 694–715.

Benderbal, H. H., Dahane, M., & Benyoucef, L. (2017). Flexibility-based multi-objective approach for machines selection in reconfigurable manufacturing system (RMS) design under unavailability constraints. *International Journal of Productions Research*, *55*(20), 6033–6051.

Bensmaine, A., Benyoucef, L., & Dahane, D. (2013). A non-dominated sorting genetic algorithm based approach for optimal machines selection in reconfigurable manufacturing environment. *Computers & Industrial Engineering*, *66*(3), 519–524.

Bensmaine, A., Benyoucef, L., & Dahane, D. (2014). A new heuristic for integrated process planning and scheduling in reconfigurable manufacturing systems. *International Journal of Productions Research*, *52*(12), 3583–3594.

Berger, I., Bourjolly, J.-M., & Laporte, G. (1992). Branch-and-bound algorithms for the multi-product assembly line balancing problem. *European Journal of Operational Research*, *58*(2), 215–222.

Borisovsky, P. A., Delorme, X., & Dolgui, A. (2013). Genetic algorithm for balancing reconfigurable machining lines. *Computers & Industrial Engineering*, *66*(3), 541–547.

Bortolini, M., Galizia, F. G., & Mora, C. (2018). Reconfigurable manufacturing systems: Literature review and research trend. *Journal of Manufacturing Systems*, *49*, 93–106.

Chen, J. C., Chen, Y.-Y., Chen, T.-L., & Kuo, Y.-H. (2019). Applying two-phase adaptive genetic algorithm to solve multi-model assembly line balancing problems in TFT–LCD module process. *Journal of Manufacturing Systems*, *52*, 86–99.

Chopra, S., & Meindl, P. (2016). *Supply chain management: strategy, planning and operation* (6th ed.). New Jersey: Person Prentice Hall.

Dou, J., Li, J., & Su, C. (2016). Bi-objective optimization of integrating configuration generation and scheduling for reconfigurable flow lines using NSGA-II. *International Journal of Advanced Manufacturing Technology*, *86*(5–8), 1945–1962.

van Zante-de Fokkert, J. I., & de Kok, T. G. (1997). The mixed and multi model line balancing problem: a comparison. *European Journal of Operational Research*, *100*(3), 399–412.

Goyal, K. K., & Jain, P. K. (2015). Design of reconfigurable flow lines using MOPSO and maximum deviation theory. *International Journal of Advanced Manufacturing Technology*, *84*, 1587–1600.

Grangeon, N., Leclaire, P., & Norre, S. (2011). Heuristics for the re-balancing of a vehicle assembly line. *International Journal of Productions Research*, *49*(22), 6609–6628.

Hoffman, T. R. (1963). Assembly line balancing with a precedence matrix. *Management Science*, *9*(4), 551–562.

Johnson, R. V. (1988). Optimally balancing large assembly lines with FABLE. *Management Science*, *34*(2), 240–253.

Katz, R. (2006). Design principles of reconfigurable machines. *International Journal of Advanced Manufacturing Technology*, *34*(5–6), 430–439.

Koren, Y. (2006). General RMS characteristics. comparison with dedicated and flexible systems. In A. Dashchenko (Ed.), *Reconfigurable manufacturing systems and transformable factories* (pp. 27–45). Springer Berlin Heidelberg.

Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., & Brussel, H. V. (1999). Reconfigurable manufacturing systems. *CIRP Annals*, *48*(2), 527–540.

Koren, Y., & Shpitalni, M. (2010). Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, *29*(4), 130–141.

Lapierre, S., Debargis, L., & Soumis, F. (2000). Balancing printed circuit board assembly line systems. *International Journal of Productions Research*, *38*(16), 3899–3911.

Makssoud, F., Battaïa, O., & Dolgui, A. (2014). An exact optimization approach for a transfer line reconfiguration problem. *International Journal of Advanced Manufacturing Technology*, *72*(5–8), 717–727.

Mehrabi, M., Ulsoy, A., & Koren, Y. (2000). Reconfigurable manufacturing systems: key to future manufacturing. *Journal of Intelligent Manufacturing*, *11*(4), 403–419.

Nazarian, E., Ko, J., & Wang, H. (2010). Design of multi-product manufacturing lines with the consideration of product change dependent inter-task times, reduced changeover and machine flexibility. *Journal of Manufacturing Systems*, *29*(1), 35–46.

Otto, A., Otto, C., & Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. *European Journal of Operational Research*, *228*(1), 33–45.

Patterson, J. H., & Albracht, J. J. (1975). Assembly-line balancing: Zero-one programming with Fibonacci search. *Operations Research*, *23*(1), 166–172.

Pereira, J. (2018). Modelling and solving a cost-oriented resource-constrained multi-model assembly line balancing problem. *International Journal of Productions Research*, *56*(11), 3994–4016.

Sancı, E., & Azizoğlu, M. (2017). Rebalancing the assembly lines: exact solution approaches. *International Journal of Productions Research*, *55*(20), 5991–6010.

Saxena, L. K., & Jain, P. K. (2012). A model and optimisation approach for reconfigurable manufacturing system configuration design. *International Journal of Productions Research*, *50*(12), 3359–3381.

Scholl, A. (1999). *Balancing and sequencing of assembly lines* (2nd ed.). Physica-Verlag Heidelberg.

Tóth, A., Knuutila, T., & Nevalainen, O. (2018). Machine configuration and workload balancing of modular placement machines in multi-product PCB assembly. *International Journal of Computer Integrated Manufacturing*, *31*(9), 815–830.

Touzout, F., & Benyoucef, L. (2018). Multi-objective sustainable process plan generation in a reconfigurable manufacturing environment: exact and adapted evolutionary approaches. *International Journal of Productions Research*, *57*(8), 1–17.

Yang, C., Gao, J., & Sun, L. (2013). A multi-objective genetic algorithm for mixed-model assembly line re-balancing. *Computers & Industrial Engineering*, *65*(1), 109–116.

Yelles-Chaouche, A. R., Gurevsky, E., Brahimi, N., & Dolgui, A. (2021). Reconfigurable manufacturing systems from an optimisation perspective: a focused review of literature. *International Journal of Productions Research*, *59*(21), 6400–6418.

Zhang, F., Zhang, Y. F., & Nee, A. Y. C. (1997). Using genetic algorithms in process planning for job shop machining. *IEEE Transactions on Evolutionary Computation*, *1*(4), 278–289.