Michels, Adalberto Sato; Lopes, Thiago Cantos; Magatão, Leandro

Article

An exact method with decomposition techniques and combinatorial Benders' cuts for the type-2 multi-manned assembly line balancing problem

Operations Research Perspectives

Provided in Cooperation with:
Elsevier

Suggested Citation: Michels, Adalberto Sato; Lopes, Thiago Cantos; Magatão, Leandro (2020) : An exact method with decomposition techniques and combinatorial Benders' cuts for the type-2 multi-manned assembly line balancing problem, Operations Research Perspectives, ISSN 2214-7160, Elsevier, Amsterdam, Vol. 7, pp. 1-16, https://doi.org/10.1016/j.orp.2020.100163

This Version is available at:
https://hdl.handle.net/10419/246431

Operations Research Perspectives 7 (2020) 100163

Contents lists available at ScienceDirect

Operations Research Perspectives

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

An exact method with decomposition techniques and combinatorial Benders' cuts for the type-2 multi-manned assembly line balancing problem

Adalberto Sato Michels, Thiago Cantos Lopes, Leandro Magatão*

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI)Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, Brazil

ARTICLE INFO

Keywords:
Multi-manned assembly line balancing
Cycle time minimization
Combinatorial Benders' cuts
Mixed-integer linear programming

ABSTRACT

Multi-manned assembly lines are widely applied to manufacturing industries that produce large-size products and are concerned with high levels of productivity. Such lines are commonly found in automotive industries, where different tasks are simultaneously performed by more than one worker on the same product in multi-operated stations, giving rise to a class of balancing problem that aims to minimize the line's cycle time. This clear practical application had made the type-2 multi-manned assembly line balancing problem to be explored in the past. However, only few small-size instances could be solved by preceding exact solution approaches, whereas large and real-life cases still lack optimality proofs since they were tackled by heuristics. In this work, a new Mixed-Integer Linear Programming model is presented and its modeling decisions discussed. Moreover, an innovative exact solution procedure employing a combination of decomposition techniques and combinatorial Benders' cuts is presented to solve large and real-life instances optimally. Tests on an extended literature dataset and a real-life assembly plant case study have demonstrated that the proposed algorithm outperforms previously developed methods in terms of solution quality by an ample margin in efficiency gains. Synergies between the algorithm's components are also revealed. Finally, the proposed exact method has been able to yield 60 optimal results out of a 108-instance dataset, with the remaining 48 solutions presenting a small integer gap (less than 2%).

1. Introduction

Assembly lines are production systems usually applied to large-scale industries of similar and standardized products. They are frequently built to fit homogeneous products and enable their mass production, hence generally designed as flow-shop layouts. In the academic research, the Assembly Line Balancing Problem (ALBP) is widely discussed in the literature, which is the main combinatorial problem that assembly lines have created [4].

In its simplest form, the Simple Assembly Line Balancing Problem (SALBP) consists in assigning a list of tasks subjected to a precedence graph to stations, whilst considering various simplification hypotheses described by Baybars [6]. One of them is to solely allow one worker to operate each station. Furthermore, a serial, straight line organization is imposed to the stations. In turn, such line produces a single product model. The most common goals in a SALBP are (i) minimizing the number of stations (SALBP-1) or (ii) minimizing the cycle time (SALBP-2) and important contributions to its literature was surveyed by Scholl and Becker [40]. Among those, algorithmic solution methods were proposed and related to practical applications: developed by Scholl and Klein [41], SALOME is an effective procedure based on a bidirectional branch-and-bound, while Bautista and Pereira [5] proposed a dynamic programming approach, which were followed by Sewell and Jacobson [42] and their branch, bound, and remember algorithm, as well as a multi-Hoffmann heuristic with enhanced properties proposed by Sternatz [44]. For an overview and improved techniques for SALBP-1, it is possible to refer to Pape [33].

Notwithstanding, by revisiting the "only one worker per station" hypothesis applied to SALBPs, one can verify that it often is not a practical limitation on, for instance, manufacturing lines found in automotive factories. These industries regularly produce cars, buses, and trucks, which are large-size products, broadly speaking. Because the physical area occupied by stations are quite large compared to an operator's working space, multiple workers can simultaneously perform tasks on the same product. Accordingly, more generalized extensions of the SALBP come to light as natural extensions: the Multi-manned Assembly Line Balancing Problem (MALBP) and the Two-sided Assembly Line Balancing Problem (TALBP). Both problems, along with an assortment of other practical extensions, are surveyed by Becker and Scholl [7].

* Corresponding author.
E-mail addresses: amichels@alunos.utfpr.edu.br (A.S. Michels), thiagolopes@alunos.utfpr.edu.br (T.C. Lopes), magatao@utfpr.edu.br (L. Magatão).

https://doi.org/10.1016/j.orp.2020.100163
Received 3 April 2020; Received in revised form 27 July 2020; Accepted 30 September 2020
Available online 02 October 2020

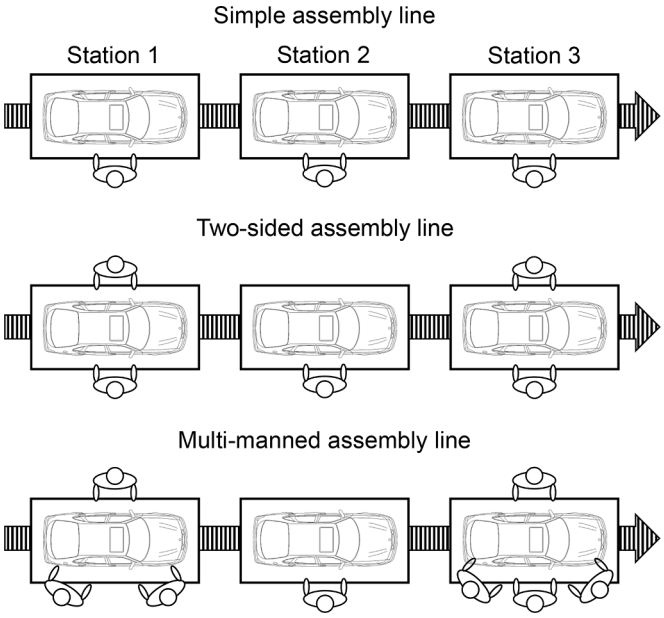


Fig. 1. Layout examples of simple, two-sided, and multi-manned assembly lines.

These above mentioned lines are illustrated in Fig. 1. It exhibits examples of assembly lines composed of three stations each: simple, two-sided, and multi-manned assembly lines are respectively depicted. As observed, both two-sided and multi-manned assembly lines admit multiple workers in each station (first and third stations), with workers simultaneously performing tasks on the same product. Notice, however, that MALBPs are more flexible than TALBPs in terms of the quantity of workers and their positioning. The main difference is that two-sided lines allow at most one operator in each side of the station (right and left), whereas multi-manned lines have their maximum number of workers set depending on product's attributes. These can be seen as size, structure, and tasks' specificities. The last divergence is that TALBPs may have to deal with tasks exclusively performed on the right or left side of the product. Both of these problems can also be solved considering analogous goal functions studied for SALBP [7].

This work focuses on the MALBP-2 variant, namely cycle time minimization. A previous MALBP-1 specialized algorithm [28], when adapted, was not capable to satisfactorily solve the proposed problem, not even generating feasible solutions for most large-size instances, and the best-known meta-heuristic for the MALBP-2 [35] produces several sub-optimal solutions. Therefore, the main contribution to the state-of-the-art herein proposed comes from the implementation of an innovative exact algorithm to optimally solve the MALBP-2 dataset. Some relevant simplification hypotheses from SALBP-2 mentioned in Baybars [6] are kept, viz. a serial, straight line is considered and such line produces a single, unique model. The remaining of the paper is organized as follows. In Section 2, the relevant literature is introduced, focusing on specific objective functions, extensions, and methodologies used by each author. In order to define the problem, Section 3 precisely explains the characteristics of a type-2 MALBP, demonstrating advantages associated to obtainable production rate improvement and line length reduction when using a multi-manned configuration. Section 4 presents the MILP model, valid inequalities, upper and lower bounds, and a brief discussion on modeling decision. Section 5 describes the development and applications of the proposed Benders' decomposition algorithm and combinatorial cuts, as well as how an initial solution is constructed for the problem at hand. Computational results retrieved from a benchmark dataset and a real-life assembly plant case study are presented and discussed in Section 6. Finally, in Section 7, a summary of the concluding remarks and further research directions are described.

2. Literature review

As a consequence of the intense use of multi-operated stations in industrial environments, several studies regarding MALBPs and TALBPs have been elaborate in the literature. To the best of the authors' knowledge, Akagi et al. [1] is the first study to take on the problem of allowing more than one worker per station in assembly lines, while attaining good production rates. For that, an approach denominated Parallel Assignment Method (PAM) was developed.

Much later, a heuristic method to address such problem was introduced by Dimitriadis [12], which was based on altering a procedure previously created by Hoffmann [16]. Given a fixed cycle time, it has been shown that the heuristic was useful in enhancing stations physical utilization. The objective of minimizing the total number of workers and stations was considered, which turns out to be the most customary goal function exploited in subsequent works since then.

Following those pioneer publications, the attention on MALBPs have been growing in the last ten years. Becker and Scholl [8] proposed the Assembly Line Balancing Problem with Variable Workplaces (VWALBP), in which a cycle time is given and working areas are minimized. Here the product is fragmented into mounting positions, while imposing in each multi-manned station that only one worker is capable of assembling them. A Mixed-Integer Linear Programming (MILP) formulation is modeled in conjunction with lower bounding techniques. Besides, a branch-and-bound algorithm based on SALOME – called VWSolver – is developed to solve large instances. In sequence, a novel efficient algorithm was implemented by Kellegöz and Toklu [19] to tackle ALBPs with multi-manned stations (Jumper), which is also based on branch-and-bound procedures. Jumper outperforms VWSolver in both computational processing times and quality of feasible solutions.

Fattahi and Roshani [13] were the first to propose a mathematical formulation that simultaneously minimizes the total number of workers and stations in a MALBP, defining the type-1 variant. In the optimization procedure, minimizing the number of workers is the primary objective and minimizing the number of stations is the secondary one. Keeping the remaining SALBP hypotheses, their model failed in solving medium and large cases, but was able to solve small-size test problems in an acceptable time limit. To settle this issue, they have developed an Ant Colony Optimization (ACO) algorithm. The ACO algorithm could find the same optimal solutions for small-size problems in a much reduced computational time, as well as feasible and near-optimal solutions for many medium and large instances. Later, Kellegöz [18] has created a better MALBP-1 mathematical model proposed by Fattahi and Roshani [13]. Additionally, a Gantt-based heuristic was developed within a Simulated Annealing (SA) framework. This procedure is able to solve medium and large-size instances, finding improved feasible solutions to a large number of instances in the tested benchmark. Therefore, it has been concluded that the GSA algorithm outperforms the ACO algorithm given by Fattahi and Roshani [13].

The examination of assembly lines with multi-operated stations regarding cost-oriented problem instances has been conducted by Kazemi and Sedighi [17] and Michels et al. [27] on real-size case studies. The former presents a heuristic method based on Genetic Algorithm (GA) that takes into account the objective of minimizing costs per production unit, whereas the latter analyzes a robotic assembly line that considers the employment of multiple robotic workers per station, conceiving robots, facilities, and tool prices. Hence, an MILP model to minimize implementation costs for the line design is developed.

A multi-objective function to address MALBPs is firstly proposed by Roshani et al. [37]. By maximizing line efficiency and minimizing the line length and smoothness index, a mathematical model is formulated.

Furthermore, an improved SA algorithm has been developed to tackle the problem. Meanwhile, a constructive heuristic based on priority rules followed by an improvement procedure based on GA has been presented by Kellegöz and Toklu [20]. Computational experiments have been conducted on MALBP instances to minimize the total number of workers in the line. At this point, Yilmaz and Yilmaz [48] have created a new mathematical formulation, aiming at the minimization of total number of workers, stations, as well as workload difference between workers. Subsequently, Yilmaz and Yilmaz [49] also studied the impacts of equipment needs and skilled workers on MALBPs. In order to solve the problem, a heuristic procedure was proposed.

For a given fixed number of stations, Roshani and Giglio [35] approached the MALBP by attempting to minimize the cycle time and the number of workers in a line as the primary and secondary objective, respectively. This strategy is further discussed in Section 4.3, as it neglects fundamental MALBP hypotheses regarding the higher importance of workers over stations, flexibility gains over SALBP solutions, and conditions found in practice. Alongside the MILP model, two metaheuristics, the indirect and direct SA algorithm (ISA and DSA, respectively), were implemented. In both terms of solution quality and computational time. Therefore, the DSA showed to be more efficient, with a better performance in solving the problem. Moreover, another SA algorithm was developed by Roshani and Nezami [36], this one undertakes the mixed-model variant of MALBP-1.

More recently, both Naderi et al. [30] and Michels et al. [28] developed and applied Benders' decomposition algorithms (BDA) [9] with combinatorial Benders' cuts (CBC) [11] to the MALBP-1 variant, reaching outstanding results. The former is able to solve a realistic five-sided MALBP with moving workers and limited workspace, whereas the latter solves various medium and large-size MALBP-1 instances optimally, proving the optimality of some solutions reported by Kellegöz [18] and improving many others in terms of solution quality. The Benders' decomposition method aspires to reformulate an original and complete monolithic model into a master problem (MP) and slave (or sub) problems (SP), transforming them into two hierarchical problems. This partition aims at removing the burden of several variables and restrictions from the MP, transferring part of the load to one or more SPs, which are then solved individually. By iteratively working between the MP and SPs, solutions can be found and evaluated much faster: values for key variables are fixed in a given solution of the MP, then used in the SPs to define the remaining ones. Each MP candidate solution is generally decomposable into multiple smaller problems, which is translated into a set of SPs that are solved individually, supposedly reducing the computational burden when compared to straightforwardly solving the monolithic problem. Each SP yields a solution that can be used to inform the MP of infeasible combinations: these are known as CBC. The MP with these added cutting planes (extra CBC restrictions) is then solved and the procedure iterates with the next result. Specifically, this decomposition method strips off complicated variables from the MP, then repeatedly corrects misled solutions by solving the SPs [28]. These corrections are made by finally applying CBC to the MP [11], pointing out infeasibilities caused by parts that were omitted in it. As BDAs are highly dependent on the problem being dealt with, straightforward adaptations of previous algorithms rarely work as well as intended. Thus, authors generally develop specific algorithms to their problems, incorporating as much particular information as possible from the problem under study.

Rahmaniani et al. [34] surveys and summarizes the use of BDAs in the literature, where only one work concerning a transfer line balancing problem is listed in the review [32]. Nonetheless, it was verified by further investigation that, besides the aforementioned works, various real-world problems have been approached by using BDAs. However, related research was scarcely developed for assembly line balancing problems. In Hazir and Dolgui [14] and Hazir and Dolgui [15], different layouts (straight lines and U-lines) are considered under uncertainty. A robust optimization model and a specialized algorithm are formulated for each case. Lastly, Akpinar et al. [2] task-dependent set-up times are taken into account. In this problem, task assignment and task sequencing decisions can be interpreted as hierarchical problems.

Finally, other recent works worth mentioning on MALBP variants are: Sahin and Kellegöz [38], which considered resource investment and developed a hybrid heuristic based on Particle Swarm Optimization (PSO) to tackle the problem; Sahin and Kellegöz [39] took into account the possibility of walking workers [43] in the line and solved large instances with a reduced integer gap by applying an Electromagnetic Field Optimization (EFO) algorithm to the problem; Lopes et al. [22] pointed out inconsistencies in a previously published paper [10] and proposes a model based hierarchical decomposition procedure to minimize the number of stations after the task-worker assignment solution has been decided in a multi-manned assembly line; Lopes et al. [23] brought more flexibility to multi-manned lines by explicitly considering continuous paced line control, relaxing the limitation of fixed, discrete, and restrictive frontiers between stations. Their model, heuristic procedure, and algorithmic lower bounds accommodate significantly shorter line lengths; Yilmaz and Yilmaz [51] proposed a mathematical model to consider assignment restrictions, i.e. positive and negative zoning, distance, station, and synchronous task restrictions. A tabu-search algorithm is also developed to effectively solve the problem; and Yadav et al. [46] improved worker and station efficiencies of a real-life automotive plant by reconfiguring the assembly line under study.

In summary, none of the reviewed works but Roshani and Giglio [35] considered the type-2 MALBP. Nevertheless, by using SA algorithms that would not guarantee optimality, they obtained sub-optimal solutions with lower levels of efficiency than its SALBP-2 counterpart. In order to bridge this gap, a new mathematical formulation with different modeling decisions is developed with valid inequalities to address the problem. Furthermore, a solution method for the problem under study is proposed. It combines the decomposition technique introduced by Lopes et al. [22] to find an initial assignment with a BDA as the strategy to find and prove optimal solutions for MALBP-2 instances. Larger benchmark instances can be solved to optimality by applying CBC [9,11] as lazy constraints while the algorithm is executing. Unlike classical Benders' decompositions with linear SPs [9], the proposed algorithm presents integer slave problems [11], which are intended for feasibility seeking, as in Michels et al. [28] and Naderi et al. [30]. Besides Roshani and Giglio [35] being the only work concerning MALBP-2, they likewise supply an comprehensive dataset to validate both the proposed model and algorithm. Therefore, these instances are provided as a benchmark for this work and the solution method herein proposed, which defines the MALBP-2 and focus on minimizing the cycle time as the primary objective and the number of stations as the secondary given a fixed number of worker. In this way, objective function results can be directly compared to evaluate the algorithm's performance.

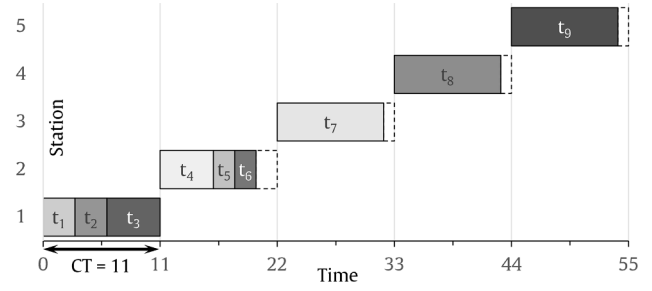
3. Problem statement

As aforementioned, multi-manned stations from assembly lines examined in this study are employed to large-scale production of a single model of large products. Such stations are sequentially positioned in a straight, serial line. Work-pieces can just be processed one at a time in each station. These pieces move forward between stations within a cycle time (CT) to be optimized (minimized), whilst their transfer times between stations are neglected. As a single product is produced in the line, its pace – and consequently its production rate – is entirely determined by the most loaded station [6].

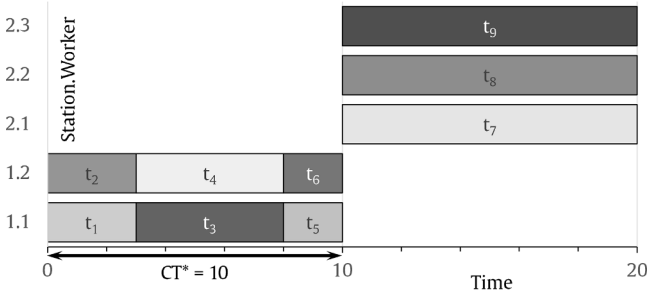
A set of indivisible tasks T must be performed in order to assemble any product. Since precedence relations must be respected, they cannot be executed in an arbitrary order. Tasks have a deterministic processing duration time (D_t) to be finished. Therefore, the sum of task processing times assigned to the same worker must not surpass the cycle time limit,

Table 1
Task duration times and precedence relations for the illustrative instance.

Task	1	2	3	4	5	6	7	8	9
Duration	3	3	5	5	2	2	10	10	10
Precedence	–	–	1,2	1,2	3,4	3,4	5	5,6	6



(a) SALBP-2 solution



(b) MALBP-2 solution

Fig. 2. Gantt diagram representations of simple and multi-manned optimal solutions for the illustrative instance.

which is imposed by the most loaded worker. However, the possibility of parallel work within each station makes it essential to schedule tasks in such a manner that precedence restrictions are still respected.

In Table 1, an illustrative instance is sampled to depict differences between SALBP and MALBP possible results. This illustrative instance will be used as a numerical example throughout the paper to explain the proposed method. For the considered precedence graph, task index numbers are ordered on the first line, their respective durations are represented by the value just below it, and their direct predecessor are given in the last line. Additionally, optimal solutions for both SALBP (Fig. 2a) and MALBP (Fig. 2b) versions of this instance are shown in the format of Gantt diagrams, with a defined maximum number of workers in the line to be five ($Nmax = 5$). Differently from multi-manned lines, the total number of workers and stations must be the same for SALBPs. In the former, notwithstanding, each work-piece is allowed to be engaged by more than one worker employed in the same station simultaneously. Due to product sizes, the maximum number of workers that a station can suit to perform different tasks concomitantly (NW) may vary. In this example, $NW = 3$.

The SALBP's solution requires 5 workers (represented by bars) assigned to 5 stations (on the y-axis) to deliver a production rate (with times on x-axis) of one product unit each 11 time units (i.e. $CT_{SALBP} = 11$) in its optimal configuration. Represented by dashed blank spaces in each station, idle times total 5 time units along the line, which translates to approximately 9.09% of the line's available task performing time. On the other hand, by permitting simultaneous operations within stations by different workers, the MALBP optimal solution could, at the same time, reduce the line length from 5 to 2 stations and improve the production rate to a cycle time of 10 time units ($CT_{MALBP} = 10$) instead of 11 to perform the same task set. Hence, an advantage of multi-manned lines over simple ones was demonstrated: the additional flexibility allowed idle time to be reduced to zero. Naturally, these improvements depend on the instance's parameters, but it is clear that efficiency gains can occur at no extra or even reduced cost when multiple workers are allowed to perform different tasks at the same time within a station.

Nevertheless, tasks must still be scheduled within each station to ensure that precedence relations are respected. These task-scheduling requirements imply on a computational burden to solve the problem, which inevitably comes along as a drawback by allowing such flexibility to acquire the presented viable advantage. Notice that, into the same station, at most one task can be executed by each worker at a given time, whilst cooperation between workers is forbidden, which means that common tasks are not considered [43,47]. Moreover, positioning and zoning restrictions do not constrain task assignments [3,8], and interferences between workers do not happen during the assemblage process [24] as long as precedence relations are respected. Workers are homogeneous [29], meaning all of them have the same regular capability and can perform any task with the same specific time duration required for its execution. As stated, even though different workers can perform different tasks synchronously, all precedence relations enforced by the precedence graph still must be satisfied. Consequently, for each station a task scheduling problem originates, with possible waiting (idle) times for workers between or before the performance of tasks. These must be taken into account, thus making MALBPs more complex and difficult to solve than SALBPs [13].

Due to high line utilization associated to assembly lines, it is assumed that balancing decisions are a long-term plan for the studied problem, which implies in a desire to maximize productivity with the available resources. In this situation, the number of workers that a company can employ is the main fixed resource, as they represent costs related to wages, equipment, labor regulations, among others. Stations and facilities, on the other hand, are a one-time investment that is relatively much cheaper when compared to workers, but should be considered nonetheless [13,18,28]. Hence, the mathematical model presented in Section 4 prioritizes productivity maximization in its objective function, i.e. minimizing cycle time, accompanied by a secondary objective of reducing facility costs, i.e. minimizing the total number of stations and, accordingly, the line length.

Ultimately, solutions that are deemed optimal for SALBPs are necessarily feasible configurations for MALBPs. It follows that the optimal cycle time solution given a fixed number of workers (and stations) achieved by a SALBP can be viewed as an upper bound for its MALBP counterpart, since simple lines are less flexible and restrict the use of workers to only one in each station for any given solution. Correspondingly, the maximum number of stations (NS) in a MALBP can be set to one unit lesser than the optimal solution found by its simpler counterpart. This adopted upper bound is a reasonable measure: as the MALBP-2's goal is to minimize both the cycle time and the total number of stations, with a much lower importance in the latter, the minimal marginal improvement taken from a SALBP solution is keeping the same productivity level while reducing the line length in at least one station. This has been similarly demonstrated by Michels et al. [28] for the MALBP-1 variant. For the illustrative example presented in Fig. 2a and b, a MALBP-2 instance starts with an upper bound for the number of stations equaling four ($NS = 4$). It is explained by the fact that, if the model is incapable of reducing at least one station in the previous solution (accusing the problem to be infeasible), then it is concluded that having the flexibility to allow multiple worker per station does not help in obtaining efficiency improvements at no additional costs, meaning that both versions (SALBP and MALBP) have the same optimal solution in terms of objective value. This reasoning is further discussed in Section 4.3.

4. MILP model

In this section, the developed Mixed-Integer Linear Programming (MILP) model is presented. It represents the type-2 Multi-manned Assembly Line Balancing Problem (MALBP-2), taking into account the characteristics recognized in Section 3. Section 4.1 outlines the main model to express the problem and Section 4.2 shows the implemented constraints that strengthens the problem's linear relaxation (valid inequalities). Furthermore, modeling decisions concerning the objective function and parameter values are discussed in Section 4.3. Tables 2 and 3 display the applied terminology to describe parameters, sets, and variables used in the formulation. Variables are created by the model depending on its sets, as detailed in Table 3.

4.1. Main model

The formal mathematical definition of the MALBP-2 is given by Expressions (1) to (13):

$$\text{minimize:} \quad \underbrace{CT}_{\text{cycle time}} + \underbrace{\frac{1}{NT+1} \cdot \sum_{s \in S} Z_s}_{\text{number of stations}} \tag{1}$$

$$\sum_{s \in S} X_{t,s} = 1 \qquad \forall\ t \in T \tag{2}$$

$$\sum_{w \in W} W_{t,w} = 1 \qquad \forall\ t \in T \tag{3}$$

$$Y_{w,s} \geq X_{t,s} + W_{t,w} - 1 \qquad \forall\ (t,s) \in TS,\ (t,w) \in TW \tag{4}$$

$$Nmax \geq \sum_{(w,s) \in WS} Y_{w,s} \tag{5}$$

$$F_{t_i,t_j} = 1 \qquad \forall\ (t_i, t_j) \in P \tag{6}$$

Table 2
Definition of parameters and sets.

Parameter	Description
NT	Number of tasks
NS	Upper limit on the number of stations
NW	Upper limit on the number of workers per station
$Nmax$	Upper limit on the number of workers in the line
D_t	Duration [time units] of task t, always a natural number
B	A big positive number
Set	**Description**
T	Tasks set t; $T = \{1, 2, ..., t, ..., NT\}$
S	Stations set s; $S = \{1, 2, ..., s, ..., NS\}$
W	Workers set w; $W = \{1, 2, ..., w, ..., NW\}$
TS	Task-Station tuple
TW	Task-Worker tuple
WS	Worker-Station tuple
TWS	Task-Worker-Station tuple
P	Precedence relations between tasks (t_i, t_j): $t_i \prec t_j$

$$F_{t_i,t_j} \geq X_{t_j,s} - \sum_{sk \in S | sk \geq s} X_{t_i,sk} \qquad \forall\ t_i,\ t_j \in T,\ s \in S \mid t_i \neq t_j \tag{7}$$

$$F_{t_i,t_j} + F_{t_j,t_i} \geq X_{t_i,s} + X_{t_j,s} + W_{t_i,w} + W_{t_j,w} - 3 \qquad \forall\ (t_i, w, s),\ (t_j, w, s) \in TWS \mid t_i \neq t_j \tag{8}$$

$$ST_t \geq CT \cdot (s-1) - B \cdot (1 - X_{t,s}) \qquad \forall\ (t,s) \in TS \tag{9}$$

$$ST_t + D_t \leq CT \cdot s + B \cdot (1 - X_{t,s}) \qquad \forall\ (t,s) \in TS \tag{10}$$

$$ST_{t_j} \geq ST_{t_i} + D_{t_i} - B \cdot \left(1 - F_{t_i,t_j}\right) \qquad \forall\ t_i,\ t_j \in T \mid t_i \neq t_j \tag{11}$$

$$Z_s \geq X_{t,s} \qquad \forall\ (t,s) \in TS \tag{12}$$

$$Z_s \geq Y_{w,s} \qquad \forall\ (w,s) \in WS \tag{13}$$

Expression (1) states the objective function considered for this problem, it is akin to the one used in Roshani and Giglio [35]. The first portion in the expression is the same, it corresponds to the line's cycle time. The second component of the objective function, however, represents the total number of stations opened in the line instead of the total number of workers, along with its weighted cost ($\frac{1}{NT+1}$). This crucial modification is further detailed in Section 4.3. Notice that, in this manner, the primary objective is to minimize the cycle time: there is a clear hierarchical order of importance between cycle time and number of stations, so that a cycle time unit is at least $1 + NT$ times costlier than a station, enforcing a disadvantageous trade-off between them. It is also important to mention that cycle time values are always integer due to the nature of task processing times: the ALBP benchmark only contains instances with integer task durations and, in any case, one can simply multiply the actual task processing times (given in seconds or minutes) of all tasks by an arbitrary large number in order to obtain integer durations, which is a common practice in the surveyed literature (Section 2).

Occurrence is stated by Constraints (2), they force each task to be assigned to a station exactly once. Analogously, Constraints (3) ensure each task to be exclusively executed by a specific worker. Constraints 4 impute appropriate values to $Y_{w,s}$ variables: whenever a task t is assigned to station s and such task t is also performed by worker w, then it is conceivable to infer that worker w from station s is employed for such tasks. Finally, Constraints (5) state that the total number of workers in the line is limited by the maximum number of available workers ($Nmax$).

The precedence relations and scheduling between tasks are satisfied by Constraints (6)–(11), which use the same reasoning recently presented by Michels et al. [28]. Task following (F_{t_i,t_j}) and starting time (ST_t) variables define if and when each task can or must start. It is mandatory for a given task t_j to follow task t_i (Constraints 6) if such pair of tasks (t_i, t_j) is included in P (the precedence set). The variables F_{t_i,t_j} exclude situations in which $t_i = t_j$, since a task cannot follow itself. In order to properly determine following variables based on task assignment and worker use, Constraints (7) and (8) work as logical ties. Constraints 7 decide whether or not t_j follows t_i between stations: task t_i follows task t_j whenever t_i is not assigned to the same station that t_j is or in any station after that. In the same station, however, one of them must follow the other if both tasks t_i and t_j are performed by the same worker

Table 3
Definition of variables.

Variable	Set	Domain	Description
CT	–	$\mathbb{R}_+$	Cycle time [time units]
$X_{t,s}$	$(t,s) \in TS$	{0, 1}	Task assigned to station: 1 if task t is assigned to station s
$Y_{w,s}$	$(w,s) \in WS$	{0, 1}	Worker assigned station: 1 if worker w is hired in station s
$W_{t,w}$	$(t,w) \in TW$	{0, 1}	Task assigned to worker: 1 if task t is executed by worker w
Z_s	$s \in S$	{0, 1}	Open station: 1 if station s is used
F_{t_i,t_j}	$t_i, t_j \in T \mid t_i \neq t_j$	{0, 1}	Following: 1 if task t_j follows task t_i
ST_t	$t \in T$	$\mathbb{R}_+$	Start time: task t starts to be performed at this time

(Constraints (8)). Introducing a sufficiently large natural number B (whose value is stated in Section 4.3), the starting time of a task t must be superior to a minimum value regarding the station s in which it is performed (Constraints (9)), as well as inferior to its upper maximum limit (Constraints (10)). Lastly, Constraints (11) state that all tasks t_i that precede t_j must be finished before t_j can start.

Finally, the combination of Constraints (12) and (13) along with the objective function assists in a logical searching process, as they forbid the existence of unoccupied or unproductive stations: they respectively state that a station must be opened for a task or a worker to be assigned there.

4.2. Valid inequalities

The model (1)–(13) represents MALBP-2. Nonetheless, in order to save some time by not visiting symmetric solutions, some ordering symmetry breaks have to be implemented into the model [45]. The problem's linear relaxation is strengthened and the issue found by Yilmaz and Yilmaz [50] in a previous paper is prevented by them. In Fattahi and Roshani [13], objective function values would be correct, but inconsistent solutions were found because an arbitrary opening order of stations was permitted. Therefore, a station can only be opened if a previous one is already opened (Constraints (14)) and a worker can only be used if a previous one is already in use (Constraints (15)). They break the symmetry between stations and workers, respectively, avoiding equivalent solutions to be taken into account by the model in respect of objective function values, thus shrinking the search-space and leading to tighter bounds. These ordering constraints were also used by Kellegöz [18], Michels et al. [28], and Naderi et al. [30] in their MALBP-1 mathematical models, provided that Constraints (14) and (15) are just applied to the model from the second station/worker onwards.

$$Z_s \leq Z_{s-1} \qquad \forall\ s \in S \mid s > 1 \tag{14}$$

$$Y_{w,s} \leq Y_{w-1,s} \qquad \forall\ (w,s) \in WS \mid w > 1 \tag{15}$$

Henceforth, the MILP formulation defined by (1)–(15) is referred to as PF (proposed formulation).

4.3. Upper and lower bound values for CT

This section explores a crucial modeling decision. It concerns the definition of an upper bound value applied to CT and its role in the objective function. It is necessary to remind that, by hypothesis, Baybars [6] define the SALBP to be a very constrained problem, restricted by several simplification assumptions. One of them being that each station can only be operated by one worker. Thus, by fixing the number of stations in a type-2 SALBP, one is, in practice, minimizing the line's cycle time while maintaining the number of workers fixed as a consequence. It makes sense; tasks are actually performed by workers rather than by physical stations. Nevertheless, more than one worker can be engaged in each station once this hypothesis is relaxed. It follows that the number of workers and stations must be expressly disassociated, allowing more flexible and efficient manufacturing configurations. Moreover, it has been repeatedly stated that weights can be attributed to workers and stations based on the explicit economical importance of each resource, in which fixed monthly costs (cumulative salaries) are generally much costlier than the one-time expense of physical parts of a station [13,18,28,30] in MALBP-1 works.

For a MALBP-2, Roshani and Giglio [35] have chosen to minimize the cycle time as the primary goal and the number of workers along the line as the secondary, all that whilst fixing the total number of stations. As a consequence, their model tends to fit as many workers as possible in the stations in order to achieve better levels of productivity. Thus, the results reported in their paper show solutions in which a SALBP-2 configuration with the same number of workers could be more efficient in terms of cycle time than the one optimized for MALBP-2. Considering that the primary objective is to minimize CT, optimized MALBP solutions should have been at least equivalent to its SALBP counterpart [28], but never worse. Taking that issue into consideration, it has been decided that the proposed formulation would fix the number of workers along the line ($Nmax$) and minimize the total number of stations as the secondary objective. The reasoning behind it is that, in this way, the problem is closer to its SALBP-2 relative (minimize cycle time) and still accepts the concept of previously developed MALBP-1 works, in which the number of workers is the resource that should be prioritize over the number of stations. This approach is also in accordance with what can be found in reality: it does not exaggerate labor costs to achieve better production rates. In fact, overall costs must be lower than (or at least equal to) its less flexible counterpart, a characteristic that cannot be found in Roshani and Giglio [35].

With the number of workers as the fixed resource, it is expected that the proposed MALBP-2 formulation cycle time is at least as good as its SALBP-2 version with the same given number of workers. That said, Constraints (16) are included in the model to represent an upper bound for the cycle time value in a MALBP-2, in which CT_{SALBP} is a newly introduced parameter for the optimized SALBP-2 variation of the same data. On the other hand, Constraints (17) and (18) are trivial lower bounds for the cycle time and the total number of stations, respectively. Moreover, $CT_{SALBP} \cdot NS$ is sufficiently large to assume the B role in the PF.

$$CT \leq CT_{SALBP} \tag{16}$$

$$CT \geq \lceil \frac{\sum_{t \in T} D_t}{Nmax} \rceil \tag{17}$$

$$\sum_{s \in S} Z_s \geq \lceil \frac{Nmax}{NW} \rceil \tag{18}$$

5. Solution method

The Benders' decomposition algorithm (Section 5.2) – hereafter referred to as BDA – and its initial solution procedure (Section 5.1) developed for the MALBP-2 are herein presented. Section 5.2.1 presents the master problem (MP), whilst Section 5.2.2 is dedicated to the slave problem (SP). In the proposed implementation, the former is associated to decisions concerning task and worker assignments to stations (high-level problem) and the latter is related to feasibility tests of task assignments and scheduling to workers in each station (lower-level problem). However, since Benders' decompositions are known to have a slow convergence process [25], the algorithm may go through enhancement modifications to accelerate its operation. Those are explicitly pointed out along with the initial solution, MP, and SP descriptions.

5.1. Initial solution decomposition

An initial feasible solution for the MALBP-2 can be obtained by following the routine presented in Sections 5.1.1 and 5.1.2. Afterwards, the task-worker-station allocation and scheduling found by this procedure feeds the Benders' decomposition algorithm, which is, in turn, presented in Section 5.2. The illustrative instance presented in Table 1 is solved step-by-step as the proposed solution method stages are presented.

5.1.1. Solving the SALBP counterpart

In the proposed solution method, the first step to solve a MALBP-2 instance is to solve its simpler counterpart, the SALBP-2. For that, the well-known exact algorithm SALOME [40] is used to find the optimal or an integer solution for a given case. By applying this step to the illustrative instance (Table 1), the solution previously presented in Fig. 2a

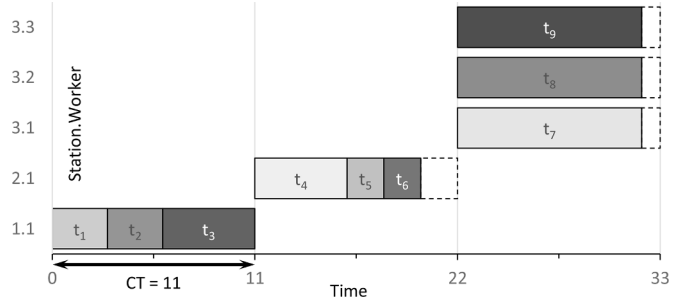


Fig. 3. Gantt diagram representation of an intermediary multi-manned feasible solution for the illustrative instance by applying the initial decomposition procedure.

can be obtained, with 5 workers placed into 5 stations, and a cycle time of 11 time units.

5.1.2. Minimizing stations

Since SALBPs assume that there is always only one worker in each station, once with a feasible task-station assignment ($X_{t,s}$) provided by SALOME at hand, it is possible to emulate that solution to a fixed task-worker assignment ($W_{t,w}$), treating such variables as parameters.

The residual problem, then, can be reduced to worker-station assignment ($Y_{w,s}$) and task scheduling (ST_t) problems. As precedence relations constraints between stations are automatically respected from the SALBP solution, this model only has to group workers whose assignments do not violate such constraints within each station, and therefore minimizing the number of stations for a given task-worker assignment.

For example, by considering once again the illustrative instance from Table 1, the initial SALBP solution in Fig. 2a, and an upper limit on the number of workers per station set to $NW = 3$, this intermediary step of minimizing stations given task-worker assignments can aggregate multiple workers in the same station while still respecting precedence relations constraints. An intermediary multi-manned feasible solution for the illustrative instance is presented in Fig. 3. The initial decomposition procedure is not able to deliver a solution with a reduced production rate (i.e. $CT = 11$, with times on x-axis), and idle times remain at 5 time units along the line, or 9.09% of the line's available task performing time. On the other hand, by allowing co-occurring operations within the third station by three different workers (represented by bars), the intermediary MALBP feasible solution was capable of reducing the line length from 5 to 3 stations (on the y-axis). Notice that all task-worker assignments persist from the input SALBP solution (Fig. 2a) and no constraint imposed by precedence relations is violated.

As demonstrated, this approach does not ensure optimality for any type of MALBPs. Nonetheless, it has been successfully applied to the MALBP-1 instances by Lopes et al. [22], as it yields good solutions in quite reduced computational times. As evidenced in Section 6.2, using such task-station-worker assignment as an initial solution for the complete Benders' decomposition approach (Section 5.2) can drastically improve faster convergence for the algorithm (*Enhancement 1*).

5.2. Benders' decomposition algorithm

For the MALBP-2, the MP tackles task and worker assignments to stations, whereas the SP takes care of task scheduling problems, orderly assigning tasks to workers. Naturally, MP and SP cannot be formulated by simply decoupling parts of the PF: Sections 5.2.1 and 5.2.2 explain how each of these models is generated. Fig. 4 is a flowchart that illustrates the path taken by the proposed BDA, when looking for an optimal solution, in which HT is a hash-table to be consulted for repetitive solutions. When the algorithm loads the initial solution, it consequently establishes an upper bound for the problem and gets a proxy on the direction that the search process should occur: for the illustrative instance, this initial solution is represented by Fig. 3, which the proposed algorithm promptly leads to an optimal MALBP-2 solution previously presented in Fig. 2b.

In order to deal with the problem stated in Section 4 by the PF (Expressions (1)–(18)), one should notice that the SP is a mixed-integer problem in this application, thus a feasibility-seeking variant [11] should be used. Once again, the SP must be employed as a feasibility check on the algorithm, as analogously proposed by recent MALBP-1 works [28,30]. Given its resources to each multi-manned station, the SPs should, particularly for the proposed BDA, assign and schedule tasks to workers for each individual station, in an effort to complete all the assigned tasks within the cycle time limit previously defined by the current MP iteration.

The second enhancement is detailed in Section 5.2.1: feasibility cuts are implemented and applied to the MP before execution starts, constraining the possibilities of task assignments (*Enhancement 2*). These are based on precedence graph analyzes [21]. Besides, each infeasible SP returns combinatorial inequalities (cutting planes) to be appended as lazy constraints into the MP, originating the third enhancement procedure to accelerate convergence [11]: the use Combinatorial Benders' Cuts (*Enhancement 3*). This is done by extending the previous concept to use such cuts. The MP is distilled from the monolithic model, i.e. the original and complete combinatorial problem, unconstrained from scheduling restriction, since it is initially taken apart from the SP. As all decision variables to calculate the objective value (CT and Z_s) are found in any given MP iteration, solving it may generate feasible integer solutions. These candidate solutions are then sent to the SP to be validated by it. If all SPs detect the tentative solution to be feasible, the current solution is established as the new incumbent. Otherwise, a set of the aforementioned CBC is returned from each infeasible SP. Finally, the BDA iterates this procedure until (i) an optimal solution is found and proven or (ii) the time limit is reached.

The last improvement prevents redundant feasibility seeking tests to occur (*Enhancement 4*). In order to do so, a hash-table [26], hereafter referred to as HT, is exploited to store SP instance data that had led to feasible solutions. If a set of task assignments for a given SP is solved and deemed to be feasible, such set of tasks is included into a HT with tested feasible problems, along with the number of workers and cycle time used to perform them. This procedure allows the SP model to skip repeated scheduling problems, since the HT can be quickly consulted by the algorithm's sub-routine beforehand. For similar reasons, every time an infeasibility is detected by an SP, such circumstance is modeled as a new set of constraints and appended to the MP, so those task and worker assignments will not be repeated. These combinatorial Benders' cuts are further explained in Section 5.2.2.

5.2.1. Master problem

Expressions (1), (2), (5), and (12)–(18) are kept in the MP, since they are part of the balancing core of the complete problem. Likewise, additional constraints are developed to adapt and strengthen the MP. Before establishing them, however, Table 4 newly introduces the terminology needed to describe the remaining of the MP model.

Independently of the number of workers assigned to a station, there are tasks that cannot be performed together in the same station due to limitations imposed by cycle time and precedence relation restrictions. In order to represent those, a set of incompatible task pairs (*Inc*) is developed. Algorithm 1 shows the steps on how this incompatibility is determined to find such task pairs. Firstly, the direct precedence relations set P must be extended to a complete (direct and indirect) precedence set P^* in order to do so. That is easily made possible by regarding all direct and indirect precedence relations in the precedence graph. With P^* at hand, the next step consists in constructing sets for each task containing all their successors and predecessor: Su_t represents direct and indirect successors of task t and Pr_t constitutes direct and

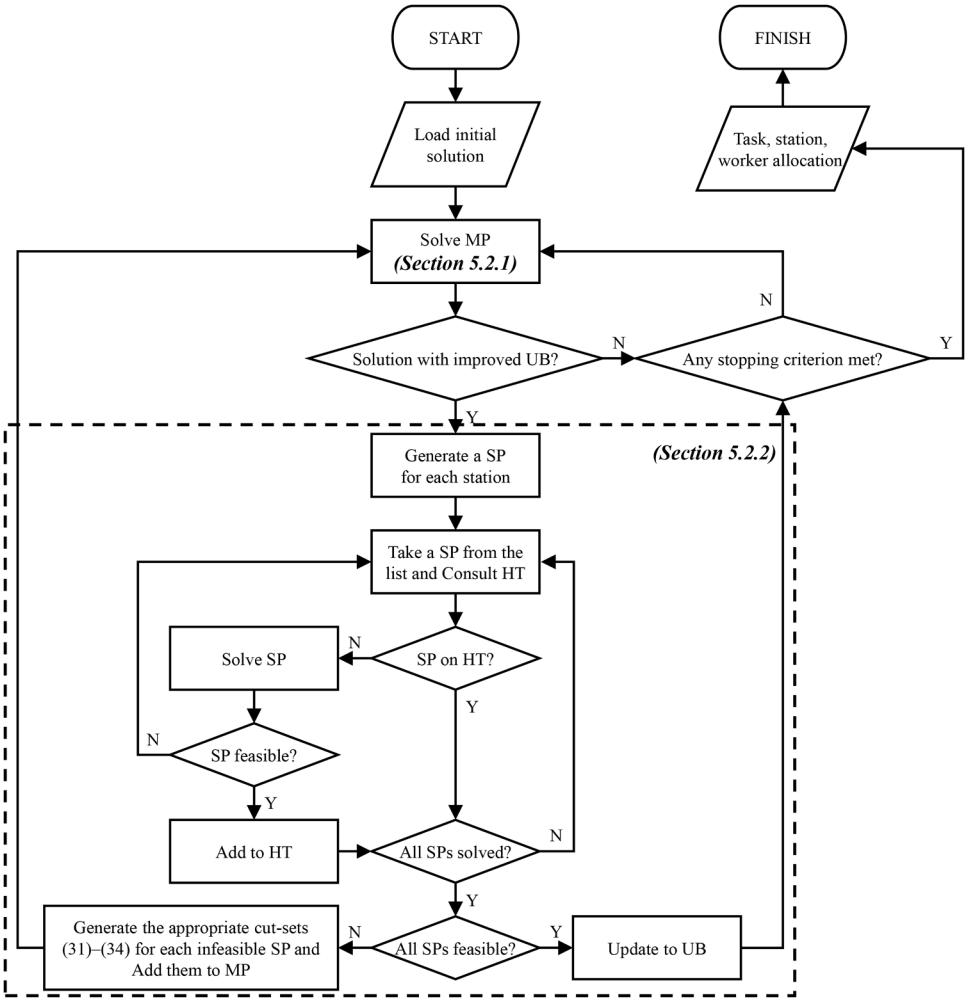


Fig. 4. General flowchart scheme of the proposed solution method.

Table 4
Definition of new parameters, sets, and variable for the MP.

Parameter	Description
δ_{t_i,t_j}	Critical path duration between tasks t_i and t_j; $t_i \prec t_j$
σ_{t_i,t_j}	Sum of task durations between tasks t_i and t_j; $t_i \prec t_j$
Set	**Description**
P^*	Set of complete extended precedence relations P: (t_i, t_j)
Su_t	Set of all direct and indirect successor of task t
Pr_t	Set of all direct and indirect predecessor of task t
Inc	Set of incompatible task pairs: (t_i, t_j)
Variable	**Description**
$A_{w,s}$	Auxiliary integer variable: detects the available time for each worker w in station s

indirect predecessors of task t. Moreover, the critical path duration and the sum of task durations between tasks t_i and t_j are represented by the two extra parameters δ_{t_i,t_j} and σ_{t_i,t_j}, respectively. A topological order is applied in order to recursively evaluate and properly compute the values of these parameters.

That stated, P^*, Su_t, Pr_t, CT_{SALBP}, and D_t can be the input for Algorithm 1, whereas δ_{t_i,t_j}, σ_{t_i,t_j}, and Inc are its output. Given a task pair (t_i, t_j) $\in P^*$ (line 3), the code runs through the complete precedence graph and establishes the critical path (δ_{t_i,t_j}, line 4) and capacity bounds (σ_{t_i,t_j}, line 5) between any two tasks that have direct or indirect precedent relations, recursively attributing correct values to the parameters. In other words, Algorithm assigns to δ_{t_i,t_j} the highest value for the sum of task durations that have to be executed between tasks t_i and t_j and to σ_{t_i,t_j} the sum of task durations taken from all tasks that are, at the same time, successors and predecessor of tasks t_i and t_j, respectively. This is a logical concept inherited from Klein [21] and likewise adopted in previous MALBP-1 works [8,28], which takes into account similarities between the SP and project scheduling problems. Lastly, by taking the task pair (t_1, t_8) of the precedence graph from Table 1 for instance, these parameters would be $\delta_{t_1,t_8} = 7$ and $\sigma_{t_1,t_8} = 14$.

If either condition $D_{t_i} + D_{t_j} + \delta_{t_i,t_j} > CT_{SALBP}$ or $D_{t_i} + D_{t_j} + \lceil \frac{\sigma_{t_i,t_j}}{NW} \rceil > CT_{SALBP}$ is verified (line 6), then the condition to flag an incompatibility is satisfied, and the task pair (t_i, t_j) is included into the incompatibility set Inc (line 7). This process goes on until the whole complete precedence graph is analyzed.

Once in possession of Inc, Constraints (19) can be appended to the MP, which should restrict specific task pairs to be assigned to the same station. Generating a set of weaker but still valid inequalities is possible by using information extracted from parameter σ_{t_i,t_j}: a very small positive number (ε) is introduced to prevent dividing by zero, Constraints (20) define what is the minimum number of workers that must be employed in any given station to perform both t_i and t_j of a task pair (t_i, t_j).

$$X_{t_i,s} + X_{t_j,s} \leq 1 \qquad \forall\ s \in S,\ (t_i, t_j) \in Inc \tag{19}$$

Input : $P^*, Su_t, Pr_t, CT_{SALBP}, D_t$
Output: $\delta_{t_i,t_j}, \sigma_{t_i,t_j}, Inc$
1 **Initialization**: $\delta_{t_i,t_j} \leftarrow 0$, $\sigma_{t_i,t_j} \leftarrow 0$, $Inc \leftarrow \{\}$
2 **Start**
3 **forall the** $(t_i, t_j) \in P^*$ **do**
4 $\quad \delta_{t_i,t_j} = \max\,[0;\ \delta_{t_i,t_k} + D_{t_k} \mid t_k \in Su_{t_i} \cap Pr_{t_j}]$
5 $\quad \sigma_{t_i,t_j} = \sum_{t_k \in Su_{t_i} \cap Pr_{t_j}} D_{t_k}$
6 $\quad$ **if** $D_{t_i} + D_{t_j} + \delta_{t_i,t_j} > CT_{SALBP} \vee D_{t_i} + D_{t_j} + \left\lceil \frac{\sigma_{t_i,t_j}}{NW} \right\rceil > CT_{SALBP}$ **then**
7 $\quad\quad Inc \leftarrow Inc \bigcup \{(t_i, t_j)\}$
8 $\quad$ **end**
9 **end**
10 **End**

Algorithm 1. Pseudo-code for the incompatibility set generation.

$$\sum_{w \in W} Y_{w,s} \geq \left\lceil \frac{\sigma_{t_i,t_j}}{CT_{SALBP} - D_{t_i} - D_{t_j} + \varepsilon} \right\rceil - NW \cdot \left(2 - X_{t_i,s} - X_{t_j,s}\right) \quad \forall\ s \in S,\ (t_i, t_j) \in P^* \tag{20}$$

Finally, precedence relations constraints (Constraints (21)) are added to the MP and the available time to execute tasks in each station (Constraints (22)) is reckoned. Notice that the available time is dependent on the cycle time and the number of workers assigned for a given station, leading to a non-linear constraint. To solve that issue, an integer auxiliary variable ($A_{w,s}$) must be introduced to evaluate how much available time each station s has at each worker position w. Constraints (23) and (24) are respectively activated as trivial lower and upper bounds for $A_{w,s}$ and limit its value to zero unless some worker w is assigned to their position. Constraints (25) and (26), on the other hand, attribute the correct amount of cycle time to $A_{w,s}$ whenever worker w is employed in a given station. Ultimately, Constraints (22) can be replaced by its linearized version: Constraints (27).

$$\sum_{s \in S} s \cdot X_{t_i,s} \leq \sum_{s \in S} s \cdot X_{t_j,s} \quad \forall\ (t_i, t_j) \in P \tag{21}$$

$$\sum_{t \in T} X_{t,s} \cdot D_t \leq CT \cdot \sum_{w \in W} Y_{w,s} \quad \forall\ s \in S \tag{22}$$

$$A_{w,s} \geq \left\lceil \frac{\sum_{t \in T} D_t}{Nmax} \right\rceil \cdot Y_{w,s} \quad \forall\ (w, s) \in WS \tag{23}$$

$$A_{w,s} \leq CT_{SALBP} \cdot Y_{w,s} \quad \forall\ (w, s) \in WS \tag{24}$$

$$A_{w,s} \geq CT - CT_{SALBP} \cdot (1 - Y_{w,s}) \quad \forall\ (w, s) \in WS \tag{25}$$

$$A_{w,s} \leq CT - \left\lceil \frac{\sum_{t \in T} D_t}{Nmax} \right\rceil \cdot (1 - Y_{w,s}) \quad \forall\ (w, s) \in WS \tag{26}$$

$$\sum_{t \in T} X_{t,s} \cdot D_t \leq \sum_{w \in W} A_{w,s} \quad \forall\ s \in S \tag{27}$$

For the proposed BDA, seeking an optimal solution is the emphasis of this reformulated MP. In order to concentrate the efforts in that direction, each feasible solution $(\bar{X}, \bar{Y}) = \{(\bar{X}_{1,1}, ..., \bar{X}_{t,s}), (\bar{Y}_{1,1}, ..., \bar{Y}_{w,s})\}$ with an improved objective value (CT^*, S^*) attained by the MP is passed on to the SP model, so it can carry on scheduling feasibility checks in each station $s \in S^*$ (Fig. 4). Therefore, the PF is decomposed into an MP that decides cycle time, task-station allocation, worker-station assignment, and station opening variables: CT, $X_{t,s}$, $Y_{w,s}$, and Z_s, respectively. Meanwhile, the SPs look for feasible task-worker assignments ($W_{t,w}$), by accounting for task starting times and ordering (ST_t and F_{t_i,t_j}) for each station.

5.2.2. Slave problem

The path taken by the BDA to solve each SP is presented in Fig. 4. Algorithm 2 details each step of that process. It gets as input an improved solution from the MP, with information regarding the task and worker assignments to stations ($\bar{X}$, $\bar{Y}$), as well as the new objective value to be tested, represented by a cycle time (CT^*) and the total number of stations (S^*) used for such solution. The current incumbent (UB), lower bound (LB), and the parameter CT_{SALBP} are also informed.

As to generate an SP for each station $s \in S^*$ (line 3), the SP model maintains Expressions (3), (6), and (11) as in Section 4, but simplifies Constraints (8) and adds Constraints (29) and (30) as valid inequalities for idle time symmetry breaks.

For that, they are separately applied to each SP (line 4), where task and worker sub-sets (T_s and W_s) are extracted from the MP solution ($\bar{X}$, $\bar{Y}$), as declared by on lines 5 and 6. Regarding each station $s \in S^*$ as a detached resource-constrained scheduling problem, Constraints (28), (29), and (30) can replace the previous PF's monolithic ones.

$$F_{t_i,t_j} + F_{t_j,t_i} \geq W_{t_i,w} + W_{t_j,w} - 1 \quad \forall\ t_i,\ t_j \in T_s,\ w \in W_s \tag{28}$$

Input : $(\overline{X}, \overline{Y}), CT^*, S^*, UB, LB, CT_{SALBP}$
Output: $Cuts, TWS_{allocation}$
Initialization: $SP_s \leftarrow infeasible, Status \leftarrow 0, Counter \leftarrow 0, HT \leftarrow \{\}$
Start
Generate an SP for each station: SP_s
forall the $s \in S^*$ **do**
  $T_s = \{t \in T \mid \overline{X}_{t,s} = 1\}$
  $W_s = \{w \in W \mid \overline{Y}_{w,s} = 1\}$
  if $\{(T_s, W_s, Obj^*)\} \subset HT \vee |W_s| = 1$ **then**
    $SP_s \leftarrow feasible$
    $Counter \leftarrow Counter + +$
  else
    Solve SP_s
    if $SP_s = feasible$ **then**
      $HT \leftarrow HT \bigcup \{(T_s, W_s, CT^*)\}$
      $Counter \leftarrow Counter + +$
    else Generate $Cuts$ for infeasible SP_s
      if $|W_s| = NW \wedge CT^* = CT_{SALBP}$ **then**
        Add Cut 31
      else
        if $|W_s| < NW \wedge CT^* = CT_{SALBP}$ **then**
          Add Cut 32
        else
          if $|W_s| = NW \wedge CT^* < CT_{SALBP}$ **then**
            Add Cut 33
          else $|W_s| < NW \wedge CT^* < CT_{SALBP}$
            Add Cut 34
          end
        end
      end
    end
  end
end
if $Counter = S^*$ **then**
  Update incumbent solution with a complete $TWS_{allocation}$: $UB \leftarrow (CT^*, S^*)$
  if $UB = LB \vee CPU \geq Time.Limit$ **then**
    $Status \leftarrow 1$
  end
end
End

Algorithm 2. Pseudo-code for the SP generated cuts.

Table 5
Summary of dataset instances.

Size (Total of instances)	Problem	*NT*	*Nmax*	*NW*
Small (32)	Mitchell	21	3; 4; 5; 7; 8; 9	2
	Heskiaoff	28	4; 6; 8; 9; 10	2; 4
	Sawyer	30	4; 6; 8; 9; 10; 12; 13; 14	2; 4
Medium (34)	Kilbridge	45	4; 6; 8; 10; 11	2; 4
	Tonge	70	12; 14; 16; 18; 19; 20; 22; 23	2; 4; 6
Large (42)	Arcus1	83	11; 12; 14; 15; 18; 19; 21	2; 4; 6
	Arcus2	111	12; 16; 18; 23; 24; 26; 27	2; 4; 6
Case study (4)	Dimitriadis	64	8; 10	2; 4

$$I_w = CT - \sum_{t \in T_s} D_t \cdot W_{t,w} \qquad \forall\ w \in W_s \tag{29}$$

$$I_w \geq I_{w-1} \qquad \forall\ w \in W_s \mid w > 1 \tag{30}$$

There are two situations that an SP is automatically deemed feasible (line 7 to 9): (i) the solution under analysis has already been evaluated before and is contained in HT or (ii) there is only one worker employed in the station, which means there is no scheduling problem in the first place. If none of these conditions is observed, then the new SP is solved (line 11). Whenever feasibility is detected to this new problem, its information concerning tasks, workers, and cycle time used in the instance is stored into HT (line 13) in order to avoid wasting time solving the same instance in the future by simply accessing HT. On the other hand, when the SP is detected to be infeasible, a set of combinatorial Benders' cuts (CBC) can be applied to the MP as lazy constraints, i.e. the initial optimization problem receives new restrictions while executing in order to cut off infeasible combinations of task and worker assignments to all stations (lines 15 to 29).

Nonetheless, deciding which set of CBCs to add depends on the number of workers ($|W_s|$) and the trial cycle time (CT^*) value used by SPs that were proven to be infeasible. The strongest set of CBCs is appended when the trial solution employs the maximum allowed workers (*NW*) on a maximum allowed cycle time (CT_{SALBP}) for a specific station (line 16). On line 17, Constraints (31) state that, if any given task assignment set is tested to be infeasible, then it cannot be entirely executed in the same station *s*, thus at least one of the tasks must be performed elsewhere. Alternatively, the SP may be infeasible, but it might not be using the maximum number of workers for that set of tasks (line 19), the trial cycle time might be lower than the SALBP's imposed one (line 22), or both (line 24).

In such cases, the set of CBCs described by Constraints (32), (33), and (34) are applied to the MP; they respectively state that a tested task assignment set cannot be totally performed in the same station unless an additional worker (represented by the $|W_s| + 1$ index) is assigned there (line 20), the cycle time is increased (line 23), or both (line 25). This wide variety of cuts could not be found in any previous work.

$$\sum_{t \in T_s} X_{t,s} \leq \sum_{t \in T} \bar{X}_{t,s} - 1 \qquad \forall\ s \in S \tag{31}$$

$$\sum_{t \in T_s} X_{t,s} \leq \sum_{t \in T} \bar{X}_{t,s} - 1 + Y_{|W_s|+1,s} \qquad \forall\ s \in S \tag{32}$$

$$\sum_{t \in T_s} X_{t,s} \leq \sum_{t \in T} \bar{X}_{t,s} - 1 + (CT - CT^*) \qquad \forall\ s \in S \tag{33}$$

$$\sum_{t \in T_s} X_{t,s} \leq \sum_{t \in T} \bar{X}_{t,s} - 1 + Y_{|W_s|+1,s} + (CT - CT^*) \qquad \forall\ s \in S \tag{34}$$

The SP evaluates a trial solution as wholly feasible only if it is feasible for all stations $s \in S^*$ (line 32), then the proposed method judges this solution sent from the MP as incumbent and the instance's *UB* is updated with the new, improved cycle time, number of stations, and a complete set of task-worker-station assignments (line 33). This process is repeated iteratively and the MP keeps searching for better solutions, either with a revised UB or newly added lazy constraints, until any stopping criterion is met, namely (i) an optimal solution is found and proven or (ii) the CPU time limit is reached (line 34).

In combination with the initial solution generation technique (Section 5.1), the BDA herein presented (Section 5.2) is what constitutes the whole proposed solution procedure, which is hereafter referred to as PM (Proposed Method).

6. Computational study

This section carries out a computational study based on the same benchmark dataset and industrial case used by Roshani and Giglio [35]. Nevertheless, the current dataset extends the previous one (with 72 instances) by considering the possibility of employing 2, 4, or 6 workers in the same station, totaling 108 instances. A well-known literature benchmark is contained in the tested instances. It is also accessible for download at <www.assembly-line-balancing.de>. They comprise information about task durations, precedence graphs, and cycle time values. The list of instances is summarized in Table 5; in it, instances are divided into three categories: small, medium, and large, according to the number of tasks (*NT*) parameter. Additionally, there are 4 case study instances regarding the problem studied by Dimitriadis [12]. These instances are run with different values of total number of workers in the line (*Nmax*) and maximum number of workers allowed in each station (*NW*). As explained in Section 4.3, upper bounds for the cycle time (*CT*) and number of stations (*NS*) have been obtained with a preliminary process, retrieving information from SALBP-2 counterpart solutions of each instance.

The computational tests are split in three parts. Firstly, Section 6.1 reports results obtained by the monolithic model (PF) presented in Section 4 when it is applied to small-size instances. These results are compared to those described in Roshani and Giglio [35]; this last mathematical model is henceforth referred to as RF (Roshani's Formulation). In addition, the same fraction of the benchmark dataset is also solved by the PM displayed in Section 5, whilst its performance is compared to the best outcome obtained by either Direct or Indirect Simulated Annealing (DSA/ISA) heuristics developed by Roshani and Giglio [35] in terms of solution quality reported by them; the latter method is henceforth referred to as RM (Roshani's Method).

Secondly, as both methods demonstrated dominance over their respective mathematical formulations, monolithic models (PF and RF) were discarded in the remainder testing process, as they are not expected to keep up with specific methods in terms of solution quality and computational processing time. Therefore, for this medium and large-size computational study conducted afterwards, only PM and RM were considered and applied to MALBP-2 instances. Moreover, in order to test the synergies between the initial decomposition (Dec) and the Benders' decomposition algorithm (BDA) that compose the PM, Dec and BDA were also applied to medium and large-size instances separately. Section 6.2 presents and discusses such results and comparisons.

These comparisons are conducted as suitably as possible: as Roshani and Giglio [35] fixed the number of stations and minimized cycle time as the primary objective, their formulation tends to fit as many workers as it can to the limited number of stations, which leads to solutions with lower efficiencies than its less flexible version. As argued in Section 4.3, the assumption of limited number of stations (line area) is not practical and ignore the fact that workers and stations have completely different costs, with those of the former being much higher. Wages are paid to the workforce at least in a monthly basis, so labor costs directly influence the final cost of the product and impose a limit on the resources one can use. Assembly line plants for the automotive industry can regularly surpass 300,000m^2 in area, with extreme examples such as the Hyundai Ulsan Factory and the Volkswagen Wolfsburg Plant with 5,050,000m^2 and 6,500,000m^2 in area, respectively. Considering the dimensions of a regular car and the additional space to perform tasks by

Table 6
Results for small-size instances: comparison between monolithic formulations (PF and RF) and solution methods (PM and RM).

Problem	*Nmax*	*NW*	PF			RF			PM							RM		
			CT	St	CPU	CT	St	CPU[1]	CT	St	CPU	DCPU	SCPU	CBC	HT	CT	St	CPU[1]
Mitchell	3	2	35	3	0.07	39	2	0.97	**35**	3	0.07	0.05	0.00	0	0	39	2	17.06
	4	2	27	3	0.19	27	3	1.60	27	3	0.16	0.02	0.01	0	0	27	3	14.33
	5	2	21	5	0.13	–	–	–	**21**	5	0.19	0.01	0.00	0	0	27	3	2.64
	7	2	16	6	0.87	16	6	5.01	16	6	0.26	0.01	0.00	0	0	16	6	12.32
	8	2	14	7	0.05	14	7	6.35	14	7	0.21	0.02	0.00	0	1	14	7	2.58
	9	2	13	8	0.10	13	8	9.32	13	8	0.26	0.01	0.00	0	0	13	8	0.29
Heskiaoff	4	2	256	3	22.01	274	2	1h	**256**	3	32.23	0.01	31.97	1326	1	274	2	6.31
		4	256	3	25.26	–	–	–	256	3	36.81	0.01	36.56	1401	1	–	–	–
	6	2	171	4	1h	183	3	1h	**171**	4	0.05	0.02	0.00	0	2	183	3	7.50
		4	171	4	1h	–	–	–	171	4	0.04	0.01	0.01	0	2	–	–	–
	8	2	129	5	1h	139	4	1h	**129**	5	0.31	0.02	0.03	20	3	139	4	6.87
		4	129	5	1h	–	–	–	129	5	0.34	0.02	0.04	35	3	–	–	–
	9	2	116	7	100.71	124	5	1402.28	**116**	7	1.53	0.02	0.30	749	14	124	5	41.66
		4	116	7	140.03	–	–	–	116	7	2.73	0.02	0.86	1148	13	–	–	–
	10	2	108	6	13.85	108	6	63.26	108	6	0.27	0.02	0.18	318	59	108	6	2.91
		4	108	6	20.99	–	–	–	108	6	0.93	0.02	0.46	804	84	–	–	–
Sawyer	4	2	81	2	446.71	82	2	1h	**81**	2	0.27	0.02	0.02	4	0	83	2	5.93
		4	81	2	108.45	–	–	–	81	2	0.42	0.02	0.01	0	0	–	–	–
	6	2	55	4	1h	56	3	1h	**55**	4	0.22	0.01	0.00	0	2	57	3	7.08
		4	55	4	1h	–	–	–	55	4	0.26	0.01	0.00	0	2	–	–	–
	8	2	41	4	1h	41	4	123.49	**41**	4	0.13	0.01	0.02	7	3	42	4	7.36
		4	41	4	1h	–	–	–	41	4	0.29	0.01	0.02	7	2	–	–	–
	9	2	37	6	1h	39	5	1h	**37**	6	1.82	0.01	0.01	0	5	40	5	7.16
		4	37	6	1h	–	–	–	37	6	2.23	0.01	0.09	24	1	–	–	–
	10	2	34	6	1h	–	–	–	34	6	6.34	0.02	0.05	80	12	34	6	32.15
		4	34	6	1h	–	–	–	34	6	8.08	0.02	0.04	80	10	–	–	–
	12	2	28	7	1h	28	7	1h	**28**	7	0.39	0.02	0.01	0	4	29	7	19.81
		4	28	7	1h	–	–	–	28	7	0.22	0.02	0.01	0	4	–	–	–
	13	2	26	8	1h	–	–	–	**26**	8	2.84	0.02	0.02	10	8	28	7	7.49
		4	26	8	1h	–	–	–	26	8	1.83	0.02	0.01	0	7	–	–	–
	14	2	25	8	16.50	25	8	1h	25	8	1.35	0.02	0.02	36	6	25	8	4.42
		4	25	8	32.05	–	–	–	25	8	1.62	0.02	0.03	27	5	–	–	–

[1] As reported in Roshani and Giglio [35] multiplied by a factor of 0.37.

multiple workers, a station is a much more plentiful resource than skilled labor. That is precisely why objective functions in the type-1 problem make sense, with workers representing higher costs than stations with a difference of one or two orders of magnitude [13]. Acknowledging that, PF and PM may have outperformed RF and RM in some instances due to such space limitation imposed by Roshani and Giglio [35]. Nevertheless, specific instances for direct comparison are identified and examined in both Sections 6.1 and 6.2.

Lastly, Section 6.3 evaluates the performance of both methods (PM and RM) when applied to a real-life automotive assembly plant case study. Their efficiency improvements are compared based on instances that contain a dataset originally published by Dimitriadis [12]. Since Roshani and Giglio [35] also solved a case by fixing the upper limit on the number of workers in the line to be $Nmax = 8$, the comparison here is decidedly straightforward. Furthermore, the other case ($Nmax = 10$) yielded a solution with workers hired for all possible positions in the station, which equally allows a direct comparison with the PM.

In all Sections 6.1, 6.2, and 6.3, each instance result is reported in a line of Tables 6 to 9. They are addressed by their precedence graph structure, *Nmax*, and *NW* values. For the results, columns headlines with CT, St, Gap, CPU, DCPU, SCPU, CBC, and HT respectively indicate: best found cycle time (CT), best found total number of stations (St) for that cycle time, integrality gap (Gap), computational processing time in total (CPU), for the initial decomposition (DCPU), for the slave problems (SCPU), total number of applied combinatorial Benders' cuts (CBC), and how many times the hash-table has been accessed (HT) for each instance. For small-size instances, Gap is not reported because PM has found the optimal solution for all of them. Some results are left unfilled (–) due to the fact that no integer solutions could be found by the BDA within the time limit for some instances. As this is an extended dataset, the same happens for some instances that were not solved in Roshani and Giglio [35]. For the case study, line efficiency (LE) is also computed.

For every instance, Gurobi 8.1 [31] with optimality focus for the MP and feasibility focus for SPs was employed as universal solver. Four threads of a 64-bit Intel™ i7-3770 CPU (3.4 GHz) with 16GB of RAM were used to run the algorithm. The PM's BDA and its interaction with Dec were coded in Microsoft Visual Basic 2019 programming language. Roshani and Giglio [35] solved their instances on a 64-bit Intel™ i3-330M CPU (2.13 GHz) with 4GB of RAM. Therefore, a factor of 0.37 (obtained by Passmark Performance Test 9.0) will be applied to their CPU time. Nonetheless, notice that computational processing time results are just mentioned, while relevant comparisons are strictly focused on solution quality.

6.1. Small-size instances

Table 6 is a summary of the comparison between monolithic models (PF and RF) and solution methods (PM and RM). In this section, they were applied to the small-size instances with a time limit set to 3600 seconds. Whenever an instance had been solved by Roshani and Giglio [35] as well, results obtained by PF are compared to those found in the referred paper. Complementary, PM results are exhibited alongside with the best result reported by either DSA or ISA in their respective paper (RM).

The PF clearly outperforms RF in this subset containing 32 instances: the former obtained 16 optimal solutions (50% of cases), whilst the latter reached optimality in only 6 out of the 16 instances tested by Roshani and Giglio [35], being 4 of them in the smallest problem. In other words, PF has improved 8 previously known integer solutions obtained by a mathematical model and has proven the optimality of 4 of them. This may be attributed to the fact that PF and RF took different

modeling decisions: as reported in Section 4, the proposed formulation prioritizes cycle time minimization taking into account information acquired from SALBP bounds.

A particular attention is given to the Sawyer family of instances. For $Nmax = 4$, solutions reported for both formulations and methods comprise 2 stations, however, PF and PM were able to prove an optimal cycle time result of 81 time units, while RF and RM were outperformed by yielding solutions with 82 and 83 cycle time units, respectively. By just considering the solution methods, PM has also outperformed RM in instances with $Nmax = 8$ and $Nmax = 12$: PM has proven cycle times of 41 and 28 time units, whereas RM had reached solutions with 42 and 29 cycle time units for the same number of workers and stations along the line.

The remaining information in Table 6 compares PM and RM results. Similar computational processing times were reported for both methods. Nevertheless, the PM presented in Section 5 has improved and proven 12 previously known integer solutions (boldfaced values in Table 6). Therefore, it can be stated that the PM has reached better results than RM's heuristics. Moreover, it has guaranteed solutions to be optimal for all small-size instances in a very low CPU time, achieving 100% of optimality proofs in this subset of instances.

6.2. Medium and large-size instances

By rapidly finding and proving optimal solutions in all small instances (Section 6.1, Table 6), the PM has been validated as an efficient and reliable method. Besides, its superiority over the monolithic model was evidenced. Hence, only specialized methods (i.e. PM and RM) are compared in this section regarding solution quality. The comparison only takes into account instances that were previously reported in Roshani and Giglio [35], focusing on computationally solving medium and large-size instances. Furthermore, in order to observe possible synergy effects caused by integrating the initial decomposition (Dec) and the Benders' decomposition algorithm (BDA) that compose the PM, they were also applied to medium and large-size instances separately.

Table 7 reports results for 34 medium-sized instances from Table 5. Out of that enlarged dataset, Roshani and Giglio [35] has only solved and reported solutions for 13 instances. In terms of solution quality, the PM has outperformed RM in 12 instances (boldfaced in Table 7), whilst tying in the last one. Out of the totality of 34 medium-sized instances, the PM has solved 15 cases to optimality (i.e. 44.12% of this subset), with a minor integer gap for the remaining 19 solutions (1.81% in the worst case and 0.77% on average).

The importance of the possibility to allow 4 workers per station was evidenced for both Kilbridge and Tonge families of instances. By defining $NW = 4$ as a parameter, it has been possible to further reduce the required number of stations along the line in all 13 instances with $NW = 2$. Since this flexibility was not considered by Roshani and Giglio [35], it can be seen as a novel contribution to the MALBP-2 literature. Furthermore, by closely analyzing the Tonge family last instance ($Nmax = 23$), one can realize that, even for the same number of workers and stations, PM demonstrated to be superior to RM in terms of cycle time, the former resulting in a solution with 156 time units, while the latter only achieved a solution with 177 time units. It is, in relative terms, a 11.86% improvement generated by the PM.

Synergies between Dec and BDA could be verified in many instances throughout the medium dataset. Although PM and BDA have been able to optimally solve the same instances, CPU time reduction was observed in the most challenging ones. In addition, PM found better integer solutions than BDA in 2 instances, and also yielded good integer solutions to 8 instances in which BDA was not capable of attaining any initial solution. It means an improved result in 29.41% of the medium-size instances when compared to the direct BDA approach. Such improvement can be attributed to Dec feeding the PM with a feasible solution at the beginning of its execution. Nonetheless, also notice that, for those same 8 instances that BDA could not find any feasible solution, PM did not simply maintained the initial solution provided by Dec, but in fact improved all of them in terms of the number of used stations.

The large-size subset from Table 5 contains the last 42 instances to be tested. The comparison between Dec, BDA, PM, and RM when they are applied to such instances is presented in Table 8. The boldfaced values represent 14 results in which the PM has outperformed RM in terms of solution quality for the instances solved and reported in Roshani and Giglio [35]. In total, PM has optimally solved 12 large-sized instances (i.e. 28.57% of them for this subset), whilst reaching integer solutions with gaps inferior to 0.14% (or 0.03% on average) in the remaining 30 ones. Synergies between Dec and BDA were strongly verified this time: (i) out of the 9 instances that both PM and BDA solved to optimality, a reduced CPU time could once again be observed for PM; (ii) out of the remaining 33 instances, BDA was unable to find a feasible solution, whereas PM yielded good integer solutions for all of them, which translates into obtaining improved results for 78.57% of the large-size instances when compared to the direct BDA approach; and (iii) the PM has improved the initial solution found by Dec in 23 instances.

These results convey the efficiency of the PM when applied to the studied problem: similar methods used in isolation in the past for MALBP-1 [23,28,30] were not sufficient to generate solutions as good as the ones produced by the PM, and the known MALBP-2 solutions previously found by RM [35] were greatly improved.

Another feature that could be examined with these tests herein conducted was to evidence the possibility to improve cycle time beyond the SALBP-2 optimal solution limit. For MALBP-1, Michels et al. [28] has stated that the results obtained for the problem's type-1 variant were an indicative that it would be more profitable to accept SALBP-1 optimal solutions as the number of workers, while trying to minimize the line length as much as possible, since no improvement could be verified in the total number of workers throughout the entire dataset. Indeed, such methodology generates great results, as corroborated by Lopes et al. [22,23]. Nevertheless, cycle time improvements could have been confirmed in 12 instances (3 medium and 9 large) for the MALBP-2 dataset, supporting with empirical evidence that the possibility depicted in Fig. 2, Section 3 exists for benchmark instances as well.

The full assignment and scheduling results concerning tasks, workers, and stations were tested for all solutions in order to carry on with a feasibility check. The starting time for each task was evaluated for consistency in regard to station and worker assignments, global cycle times, and order imposed by precedence relations. These assignment files are made available in the paper's supporting information for reproducibility purposes, filling their function of validating the proposed PM's reliability.

Finally, contrary to the BDA applied to MALBP-1 [28], in most cases the PM has a tendency to spend the majority of its available CPU time running the MP. The latter is more similar to the case of the BDA proposed by Naderi et al. [30], which spends more than 98% of its computational processing time solving the MP.

6.3. Real-life assembly plant case study

The PM is lastly applied to the data of an industrial assembly plant originally published by Dimitriadis [12] and its results are compared to those obtained by the RM: Table 9 summarizes the results of such comparison. The number of tasks to be performed in this case study is 64, with a total duration time of 65693 time units. In this study, Roshani and Giglio [35] also fixed the total number of workers, lines with 8 and 10 total workers were tested, allowing at most 2 or 4 workers per station in each case.

According to these results, the PM methodology is once again more effective than RM when it comes to determining lower values of cycle time. In fact, for $NW = 2$, PM reduced the cycle time (improved the line efficiency) for $Nmax = 8$ and $Nmax = 10$ from 8310 (98.80%) to 8212 (99.99%) and from 6650 (98.60%) to 6571 (99.97%), respectively,

Table 7
Results for medium-size instances: comparison between solution methods (PM and RM).

Problem	*Nmax*	*NW*	Dec			BDA			PM							RM		
			CT	St	DCPU	CT	St	CPU	CT	St	Gap	CPU	SCPU	CBC	HT	CT	St	CPU[1]
Kilbridge	4	2	138	3	0.09	138	3	1h	**138**	3	0.02%	1h	1h	1128	1	146	2	9.60
		4	138	3	0.06	138	2	1.95	138	2	0.00%	2.46	2.03	0	1	–	–	–
	6	2	92	5	0.04	92	4	95.74	**92**	4	0.00%	96.66	91.90	5680	837	98	3	8.64
		4	92	5	0.06	92	3	0.11	92	3	0.00%	1.05	0.65	5	1	–	–	–
	8	2	69	5	0.04	69	5	2.35	**69**	5	0.00%	2.68	2.30	185	22	75	4	7.87
		4	69	5	0.04	69	4	0.14	69	4	0.00%	0.55	0.27	10	2	–	–	–
	10	2	56	7	0.03	56	6	0.54	**56**	6	0.00%	0.96	0.26	35	3	60	5	7.47
		4	56	7	0.03	56	5	0.15	56	5	0.00%	0.29	0.04	7	3	–	–	–
	11	2	55	7	0.06	55	6	1h	55	6	1.81%	1h	6.15	6216	992	55	6	7.35
		4	55	7	0.04	55	5	1h	55	5	1.81%	1h	427.77	251370	55578	–	–	–
Tonge	12	2	294	9	0.11	294	7	1h	**294**	7	0.34%	1h	76.59	2106	180	311	6	46.14
		4	294	9	0.11	294	6	1h	294	6	0.34%	1h	68.61	1008	40	–	–	–
		6	294	9	0.11	294	6	1h	294	6	0.34%	1h	609.12	5319	913	–	–	–
	14	2	251	10	0.12	251	8	514.44	**251**	8	0.00%	282.67	3.34	290	43	280	7	33.78
		4	251	10	0.11	251	7	965.28	251	7	0.00%	901.05	225.78	2430	483	–	–	–
		6	251	10	0.11	251	7	773.48	251	7	0.00%	409.16	25.01	690	36	–	–	–
	16	2	221	14	0.16	220	10	733.06	**220**	10	0.00%	360.79	42.96	10346	1231	240	8	33.17
		4	221	14	0.17	220	9	1928.28	220	9	0.00%	1022.80	66.70	5166	487	–	–	–
		6	221	14	0.16	221	7	1h	220	9	0.01%	1h	51.02	4998	298	–	–	–
	18	2	196	14	0.09	–	–	1h	**196**	13	0.51%	1h	6.45	4381	257	225	9	37.41
		4	196	13	0.11	–	–	1h	196	12	0.52%	1h	19.64	5447	276	–	–	–
		6	196	13	0.10	–	–	1h	196	11	0.51%	1h	14.85	4108	183	–	–	–
	19	2	186	14	0.16	–	–	1h	**186**	12	0.53%	1h	5.42	4200	161	206	10	147.49
		4	186	14	0.16	–	–	1h	186	11	0.54%	1h	70.82	7518	607	–	–	–
		6	186	14	0.16	–	–	1h	186	11	0.54%	1h	45.60	7140	385	–	–	–
	20	2	177	14	0.11	177	12	235.86	**177**	12	0.00%	228.95	14.29	9264	1152	197	10	70.78
		4	177	12	0.11	177	11	2479.63	177	11	0.00%	1581.09	63.97	12924	1704	–	–	–
		6	177	12	0.11	177	11	3567.05	177	11	0.00%	2692.13	78.77	14664	1854	–	–	–
	22	2	162	16	0.24	–	–	1h	**162**	15	1.24%	1h	11.45	13536	1206	178	12	117.11
		4	162	16	0.24	–	–	1h	162	11	1.22%	1h	40.16	11632	686	–	–	–
		6	162	16	0.24	162	11	1h	162	11	1.23%	1h	20.44	9392	518	–	–	–
	23	2	156	14	0.09	156	12	1h	**156**	12	0.64%	1h	10.62	23016	5701	177	12	22.75
		4	156	14	0.09	156	10	1h	156	10	1.24%	1h	59.62	25844	2724	–	–	–
		6	156	14	0.09	156	12	1h	156	10	1.25%	1h	22.22	18130	1479	–	–	–

[1] As reported in Roshani and Giglio [35] multiplied by a factor of 0.37.

with the same numbers of workers in both cases and one less stations in the former. Moreover, when $NW = 4$ is allowed, line length is reduced to 3 stations in both situations, with improvements in cycle time (6571 to 6570) and line efficiency (99.97% to 99.99%) for $Nmax = 10$.

7. Conclusions

The type-2 Multi-manned Assembly Line Balancing Problem (MALBP-2) has been addressed in this study. The objective is to minimize the cycle time and the total number of stations as ranked goals. The existing literature on MALBP-2 was very limited, and efficient exact solution methods were only available for the type-1 variant (MALBP-1). This paper's main contribution was presenting an innovative method to optimality solve larger MALBP-2 instances by decomposing the original problem. It proposed an initial solution procedure and implemented a Benders' decomposition algorithm with the application of multiple combinatorial Benders' cuts during its execution. The results showed that the proposed algorithm is very efficient in comparison to an adapted version of an exact method for the MALBP-1 and to a metaheuristic for the MALBP-2.

In order to solve the optimization problem, a new Mixed-Integer Linear Programming (MILP) model was developed (Section 4.1), along with several symmetry break constraints (i.e. valid inequalities, Section 4.2) and strong bounds (Section 4.3). The proposed formulation (PF) outperformed previous monolithic mathematical formulations in terms of solution quality and computational processing time (Section 6.1). By studying MALBP-1 specific solutions methods, it was possible to infer that the type-2 variant was also hierarchically divisible into a Master Problem (MP) and a group of Slave Problems (SPs). By reformulating the original monolithic model and using problem specific knowledge, a new Benders' Decomposition Algorithm (BDA) with an initial feasible solution procedure is forged (Section 5). The solutions obtained by the proposed method (PM) were compared to previously developed methods by applying the PM to an extended benchmark dataset and a real-life assembly plant case study (Section 6.1). From the 46 instances that an integer solution was known in the literature, the PM was able to produce improved results in most cases, totaling 38 new integer solutions, of which 29 were proven optimal. In total, 60 optimal solutions were obtained out of a dataset with 108 instances, resulting in 55.56% of optimal solutions in the proposed dataset (Section 6.2). Lastly, better integer solutions were found in all case study instances (4), of which 2 were deemed optimal (Section 6.3).

Industries manufacturing large-size products often allow multiple tasks to be simultaneously performed by different workers in the same station. Given the resources, the managers aim to achieve the best possible productivity level. This is a notable realistic feature, which is viewed as a natural extension of the problem's simpler version and is widely employed in the literature. Nonetheless, incorporating more practical extensions is a desirable modification. For instance, equipment selection, worker heterogeneity, product variability (multi and mixed-model lines), and line layouts (U-line, parallel stations) could be valuable features to be added to the PM. Further research should concentrate in doing so, as well as implementing balancing and project scheduling heuristics for the master and slave problems to mitigate computational burden.

Table 8
Results for large-size instances: comparison between solution methods (PM and RM).

Problem	*Nmax*	*NW*	Dec			BDA			PM							RM		
			CT	St	DCPU	CT	St	CPU	CT	St	Gap	CPU	SCPU	CBC	HT	CT	St	CPU[1]
Arcus1	11	2	7084	9	0.09	–	–	1h	**7084**	9	0.00%	897.11	58.97	171	7	7909	6	311.65
		4	7084	9	0.09	–	–	1h	7084	9	0.00%	710.37	59.31	189	9	–	–	–
		6	7084	9	0.09	–	–	1h	7084	9	0.00%	1911.08	62.45	153	9	–	–	–
	12	2	6412	11	0.10	–	–	1h	**6412**	11	0.02%	1h	57.36	693	17	7688	6	83.51
		4	6412	11	0.10	–	–	1h	6412	10	0.03%	1h	87.91	220	5	–	–	–
		6	6412	11	0.10	–	–	1h	6412	10	0.03%	1h	58.17	253	9	–	–	–
	14	2	5441	12	0.08	5441	10	136.83	**5441**	10	0.00%	124.42	1.92	180	5	6412	8	449.18
		4	5441	11	0.09	5441	10	193.60	5441	10	0.00%	181.12	121.97	1012	18	–	–	–
		6	5441	11	0.08	5441	10	260.13	5441	10	0.00%	164.33	11.72	0	4	–	–	–
	15	2	5104	11	0.20	5104	10	347.04	**5104**	10	0.00%	221.40	48.24	1309	85	6194	8	81.15
		4	5104	11	0.29	5104	9	1658.65	5104	9	0.00%	996.66	115	913	48	–	–	–
		6	5104	11	0.19	5104	9	1413.57	5104	9	0.00%	1270.48	350.75	1287	69	–	–	–
	18	2	4317	17	1.46	–	–	1h	**4317**	17	0.14%	1h	0.42	102	4	4811	10	85.61
		4	4317	17	1.45	–	–	1h	4317	17	0.14%	1h	0.16	136	3	–	–	–
		6	4317	17	1.45	–	–	1h	4317	16	0.14%	1h	0.16	85	5	–	–	–
	19	2	4068	16	0.32	–	–	1h	**4068**	16	0.03%	1h	1.73	608	5	4494	12	61.93
		4	4068	16	0.32	–	–	1h	4068	16	0.03%	1h	11.28	912	14	–	–	–
		6	4068	16	0.31	–	–	1h	4068	16	0.03%	1h	8.88	496	23	–	–	–
	21	2	3691	18	3.20	3691	14	1144.26	**3691**	14	0.00%	275.22	5.24	3924	175	4320	12	252.04
		4	3691	17	3.20	3691	14	1155.77	3691	14	0.00%	555.81	16.80	2822	58	–	–	–
		6	3691	17	3.20	3691	14	1240.87	3691	14	0.00%	187.77	7.11	527	29	–	–	–
Arcus2	12	2	12534	9	9.25	–	–	1h	**12534**	9	0.01%	1h	3412.53	2835	57	13350	6	158.48
		4	12534	9	9.60	–	–	1h	12534	9	0.01%	1h	3277.14	1800	20	–	–	–
		6	12534	9	9.66	–	–	1h	12534	9	0.01%	1h	3241.73	2106	67	–	–	–
	16	2	9412	12	6.75	–	–	1h	**9412**	12	0.01%	1h	0.01	0	12	10889	8	219.91
		4	9412	12	6.80	–	–	1h	9412	12	0.01%	1h	0.01	0	12	–	–	–
		6	9412	12	6.76	–	–	1h	9412	12	0.01%	1h	34.62	300	16	–	–	–
	18	2	8377	14	1’	–	–	1h	**8377**	14	0.02%	1h	211.07	840	23	9726	10	193.71
		4	8377	13	1’	–	–	1h	8377	11	0.02%	1h	1197.72	11414	48	–	–	–
		6	8377	13	1’	–	–	1h	8377	11	0.02%	1h	321.29	1001	10	–	–	–
	23	2	6561	17	1’	–	–	1h	**6560**	17	0.03%	1h	1815.44	16150	48	7467	13	161.03
		4	6561	17	1’	–	–	1h	6560	17	0.06%	1h	974.15	7871	29	–	–	–
		6	6561	17	1’	–	–	1h	6560	17	0.05%	1h	1428.07	9554	37	–	–	–
	24	2	6313	18	1’	–	–	1h	**6310**	17	0.02%	1h	2635.00	17280	75	7235	13	793.65
		4	6313	18	1’	–	–	1h	6310	17	0.03%	1h	2661.06	15750	121	–	–	–

[1] As reported in Roshani and Giglio [35] multiplied by a factor of 0.37.

Table 9
Results for real-life case study instances: comparison between solution methods (PM and RM).

Problem	*Nmax*	*NW*	PM								RM			
			CT	St	LE	CPU	DCPU	SCPU	CBC	HT	CT	St	LE	CPU[1]
Dimitriadis	8	2	8212	4	99.99%	7.27	9.66	5.52	0	3	8310	5	98.80%	–
		4	8212	3	99.99%	1h	10.01	1h	108	2	–	–	–	–
	10	2	6571	5	99.97%	1h	1’	95.29	1945	155	6650	5	98.60%	–
		4	6570	3	99.99%	191.55	1’	191.49	6	4	–	–	–	–

[1]Not reported in Roshani and Giglio [35].

CRediT authorship contribution statement

Adalberto Sato Michels: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Visualization. **Thiago Cantos Lopes:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - review & editing, Visualization. **Leandro Magatão:** Conceptualization, Validation, Resources, Writing - review & editing, Supervision, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors would like to thank the financial support from Fundação Araucária (Agreement 041/2017 FA–UTFPR–RENAULT), and CNPq (Grants 406507/2016-3 and 307211/2017-7). This study was also financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.orp.2020.100163.

References

[1] Akagi F, Osaki H, Kikuchi S. A method for assembly line balancing with more than one worker in each station. Int J Prod Res 1983;21(5):755–70. https://doi.org/10.

1080/00207548308942409.
[2] Akpinar S, Elmi A, Bekta T. Combinatorial Benders cuts for assembly line balancing problems with setups. Eur J Oper Res 2017;259(2):527–37. https://doi.org/10.1016/j.ejor.2016.11.001.
[3] Bartholdi JJ. Balancing two-sided assembly lines: a case study. Int J Prod Res 1993;31(10):2447–61. https://doi.org/10.1080/00207549308956868.
[4] Battaïa O, Dolgui A. A taxonomy of line balancing problems and their solution approaches. Int J Prod Econ 2013;142(2):259–77. https://doi.org/10.1016/j.ijpe.2012.10.020.
[5] Bautista J, Pereira J. A dynamic programming based heuristic for the assembly line balancing problem. Eur J Oper Res 2009;194(3):787–94. https://doi.org/10.1016/j.ejor.2008.01.016.
[6] Baybars . A survey of exact algorithms for the simple assembly line balancing problem. Manage Sci 1986;32(8):909–32. https://doi.org/10.1287/mnsc.32.8.909.
[7] Becker C, Scholl A. A survey on problems and methods in generalized assembly line balancing. Eur J Oper Res 2006;168(3):694–715. https://doi.org/10.1016/j.ejor.2004.07.023.
[8] Becker C, Scholl A. Balancing assembly lines with variable parallel workplaces: problem definition and effective solution procedure. Eur J Oper Res 2009;199(2):359–74. https://doi.org/10.1016/j.ejor.2008.11.051.
[9] Benders JF. Partitioning procedures for solving mixed-variables programming problems. Numer Math 1962;4(1):238–52. https://doi.org/10.1007/BF01386316.
[10] Chen YY. A hybrid algorithm for allocating tasks, operators, and workstations in multi-manned assembly lines. J Manuf Syst 2017;42:196–209. https://doi.org/10.1016/j.jmsy.2016.12.011.
[11] Codato G, Fischetti M. Combinatorial Benders' cuts for mixed-integer linear programming. Oper Res 2006;54(4):756–66. https://doi.org/10.1287/opre.1060.0286.
[12] Dimitriadis SG. Assembly line balancing and group working: a heuristic procedure for workers' groups operating on the same product and workstation. Comput Oper Res 2006;33(9):2757–74. https://doi.org/10.1016/j.cor.2005.02.027.
[13] Fattahi P, Roshani A. A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem. Int J Adv ManufTechnol 2011;53:363–78. https://doi.org/10.1007/s00170-010-2832-y.
[14] Hazir O, Dolgui A. Assembly line balancing under uncertainty: robust optimization models and exact solution method. Comput Ind Eng 2013;65:261–7. https://doi.org/10.1016/j.cie.2013.03.004.
[15] Hazir O, Dolgui A. A decomposition based solution algorithm for U-type assembly line balancing with interval data. Comput Oper Res 2015;59:126–31. https://doi.org/10.1016/j.cor.2015.01.010.
[16] Hoffmann TR. Assembly line balancing with a precedence matrix. Manage Sci 1963;9(4):551–62. https://doi.org/10.1287/mnsc.9.4.551.
[17] Kazemi A, Sedighi A. A cost-oriented model for balancing mixed-model assembly lines with multi-manned workstations. Int J Serv OperManage 2013;16(3):289. https://doi.org/10.1504/IJSOM.2013.056765.
[18] Kellegöz T. Assembly line balancing problems with multi-manned stations: a new mathematical formulation and Gantt based heuristic method. Annals of Oper Res 2017;253(1):377–404. https://doi.org/10.1007/s10479-016-2156-x.
[19] Kellegöz T, Toklu B. An efficient branch and bound algorithm for assembly line balancing problems with parallel multi-manned workstations. Comput Oper Res 2012;39(12):3344–60. https://doi.org/10.1016/j.cor.2012.04.019.
[20] Kellegöz T, Toklu B. A priority rule-based constructive heuristic and an improvement method for balancing assembly lines with parallel multi-manned workstations. Int J Prod Res 2015;53(3):736–56. https://doi.org/10.1080/00207543.2014.920548.
[21] Klein R. Scheduling of resource-constrained projects. 1st ed. Springer US; 2000. https://doi.org/10.1007/978-1-4615-4629-0.
[22] Lopes TC, Michels AS, Magatão L. A note to: a hybrid algorithm for allocating tasks, operators, and workstations in multi-manned assembly lines. J Manuf Syst 2019;52:205–8. https://doi.org/10.1016/j.jmsy.2019.06.003.
[23] Lopes TC, Pastre GV, Michels AS, Magatão L. Flexible multi-manned assembly line balancing problem: model, heuristic procedure, and lower bounds for line length minimization. Omega (United Kingdom) 2019. https://doi.org/10.1016/j.omega.2019.04.006.
[24] Lopes TC, Sikora CGS, Molina RG, Schibelbain D, Rodrigues LCA, Magatão L. Balancing a robotic spot welding manufacturing line: an industrial case study. Eur J Oper Res 2017;263(3):1033–48. https://doi.org/10.1016/j.ejor.2017.06.001.
[25] Magnanti TL, Wong RT. Accelerating benders decomposition: algorithmic enhancement and model selection criteria. Oper Res 1981;29(3):464–84. https://doi.org/10.2307/170108.
[26] Maurer WD, Lewis TG. Hash table methods. ACM Comput Surv (CSUR) 1975;7(1):5–19. https://doi.org/10.1145/356643.356645.
[27] Michels AS, Lopes TC, Sikora CGS, Magatão L. The robotic assembly line design (RALD) problem: model and case studies with practical extensions. Comput Ind Eng 2018;120(February):320–33. https://doi.org/10.1016/j.cie.2018.04.010.
[28] Michels AS, Lopes TC, Sikora CGS, Magatão L. A Benders' decomposition algorithm with combinatorial cuts for the multi-manned assembly line balancing problem. Eur J Oper Res 2019;278(3):796–808. https://doi.org/10.1016/j.ejor.2019.05.001.
[29] Moreira MCO, Miralles C, Costa AM. Model and heuristics for the assembly line worker integration and balancing problem. Comput Oper Res 2015;54:64–73. https://doi.org/10.1016/j.cor.2014.08.021.
[30] Naderi B, Azab A, Borooshan K. A realistic multi-manned five-sided mixed-model assembly line balancing and scheduling problem with moving workers and limited workspace. Int J Prod Res 2019;57(3):643–61. https://doi.org/10.1080/00207543.2018.1476786.
[31] Optimization, G. (2019). Gurobi optimizer reference manual.
[32] Osman H, Baki MF. Balancing transfer lines using Benders decomposition and ant colony optimisation techniques. Int J Prod Res 2014;52(5):1334–50. https://doi.org/10.1080/00207543.2013.842017.
[33] Pape T. Heuristics and lower bounds for the simple assembly line balancing problem type 1: overview, computational tests and improvements. Eur J Oper Res 2015;240:32–42. https://doi.org/10.1016/j.ejor.2014.06.023.
[34] Rahmaniani R, Crainic TG, Gendreau M, Rei W. The Benders decomposition algorithm: a literature review. Eur J Oper Res 2017;259(3):801–17. https://doi.org/10.1016/j.ejor.2016.12.005.
[35] Roshani A, Giglio D. Simulated annealing algorithms for the multi-manned assembly line balancing problem: minimising cycle time. Int J Prod Res 2017;55(10):2731–51. https://doi.org/10.1080/00207543.2016.1181286.
[36] Roshani A, Nezami FG. Mixed-model multi-manned assembly line balancing problem: a mathematical model and a simulated annealing approach. Assembly Autom 2017;37(1):34–50. https://doi.org/10.1108/AA-02-2016-016.
[37] Roshani A, Roshani A, Roshani A, Salehi M, Esfandyari A. A simulated annealing algorithm for multi-manned assembly line balancing problem. J Manuf Syst 2013;32(1):238–47. https://doi.org/10.1016/j.jmsy.2012.11.003.
[38] Sahin M, Kellegöz T. A new mixed-integer linear programming formulation and particle swarm optimization based hybrid heuristic for the problem of resource investment and balancing of the assembly line with multi-manned workstations. Comput Ind Eng 2019;133(March):107–20. https://doi.org/10.1016/j.cie.2019.04.056.
[39] Sahin M, Kellegöz T. Balancing multi-manned assembly lines with walking workers: problem definition, mathematical formulation, and an electromagnetic field optimisation algorithm. Int J Prod Res 2019;7543. https://doi.org/10.1080/00207543.2019.1566672.
[40] Scholl A, Becker C. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur J Oper Res 2006;168(3):666–93. https://doi.org/10.1016/j.ejor.2004.07.022.
[41] Scholl A, Klein R. SALOME: A bidirectional branch-and-bound procedure for assembly line balancing. INFORMS J Comput 1997;9(4):319–34. https://doi.org/10.1287/ijoc.9.4.319.
[42] Sewell EC, Jacobson SH. A branch, bound, and remember algorithm for the simple assembly line balancing problem. INFORMS J Comput 2012;24(3):433–42. https://doi.org/10.1287/ijoc.1110.0462.
[43] Sikora CGS, Lopes TC, Magatão L. Traveling worker assembly line (re)balancing problem: model, reduction techniques, and real case studies. Eur J Oper Res 2017;259:949–71. https://doi.org/10.1016/j.ejor.2016.11.027.
[44] Sternatz J. Enhanced multi-Hoffmann heuristic for efficiently solving real-world assembly line balancing problems in automotive industry. Eur J Oper Res 2014;235:740–54. https://doi.org/10.1016/j.ejor.2013.11.005.
[45] Walsh T. General symmetry breaking constraints. Principles and practice of constraint programming - CP. 4204. 2006. p. 650–64. https://doi.org/10.1007/11889205_46.
[46] Yadav A, Kumar S, Agrawal S. Reconfiguration of assembly line balancing–an automobile case study solved by the exact solution procedure. Benchmarking 2020:1–29. https://doi.org/10.1108/BIJ-08-2019-0386. In press
[47] Yazgan HR, Beypinar I, Boran S, Ocak C, et al. A new algorithm and multi-response Taguchi method to solve line balancing problem in an automotive industry. Int J Adv ManufTechnol 2011;57:379–92. https://doi.org/10.1007/s00170-011-3291-9.
[48] Yilmaz H, Yilmaz M. Multi-manned assembly line balancing problem with balanced load density. Assembly Autom 2015;35(1):137–42. https://doi.org/10.1108/AA-05-2014-041.
[49] Yilmaz H, Yilmaz M. A multi-manned assembly line balancing problem with classified teams: a new approach. Assembly Autom 2016;36(1):51–9. https://doi.org/10.1108/AA-04-2015-035.
[50] Yilmaz H, Yilmaz M. Note to: a mathematical model and ant colony algorithm for multi-manned assembly line balancing problem. Int J Adv ManufTechnol 2016;89(5–8):1935–9. https://doi.org/10.1007/s00170-016-9223-y.
[51] Yilmaz H, Yilmaz M. A mathematical model and tabu search algorithm for multi-manned assembly line balancing problems with assignment restrictions. Eng Optim 2020;52(5):856–74. https://doi.org/10.1080/0305215X.2019.1618288.