Accepted Manuscript

A Benders' decomposition algorithm with combinatorial cuts for the multi-manned assembly line balancing problem

Adalberto Sato Michels, Thiago Cantos Lopes,
Celso Gustavo Stall Sikora, Leandro Magatão

PII: S0377-2217(19)30384-4
DOI: https://doi.org/10.1016/j.ejor.2019.05.001
Reference: EOR 15799

To appear in: *European Journal of Operational Research*

Received date: 21 September 2018
Revised date: 1 May 2019
Accepted date: 1 May 2019

Please cite this article as: Adalberto Sato Michels, Thiago Cantos Lopes, Celso Gustavo Stall Sikora, Leandro Magatão, A Benders' decomposition algorithm with combinatorial cuts for the multi-manned assembly line balancing problem, *European Journal of Operational Research* (2019), doi: https://doi.org/10.1016/j.ejor.2019.05.001

Highlights

- The Multi-manned Assembly Line Balancing Problems is studied and solved
- Strong symmetry break constraints are implemented in a new mixed-integer formulation
- A Benders' decomposition algorithm is proposed to solve medium and large-size cases
- The proposed algorithm solved 117 out of 131 instances of the benchmark to optimality
- Compared to previously presented methods, 44 new best solutions were reached

A Benders' decomposition algorithm with combinatorial cuts for the multi-manned assembly line balancing problem

Adalberto Sato Michels[a], Thiago Cantos Lopes[a], Celso Gustavo Stall Sikora[a,b], Leandro Magatão[a*]

a: Graduate Program in Electrical and Computer Engineering (CPGEI)

Federal University of Technology – Paraná (UTFPR), Curitiba, Brazil

b: Institute for Operations Research, Hamburg Business School

University of Hamburg, Hamburg, Germany

Abstract

Multi-manned assembly lines are commonly found in industries that manufacture large-size products (e.g. automotive industry), in which multiple workers are assigned to the same station in order to perform different operations simultaneously on the same product. Although the balancing problem of multi-manned assembly lines had been modelled before, the previously presented exact mathematical formulations are only able to solve few small-size instances, while larger cases are solved by heuristics or metaheuristics that do not guarantee optimality. This work presents a new Mixed-Integer Linear Programming model with strong symmetry break constraints and decomposes the original problem into a new Benders' Decomposition Algorithm to solve large instances optimally. The proposed model minimises the total number of workers along the line and the number of opened stations as weighted primary and secondary objectives, respectively. Besides, feasibility cuts and symmetry break constraints based on combinatorial Benders' cuts and model's parameters are applied as lazy constraints to reduce search-space by eliminating infeasible sets of allocations. Tests on a literature dataset have shown that the proposed mathematical model outperforms previously developed formulations in both solution quality and computational processing time for small-size instances. Moreover, the proposed Benders' Decomposition Algorithm yielded 117 optimal results out of a 131-instances dataset. Compared to previously presented methods, this translates to 19 and 25 new best solutions reached for medium and large-size instances, respectively, of which 19 and 23 are optimal solutions.

Keywords: Combinatorial optimisation; Multi-manned assembly line balancing; Benders' decomposition; Combinatorial Benders' cuts; Mixed-integer linear programming

1. Introduction

Production systems used in high-volume industries of standardised products are frequently based on flow-shop layouts, which are product-oriented designs. Assembly lines in a flow-shop configuration are generally dedicated to make homogeneous products, enabling their mass production. Along with

*Corresponding author

Email address: magatao@utfpr.edu.br (Leandro Magatão[a])

this real-world usage, assembly lines have given rise to a combinatorial problem widely discussed in the literature (Battaïa & Dolgui, 2013): the Assembly Line Balancing Problem (ALBP).

Considering several restrictive assumptions described by Baybars (1986), the problem of assigning a list of tasks subjected by a precedence graph to stations is called Simple Assembly Line Balancing Problem (SALBP). Allowing only one worker in each station is one of the restrictions. Moreover, these stations are organised in a straight, serial line that produces a unique model of a single product. The importance of ALBP is shown in the literature by the high number of published papers that still contribute to practical applications. In order to optimise (minimise) the number of stations (SALBP-1) or the cycle time (SALBP-2), several algorithmic solution methods were proposed: SALOME – an efficient bidirectional branch-and-bound procedure – was developed (Scholl & Klein, 1997) followed by a dynamic programming approach (Bautista & Pereira, 2009), a branch, bound, and remember algorithm (Sewell & Jacobson, 2012), and an enhanced multi-Hoffmann heuristic (Sternatz, 2014). These and other techniques were gathered in an overview and improved for SALBP-1 by Pape (2015).

However, assembly lines applied to automotive industry, for instance, commonly process large-size products, such as cars and buses. In these lines, the SALBP's hypothesis of allowing only one worker in each station often is not a practical limitation. As product size is rather large, it becomes admissible to assign more than one worker to each station and perform tasks simultaneously in different sectors of the same product, giving rise to natural extensions and more generalised versions of the SALBP: the Multi-manned Assembly Line Balancing Problem (MALBP) and the Two-sided Assembly Line Balancing Problem (TALBP), which are surveyed by Becker & Scholl (2006) along with a variety of practical extensions. Figure 1 depicts both above-mentioned lines, it shows simple, two-sided, and multi-manned assembly lines with three stations each. Nevertheless, two-sided and multi-manned assembly lines permit more than one worker in each station (i.e. stations 1 and 3), with workers performing tasks on the same product at the same time. The main difference between MALBP and TALBP is the flexibility on the quantity of workers and their positioning. TALBP allows at most two operators, each of them at the station's right or left side, whereas in MALBPs the number of maximum workers depends on product's attributes, such as size, structure, and tasks' precedence relations. Another divergence is that TALBPs might have to deal with tasks that can be performed exclusively on the right or left side of the product.

This work focuses on the MALBP variant; advantages of using the multi-manned configuration are associated to workforce and line length reduction, which are further exemplified in Section 2. Other simplification hypotheses from SALBP are kept, the most important ones to be mentioned are: (i) a straight, serial line is considered and (ii) the line produces a unique model of a single product.

In industrial environments, the use of multi-operated stations is intense. Consequently, numerous studies concerning MALBPs and TALBPs have recently been elaborated. To the best of the authors' knowledge, the Parallel Assignment Method (PAM) developed by Akagi et al. (1983) takes place as the first study in the literature to tackle the problem of achieving higher production rates in assembly lines with more than one worker in each station. Many years later, Dimitriadis (2006) proposed a heuristic method based on modifying a procedure created by Hoffmann (1963). The heuristic has shown to be

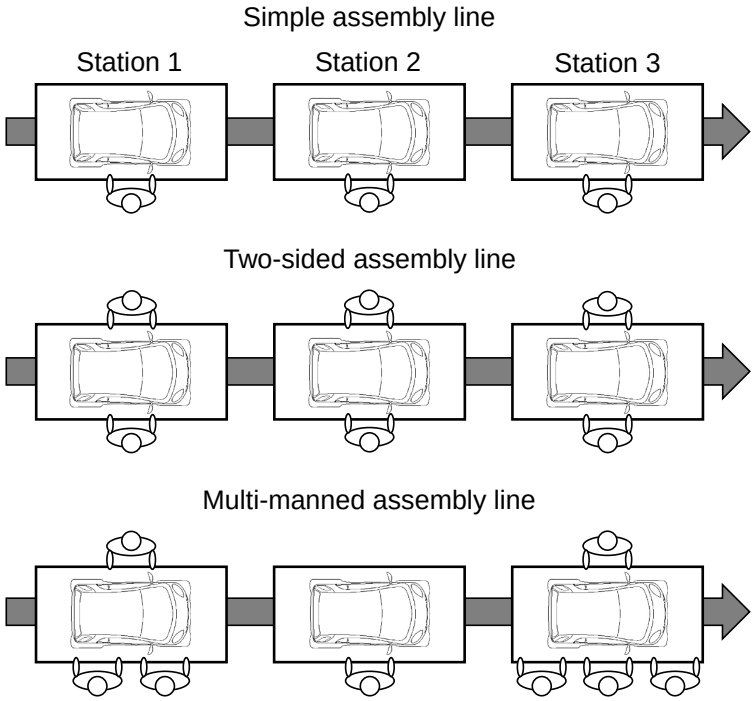

Figure 1: Configuration examples of a simple assembly line, a two-sided assembly line, and a multi-manned assembly line.

effective in enhancing space utilisation, with the objective of minimising the total number of workers and stations given a cycle time, which is still the most usual goal function employed in various works. Succeeding those papers, Becker & Scholl (2009) introduced the Assembly Line Balancing Problem with Variable Workplaces (VWALBP). In this problem, working areas are minimised given a cycle time, while work-pieces are divided into mounting positions and only a single worker is able to assemble them in each multi-manned station. A Mixed-Integer Linear Programming (MILP) model is generated including lower bounding techniques and a branch-and-bound algorithm named VWSolver (based on SALOME) is implemented to solve larger instances. Concomitantly, two-sided assembly lines were firstly explored by Bartholdi (1993), and its variants concerning mixed-model lines (Özcan & Toklu, 2009) and stochastic task times (Özcan, 2010) were further developed.

Following those publications, works on MALBPs have been increasing yearly; Moon et al. (2009) included the feature of variably skilled workers into MALBPs, proposed a mathematical formulation, and solved large-size instances with a Genetic Algorithm (GA). Cevikcan et al. (2009) devised the application of multi-manned stations for mixed-model assembly lines with zoning constraints. Due to the complexity of the mathematical model, a five-phase heuristic was developed to solve it. However, none of them used exact algorithms, and their adopted methods would find near-optimal feasible solutions. The first mathematical model that minimises the total number of workers and stations simultaneously in a MALBP was proposed by Fattahi & Roshani (2011). They consider a single model line, in which the number of workers and stations are the primary and secondary objectives in the optimisation procedure, respectively. Their model could solve small-size instances in a reasonable amount of time, but failed in solving larger cases. For that reason, an Ant Colony Optimisation

(ACO) algorithm has been developed to find feasible and near-optimal solutions for medium and large test problems. A novel efficient branch-and-bound algorithm called Jumper was developed by Kellegöz & Toklu (2012) to solve ALBPs with parallel multi-manned stations. Their algorithm outperforms the VWSolver in both quality of feasible solutions and computational processing times. Kazemi & Sedighi (2013) and Michels et al. (2018) examine real-size cost-oriented problem instances for assembly lines with multi-operated stations: the first paper takes into consideration the objective of minimising total cost per production unit by presenting a heuristic method based on GA, whilst the latter one develops a MILP model to minimise design implementation costs (robots, station facilities, and equipment) of a robotic line that conceives the use of multiple robots per station. Roshani et al. (2013) addressed the MALBP with a multi-objective function in their mathematical model, which maximises smoothness index and line efficiency, whereas minimising the line length. Moreover, an improved Simulated Annealing (SA) algorithm was proposed to solve the problem. Kellegöz & Toklu (2015) presented a constructive heuristic based on priority rules, a GA based improvement procedure, and conducted computational experiments on MALBP instances with the objective of minimising the total number of workers in the line. Yilmaz & Yilmaz (2015) aimed at the minimisation of number of workers, stations, and workload difference between workers with a mathematical formulation. Yilmaz & Yilmaz (2016a) also analysed the impacts of MALBPs with skilled workers and equipment needs, and for that a heuristic procedure was proposed for solving the problem. Roshani & Giglio (2017) approached the MALBP by trying to minimise the cycle time of a line as the primary objective, for a given number of stations. Besides the MILP model, two meta-heuristics based on SA algorithm were developed: the indirect and direct SA (ISA and DSA, respectively). The DSA performance in solving the problem showed to be better in terms of quality and computational time. In order to reduce the required workspace for shop operations, Chen (2017) developed a hybrid heuristic approach based on SA algorithms with specific practical extensions for the automotive industry, prioritising the minimisation of stations. Kellegöz (2017) has improved the mathematical formulation proposed by Fattahi & Roshani (2011) to minimise the total number of workers and stations in a MALBP. In addition, a Gantt-based heuristic is proposed within a SA algorithm to solve medium and large-size instances. This procedure outperforms the ACO algorithm presented by Fattahi & Roshani (2011) and finds better feasible solutions to most instances in the tested benchmark. Lastly, another SA algorithm is implemented by Roshani & Nezami (2017), this time to undertake the mixed-model MALBP with the minimisation of number of workers and stations as primary and secondary objectives, respectively.

Table 1 provides a summarised literature review and, by comparing the proposed method with previously published papers, it is possible to situate the proposed work's contribution into the literature. Although Becker & Scholl (2009) and Kellegöz & Toklu (2012) have developed exact methods to solve ALBPs with parallel workplaces, they had only considered the minimisation of working areas and the number of worker as a consequence, which is a different concept. The main concern from the literature appears to be problems with single model lines in which both total number of workers and stations are minimised. This is due to the reason that, as stated in several published articles (e.g. Fattahi & Roshani (2011); Kellegöz (2017); Roshani & Nezami (2017)), minimising the number

of workers might be more important reducing the number of stations. For that, mathematical models were developed along with heuristic (Yilmaz & Yilmaz, 2016a), genetic algorithm (Moon et al., 2009), ant colony optimisation (Fattahi & Roshani, 2011; Yilmaz & Yilmaz, 2016b), and simulated annealing (Roshani et al., 2013; Kellegöz, 2017) methods. However, none of these methods can guarantee optimality for medium and large-size instances. In order to fill such gap, a new mathematical formulation is developed with search-space reduction constraints and symmetry breaks to address the problem. Furthermore, a Benders' decomposition algorithm is proposed as an innovative exact method for the problem under study. By applying Benders' combinatorial cuts (Benders, 1962; Codato & Fischetti, 2006) techniques as lazy constraints, larger benchmark instances can be solved to optimality. Differently from the classical Benders' decomposition, the proposed algorithm presents an integer slave problem intended for feasibility seeking. These works presented in Table 1, in particular Fattahi & Roshani (2011) and Kellegöz (2017), will serve as a benchmark for this paper and the decomposition procedure herein proposed, which focus on minimising the total number of workers as the primary objective and the number of stations as the secondary one in a MALBP. In this way, a direct performance comparison of the objective function results is possible for each instance. Besides, these works (Fattahi & Roshani (2011) and Kellegöz (2017)) are the most recent ones concerning MALBPs with such minimisation objective and they also provide an extensive dataset to validate the proposed model and algorithm.

Table 1: Literature overview for the Multi-manned Assembly Line Balancing Problem (MALBP).

	Product diversity		Goal function				Solution method					
Author(s) (Year)	Single-model	Mixed-model	Production rate	Number of workers	Number of stations	Cost-oriented	Mathematical formulation	Heuristic	Genetic algorithm	Ant colony optimisation	Simulated annealing	Exact method
Akagi et al. (1983)	•		•					•				
Dimitriadis (2006)	•			•				•				
Becker & Scholl (2009)	•			•				•				•
Moon et al. (2009)	•			•	•		•		•			
Cevikcan et al. (2009)		•	•				•	•				
Fattahi & Roshani (2011)	•			•	•		•			•		
Kellegöz & Toklu (2012)	•			•								•
Kazemi & Sedighi (2013)		•				•	•		•			
Roshani et al. (2013)	•			•	•						•	
Kellegöz & Toklu (2015)	•			•			•	•				
Yilmaz & Yilmaz (2015)	•			•	•		•					
Yilmaz & Yilmaz (2016a)	•			•	•		•	•				
Roshani & Giglio (2017)	•		•				•				•	
Chen (2017)	•			•	•		•				•	
Kellegöz (2017)	•			•	•		•				•	
Roshani & Nezami (2017)		•		•	•		•				•	
Proposed paper (2018)	•			•	•		•					•

The remaining of the paper is organised as follows. In Section 2, a deeper definition of MALBP is given in order to explain the problem. Section 3 presents the MILP model and the additional constraints to reduce the problem's search-space. Section 4 reviews the development and applications of Benders' decomposition and combinatorial cuts. It also describes in detail the proposed algorithm. Computational results retrieved from this study are presented and discussed in Section 5. Lastly, in Section 6, concluding remarks are summarised and further research directions are suggested.

2. Problem statement

As mentioned, the assembly lines considered in this study are dedicated to mass production of a single model of a unique product. Their stations are positioned sequentially in a serial, straight line. Only one work-piece can be processed at a given time in each station. Contrary to unpaced and mixed-model lines, in which processing time oscillations, starvations, and blockages are relevant factors (Lopes et al., 2018), these pieces are moved forward between stations with a previously known and fixed cycle time (CT), while their transportation times are neglected. As the line produces a single product, its pace is exclusively determined by the most loaded station or the defined cycle time (Baybars, 1986).

In order to assemble any product, a set of tasks T must be performed. These tasks are indivisible and must respect precedence restrictions in their execution order. Each of them takes a deterministic duration time (D_t) to be completed, thus, the sum of these duration times assigned to the same worker must not exceed the defined cycle time. Nevertheless, due to parallel work within stations, it is also necessary for tasks to be scheduled in such a way that precedence relations are respected.

A sample instance is used to illustrate the difference of SALBP and MALBP. The precedence graph containing task indexes (number inside the circle) and durations (value on the top right corner of the circle), as well as optimal solutions for a SALBP and MALBP with a defined $CT = 10$ are presented in Figure 2. For the multi-manned line, each station is allowed to be occupied by more than one worker simultaneously working on the same work-piece. However, the maximum number of operators (NW, with $NW = 3$ for the illustrative instance) admitted to perform different tasks concomitantly may vary due to the product size. The configuration of SALBP's optimal solution necessitates 6 workers assigned to 6 stations, totalling an idle time of 10 time units, or approximately 16.67% of the line's available working time, represented by the blank spaces in each station. On the other hand, by permitting more than one worker per station, the MALBP solution was able to not only reduce the line length from 6 to 2 stations, but also assign 5 workers instead of 6 to perform the task set. In this illustrative instance, it was possible to verify an advantage of multi-manned lines over simple ones by bringing the idle time down to zero. This efficiency improvement arises from allowing multiple workers to perform tasks simultaneously. Naturally, it depends on the instance's parameters.

Nonetheless, this viable advantage comes along with the drawback of computational burden in solving the problem. Notice that the task indivisibility attribute is still valid and must be respected. Into the same station, each worker can only execute at most one task at a given time, and no cooperation is allowed between workers, i.e. no common task (Yazgan et al., 2011; Sikora et al., 2017) can

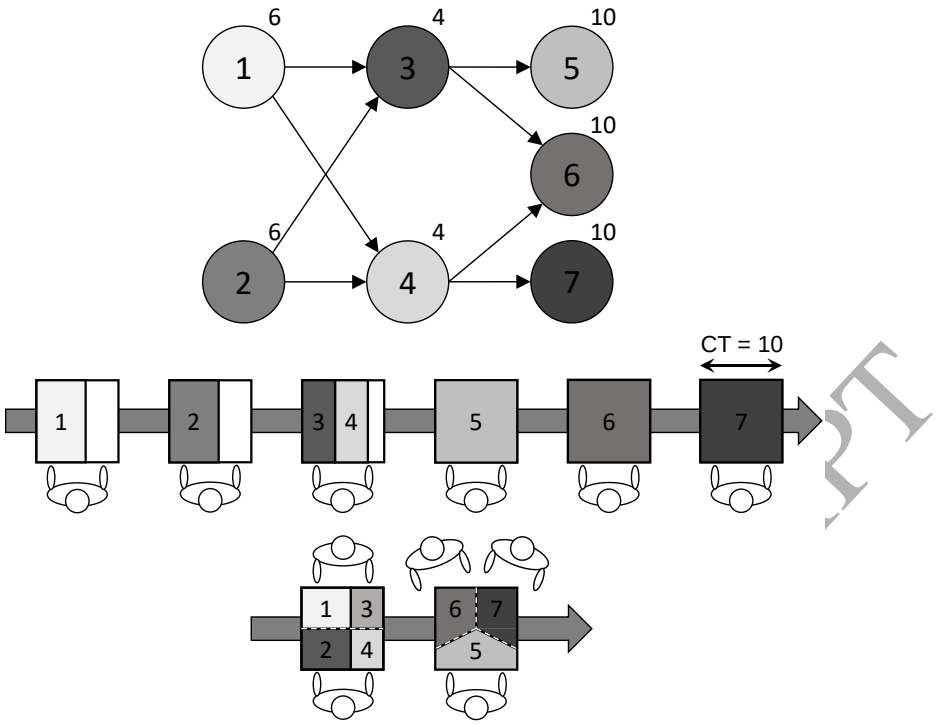

Figure 2: Precedence graph, SALBP, and MALBP optimal solutions for the illustrative instance.

be performed by two or more workers together. Furthermore, tasks are not constrained by positioning and zoning restriction (Bartholdi, 1993; Becker & Scholl, 2009), i.e. no interference occurs between workers during the assembly process (Lopes et al., 2017). Lastly, there is no heterogeneity among workers (Moreira et al., 2015), i.e. all workers have the same capacity and can perform a task with the same specific time required for its execution. Whilst different tasks can be performed by different workers synchronously, they still must satisfy all precedence relations imposed by the precedence graph. Therefore, a task scheduling problem arises for each station with conceivable waiting times for workers before or between the execution of tasks, making MALBPs more complex than SALBPs (Fattahi & Roshani, 2011).

For the studied problem, it is assumed that the balancing decision is a long-term plan due to the high costs and line utilisation associated to it. That said, it is considered that the cost of a worker is much greater than the cost of opening an additional station, since each worker has some associated costs such as wages, equipment, labour regulations, among others. Hence, the primary objective of the mathematical model presented in Section 3 is to minimise the total number of workers, accompanied by the secondary objective of minimising the total number of stations, that is, the line length.

Ultimately, this work presumes that optimal SALBP solutions are feasible configurations for MALBPs. Naturally, the necessary number of workers (and stations) to achieve an optimal solution for the SALBP can be accepted as an upper bound for the MALBP, since SALBPs are more restrictive and assume the number of workers and stations to be the same in a given solution. Likewise, it is reasonable to adopt the upper bound for the total number of stations (NS) in a MALBP to be one unit lesser than its simpler counterpart optimal solution. As the MALBP's objective is

to minimise both the number of workers and stations, with a higher weight in the former, the minimal marginal improvement taken from a SALBP solution is reducing the line length in one station by reallocating a worker in some of the remaining stations. For instance, the upper bound for the number of stations in a MALBP would be considered to be five ($NS = 5$) for the illustrative example presented in Figure 2, since that is the SALBP's optimal number of stations minus one. It is justified by the reasoning that, if the model is not even able to reduce one station in the previous solution by pointing out an infeasibility, then it is concluded that allowing more than one worker per station cannot contribute to efficiency improvements, and optimal solutions of both versions (SALBP and MALBP) are coincident in objective value. This fact is further analysed in Section 3.3.

3. Mathematical formulation

This section contains a Mixed-Integer Linear Programming (MILP) model developed to represent the Multi-manned Assembly Line Balancing Problem (MALBP) considering the characteristics identified in Section 2. Section 3.1 presents the main model to represent the problem and Section 3.2 exhibits the implemented symmetry breaks that strengthens the problem's linear relaxation. Table 2 informs the applied terminology to describe parameters and sets used in the formulation. The variables are detailed in Table 3, they are created by the model depending on the sets.

Table 2: Terminology: names of parameters and sets, their meaning, and [dimensional units].

Parameter	Meaning
NT	Number of tasks
NS	Maximum number of stations
NW	Maximum number of workers per station
CT	Cycle time [time units]
D_t	Duration [time units] of task t
$WCost$	Worker cost [monetary units]
$SCost$	Station cost [monetary units]
$BigM$	A sufficiently large positive number
Set	**Meaning**
T	Set of tasks t; $T = \{1, 2, ..., t, ..., NT\}$
S	Set of stations s; $S = \{1, 2, ..., s, ..., NS\}$
W	Set of workers w; $W = \{1, 2, ..., w, ..., NW\}$
TS	Set of feasible Task-Station elements
TW	Set of feasible Task-Worker elements
WS	Set of feasible Worker-Station elements
TWS	Set of feasible Task-Worker-Station elements
P	Set of precedence relations between two tasks t_i and t_j: (t_i, t_j)

3.1. Main model

The objective function considered in Expression 1 for this problem is similar to the ones used in Fattahi & Roshani (2011) and Kellegöz (2017). The first component in the objective function corresponds to the total number of workers employed in line and their weighted cost ($WCost$). The remaining of the expression represents the total number of stations used in the line, along with its

Table 3: Terminology: definition of model's variables.

Variable	Set	Domain	Meaning
$X_{t,s}$	$(t,s) \in TS$	$\{0,1\}$	Task-station assignment: set to 1 if task t is performed in station s
$Y_{w,s}$	$(w,s) \in WS$	$\{0,1\}$	Worker-station assignment: set to 1 if worker w is used in station s
$W_{t,w}$	$(t,w) \in TW$	$\{0,1\}$	Task-worker assignment: set to 1 if task t is performed by worker w
Z_s	$s \in S$	$\{0,1\}$	Station opened: set to 1 if station s needs to be used
F_{t_i,t_j}	$t_i, t_j \in T \mid t_i \neq t_j$	$\{0,1\}$	Follow variable: set to 1 if task t_i is followed by task t_j
ST_t	$t \in T$	$\mathbb{Z}_+$	Starting time: time in which task t starts to be performed
$A_{t,w,s}$	$(t,w,s) \in TWS$	$\{0,1\}$	Auxiliary variable: set to 1 for mimicking variables $Y_{w,s}$
$I_{w,s}$	$(w,s) \in WS$	$\mathbb{Z}_+$	Idle time: total time that worker w spends idle in station s

weighted cost ($SCost$). Remind that $WCost$ is a positive number much larger than $SCost$, therefore, the primary objective is to minimise the total number of workers.

$$\text{Minimise: } \underbrace{WCost \cdot \sum_{(w,s) \in WS} Y_{w,s}}_{workers\ cost} + \underbrace{SCost \cdot \sum_{s \in S} Z_s}_{stations\ cost} \tag{1}$$

In the model's constraints, Equations 2 are the occurrence constraint, forcing each task to be allocated to a station exactly once. Equations 3 are analogous to the previous one, it ensures that each task is exclusively performed by one worker. Equations 4 assign appropriate values to $Y_{w,s}$ variables: if a task t is allocated to station s, and this same task t is performed by worker w, then it is possible to induce which worker w from station s is employed for the activity. The precedence relations between tasks allocated in different stations are satisfied by Inequalities 5.

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

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

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

$$\sum_{s \in S} s \cdot X_{t_i,s} \leq \sum_{s \in S} s \cdot X_{t_j,s} \qquad \forall\ (t_i,t_j)\ \in\ P \tag{5}$$

The task scheduling core of the problem is based on the concept of task following, i.e. when a task can only start after other is finished. How task following and task starting time variables behave in the formulation is hereafter presented. If a pair of tasks (t_i, t_j) is contained in the precedence set P, it is mandatory that task t_j follows task t_i (Equations 6). Logically, tasks with the same index cannot follow one another, and so is the set F_{t_i,t_j} accordingly defined. Inequalities 7 and 8 are logical ties to properly decide following variables depending on task allocation and worker use. Between stations, if a task t_i is not allocated to the station that t_j is or in any station after that, then t_j follows t_i (Inequalities 7). Into the same station, if tasks t_i and t_j are performed by the same worker, then one of them must follow the other (Inequalities 8). Contrary to previously presented mathematical formulations that use a relative order time reasoning (Fattahi & Roshani, 2011; Kellegöz, 2017),

this follow variable (F_{t_i,t_j}) concept allows the proposed model to be less dependent on Big-M based constraints. Nonetheless, the definition of task starting times still relies on few formulations containing Big-M strategies (Hillier & Lieberman, 2015). In all cases, the $CT \cdot NS$ numerical value is a valid, sufficiently large, value for the parameter $BigM$ (Table 2). Inequalities 9 bind the starting time of a task t to a minimum value (lower bound) regarding station s in which it is allocated. From the other side, Inequalities 10 limit the maximum starting time (upper bound) that a task t can begin to be performed in station s. Complementary, the last task t must be finished up to the last opened station (Inequalities 11), and all tasks t_i that precede t_j must be completed before t_j starts to be performed (Inequalities 12).

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

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

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

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

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

$$ST_t + D_t \leq CT \cdot \sum_{s \in S} Z_s \qquad \forall\ t\ \in\ T \tag{11}$$

$$ST_{t_j} \geq ST_{t_i} + D_{t_i} - BigM \cdot (1 - F_{t_j,t_i}) \qquad \forall\ t_i,\ t_j\ \in\ T \mid t_i \neq t_j \tag{12}$$

3.2. Symmetry break constraints

The model 1–12 represents the problem. However, some ordering symmetry breaks were implemented into the model to strengthen the problem's linear relaxation and avoid wasting much time visiting symmetric solutions (Walsh, 2006). Inequalities 13 state that a station can only be opened if there is a task allocated to it. Inequalities 14 are similar, but for a worker assigned to that station. The combination of these inequalities assists the searching process for tighter bounds, as they prohibit the existence of unproductive or unoccupied stations. Inequalities 15 state that a station can only be opened if a previous one is already opened, they prevent the issue found by Yilmaz & Yilmaz (2016b) in a previous paper (Fattahi & Roshani, 2011), in which an arbitrary opening order of stations was allowed, leading to inconsistent solutions. Analogously, Inequalities 16 break the symmetry between workers by stating that a worker can only be used if a previous one is already in use, avoiding equivalent solutions (in terms of objective function value) to be taken into consideration by the model and,

therefore, shrinking the search-space. Notice that Inequalities 15 and 16 are only applied to the model from the second station/worker onwards, as it also was used by Kellegöz (2017).

$$Z_s \geq X_{t,s} \qquad \forall\ (t,s)\ \in\ TS \tag{13}$$

$$Z_s \geq Y_{w,s} \qquad \forall\ (w,s)\ \in\ WS \tag{14}$$

$$Z_s \leq Z_{s-1} \qquad \forall\ s\ \in\ S \mid s > 1 \tag{15}$$

$$Y_{w,s} \leq Y_{w-1,s} \qquad \forall\ (w,s)\ \in\ WS \mid w > 1 \tag{16}$$

Finally, idle time symmetry breaks are also added to the formulation. An auxiliary variable ($A_{t,w,s}$) that mimics the worker-station assignment ($Y_{w,s}$) is necessary for this part of the formulation. Inequalities 17 and 18 are the bounds to define appropriate values for $A_{t,w,s}$. This is necessary to calculate idle times associated to each worker along stations ($I_{w,s}$), which is done by Inequalities 19 and 20. Inequalities 21 conduct the symmetry break based on idle time information: the first worker's idle time must be lesser or equal to the second's and so on. This also enables the reduction of search-space, since equivalent solutions would be disregarded by the model due to rules concerning idle time differences between workers imposed by Inequalities 21.

$$A_{t,w,s} \geq X_{t,s} + W_{t,w} - 1 \qquad \forall\ (t,w,s)\ \in\ TWS \tag{17}$$

$$\sum_{(t,w,s)\in TWS} A_{t,w,s} \cdot D_t \leq CT \qquad \forall\ (w,s)\ \in\ WS \tag{18}$$

$$I_{w,s} \geq CT \cdot Y_{w,s} - \sum_{(t,w,s)\in TWS} A_{t,w,s} \cdot D_t \qquad \forall\ (w,s)\ \in\ WS \tag{19}$$

$$\sum_{(w,s)\in WS} I_{w,s} + \sum_{t\in T} D_t \leq CT \cdot \sum_{(w,s)\in WS} Y_{w,s} \tag{20}$$

$$I_{w,s} \geq I_{w-1,s} \qquad \forall\ (w,s)\ \in\ WS \mid w > 1 \tag{21}$$

The MILP formulation defined by 1–21 is henceforth referred to as PM (proposed model).

3.3. Upper bound value for NS

This section mathematically formalises the modelling decisions for upper bound values applied to NS. By hypothesis, the SALBP is a very restrictive problem, constrained by several simplification hypotheses (Baybars, 1986). One of them states that each station is operated by one worker. Thus, by minimising the number of stations, one is, in practice, minimising the number of workers as a consequence. When such hypothesis is relaxed, more than one worker can be allowed in each station, and the number of workers and stations are explicitly minimised separately. Furthermore, it is possible

to attribute weights to workers and stations based on the importance (cost) of each resource, in which cumulative wages generally are much more costly than the physical parts of a station (Fattahi & Roshani, 2011; Kellegöz, 2017).

That stated, it is known that the number of workers and stations in both SALBP and MALBP cases are integers. For SALBP, x_S and y_S represent the number of workers and stations in a given configuration, respectively. Analogously, let x_M and y_M respectively represent the number of workers and stations in a given configuration of MALBPs. Naturally, x_S and y_S will always assume the same integer value ($x_S = y_S$) in any solution, since it is, by hypothesis, mandatory for a SALBP to have the same number of workers and station. Conversely, x_M is considered to be greater or equal to y_M ($x_M \geq y_M$), because each opened station must have at least one worker performing operations in it. Moreover, the worker component weighted cost (w_1) is much larger than its station counterpart (w_2): $w_1 >> w_2$. Therefore, an objective function considering SALBP hypotheses can be expressed as $S(x_S, y_S) = w_1 \cdot x_S + w_2 \cdot y_S \mid x_S = y_S$ and a MALBP objective function as $M(x_M, y_M) = w_1 \cdot x_M + w_2 \cdot y_M \mid x_M \geq y_M$.

In terms of optimal solutions, the number of workers and stations are represented by x_S^*, y_S^*, x_M^*, and y_M^* for SALBP and MALBP cases, respectively. As MALBPs are less restrictive than SALBPs, their optimal solutions for the minimisation problem cannot be worse than their simpler counterpart in any given instance: Proposition 1.

Proposition 1. *$M(x_M^*, y_M^*) \leq S(x_S^*, y_S^*)$, which can be subdivided into two cases:*

(i) $M(x_M^*, y_M^*) = S(x_S^*, y_S^*)$

(ii) $M(x_M^*, y_M^*) < S(x_S^*, y_S^*)$

Proof. In Proposition 1, case (i), the SALBP solution has, by hypothesis, the same number of workers and stations ($x_S^* = y_S^*$). Therefore, in order to have an optimal MALBP solution equivalent to a SALBP one, x_M^* must be equal to x_S^*, and y_M^* must be equal to y_S^*, that is: $M(x_M^*, y_M^*) = S(x_S^*, y_S^*) \Leftrightarrow x_M^* = x_S^* = y_M^* = y_S^*$, which is perfectly feasible. For case (ii), in which the optimal MALBP solution value of a given instance is exclusively lower than the optimal SALBP one, at least one of the following conditions must happen: (a) fewer workers ($x_M^* < x_S^*$), which would inevitably lead to fewer stations (a station without any worker/operation cannot be opened), or (b) same number of workers, but fewer stations ($x_M^* = x_S^* \wedge y_M^* < y_S^*$), which still meets the MALBP hypothesis of allowing more than one worker per station and is also reckoned possible by an illustrative example (Figure 2).

Given the assumption that the worker cost component receives a much higher weight in the objective function than the station one ($w_1 >> w_2$), it can be concluded by Proposition 1, case (ii), that the minimal marginal improvement in a MALBP solution over a SALBP one comes from the reduction of the line length in one station unit ($x_M^* \leq x_S^*,\ y_M^* < y_S^*$), and from case (i) that, in the worst case, the MALBP optimal solution is equivalent to the SALBP one, which produces the following corollary that is used in the proposed mathematical model.

Corollary. The upper bound for the number of stations (NS) in a MALBP model can be set to

a value one unit lesser than the optimal solution to its SALBP counterpart (taking $w_1 >> w_2$ into account). If the model is found to be infeasible, it is concluded that the MALBP optimal solution is exactly the same as the configuration yielded by its SALBP version.

4. Benders' decomposition algorithm

The Benders' decomposition algorithm (hereafter referred to as BDA) developed for the MALBP is presented in this section. The Benders' decomposition (Benders, 1962) is a method based on reformulating the original monolithic model into two hierarchical problems: the master problem (MP) and the slave (or sub) problems (SP). The partition aims at freeing the MP from several variables and restrictions, which are then solved as SPs. A Benders' decomposition works iteratively: a solution of the MP (with fixed values for the key variables) is then used in the SPs. The SP is generally decomposable in multiple problems, reducing the computational burden comparing to a monolithic problem. The results of such divided problems are used to inform the MP by using cutting planes. The MP with extra restrictions is solved and the procedure repeats with the next answer. In other words, the decomposition strips off difficult variables from the MP and then iteratively corrects misled solutions by solving the parts that are omitted in the MP. In the proposed implementation, the MP is related to the high-level decisions of task-station and worker-station assignments, whereas SP is associated to feasibility tests on the lower-level problem of task-worker assignment and task-scheduling in each station. Besides, the MP is enhanced by graph-based feasibility cuts described in Section 4.1. Extending the concept to use Combinatorial Benders' Cuts (Codato & Fischetti, 2006), the MP is distilled from the original and complete combinatorial problem (monolithic model), which is equivalent to being significantly relaxed, since it is initially separated from the SP. Once all decisions variables to compute the objective value ($Y_{w,s}$ and Z_s) are contained in the MP, solving it might yield feasible integer solutions, which are sent to the SP to be validated (or not) by it. If feasibility is detected by the SP, the current solution is accepted as an incumbent one. Otherwise, the SP returns combinatorial inequalities to be added as lazy constraints into the MP. The BDA iterates this procedure until an optimal solution is found and proven.

The use of BDAs in the literature is reviewed and summarised by Rahmaniani et al. (2017). Several real-world problems were approached by using Benders' decomposition method. Nevertheless, when it comes to line balancing problems, only one work is listed in the survey (Osman & Baki, 2014), which concerns transfer lines. Further research regarding assembly line balancing problems was scarcely developed. In Hazir & Dolgui (2013) and Hazir & Dolgui (2015), straight and U-type layouts are considered under uncertainty, formulating a robust optimisation model and algorithm. Lastly, Akpinar et al. (2017) takes into account set-up times that are dependent on task sequencing in each station, interpreting task assignment and sequencing decisions as hierarchical problems. None of them involved assembly lines with multi-manned stations.

For the MALBP, the MP represents task-station assignments and worker-station allocation problems, whilst the SP takes care of the task-worker scheduling problem. Notice that, in this application, SP is not a continuous problem, thus a feasibility-seeking variant (Benders, 1962) must be used in or-

der to solve the problem previously stated in Section 3 by the PM (Expressions 1–21). Therefore, the slave problem should be used as a feasibility check on the system, as it was stated by Côté et al. (2014) and Fakhri et al. (2017), who applied the method in the strip packing problem and the capacitated fixed charge multiple knapsack problem, respectively. In particular for the proposed BDA, the SPs are task-worker scheduling problems solved individually for each multi-manned station. As Benders decompositions might go through a slow convergence process (Magnanti & Wong, 1981), some algorithm enhancements are deemed necessary to accelerate such operation and so they are pointed out along with the MP and SP descriptions. Also for that reason, whenever an infeasibility is detected, such condition is modelled as a new restriction and added to the MP. These combinatorial Benders' cuts are further explained in Section 4.2. Moreover, feasibility cuts based on precedence graph analyses are implemented in the MP (Section 4.1), limiting the possibilities of task allocations.

4.1. Master problem

For the MP, Expressions 1, 2, 5, and 13–16 are maintained, and additional constraints are developed based on a set of incompatible task pairs (Inc), which are tasks that cannot be executed in the same station due to precedence relations and cycle time restrictions, independently on the number of workers assigned there (*Enhancement 1*). How this analysis is conducted to define such task pairs is hereafter presented. In order to do so, the precedence relations set P must be extended to a complete set P^* by considering all direct and indirect precedence relations. By constructing this complete set P^*, it is possible to create successors and predecessor sets for each task: Suc_t and Pre_t are sets that represent all direct and indirect successors and predecessors of a task t, respectively. Furthermore, two extra parameters are needed to represent the critical path duration (δ_{t_i,t_j}) and the sum of task durations (σ_{t_i,t_j}) between tasks t_i and t_j. When computed, these parameters are recursively evaluated in a topological order.

Given three tasks t_i, t_j, and t_k, such that $t_i \neq t_j \neq t_k$, Equations 22 recursively attribute to parameter δ_{t_i,t_j} the critical path between two tasks by employing an algorithmic procedure, that is, they run through the precedence graph and establishes what is the longest sum of task durations that have to be performed between tasks t_i and t_j. This concept was adopted from project scheduling problems (Klein, 2000). Equations 23 are needed to further calculate capacity bounds, another logic inherited from the resource constrained project scheduling problems (Klein, 2000). The sum of task durations of all tasks that are successors of task t_i and predecessors of task t_j is assigned to the parameter σ_{t_i,t_j}. Taking the precedence graph from Figure 2 as an example, these parameters for the task pair (t_1,t_6) would be $\delta_{t_1,t_6} = 4$ and $\sigma_{t_1,t_6} = 8$.

$$\delta_{t_i,t_j} = \max\left[0;\ \delta_{t_i,t_k} + D_{t_k} \mid t_k \in Suc_{t_i} \cap Pre_{t_j}\right] \qquad \forall\ (t_i,t_j)\ \in\ P^* \tag{22}$$

$$\sigma_{t_i,t_j} = \sum_{t_k \in Suc_{t_i} \cap Pre_{t_j}} D_{t_k} \qquad \forall\ (t_i,t_j)\ \in\ P^* \tag{23}$$

In order to define which task pairs (t_i,t_j) are incompatible, only task pairs $(t_i,t_j) \in P^*$ are considered. If either Inequalities 24 or Inequalities 25 are verified, the incompatibility condition

is satisfied, and the task pair (t_i, t_j) are added to the incompatibility set Inc. Once in hand of Inc, Inequalities 26 are added to the MP, restricting specific task pairs to be allocated to the same station. Whenever Inequalities 25 are not satisfied, it is possible to generate weaker – but still valid – restrictions. Inequalities 27 state the minimum number of workers that a station requires in order to perform both tasks of a task pair (t_i, t_j), being ε a very small positive number to avoid dividing by zero. Finally, the available time to perform tasks in each station is given by the number of workers assigned there (Inequalities 28).

$$D_{t_i} + D_{t_j} + \delta_{t_i,t_j} > CT \tag{24}$$

$$D_{t_i} + D_{t_j} + \left\lceil \frac{\sigma_{t_i,t_j}}{NW} \right\rceil > CT \tag{25}$$

$$X_{t_i,s} + X_{t_j,s} \leq 1 \qquad \forall\ s\ \in\ S,\ (t_i, t_j)\ \in\ Inc \tag{26}$$

$$\sum_{w \in W} Y_{w,s} \geq \left\lceil \frac{\sigma_{t_i,t_j}}{CT - D_{t_i} - D_{t_j} + \varepsilon} \right\rceil - NW \cdot (2 - X_{t_i,s} - X_{t_j,s}) \qquad \forall\ s\ \in\ S,\ (t_i, t_j)\ \in\ P^* \tag{27}$$

$$\sum_{t \in T} X_{t,s} \cdot D_t \leq CT \cdot \sum_{w \in W} Y_{w,s} \qquad \forall\ s\ \in\ S \tag{28}$$

This reformulated part of BDA focus on finding an optimal solution for the problem. Any feasible solution $(\tilde{X}, \tilde{Y}) = \{(\tilde{X}_{1,1}, ..., \tilde{X}_{t,s}), (\tilde{Y}_{1,1}, ..., \tilde{Y}_{w,s})\}$ found by the MP is passed to SP for scheduling feasibility check in each station s. That way, the monolithic model is decomposed in an MP that decides allocation variables $(X_{t,s})$ and the number of total workers and stations ($Y_{w,s}$ and Z_s) used along the line, while the SP seeks for feasible task-worker assignments $(W_{t,w})$ by considering task starting times and ordering (ST_t and F_{t_i,t_j}) for each station.

4.2. Slave problem

The SP keeps Expressions 3, 6, and 12 as in Section 3, but modifies Inequalities 8, 19, and 21 for simpler ones. For that, they are applied to each station s separately by using task and worker sub-sets T_s and W_s, which are dependent on the solution $(\tilde{X}, \tilde{Y})$ sent from MP, as expressed by Equations 29 and 30. Thinking of each station as a separate resource-constrained scheduling sub-problem, Inequalities 31, 32, and 33 respectively substitute the previous monolithic ones without any loss of functionality. Slave problems in which only one worker is employed in the station are automatically deemed feasible, since there is no scheduling problem to begin with in such cases.

$$T_s = \{t\ \in\ T \mid \tilde{X}_{t,s} = 1\} \qquad \forall\ s\ \in\ S \tag{29}$$

$$W_s = \{w\ \in\ W \mid \tilde{Y}_{w,s} = 1\} \qquad \forall\ s\ \in\ S \tag{30}$$

$$F_{t_i,t_j} + F_{t_j,t_i} \geq W_{t_i,w} + W_{t_j,w} - 1 \qquad \forall\ t_i,\ t_j\ \in\ T_s \tag{31}$$

$$I_w = CT - \sum_{t \in T_s} D_t \cdot W_{t,w} \qquad \forall\ w\ \in\ W_s \tag{32}$$

$$I_w \geq I_{w-1} \qquad \forall\ w\ \in\ W_s \mid w > 1 \tag{33}$$

In order to prevent redundant feasibility seeking tests, a hash-table (Maurer & Lewis, 1975) is employed to store SP instance information that had led to feasible solutions (*Enhancement 2*). Whenever a task allocation set for a given station is tested and found to be feasible, that set of tasks and number of workers used to perform them is included into a tested problem hash-table, so the SP model does not need to solve repeated scheduling problems, since the algorithm can quickly consult this hash-table beforehand. If the MP's solution is evaluated as feasible for all stations by the SP, then such solution is considered incumbent by the proposed BDA and the algorithm returns to the MP with a new UB. Otherwise, the submitted MP's solution might be detected as infeasible. If that is the case, Benders' combinatorial cuts are applied to the MP as lazy constraints, that is, new restrictions are added to the initial optimisation problem in order to cut off infeasible task allocation and worker assignment possibilities (*Enhancement 3*).

The type of cut depends on the number of workers that the SP instance had employed and was proven infeasible. Stronger cuts can be added when the trial solution employs the maximum allowed workers for a specific station. Inequalities 34 state that, if a given tested task allocation set cannot be performed in the same station s, then at least one of them must be performed elsewhere. Alternatively, the allocation may be infeasible, but the maximum number of workers is not being used for that combination of tasks. In such cases, the cut described by Inequalities 35 are applied to the MP; it states that a tested task allocation set cannot be entirely performed in the same station unless an additional worker (represented by the $|W_s| + 1$ index) is assigned there.

$$\sum_{t \in T_s} X_{t,s} \leq \sum_{t \in T} \tilde{X}_{t,s} - 1 \qquad \forall\ s\ \in\ S \tag{34}$$

$$\sum_{t \in T_s} X_{t,s} \leq \sum_{t \in T} \tilde{X}_{t,s} - 1 + Y_{|W_s|+1,s} \qquad \forall\ s\ \in\ S \tag{35}$$

After solving all SPs, the BDA returns to MP and keeps searching for better solutions with a revised UB or newly added lazy constraints. This process is repeated iteratively until an optimal solution is found and proven or the computational processing time limit is reached.

4.3. BDA pseudo code

A summarised pseudo-code of the proposed BDA is presented in Algorithm 1. This is the implementation used to obtain the results reported in the computational study performed in Section 5. Initially, MALBP's parameters and computational processing time limit are input in order to build the optimisation problem. After that, the algorithm starts solving the MP (line 5). Tight upper bounds for the number of workers along the line ($ObjW$) and the number of opened stations ($ObjS$) based on SALBP results are taken into consideration to shrink the search-space (*Enhancement 4*).

Any time a new solution is found by MP, it is sent to feasibility check at the SP (line 6). Task-worker scheduling is conducted for each station (lines 8 and 9). When the combination of tasks and number of workers is feasible for a given station, such combination is added to a list coded as a hash-table (lines 10 and 11). Otherwise, if an infeasibility is detected, either Cut 34 or Cut 35 is added to the MP as a lazy constraint depending on the cardinality of W_s (lines 14 to 18). If all SP stations are proven to be feasible, this tested solution is considered to be the current incumbent solution (lines 21 and 22). Finally, the algorithm stops processing if optimality is proven or if time limit is reached (lines 23 and 24).

Algorithm 1: BDA's pseudo-code for the MALBP.

Input : $NT, NS, NW, CT, D_t, P, Time.Limit$
Output: $ObjW, ObjS, LB, CPU, TWS_{allocation}$
1 **Initialisation**: $Status \leftarrow 0$, $Incumbent \leftarrow \{max.value, max.value, 0, -\}$, $Hash \leftarrow \{\}$
2 **Start**
3 **while** $Status = 0$ **do**
4 Timer.Start, compute CPU time
5 Solve MP, compute $ObjW, ObjS, LB, TWS_{allocation}$
6 **if** $(\tilde{X}, \tilde{Y})_{new}$ **then**
7 $Counter \leftarrow 0$
8 **foreach** $s \in \{1, ..., ObjS\}$ **do**
9 Solve SP $(\tilde{X}, \tilde{Y})_s$, compute feasibility
10 **if** $feasible$ **then**
11 $Hash \leftarrow Hash \bigcup \{(\tilde{X}, \tilde{Y})_s\}$
12 $Counter \leftarrow Counter++$
13 **else**
14 **if** $|W_s| = NW$ **then**
15 Add Cut 34 to MP
16 **else**
17 Add Cut 35 to MP (in case $|W_s| < NW$)
18 **end**
19 **end**
20 **end**
21 **if** $Counter = ObjS$ **then**
22 $Incumbent(ObjW, ObjS, LB, TWS_{allocation}) \leftarrow (\tilde{X}, \tilde{Y})_{new}$
23 **if** $(ObjW + ObjS = LB) \vee (CPU \geq Time.Limit)$ **then**
24 $Status \leftarrow 1$
25 **end**
26 **end**
27 **end**
28 **end**
29 **End**

5. Computational study

This section presents a computational study that was carried out in the same benchmark dataset used by Fattahi & Roshani (2011) and Kellegöz (2017) combined. Thus, both datasets were used, totalling 131 instances. All tested instances are contained in a well-known literature benchmark. They are available for download at <www.assembly-line-balancing.de> with information about task durations, precedence graphs, and cycle time values. Some of these instances' features can be observed in Table 4: the number of tasks (NT) is the chosen parameter to divide instances into three categories related to size (small, medium, and large), and instances are solved with different cycle time (CT) values and maximum number of workers (NW) allowed in each station. Upper bounds for the number of stations (NS) for each instance has been obtained in a pre-processing step: (i) the SALBP version of each problem is solved using SALOME (Scholl & Klein, 1997), which takes less than a second; (ii) NS is set to one unit lesser than such value for the MALBP. In order to ease results' visualisation and respect parameters' order of magnitude, $WCost$ and $SCost$ were set to 100 and 1, respectively.

Table 4: Summary of dataset instances.

Size (Total of instances)	Problem	NT	CT	NW
Small (50)	Mitchell	21	14; 15; 21; 26; 35; 39	2
	Heskiaoff	28	138; 205; 216; 256; 324; 342	2; 4
	Sawyer	30	25; 27; 30; 36; 41; 54; 75	2; 4
	Kilbridge	45	57; 79; 92; 110; 138 184	2; 4; 6
Medium (45)	Tonge	70	176; 364; 410; 468; 527	2; 4; 6
	Arcus1	83	5048; 5853; 6842; 7571; 8412; 8998; 10816	2; 4; 6
	Mukherje	94	176; 248; 351	2; 4; 6
Large (36)	Arcus2	111	5755; 8847; 10027; 10743; 11378; 17067	2; 4; 6
	Barthol2	148	84; 106; 170	2; 4; 6
	Barthold	148	403; 513; 805	2; 4; 6

The computational study is divided in two parts. Firstly, small-size instances are solved in Section 5.1 for the monolithic model presented in Section 3 (PM) and results are compared to those obtained by Kellegöz (2017); this last model is henceforth referred to as KM (Kellegöz's Model). In addition, the BDA exhibited in Section 4 is also applied to the same fraction of the benchmark dataset and its performance is compared to the Ant-Colony Algorithm (ACO) and the Gantt Simulated Annealing (GSA) heuristic developed by Fattahi & Roshani (2011) and Kellegöz (2017) in terms of solution quality reported by them.

Afterwards, as both BDA, ACO, and GSA demonstrated dominance over monolithic models in terms of solution quality and computational processing time, mathematical formulations (PM and KM) were discarded in the remainder testing process. Therefore, for the second part of this computational study, only BDA, ACO, and GSA were considered to be applied to the remaining dataset (medium and large-size instances). Such results and comparisons are presented in Section 5.2.

In both Sections 5.1 and 5.2, each instance result is reported in a line of Tables 5 to 7, addressed by the problem, CT, and NW values. St indicates solution status, reporting optimal (*) or integer (I) solutions. The objective function value is represented by the Obj column. For the BDA, this Obj

column is branched into upper bound (UB), lower bound (LB), and Gap values. Total computational processing times (CPU) and computational processing times for the slave problems (SCPU) are informed in seconds. As CPU of some instances were not reported in Fattahi & Roshani (2011), they are left in unfilled (–). Lastly, the number of added cuts (Cut1 and Cut2 for Inequalities 34 and 35, respectively) and the number of times that the algorithm accessed the hash-table (HT) are reported.

To all instances, Gurobi 8.1 (Gurobi Optimization, 2019) was selected as universal solver due to implementation readiness, focusing on optimality for the MP and feasibility for the SP. A 64 bit Intel™ i7-3770 CPU (3.4 GHz) with 16 GB of RAM was employed using four threads. The BDA was coded in Microsoft Visual Basic 2015 programming language.

5.1. Small-size instances

Both PM and BDA were applied to the small-size instances presented in Table 4 with a time limit set to 3600 seconds. Table 5 summarises the comparison between monolithic models and algorithms for the small-size dataset. The results obtained by PM are compared to those reported by KM whenever such instance has also been solved by Kellegöz (2017), whilst BDA results are displayed alongside with the best result found by either ACO or GSA in their respective papers. This subset contains 50 instances, in which PM clearly outperforms KM. The PM obtained 46 optimal solutions, whereas in the 26 instances tested by Kellegöz (2017), KM reached optimality in only 12 instances. In other words, PM has improved and proven the optimality of 6 solutions (boldfaced in Table 5) and proven the optimality of 7 previously known integer solutions (italicised in Table 5) obtained by a mathematical model. As reported in Section 3, this might be due to the fact that modelling decisions were different between PM and KM: the formulation is less dependent on Big-M constraints, a follow variable concept was employed instead of a relative order time one, and symmetry break constraints were implemented.

BDA, ACO, and GSA results are reported in the remaining of the comparison by Table 5. Equivalent solutions and computational processing times are verified for both methods, however, the BDA presented in Section 4 is able to concede optimality proofs. Therefore, the BDA not only has reached results as good as the ACO algorithm and the GSA heuristic, but it also has guaranteed solutions to be optimal for 49 out of 50 small-size instances in a very reduced CPU time.

Table 5: Results comparison for small-size instances between monolithic models (PM and KM), BDA, ACO algorithm and GSA heuristic.

Problem	*CT*	*NW*	PM			KM			BDA								ACO/GSA	
			St	Obj	CPU	St	Obj	CPU[1]	St	UB	LB	Gap	CPU	SCPU	Cut1	Cut2	HT	Obj	CPU[1]
Mitchell	14	2	*	807	0.09	*	807	3.40	*	807	807	0%	0.01	0.00	0	0	0	807	0.88
	15	2	*	807	0.08	–	–	–	*	807	807	0%	0.01	0.00	0	0	0	807	–
	21	2	*	505	0.17	–	–	–	*	505	505	0%	0.01	0.00	0	0	0	505	–
	26	2	*	504	0.06	*	504	0.44	*	504	504	0%	0.01	0.00	0	0	0	504	0.93
	35	2	*	303	0.05	–	–	–	*	303	303	0%	0.01	0.00	0	0	0	303	–
	39	2	*	302	0.04	*	302	0.44	*	302	302	0%	0.01	0.00	0	0	0	302	0.94
Heskiaoff	138	2	*	*805*	42.26	I	805	1h	*	805	805	0%	1.40	1.37	575	0	86	805	0.95
		4	*	804	0.80	*	804	19.98	*	804	804	0%	0.01	0.01	0	0	0	804	1.26
	205	2	*	503	0.36	–	–	–	*	503	503	0%	0.73	0.71	87	0	2	503	–
		4	*	503	2.23	–	–	–	*	503	503	0%	0.73	0.72	0	45	0	503	–
	216	2	*	503	0.38	–	–	–	*	503	503	0%	0.19	0.18	72	0	7	503	–
		4	*	503	1.91	–	–	–	*	503	503	0%	0.03	0.02	0	0	0	503	–
	256	2	*	**403**	11.10	I	503	1h	*	403	403	0%	1.59	1.46	1326	0	0	403	1.04
		4	*	**403**	10.12	I	502	1h	*	403	403	0%	2.78	2.50	0	1401	0	403	1.21
	324	2	*	402	0.07	–	–	–	*	402	402	0%	0.02	0.02	0	0	0	402	–
		4	*	402	0.07	–	–	–	*	402	402	0%	0.02	0.01	0	0	0	402	–
	342	2	*	302	0.28	*	302	53.43	*	302	302	0%	0.25	0.24	0	0	0	302	1.07
		4	*	302	0.49	*	302	17.04	*	302	302	0%	0.32	0.31	0	8	0	302	1.21
Sawyer	25	2	*	*1408*	2.42	I	1408	1h	*	1408	1408	0%	0.03	0.01	8	0	0	1408	1.46
		4	*	*1408*	3.94	I	1408	1h	*	1408	1408	0%	0.20	0.17	0	272	49	1408	1.79
	27	2	*	*1308*	14.34	I	1308	1h	*	1308	1308	0%	0.04	0.02	32	0	3	1308	1.19
		4	*	*1308*	9.06	I	1308	1h	*	1308	1308	0%	0.02	0.01	0	0	0	1308	1.52
	30	2	*	1206	25.38	–	–	–	*	1206	1206	0%	0.06	0.02	14	0	0	1207	–
		4	*	1206	73.25	–	–	–	*	1206	1206	0%	0.10	0.01	0	0	0	1207	–
	36	2	*	1006	8.26	–	–	–	*	1006	1006	0%	0.01	0.01	0	0	0	1006	–
		4	*	1006	6.99	–	–	–	*	1006	1006	0%	0.01	0.01	0	0	0	1006	–
	41	2	*	804	1.60	–	–	–	*	804	804	0%	0.07	0.01	0	0	1	806	–
		4	*	804	2.49	–	–	–	*	804	804	0%	0.05	0.03	0	0	0	806	–
	54	2	*	*704*	54.34	I	704	1h	*	704	704	0%	0.03	0.02	0	0	0	704	1.13
		4	*	*704*	5.88	I	704	1h	*	704	704	0%	0.01	0.01	0	0	0	704	1.44
	75	2	*	503	0.17	*	503	9.42	*	503	503	0%	0.01	0.01	0	0	0	503	1.28
		4	*	503	5.88	*	503	5.99	*	503	503	0%	0.01	0.01	0	0	0	503	1.70
Kilbridge	57	2	*	*1006*	3184.13	I	1006	1h	*	1006	1006	0%	3.43	3.15	390	0	20	1006	1.53
		4	*	**1005**	6.66	I	1105	1h	*	1005	1005	0%	0.53	0.48	5	50	9	1005	2.21
		6	*	1005	5.07	*	1005	2843.85	*	1005	1005	0%	3.02	2.96	0	45	3	1005	2.22
	79	2	*	704	27.90	–	–	–	*	704	704	0%	1.03	0.70	165	0	19	705	–
		4	I	803	1h	–	–	–	*	703	703	0%	0.33	0.28	0	0	0	705	–
		6	I	803	1h	–	–	–	*	703	703	0%	2.36	2.20	0	50	0	705	–
	92	2	I	604	1h	–	–	–	*	604	604	0%	93.82	91.75	4536	0	866	604	–
		4	*	603	8.82	–	–	–	*	603	603	0%	0.61	0.57	0	0	0	604	–
		6	*	603	48.31	–	–	–	*	603	603	0%	0.54	0.46	0	4	0	604	–
	110	2	*	603	0.89	*	603	663.27	*	603	603	0%	0.75	0.72	3	0	0	603	1.95
		4	*	603	1.66	*	603	323.27	*	603	603	0%	0.03	0.02	0	0	0	603	2.60
		6	*	603	2.07	*	603	63.88	*	603	603	0%	0.05	0.04	0	0	0	603	2.73
	138	2	I	403	1h	–	–	–	I	403	402	0.25%	1h	3551.78	987	0	1	403	–
		4	*	402	13.46	–	–	–	*	402	402	0%	10.41	10.39	0	3	0	403	–
		6	*	402	42.44	–	–	–	*	402	402	0%	21.50	21.45	0	12	0	403	–
	184	2	*	**302**	4.13	I	402	1h	*	302	302	0%	0.53	0.51	4	0	0	302	2.15
		4	*	**302**	9.29	I	402	1h	*	302	302	0%	0.66	0.64	0	4	0	302	4.25
		6	*	**302**	14.71	I	402	1h	*	302	302	0%	0.47	0.44	0	24	0	302	3.84

[1]As reported in Kellegöz (2017).

ACCEPTED MANUSCRIPT

5.2. Medium and large-size instances

As BDA has been validated as a reliable and efficient method in Section 5.1 by quickly obtaining optimal solutions in all but one instance, the superiority of the proposed algorithm over the monolithic model was evidenced. Hence, this section focuses on computationally solving medium and large-size instance only with specialised methods (i.e. BDA, ACO, and GSA heuristic) and comparing their results in regard to solution quality that were previously reported in Fattahi & Roshani (2011) and Kellegöz (2017).

Table 6 reports the results for 45 medium-sized instances from Table 4. In terms of solution quality, the proposed BDA has outperformed ACO algorithm and GSA heuristic in 19 instances (boldfaced values in Table 6), while tying in the remaining 26. Nonetheless, it is important to notice that none of these instances had been directly solved to optimality previously, since their results would rather be compared to calculated theoretical LBs. Out of the 45 medium-sized instances, the proposed BDA has proven the optimality of 40 solutions, with a small integer gap for the remaining 5 cases.

The last 36 instances to be tested from Table 4 are contained in the large-size subset. Table 7 presents the comparison between BDA, ACO algorithm, and GSA heuristic when both are applied to such instances. The boldfaced values represent the 23 out of 36 results in which BDA has outperformed ACO and GSA in terms of solution quality and also proven optimality for the instance. Moreover, 2 previous best-known integer solutions were improved by BDA, whereas in 1 other instance GSA performed better. Out of the remaining 10 instances in which both methods tied, there are 5 newly proven optimal solutions obtained by BDA.

Both BDA, ACO algorithm, and GSA heuristic were able to reach the same number of workers as the optimal SALBP value in their solutions for the whole solved dataset (Tables 5, 6, and 7). Instances that the proposed BDA outperformed ACO algorithm and GSA heuristic were solved with a reduced number of stations, which indicates a tendency that it is more profitable to accept the SALBP optimal solution as the number of workers, and try to minimise as much as possible the line length (i.e. the number of multi-manned stations). An evidence to support this methodology is that the possibility to also reduce the number of workers in a multi-manned assembly line presented in Section 2 was not verified in any instance of the benchmark. Finally, it was verified that the BDA has a tendency in spending the majority of its computational processing time solving SPs in most cases. Besides, Cut1 is less frequently added than Cut2 when the maximum number of workers is increased and the hash-table is more often consulted for lower values of cycle time.

In order to conduct a feasibility check, the task-worker-station allocation results were tested for all newly found solutions. Task starting and ending times for each worker were examined for consistency regarding station and global cycle times, as well as precedence relation imposed orders. Filling their purpose to validate the proposed BDA's reliability, these tests are made available along with task-station-worker allocation results in the supporting information files for reproducibility purposes.

Table 6: Results comparison for medium-size instances between BDA, ACO algorithm, and GSA heuristic.

Problem	*CT*	*NW*	BDA									ACO/GSA	
			St	UB	LB	Gap	CPU	SCPU	Cut1	Cut2	HT	Obj	CPU[1]
Tonge	176	2	*	2112	2112	0.00%	16.87	3.86	4668	0	755	2112	14.54
		4	*	2110	2110	0.00%	36.99	30.86	610	5940	1217	2110	27.82
		6	*	2110	2110	0.00%	56.15	50.26	0	7720	1683	2110	41.17
	364	2	*	**1005**	1005	0.00%	19.89	19.57	534	0	104	1007	–
		4	*	**1004**	1004	0.00%	0.95	0.76	0	24	6	1007	–
		6	*	**1004**	1004	0.00%	17.51	16.99	0	162	53	1007	–
	410	2	*	905	905	0.00%	62.74	62.68	35	0	1	905	12.07
		4	*	**903**	903	0.00%	13.94	13.86	0	16	2	904	24.71
		6	*	**903**	903	0.00%	91.80	91.68	0	96	12	904	37.10
	468	2	*	804	804	0.00%	0.18	0.14	0	0	0	804	–
		4	*	**803**	803	0.00%	273.54	273.41	0	48	8	804	–
		6	*	**803**	803	0.00%	161.06	160.93	0	52	2	804	–
	527	2	*	704	704	0.00%	0.18	0.15	4	0	0	704	13.35
		4	*	703	703	0.00%	119.85	119.82	0	57	9	703	26.35
		6	*	703	703	0.00%	575.63	575.57	0	132	32	703	38.75
Arcus1	5048	2	*	1610	1610	0.00%	23.94	5.44	970	0	53	1610	17.95
		4	*	1610	1610	0.00%	153.58	99.78	0	1200	56	1610	34.42
		6	*	1610	1610	0.00%	122.59	92.61	0	1120	65	1610	50.92
	5853	2	*	**1408**	1408	0.00%	8.17	0.34	70	0	8	1410	–
		4	*	**1408**	1408	0.00%	12.25	0.86	0	80	4	1410	–
		6	*	**1408**	1408	0.00%	13.66	0.54	0	20	1	1410	–
	6842	2	*	**1207**	1207	0.00%	3.45	3.06	472	0	34	1208	–
		4	*	**1207**	1207	0.00%	52.21	51.72	0	544	55	1208	–
		6	*	**1207**	1207	0.00%	160.19	159.68	0	656	80	1208	–
	7571	2	*	1106	1106	0.00%	1.18	1.05	18	0	3	1106	17.20
		4	*	1106	1106	0.00%	5.90	5.79	0	48	3	1106	32.57
		6	*	1106	1106	0.00%	38.80	38.66	0	90	26	1106	47.78
	8412	2	*	1006	1006	0.00%	0.22	0.09	0	0	0	1006	–
		4	*	1006	1006	0.00%	0.32	0.22	0	0	0	1006	–
		6	*	1006	1006	0.00%	0.22	0.12	0	0	0	1006	–
	8998	2	*	**905**	905	0.00%	30.60	30.38	12	0	2	906	–
		4	*	**905**	905	0.00%	2.88	2.72	0	12	0	906	–
		6	*	**905**	905	0.00%	0.67	0.50	0	18	3	906	–
	10816	2	*	**804**	804	0.00%	87.83	84.74	485	0	54	805	17.10
		4	*	**804**	804	0.00%	2.46	0.03	0	0	0	805	32.78
		6	*	**804**	804	0.00%	14.17	11.27	0	170	4	805	47.45
Mukherje	176	2	*	2516	2516	0.00%	52.07	12.39	4368	0	392	2516	30.28
		4	I	2513	2512	0.04%	1h	3479.76	20072	13793	2744	2513	60.99
		6	I	2513	2512	0.04%	1h	3478.79	312	11089	453	2513	95.78
	248	2	*	1810	1810	0.00%	29.61	17.84	6750	0	1073	1810	30.71
		4	I	1808	1807	0.06%	1h	3523.20	528	7112	982	1808	63.44
		6	*	1807	1807	0.00%	785.02	78.75	56	868	59	1807	98.48
	351	2	I	1308	1307	0.08%	1h	3495.71	10800	0	550	1308	31.02
		4	I	1307	1306	0.08%	1h	3520.48	3115	4095	332	1307	64.09
		6	*	1306	1306	0.00%	188.23	187.51	0	90	6	1306	99.53

[1] As reported in Kellegöz (2017).

Table 7: Results comparison for large-size instances between BDA, ACO algorithm, and GSA heuristic.

Problem	*CT*	*NW*	BDA								ACO/GSA	
			St	UB	LB	Gap	CPU	SCPU	Cut1	Cut2	HT	Obj	CPU[1]
Arcus2	5755	2	*	**2714**	2714	0.00%	331.42	303.84	5888	0	554	2716	26.22
		4	*	**2712**	2712	0.00%	1900.07	1880.69	1391	3965	572	2713	48.78
		6	*	**2712**	2712	0.00%	861.55	840.12	0	2236	183	2713	73.00
	8847	2	*	**1811**	1811	0.00%	2593.92	2521.77	25839	0	4116	1812	–
		4	*	**1810**	1810	0.00%	304.24	287.28	55	1562	23	1812	–
		6	*	**1810**	1810	0.00%	12.06	1.53	0	253	0	1812	–
	10027	2	*	**1609**	1609	0.00%	151.24	150.42	2241	0	312	1610	–
		4	*	**1607**	1607	0.00%	1147.94	1134.54	90	2808	240	1610	–
		6	*	**1607**	1607	0.00%	676.12	669.81	0	837	37	1610	–
	10743	2	*	**1508**	1508	0.00%	328.60	303.63	1323	0	34	1509	31.41
		4	*	**1507**	1507	0.00%	610.87	608.03	8	1040	98	1508	61.87
		6	*	**1507**	1507	0.00%	558.84	555.84	0	544	32	1508	92.53
	11378	2	*	**1407**	1407	0.00%	194.89	193.96	904	0	266	1409	–
		4	*	**1406**	1406	0.00%	1298.10	1296.82	1547	1330	388	1409	–
		6	*	**1406**	1406	0.00%	1636.27	1634.42	0	3374	742	1409	–
	17067	2	*	905	905	0.00%	0.26	0.17	5	0	0	905	29.44
		4	*	**904**	904	0.00%	119.42	119.11	0	195	9	905	57.68
		6	*	**904**	904	0.00%	339.91	339.42	0	720	25	905	84.96
Barthol2	84	2	I	5127	5126	0.02%	3599.25	2.51	4576	0	14	*5126*	68.00
		4	I	5116	5113	0.06%	3602.33	1839.28	90272	13648	1460	5116	138.64
		6	I	5114	5111	0.06%	3608.81	2603.57	11102	46060	628	5114	224.44
	106	2	I	4121	4020	2.45%	3600.22	2.90	2688	0	186	4121	68.96
		4	I	*4111*	4010	2.46%	3615.90	2175.32	10751	1664	256	4113	145.09
		6	I	*4110*	4010	2.43%	3601.54	3192.26	1896	7200	245	4112	234.68
	170	2	*	2513	2513	0.00%	20.37	2.33	143	0	9	2513	67.18
		4	*	**2507**	2507	0.00%	199.16	196.54	40	24	16	2508	144.08
		6	*	**2506**	2506	0.00%	169.62	168.35	24	16	4	2508	232.94
Barthold	403	2	*	1407	1407	0.00%	218.58	217.96	140	0	7	1407	66.10
		4	*	**1404**	1404	0.00%	70.28	70.02	0	5	0	1405	143.39
		6	*	**1404**	1404	0.00%	1004.29	1002.56	25	120	12	1405	228.79
	513	2	*	1106	1106	0.00%	2.69	1.90	0	0	0	1106	66.57
		4	*	**1103**	1103	0.00%	1708.59	1708.24	168	16	17	1104	142.99
		6	*	**1103**	1103	0.00%	3797.85	3794.86	120	300	28	1104	226.17
	805	2	*	704	704	0.00%	75.13	74.83	8	0	0	704	66.55
		4	I	703	702	0.14%	3640.35	3638.28	183	174	0	703	140.12
		6	I	703	702	0.14%	3701.60	3700.92	3	333	0	703	216.74

[1]As reported in Kellegöz (2017).

6. Conclusions

The Multi-manned Assembly Line Balancing Problem (MALBP) with the objective of minimising the number of workers and stations has been addressed in this study. The existing literature on MALBPs indicated a lack of efficient exact solution methods for these problems, since past mathematical formulations were only able to solve some instances with up to 45 tasks. This paper's main contribution is solving to optimality MALBPs up to 148 tasks by decomposing the original problem and implementing a Benders' decomposition algorithm employing combinatorial cuts during its execution.

A new Mixed-Integer Linear Programming (MILP) model was developed along with several valid inequalities that work as symmetry break constraints to solve the optimisation problem. This proposed formulation outperforms previously presented monolithic mathematical formulations in terms of solution quality and computational processing time. By analysing the MALBP's structure, it is possible to infer that the problem is divisible hierarchically into a Master Problem (MP) and a Slave Problem (SP), and hence forging a Benders' Decomposition Algorithm (BDA). After adapting several logical cuts inherited from project scheduling problems, the MP solves task-station and worker-station assignment problems, whilst the SP deals with the task-worker scheduling problem for each station, detects infeasibility, and generates combinatorial Benders' cuts to be added into MP as lazy constraints during BDA's execution. The proposed BDA was compared to previously developed methods and was shown to produce improved results while maintaining reasonable CPU time. In total, 42 new optimal solutions were obtained, 2 integer solutions were improved, and 18 previously known solutions were proven optimal out of a dataset with 131 instances.

Allowing multiple workers to perform different tasks simultaneously in the same station is a natural extension of the simpler version of the problem, as well as a notable realistic feature widely employed in industries manufacturing large-size products. Nonetheless, incorporating more practical extensions such as line layout (U-line, parallel stations), product variety (multi and mixed model lines), and zoning restriction in the BDA is a desirable modification. Further research should focus on doing so and, in order to mitigate computational burden, might include balancing and project scheduling heuristics for the master and slave problems, respectively.

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).

References

Akagi, F., Osaki, H., & Kikuchi, S. (1983). A method for assembly line balancing with more than one worker in each station. *International Journal of Production Research*, *21*, 755–770. doi:10.1080/00207548308942409.

Akpinar, S., Elmi, A., & Bekta, T. (2017). Combinatorial Benders cuts for assembly line balancing problems with setups. *European Journal of Operational Research*, *259*, 527–537. doi:10.1016/j.ejor.2016.11.001.

Bartholdi, J. J. (1993). Balancing two-sided assembly lines: A case study. *International Journal of Production Research*, *31*, 2447–2461. doi:10.1080/00207549308956868.

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, *142*, 259–277. doi:10.1016/j.ijpe.2012.10.020.

Bautista, J., & Pereira, J. (2009). A dynamic programming based heuristic for the assembly line balancing problem. *European Journal of Operational Research*, *194*, 787–794. doi:10.1016/j.ejor.2008.01.016.

Baybars, . (1986). A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem. *Management Science*, *32*, 909–932. doi:10.1287/mnsc.32.8.909.

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, *168*, 694–715. doi:10.1016/j.ejor.2004.07.023.

Becker, C., & Scholl, A. (2009). Balancing assembly lines with variable parallel workplaces: Problem definition and effective solution procedure. *European Journal of Operational Research*, *199*, 359–374. doi:10.1016/j.ejor.2008.11.051.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*, 238–252. doi:10.1007/BF01386316.

Cevikcan, E., Durmusoglu, M. B., & Unal, M. E. (2009). A team-oriented design methodology for mixed model assembly systems. *Computers & Industrial Engineering*, *56*, 576–599. doi:10.1016/j.cie.2007.11.002.

Chen, Y. Y. (2017). A hybrid algorithm for allocating tasks, operators, and workstations in multi-manned assembly lines. *Journal of Manufacturing Systems*, *42*, 196–209. doi:10.1016/j.jmsy.2016.12.011.

Codato, G., & Fischetti, M. (2006). Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. *Operations Research*, *54*, 756–766. doi:10.1287/opre.1060.0286.

Côté, J.-f., Dell'Amico, M., & Iori, M. (2014). Combinatorial Benders' Cuts for the Strip Packing Problem. *Operations Research*, *62*, 643–661. doi:10.1287/opre.2013.1248.

Dimitriadis, S. G. (2006). Assembly line balancing and group working: A heuristic procedure for workers' groups operating on the same product and workstation. *Computers & Operations Research*, *33*, 2757–2774. doi:10.1016/j.cor.2005.02.027.

Fakhri, A., Ghatee, M., Fragkogios, A., & Saharidis, G. K. D. (2017). Benders decomposition with integer subproblem. *Expert Systems with Applications*, *89*, 20–30. doi:10.1016/j.eswa.2017.07.017.

Fattahi, P., & Roshani, A. (2011). A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem. *International Journal of Advanced Manufacturing Technology*, *53*, 363–378. doi:10.1007/s00170-010-2832-y.

Gurobi Optimization (2019). Gurobi Optimizer reference manual.

Hazir, Ö., & Dolgui, A. (2013). Assembly line balancing under uncertainty: Robust optimization models and exact solution method. *Computers & Industrial Engineering*, *65*, 261–267. doi:10.1016/j.cie.2013.03.004.

Hazir, Ö., & Dolgui, A. (2015). A decomposition based solution algorithm for U-type assembly line balancing with interval data. *Computers & Operations Research*, *59*, 126–131. doi:10.1016/j.cor.2015.01.010.

Hillier, F., & Lieberman, G. (2015). *Introduction to Operations Research*. (10th ed.). Heidelberg: Mc Graw Hill.

Hoffmann, T. (1963). Assembly line balancing with a precedence matrix. *Management Science*, *9*, 551–562. doi:10.1287/mnsc.9.4.551.

Kazemi, A., & Sedighi, A. (2013). A cost-oriented model for balancing mixed-model assembly lines with multi-manned workstations. *International Journal of Services and Operations Management*, *16*, 289. doi:10.1504/IJSOM.2013.056765.

Kellegöz, T. (2017). Assembly line balancing problems with multi-manned stations: a new mathematical formulation and Gantt based heuristic method. *Annals of Operations Research*, *253*, 377–404. doi:10.1007/s10479-016-2156-x.

Kellegöz, T., & Toklu, B. (2012). An efficient branch and bound algorithm for assembly line balancing problems with parallel multi-manned workstations. *Computers & Operations Research*, *39*, 3344–3360. doi:10.1016/j.cor.2012.04.019.

Kellegöz, T., & Toklu, B. (2015). A priority rule-based constructive heuristic and an improvement method for balancing assembly lines with parallel multi-manned workstations. *International Journal of Production Research*, *53*, 736–756. doi:10.1080/00207543.2014.920548.

Klein, R. (2000). *Scheduling of Resource-Constrained Projects*. (1st ed.). Springer US. doi:10.1007/978-1-4615-4629-0.

Lopes, T. C., Michels, A. S., Sikora, C. G. S., Molina, R. G., & Magatão, L. (2018). Balancing and cyclically sequencing synchronous, asynchronous, and hybrid unpaced assembly lines. *International Journal of Production Economics*, *203*, 216–224. doi:10.1016/j.ijpe.2018.06.012.

Lopes, T. C., Sikora, C. G. S., Molina, R. G., Schibelbain, D., Rodrigues, L. C. A., & Magatão, L. (2017). Balancing a robotic spot welding manufacturing line: An industrial case study. *European Journal of Operational Research*, *263*, 1033–1048. doi:10.1016/j.ejor.2017.06.001.

Magnanti, T. L., & Wong, R. T. (1981). Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria. *Operations Research*, *29*, 464–484. doi:10.2307/170108.

Maurer, W. D., & Lewis, T. G. (1975). Hash Table Methods. *ACM Computing Surveys (CSUR)*, *7*, 5–19. doi:10.1145/356643.356645.

Michels, A. S., Lopes, T. C., Sikora, C. G. S., & Magatão, L. (2018). The Robotic Assembly Line Design (RALD) problem: Model and case studies with practical extensions. *Computers & Industrial Engineering*, *120*, 320–333. doi:10.1016/j.cie.2018.04.010.

Moon, I., Logendran, R., & Lee, J. (2009). Integrated assembly line balancing with resource restrictions. *International Journal of Production Research*, *47*, 5525–5541. doi:10.1080/00207540802089876.

Moreira, M. C. O., Miralles, C., & Costa, A. M. (2015). Model and heuristics for the Assembly Line Worker Integration and Balancing Problem. *Computers & Operations Research*, *54*, 64–73. doi:10.1016/j.cor.2014.08.021.

Osman, H., & Baki, M. F. (2014). Balancing transfer lines using Benders decomposition and ant colony optimisation techniques. *International Journal of Production Research*, *52*, 1334–1350. doi:10.1080/00207543.2013.842017.

Özcan, U. (2010). Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm. *European Journal of Operational Research*, *205*, 81–97. doi:10.1016/j.ejor.2009.11.033.

Özcan, U., & Toklu, B. (2009). Balancing of mixed-model two-sided assembly lines. *Computers & Industrial Engineering*, *57*, 217–227. doi:10.1016/j.cie.2008.11.012.

Pape, T. (2015). Heuristics and lower bounds for the simple assembly line balancing problem type 1: Overview, computational tests and improvements. *European Journal of Operational Research*, *240*, 32–42. doi:10.1016/j.ejor.2014.06.023.

Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, *259*, 801–817. doi:10.1016/j.ejor.2016.12.005.

Roshani, A., & Giglio, D. (2017). Simulated annealing algorithms for the multi-manned assembly line balancing problem: minimising cycle time. *International Journal of Production Research*, *55*, 2731–2751. doi:10.1080/00207543.2016.1181286.

Roshani, A., & Nezami, F. G. (2017). Mixed-model multi-manned assembly line balancing problem: A mathematical model and a simulated annealing approach. *Assembly Automation*, *37*, 34–50. doi:10.1108/AA-02-2016-016.

Roshani, A., Roshani, A., Roshani, A., Salehi, M., & Esfandyari, A. (2013). A simulated annealing algorithm for multi-manned assembly line balancing problem. *Journal of Manufacturing Systems*, *32*, 238–247. doi:10.1016/j.jmsy.2012.11.003.

Scholl, A., & Klein, R. (1997). SALOME: A bidirectional branch-and-bound procedure for assembly line balancing. *INFORMS Journal on Computing*, *9*, 319–334. doi:10.1287/ijoc.9.4.319.

Sewell, E. C., & Jacobson, S. H. (2012). A Branch, Bound, and Remember Algorithm for the Simple Assembly Line Balancing Problem. *INFORMS Journal on Computing*, *24*, 433–442. doi:10.1287/ijoc.1110.0462.

Sikora, C. G. S., Lopes, T. C., & Magatão, L. (2017). Traveling worker assembly line (re)balancing problem: model, reduction techniques, and real case studies. *European Journal of Operational Research*, *259*, 949–971. doi:10.1016/j.ejor.2016.11.027.

Sternatz, J. (2014). Enhanced multi-Hoffmann heuristic for efficiently solving real-world assembly line balancing problems in automotive industry. *European Journal of Operational Research*, *235*, 740–754. doi:10.1016/j.ejor.2013.11.005.

Walsh, T. (2006). General Symmetry Breaking Constraints. *Principles and Practice of Constraint Programming - CP*, *4204*, 650–664. doi:10.1007/11889205_46.

Yazgan, H. R., Beypinar, I., Boran, S., & Ocak, C. (2011). A new algorithm and multi-response Taguchi method to solve line balancing problem in an automotive industry. *International Journal of Advanced Manufacturing Technology*, *57*, 379–392. doi:10.1007/s00170-011-3291-9.

Yilmaz, H., & Yilmaz, M. (2015). Multi-manned assembly line balancing problem with balanced load density. *Assembly Automation*, *35*, 137–142. doi:10.1108/AA-05-2014-041.

Yilmaz, H., & Yilmaz, M. (2016a). A multi-manned assembly line balancing problem with classified teams: A new approach. *Assembly Automation*, *36*, 51–59. doi:10.1108/AA-04-2015-035.

Yilmaz, H., & Yilmaz, M. (2016b). Note to: a mathematical model and ant colony algorithm for multi-manned assembly line balancing problem. *International Journal of Advanced Manufacturing Technology*, *89*, 1935–1939. doi:10.1007/s00170-016-9223-y.