Computers and Operations Research 99 (2018) 249–261

Contents lists available at ScienceDirect

Computers and Operations Research

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

A memetic algorithm for the cost-oriented robotic assembly line balancing problem

Jordi Pereira[a,*], Marcus Ritt[b], Óscar C. Vásquez[c]

[a] Faculty of Engineering and Sciences, Universidad Adolfo Ibáñez, Av. Padre Hurtado 750, Office C216, Viña del Mar, Chile
[b] Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil
[c] Industrial Engineering Department, Universidad de Santiago de Chile, Chile

ARTICLE INFO

Article history:
Received 1 August 2017
Revised 16 May 2018
Accepted 2 July 2018
Available online 6 July 2018

Keywords:
Line balancing
Cost-oriented line balancing
Robotic assembly line
Hybrid algorithms

ABSTRACT

In order to minimize costs, manufacturing companies have been relying on assembly lines for the mass production of commodity goods. Among other issues, the successful operation of an assembly line requires balancing work among the stations of the line in order to maximize its efficiency, a problem known in the literature as the assembly line balancing problem, ALBP.

In this work, we consider an ALBP in which task assignment and equipment decisions are jointly considered, a problem that has been denoted as the robotic ALBP. Moreover, we focus on the case in which equipment has different costs, leading to a cost-oriented formulation. In order to solve the problem, which we denote as the cost-oriented robotic assembly line balancing problem, cRALBP, a hybrid metaheuristic is proposed. The metaheuristic embeds results obtained for two special cases of the problem within a genetic algorithm in order to obtain a memetic algorithm, applicable to the general problem. An extensive computational experiment shows the advantages of the hybrid approach and how each of the components of the algorithm contributes to the overall ability of the method to obtain good solutions.

1. Introduction

Mass production of commodity goods is often performed in a production environment known as an assembly line. An assembly line organizes work stations serially along a common transportation method, like a conveyor belt. The assembly of the product is divided into elementary operations, and these operations, usually referred to as tasks, are performed in one of the work stations in a pre-established sequence. Given the predefined nature of the process, assembly lines are able to increase productivity by an efficient use of division of labor, operation specialization and process standardization.

Among the different problems concerned with assembly line design and operation, assembly line balancing is one of the most thoroughly studied problems in the literature (see Baybards (1986), Becker and Scholl (2006), Dolgui and Battaïa (2013), Rekiek et al. (2002), Scholl and Becker (2006), for a review of previous work). ALBPs deal with the assignment of the tasks to the work stations in order to optimize some efficiency criterion, while satisfying resource (task times) and technological (precedence) constraints.

The most simplified formulation of the problem is known in the literature as the simple ALBP or SALBP (see Morrison et al. (2014), Otto et al. (2013), Pape (2015), Pereira (2015), for some recent work). The SALBP considers that all stations are equally equipped and efficiency is measured in terms of idle time of the assembly line, that is, the difference between the total time allotted to the stations and the minimum time required to perform all of the tasks. A large body of literature has considered the resolution of the SALBP and several heuristic (Pape, 2015; Sternatz, 2014) and exact (Morrison et al., 2014) procedures are available. While the SALBP is strongly NP-hard, since it subsumes the bin packing problem as a special case (Garey and Johnson, 1979), the literature reports the solution to optimality of large instances with up to 1000 tasks.

As the SALBP is a simplification of many real-life line balancing problems, and it can be considered to be solvable for practical purposes, the research community has been increasingly focused on more general formulations that incorporate additional requirements found in different industrial settings. This work follows this line of research and studies a formulation in which the stations are not equally equipped. Such problems have been denoted

* Corresponding author.
E-mail addresses: jorge.pereira@uai.cl (J. Pereira), marcus.ritt@inf.ufrgs.br (M. Ritt).

https://doi.org/10.1016/j.cor.2018.07.001
0305-0548/

in the literature with different names, like the robotic assembly line balancing problem, RALBP (Rubinovitz et al., 1993) or the assembly line worker assignment and balancing problem, ALWABP (Miralles et al., 2007), among others. These problems extend the SALBP by considering that the time required to perform an operation depends on the equipment of the station performing it. Thus, the problem integrates assembly line design decisions, specifically equipment selection, within the SALBP, introducing a better representation of the operation times at the expense of an additional level of difficulty.

A related but different approach to account for heterogeneity among stations assumes that tasks have varying degrees of difficulty and require operators with different skill levels. Consequently, task times do not change among operators but operators can or cannot perform operations according to their skill levels. This problem is known in the literature as the cost-oriented ALBP (Amen, 2000; 2006) and aims at finding an assignment of operations to stations such that the wage of the operators (a parameter correlated to their skill level) is minimized. Hence, the formulation introduces a cost-oriented objective to the problem not present in the SALBP or in the RALBP.

While robots differ on required operation times and in their costs, both on their buying price and operation costs, line balancing problems integrating costs and variable processing times have not been extensively considered in the literature see Nilakantan et al. (2017), for a similar recent work. This work puts forward an alternative formulation to the one previously proposed, which we will denote as the cost-oriented robotic assembly line balancing problem, cRALBP. The proposed formulation considers explicitly fixed and variable costs, and accounts for throughput requirements by a cycle time constraint not present in Nilakantan et al. (2017).

In addition to the general model, two special cases of the problem are separately considered: the case in which the order of operation of the tasks is known in advance, and the case in which the number of stations is two. The first problem is shown to be polynomially solvable using dynamic programming, and the second is shown to be strongly NP-hard, but solvable to optimality within very short running times by a variation of a state-of-the-art heuristic for the SALBP (Fleszar and Hindi, 2003; Sternatz, 2014). The algorithms proposed for these problems are then combined with a genetic algorithm in order to develop a hybrid metaheuristic approach, a memetic algorithm. The effectiveness of each component of the proposed approach is tested on a set of instances derived from instances previously found in the literature (Gao et al., 2009; Otto et al., 2013) showing the advantages of the combined approach.

The remainder of the paper is structured as follows: Section 2.2 provides an in-depth review of related work, puts forward the problem formulation and provides several lower bounds for the problem. Section 3 studies the special cases and provides specific algorithms for their resolution. Section 4 details the proposed solution methods. Section 5 reports the computational experiments performed. Finally, Section 6 provides some conclusions and outlines future work.

2. The cost-oriented robotic assembly line balancing problem

2.1. Literature review

According to Boysen et al. (2007), the cRALBP can be denoted by the tuple [*pa, cum* | *equip* | *Co*]. The first field, associated to the task characteristics, indicates that there are several processing alternatives, attribute *pa*, and that a cumulative constraint, attribute *cum*, corresponding to the cycle time, has to be fulfilled. The second field, associated to station characteristics, specifies that exactly one equipment among the alternatives should be selected, and the third field refers to the cost-oriented nature of the objective function. Moreover, according to Dolgui and Battaïa (2013), the problem can be seen both as a line balancing problem with additional task attributes, the processing alternatives, as well as a line balancing problem with additional station attributes.

Due to the large body of research geared towards the formulation and resolution of different assembly line balancing problems, we refer the interested reader to the available state-of-the-art reviews and classification schemes (Becker and Scholl, 2006; Boysen et al., 2007; Dolgui and Battaïa, 2013; Scholl and Becker, 2006) and focus our description of the literature on previous work devoted to problems with similar characteristics to the cRALBP, namely models with processing alternatives and cost-oriented objectives.

Several formulations with processing alternatives have been proposed in the literature. The robotic assembly line balancing problem, RALBP (Rubinovitz et al., 1993), is one of the seminal works in which both the assignment of tasks and the equipment of the stations from a list of processing alternatives are simultaneously considered. The RALBP considers a fixed number of stations and the objective is to maximize throughput, i.e., to minimize the cycle time, by assigning tasks to stations and simultaneously choosing the best equipment (the robot minimizing the total task time) for each station. This formulation has been further explored in subsequent works (Borba et al., 2018; Gao et al., 2009; Levitin et al., 2006) sometimes with different names (Bukchin and Tzur, 2000) in order to avoid referencing alternative equipment as robots. In addition to the basic formulation, the literature covers several works that consider extended formulations, including U-shaped assembly lines (Nilakantan and Ponnambalam, 2016), mixed-model assembly lines (Çil et al., 2017) or robotic lines with additional design decisions, like locating the picking points for the assembled parts (Daoud et al., 2014).

While RALBP models have efficiency-based objectives, i.e., minimizing the number of stations or the cycle time of the line, some work has studied cost-oriented goals. Yoosefelahi et al. (2012) study a multiobjective RALBP with an efficiency-based objective, cycle time minimization, as well as two cost-oriented objectives, fixed and variable cost minimization. Recently, Nilakantan et al. (2017) have considered cost minimization with selection of robotic alternatives. The authors consider the minimization of an aggregate of task-to-robot costs, but do not define throughput constraints, either by limiting the cycle time or the number of stations of the line. Consequently, the proposed model minimizes variable operation costs but may fail to satisfy productivity requirements as minimizing variable operation costs may lead to arbitrarily low production rates.

Another formulation closely related to the RALBP is the assembly line and worker assignment balancing problem, ALWABP (Miralles et al., 2007). While the RALBP usually supposes that multiple units of each processing alternative are available, the ALWABP considers that each processing alternative is a worker from the available workforce. Therefore, the solution of the ALWABP should assign each worker to one of the stations of the assembly line, as opposed to the selection of the best processing alternative for each station as in the RALBP. The ALWABP has also been extensively studied in subsequent works (Blum and Miralles, 2011; Borba and Ritt, 2014; Moreira et al., 2012; Mutlu et al., 2013; Polat et al., 2016; Vilà and Pereira, 2014, among others), and extensions have also been proposed, like parallel stations (Araújo et al., 2012), multiple objectives (Moreira et al., 2017; Zacharia and Nearchou, 2016), stochastic worker availability (Ritt et al., 2016), robust optimization oriented approaches (Moreira et al., 2015a), as well as more elaborate formulations which account for the current as well as complementary, additional, workforce (Moreira et al., 2015b).

A third formulation in which equipment decisions are considered is the cost-oriented assembly line balancing problem, cALBP (Amen, 2000; 2006). In the cALBP, the tasks have different difficulty levels and require a worker with a minimum skill level to perform them. Hence, the operation time for a task-worker pair is infinite if the skill level does not reach the difficulty degree of a task, and a fixed value in any other case. As workers with different skill levels have different wages, the objective is to find an assignment of tasks to stations such that the total cost of the workers is minimized, while ensuring that each station is manned by a worker with an adequate skill level to perform all of its assigned tasks. Note that the cALBP differs from the RALBP and from the ALWABP both in its cost-oriented objective, and in its definition of processing alternatives.

The RALBP, the ALWABP and the cALBP consider the selection of one alternative among a set of available alternatives for each station, but the literature also covers work on the selection of multiple pieces of equipment per station (see Corominas et al. (2011), Gadidov and Wilhelm (2000), Pekin and Azizoglu (2008), Triki et al. (2017), among others). The models proposed in these works differ on how equipment decisions affect task assignments, but share a cost-oriented objective (see Hazir et al. (2015), for a review of work addressing different cost-oriented issues in the ALBP). In this case, the cost-oriented objective reflects the trade-offs involved in equipment decisions. Notice that some works consider that the operation times depend on the equipment selection (Gadidov and Wilhelm, 2000; Pekin and Azizoglu, 2008; Triki et al., 2017) as in the RALBP or ALWABP, while others consider that tasks need some specific equipment configuration to be performed (Corominas et al., 2011), similar to the cALBP. Moreover, some multi-objective approaches exist, like minimizing costs while maximization of the throughput of the assembly line (Pekin and Azizoglu, 2008; Triki et al., 2017). Note that the selection of multiple equipment per station can be modeled as the selection of a single equipment per station if the set of processing alternatives contains all the combinations of equipments that can be assigned to a station.

While this review has focused on the different formulation alternatives previously considered, these problems have been solved using different exact and heuristic procedures. We refer to Dolgui and Battaïa (2013) for a more detailed comparison of different methods, as well as to Tasan and Tunali (2008) for a review of works using genetic algorithms, which is one of the more frequently used metaheuristics in the literature to tackle this type of problems.

2.2. Model formulation

A formal definition of the problem studied in this paper follows. We consider the problem of assigning a partially ordered set of tasks T to a linearly ordered set of stations S. Each station is manned with a single robot from a given set of robots R, each $r \in R$ with a fixed cost c_r^f. Please note that the same type of robot can be assigned to several stations and some robots in R may not be used. Each task $t \in T$ has a set of predecessors $P(t)$, an operation time p_{rt}, and a variable cost c_{rt} that depends on the robot $r \in R$, assigned to the station performing the task. Denote the total execution time of the tasks assigned to a station as its *station load*. In order to satisfy a desired production rate, the station loads of the stations cannot exceed a given cycle time c. Moreover, the assignment of tasks to stations must respect the partial order of tasks, i.e., if task t precedes task t', $t \in P(t')$, then t has to be assigned to the same or an earlier station than t'. The objective is to minimize the total cost, i.e., the sum of fixed and variable costs, which we will consider to have been discounted and calculated for a common time frame (e.g., the life cycle of the assembly line), while satisfying a desired production rate (that is, if robot $r \in R$ is assigned to station $s \in S$ and S_s denotes the set of tasks assigned to station s, $\sum_{t \in S_s} p_{rt} \leq c$ must hold for each station).

Table 1
Fixed cost $c_\cdot^f$ of the robots and variable costs c_t^v of performing each operation with a given robot. The solution of Fig. 1 would cost 59 units summing the fixed costs 21 + 13, and variable costs $(2+3+4)+(5+6+5)$.

Cost	Robot 1	Robot 2	Robot 3
$c_\cdot^f$	21	6	13
$c_{\cdot 1}^v$	2	8	4
$c_{\cdot 2}^v$	3	10	4
$c_{\cdot 3}^v$	4	10	4
$c_{\cdot 4}^v$	3	5	5
$c_{\cdot 5}^v$	2	6	6
$c_{\cdot 6}^v$	3	5	5

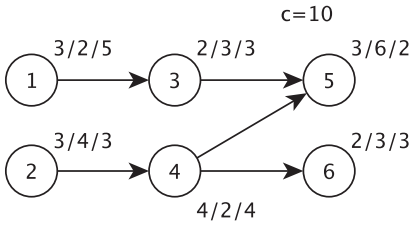

Fig. 1. Precedence graph and operation times of the tasks. A feasible (non-necessarily optimal) solution could perform tasks 1, 2 and 3 at station 1 manned by a robot of type 1 and perform the remaining tasks at station 2 manned by a robot of type 3.

As $|S| \leq |T|$ holds, if no data on the maximum number of stations is given, we can assume $|S| = |T|$. Note that the sets of precedence relations are sometimes represented using a precedence graph $G(T, A)$ in which each vertex $t \in T$ corresponds to a task and each arc corresponds to a precedence relation, that is, if $t \in P(t')$, then $(t, t') \in A$. Moreover, define $P^*(t')$ as the set of all predecessors of task t', $t \in P^*(t')$ if there is a path from t to t' in $G(T, A)$, and $F^*(t')$ as the set of all successors of task t', a path in $G(T, A)$ from t' to t exists.

Fig. 1 and Table 1 provide a sample instance with six tasks and three different robots, and one of its feasible solutions. Fig. 1 describes the precedence relations as a directed acyclic graph $G(T, A)$ in which each node corresponds to a task, and an arc leaving node t entering node t' indicates that task t precedes task t'. Fig. 1 also provides the cycle time of the example, $c = 10$, and the operation time of each task if performed by each of the three different robots. For example, task 1 requires 3, 2 or 5 time units if performed, respectively, by robot 1, 2 or 3. Table 1 reports the fixed costs, $c_\cdot^f$, of each robot and the variable costs $c_{\cdot t}^v$ of each robot and task. A feasible solution to this instance would be to assign tasks 1, 2 and 3 to station 1 manned by a robot of type 1 and to assign the remaining tasks to station 2 and equip it with a unit of robot 3. Consequently, the station loads would be 8 and 9 time units respectively. The fixed cost of such an assignment would be 34 and the variable cost 25 for a total cost of 59 units.

The described problem can be formulated as a binary program as follows. We use binary variables y_{rs} to indicate the assignment of a robot $r \in R$ to station $s \in S$, and binary variables x_{rst} to indicate the execution by robot $r \in R$ at station $s \in S$ of task $t \in T$. Then we have to

$$\text{Minimize cost} = \sum_{r \in R} \sum_{s \in S} c_r^f y_{rs} + \sum_{r \in R} \sum_{s \in S} \sum_{t \in T} c_{rt}^v x_{rst}, \tag{1}$$

$$\text{subject to } \sum_{t \in T} p_{rt} x_{rst} \leq c y_{rs}, \qquad \forall r \in R, \ \forall s \in S, \tag{2}$$

$$\sum_{r\in R}\sum_{s\in S} x_{rst} = 1, \qquad \forall t \in T, \tag{3}$$

$$\sum_{r\in R} y_{rs} \leq 1, \qquad \forall s \in S, \tag{4}$$

$$\sum_{r\in R}\sum_{s\in S} s x_{rst} - \sum_{r\in R}\sum_{s\in S} s x_{rst'} \leq 0, \qquad \forall t \in P(t'), \tag{5}$$

$$x_{rst} \in \{0,1\}, \qquad \forall r \in R,\ \forall s \in S,\ \forall t \in T, \tag{6}$$

$$y_{rs} \in \{0,1\}, \qquad \forall r \in R,\ \forall s \in S. \tag{7}$$

The objective function (1) minimizes the total sum of fixed and variable costs. Constraint set (2) ensures the desired throughput (the tasks assigned to a station can be performed within the allotted cycle time by the robot equipping the station). Constraint set (2) also ensures that tasks assigned to a station must be performed by the robot available at that station, while constraint set (3) ensures that each task is assigned to a station. Constraints (4) state that a station can be manned by a single robot. Note that some stations may remain unused, and no robot will be assigned to them. Constraint set (5) ensures the precedence constraints, and constraints (6) and (7) define the domain of the variables. The model has $O(|R|\,|S|\,|T|)$ binary variables and $O(|R|\,|S| + |T|^2)$ constraints.

2.3. Lower bounds

The linear relaxation of the previous model defined by the continuous domain of (6) and (7) constitutes a valid lower bound on the optimal solution. Moreover, the optimal solution of the relaxation can be easily computed, as Lemma 2.1 points out.

Lemma 2.1. *Consider the linear relaxation of the proposed formulation defined by the continuous domain of (6) and (7). If the cardinality of the ordered set of stations $|S|$ does not impose any constraint on the task assignments, i.e., $|S| = |T|$, then the value of any optimal solution is given by*

$$\sum_{t\in T} \min_{r\in R}\left\{c^v_{rt} + \frac{p_{rt}}{c} c^f_r\right\}. \tag{8}$$

Proof. We first note that any optimal solution of the relaxation of the proposed formulation defined by the continuous domain of (6) and (7) satisfies (2) in strict equality (i.e., there is no need to set y_{rs} to a positive value other than satisfying the constraints if a variable x_{rst} is non-zero). Consequently we replace y_{rs} in the objective function (1) and have:

$$\sum_{r\in R}\sum_{s\in S} c^f_r \sum_{t\in T} \frac{p_{rt}}{c} x_{rst} + \sum_{r\in R}\sum_{s\in S}\sum_{t\in T} c^v_{rt} x_{rst}$$
$$= \sum_{r\in R}\sum_{s\in S}\sum_{t\in T}\left(\frac{c^f_r p_{rt}}{c} + c^v_{rt}\right) x_{rst},$$

where the parameters are independent in s.

Now consider the set of assignment constraints stated by (3) and note that any type r of robot can be assigned to any, and possibly to several, stations. Thus, the contribution to the objective function (1) of the sum over the fraction optimal variables x_{rst} for a fixed t' is equal to $c^f_{r'}\frac{p_{r't'}}{c} + c^v_{r't'}$, where $r' = \arg\min_{r\in R} c^f_r \frac{p_{rt'}}{c} + c^v_{rt'}$. Each task can be assigned to a single station following a topological ordering of the tasks, as it does not impose any constraint on the task assignments and satisfies the set of precedence constraints (5) and robot to station assignment constraints (4). Thus, the value of any optimal solution equals (8) which concludes the proof. Note that the optimal solution is not necessarily unique. □

This bound is easy to compute but much like the equivalent linear relaxation of its SALBP counterpart (Ritt and Costa, 2015) it is not a tight lower bound. Therefore, solving the ILP model (1)–(7) is ineffective. Consequently, we provide an alternative bound based on a Dantzig–Wolfe type reformulation of the problem (Peeters and Degraeve, 2006). This reformulation has reported good lower bounds for the SALBP (Pereira, 2015) at the expense of additional computing requirements, and it has been successfully applied to other line balancing problems (see Bautista and Pereira (2011), for an example).

Let $\mathcal{Q}^{(r)}$ be the set of all feasible station assignments of robot r, that is, the set of all possible subsets of tasks that can be performed by robot r at a single station while fulfilling the cycle time and precedence constraints among the assigned tasks, and let $\vec{q}_l = (q_{l1}, \ldots, q_{l|T|})$ be one of these station assignments:

$$\mathcal{Q}^{(r)} = \left\{\vec{q}_l \in \{0,1\}^{|T|} : \left(\sum_{t\in T} p_{rt} q_{lt} \leq c\right) \wedge \Big(q_{lt''} + q_{lt'} \leq q_{lt} + 1\ \forall t, t', t'' \in T : t \in P^*_{t'} \cap F^*_{t''}\Big)\right\}. \tag{9}$$

Consequently, the set of all feasible assignments corresponds to $\mathcal{Q} = \mathcal{Q}^{(1)} \cup \cdots \cup \mathcal{Q}^{(|R|)}$.

Define a set of binary variables $z_{\vec{q}_l,s}$ whose value is 1 if assignment $\vec{q}_l$ is assigned to station s and 0, otherwise. The total cost $\hat{c}_{\vec{q}_l}$ of the assignment corresponds to the minimum sum of variable and fixed costs over each different robot:

$$\hat{c}_{\vec{q}_l} = \min_{r\in R:\vec{q}_l\in\mathcal{Q}^{(r)}}\left\{c^f_r + \sum_{t\in T} c^v_{rt} q_{lt}\right\}. \tag{10}$$

Then equations (11)–(15) provide an alternative formulation for the cRALBP:

$$\text{Minimize Cost} = \sum_{s\in S}\sum_{\vec{q}_l\in\mathcal{Q}} \hat{c}_{\vec{q}_l} z_{\vec{q}_l,s}, \tag{11}$$

subject to

$$\sum_{s\in S}\sum_{\vec{q}_l\in\mathcal{Q}} q_{lt} z_{\vec{q}_l,s} = 1, \quad \forall t \in T, \tag{12}$$

$$\sum_{\vec{q}_l\in\mathcal{Q}} z_{\vec{q}_l,s} \leq 1, \quad \forall s \in S, \tag{13}$$

$$\sum_{s\in S}\sum_{\vec{q}_l\in\mathcal{Q}} s q_{lt} z_{\vec{q}_l,s} \leq \sum_{s\in S}\sum_{\vec{q}_l\in\mathcal{Q}} s q_{lt'} z_{\vec{q}_l,s}, \quad \forall t' \in T,\ t \in P(t'), \tag{14}$$

$$z_{\vec{q}_l,s} \in \{0,1\}, \quad s \in S,\ \vec{q}_l \in \mathcal{Q}. \tag{15}$$

Objective (11) minimizes the cost of the selected station assignments. Constraint set (12) ensures that each task belongs to one of the selected station assignments. Constraint set (13) ensures that at most one assignment per station is selected. Constraint set (14) corresponds to the precedence constraints and (15) defines the domain of the variables to be binary.

In order to obtain a relaxation based on the formulation, constraint set (15) is relaxed to its continuous domain. Moreover, precedence constraints (14) are removed as they complicate the pricing problem without significantly improving the lower bound (Peeters and Degraeve, 2006; Pereira, 2015). Once the precedence constraints are removed, the station index can be dropped from the set of variables, leading to the following set covering problem formulation (16)–(18)

$$\text{Minimize Cost} = \sum_{\vec{q}_l\in\mathcal{Q}} \hat{c}_{\vec{q}_l} z_{\vec{q}_l}, \tag{16}$$

subject to

$$\sum_{\vec{q_l} \in \mathcal{Q}} q_{lt} z_{\vec{q_l}} \geq 1, \quad \forall t \in T, \tag{17}$$

$$0 \leq z_{\vec{q_l}} \leq 1, \quad \vec{q_l} \in \mathcal{Q}. \tag{18}$$

The cardinality of $\mathcal{Q}$ makes its explicit enumeration inefficient. Since the number of task packings without precedences is bounded by $\binom{|T|+c}{c}$ (Fernandez de la Vega and Lueker, 1981), $|Q|$ is bounded by $|R|\binom{|T|+c}{c}$. This bound is pseudo-polynomial in the size of the input, and thus the number of feasible assignments may be exponential as it is the case in many other packing problems (Valério De Carvalho, 2002). As it is impractical to enumerate all the variables, a column generation approach is used. The model is initialized with some station assignments, and additional feasible station assignments (9), are added solving a pricing problem until no more variables with negative reduced costs exists. Specifically, a pricing problem per robot is used. Let $\pi = (\pi_1, \ldots, \pi_{|T|})$ be the vector of dual prices of the assignment constraints (17). Then for each robot the pricing problem corresponds to (19) subject to the domain defined in (9).

$$\text{Minimize} \quad c_r^f + \sum_{t \in T} \left(c_{rt}^v - \pi_t\right) q_{lt}. \tag{19}$$

The pricing problem can be solved using a specially designed branch-and-bound algorithm (Peeters and Degraeve, 2006) or an off-the-shelf integer linear programming solver with objective (19) and the constraints defined by (9), as it is the case in the present work.

3. Two special cases

In this section we discuss two special cases of problem. The first case corresponds to a problem where the order of the operations is known, i.e., tasks are linearly ordered. The second case is the problem with only two stations, i.e., $|S| = 2$. The methods developed to solve each special case will be later used to evaluate feasible task orderings as well as to define a local search procedure.

3.1. Linearly ordered task set

Suppose that the tasks have to be executed in a specific order, that is $T = \{t_1, t_2, \ldots, t_{|T|}\}$ and $P(t_i) = \{t_{i-1}\}$ for all $i \in 2, \ldots, |T|$. In other words, the tasks are linearly ordered.

For linearly ordered instances, an easy pseudo-polynomial dynamic program with $O(|R|^2\, c\, |T|)$ time complexity computes the optimal solution. Let $C(r, l, i)$ be the minimal total cost of performing tasks $t_i, t_{i+1}, \ldots, t_{|T|}$ if the current station is manned by robot $r \in R$ and has station load l. In order to compute the cost $C(r, l, i)$ the recurrence decides whether task i is executed by robot r, if the resulting station load does not exceed the cycle time, or if the current station will be closed, and the task will be executed on the next station by a new robot. The optimal cost corresponds to $\min_{r \in R}\{C(r, 0, 1) + c_r^f\}$ and the cost function satisfies the recurrence

$$C(r, l, |T| + 1) = 0,$$

$$C(r, l, i) = \begin{cases} \min \left\{ \begin{array}{l} c_{rt_i}^v + C(r, l + p_{rt_i}, i + 1), \\ \min_{r' \in R: p_{r't_i} \leq c} \{c_{r't_i}^v + c_{r'}^f + C(r', p_{r't_i}, i + 1)\} \end{array} \right\} & \text{if } l + p_{rt_i} \leq c, \\ \min_{r' \in R: p_{r't_i} \leq c} \left\{ c_{r't_i}^v + c_{r'}^f + C(r', p_{r't_i}, i + 1) \right\} & \text{otherwise.} \end{cases} \tag{20}$$

To see the correctness of recurrence (20), note that for any robot r, load l and initial task i the optimal cost $C(r, l, i)$ will be achieved by one of two possible choices: i) if there is enough idle time on the current station, i.e., $l + p_{rt_i} \leq c$, then task T_i may be executed on it. This generates a variable cost $c_{rt_i}^v$, and leaves the residual problem of executing optimally the remaining tasks $i + 1, \ldots$ starting at the current station with robot r and an increased load of $l + p_{rt_i}$; ii) alternatively, even if there is enough idle time, the current station may be closed, and task T_i can be executed on a new station manned by some robot r'. This option has variable cost $c_{r't_i}^v$ for executing T_i, fixed cost $c_{r'}^f$ for robot r', and leaves the residual problem of executing optimally the remaining tasks $i + 1, \ldots$ starting at the new station with robot r and a load of $p_{r't_i}$. The minimum over all choices of r' on the new station then gives the optimal cost. Finally, the minimum over the two possible choices will lead to the optimal cost.

The correctness of recurrence (20) now follows from an easy backward induction over i. The corresponding dynamic programming table has $O(|R|\, c\, |T|)$ entries, and each entry can be computed in time $O(|R|)$. Thus, solving the dynamic program takes total time of $O(|R|^2\, c\, |T|)$.

However, we claim that this special case can be solved in polynomial time in the input size of cRALBP $|R||T|$ by dynamic programming as shown in Theorem 3.1.

Theorem 3.1. *The cRALBP with a given specific order to execute the tasks admits a polynomial time algorithm in the input size.*

Proof. A polynomial time algorithm for the cRALBP with a given specific order to execute the tasks is obtained by a more efficient dynamic program, deciding for each station which prefix of the remaining tasks and which robot will be assigned to it. Let $C(i)$ be the minimal total cost of executing tasks $t_i, t_{i+1}, \ldots, t_{|T|}$. These costs satisfy the recurrence

$$C(|T| + 1) = 0,$$

$$C(i) = \min_{\substack{r \in R, k \in \{i+1, \ldots, |T|+1\}: \\ \sum_{i \leq j < k} p_{rt_j} \leq c}} \left\{ c_r^f + \sum_{i \leq j < k} c_{rt_j}^v + C(k) \right\} \tag{21}$$

To see the correctness of recurrence (21), note that for initial task i some prefix $T_i, \ldots, T_{k-1}$, $k > i$, of the tasks must be executed on the first station by some robot r. This has fixed cost c_r^f and variable cost $\sum_{i \leq j < k} c_{rt_j}^v$. The station is then closed, and we are left with the residual problem of executing the remaining tasks $k, k + 1, \ldots$ starting at a free station. The optimal cost is obtained by minimizing over all feasible choices of r and k, which must satisfy $\sum_{i \leq j < k} p_{rt_j} \leq c$.

The corresponding dynamic programming table has $O(|T|)$ entries and each entry takes time $O(|R|\, |T|)$ to compute. Thus the optimal assignment of robots and tasks to stations can be found in time $O(|T|^2\, |R|)$. □

This formulation can be seen as a shortest path problem in a directed acyclic graph (DAG) in which the nodes correspond to the states of the recurrence relation and the transitions of recurrence (21) correspond to the arcs of the shortest path. Then, the optimal policy corresponds to the shortest path from state 1 to a final state $|T| + 1$.

Notice that the formulation can be used to evaluate the optimal assignment of tasks and robots to workstations if the sequence of operations is fixed beforehand. For instance, suppose that the order of operation of the example instance described in Fig. 1 and Table 1 is defined as $T = \{1, 2, \ldots, |T|\}$. Then, we can construct a DAG to find the optimal policy, see Fig. 2. The optimal assignment of tasks and robots to stations is to perform task 1 at the first station manned by a robot of type 2, tasks 2, 3 and 4 at station number 2 manned by a robot of type 3 and the last two tasks at station number 3 manned by a robot of type 2. The total cost of

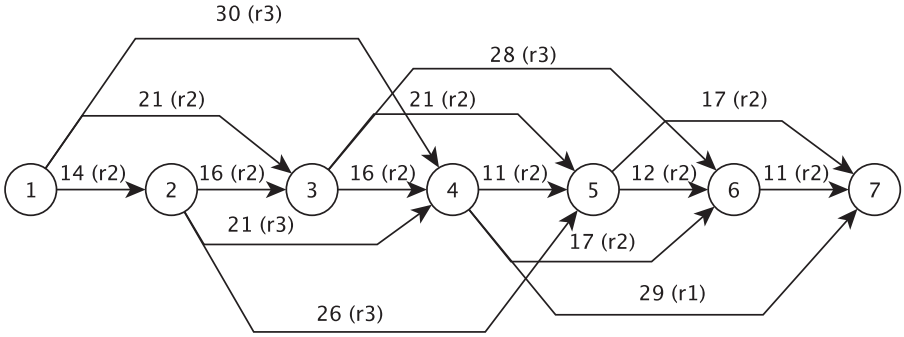

Fig. 2. DAG representation of the dynamic programming recurrence. A solution corresponds to a path between node 1 and node 7, and the optimal assignment of tasks and robots to stations is the least cost path. The cost of each arc (tt') corresponds to the minimum cost assignment, including fixed and variable costs, of performing tasks t to $t'-1$. The cost, as well as the robot generating.

this solution is 57, which improves upon the solution provided in Section 2.2.

3.2. The two-station problem

We now consider the special case in which we aim at finding an optimal solution with only two stations, that is $|S| = 2$, if one exists. This particular problem will be used to define the local neighborhood of the proposed solution procedure.

A subset $P \subseteq T$ of tasks is a *valid prefix* if $t \in P \Rightarrow t' \in P$, for all $t' \in P(t)$ and $\Sigma_{t \in P} p_{rt} \leq c$ for some $r \in R$. In other words, a valid prefix corresponds to a valid assignment of tasks to the first station according to precedences and the cycle time constraint for one or more robots. Let $\mathcal{P}$ be the set of all valid prefixes. For a subset of tasks P, let $c(P, r) = c_r^f + \sum_{t \in P} c_{rt}^v$ and $p(P, r) = \sum_{t \in P} p_{rt}$ be, respectively, the total cost and the total operation time required to execute the tasks in P using a robot of type $r \in R$. Furthermore, let $\overline{P} = T \setminus P$. Then, the two-station subproblem is equivalent to

$$\min_{P \in \mathcal{P}} \min_{r, r' \in R} \left\{ c(P, r) + c(\overline{P}, r') \mid p(P, r) \leq c \wedge p(\overline{P}, r') \leq c \right\}. \tag{22}$$

In other words, the problem can be solved by enumerating all valid prefixes for the first station, defining the second station assignment as its complement, checking their cost and feasibility according to the cycle time constraint, and selecting the assignment that minimizes the aggregated cost (22). Before discussing how to solve the problem, we show that the problem is NP-hard in strong sense even when limited to two stations, see Theorem 3.2. Our proof is based on the Partially Ordered Knapsack (POK), which is known to be NP-complete in the strong sense (Johnson and Niemi, 1983).

Theorem 3.2. *The cost-oriented Robotic Assembly Line Balancing Problem (cRALBP) is NP-hard in strong sense even when the number of stations, $|S|$, is two.*

Proof. The cost-oriented Robotic Assembly Line Balancing Problem (cRALBP) is clearly in *NP*, as the feasibility of a solution can be checked in polynomial time. To show that the cRALBP is NP-hard in strong sense, we consider the partially ordered knapsack (POK) (Johnson and Niemi, 1983):

> **Instance:** Given a directed acyclic graph $G(V, A')$, a weight $w(v) \in \mathbb{Z}_0^+$ and a value $p(v) \in \mathbb{Z}_0^+$ for each vertex $v \in V$, a knapsack capacity $B \in \mathbb{Z}^+$, and a bound $C \in \mathbb{Z}^+$.
>
> **Question:** Is there a subset $V' \subseteq V$, closed under predecessor, such that $\sum_{v \in V'} w(v) \leq B$ and $\sum_{v \in V'} p(v) \geq C$?

Note that the POK is NP-complete in the strong sense even if $p(v) = w(v)$. Therefore, we consider $p(v) = w(v)$, $\forall v \in V$ and use $p(v)$ and $w(v)$ indistinctly.

We construct an instance $\mathcal{I}$ of cRALBP as follows: a cycle time equal to $B \in \mathbb{Z}^+$, two types of robots 1 and 2, a directed acyclic graph $G'(T, A)$, where the nodes describe the set of tasks $T = V \cup \{\alpha, \omega\}$ and the arcs describe the set of precedence relations. In addition to the set A' of precedences from $G(V, A')$, the set A contains an arc from α to each vertex in V, and an arc from each vertex in V to ω. Each task $v \in V$ has an operation time equal to $p(v)$ when it is performed by a robot of type 1 and zero when it is performed by a robot of type 2, while the variable cost is zero when it is performed by a robot of type 1 and $w(v)$ when it is performed by a robot of type 2. The variable cost of task α is 0 for each robot, and its operation time is 0 when performed by a robot of type 1 and $B+1$ when it is performed by a robot of type 2. The variable cost of task ω in each robot is 0, and its operation time is $B+1$ when it is performed by a robot of type 1 and zero when performed by a robot of type 2. The fixed cost of all types of robots is B.

We claim that instance $\mathcal{I}$ has a solution S with cost at most $C = B + \sum_{v \in V} p(v) \in \mathbb{Z}^+$ if and only if there exists a solution to the POK instance.

For the easy direction, given a solution $V' \subseteq V$ to the POK instance we construct an assignment consisting of tasks $V' \cup \{\alpha\}$ on the first station manned by a robot of type 1 and the remaining tasks on a second station manned by a robot of type 2. Straightforward verification shows that the resulting solution has the required value C.

For the hard direction, given the instance $\mathcal{I}$, we consider a solution to the cRALBP of total cost C. Its cost cannot be smaller. We note that any feasible solution must have a minimum of two stations by cycle time B, the first manned by a robot of type 1 and whose assignment includes task α, and the second manned by robot of type 2 and whose assignment includes task ω. Another feasible solution with three or more stations has a value greater that C because the fixed cost of a robot is higher than any savings in variable costs. Thus, in any feasible solution, the first station contains task α plus some additional tasks of V closed under precedence relations, the set V', the second station contains the remaining tasks of V and task ω and fulfills precedence constraints. The cost of the feasible solution is equal to $2B + \sum_{v \in V} p(v) - \sum_{v \in V'} p(v)$. Then, the optimal solution is equivalent to the subset V' maximizing $\sum_{v \in V'} p(v)$ while the cycle time constraint at the first station $\sum_{v \in V'} w(v) \leq B$ is satisfied. Therefore, the optimal solution with a value $B + \sum_{v \in V} w(v)$, corresponds to the solution of the POK instance. This concludes the proof. □

While enumerating $\mathcal{P}$ is theoretically difficult, as shown by Theorem 3.2, enumerating all station assignments for each station in turn has been successfully used as a subroutine of the Hoffmann heuristic (Fleszar and Hindi, 2003; Hoffmann, 1963), as well

as in other state-of-the-art exact and heuristic procedure for several ALBPs, (see Bautista and Pereira (2009), Sewell and Jacobson (2012), Sternatz (2014), Vilà and Pereira (2014), among others). Moreover, as the assignment of the second station corresponds to the complement of the assignment, only one station needs to be enumerated in order to obtain the optimal solution for the two-station case. Algorithm 1 details the implementation.

Algorithm 1 Station enumeration (based on Fleszar and Hindi (2003)).

Input: An instance of the cRALBP.
Output: An optimal assignment of tasks to the first station, $\mathcal{A}^*$.

```
1: function Cost(A)
2:   Q = {r ∈ R : c ≥ p(A, r)}, Q' = {r ∈ R : c ≥ p(Ā, r)}
3:   if Q = ∅ ∨ Q' = ∅ then
4:     return ∞
5:   end if
6:   return min_{r∈Q, r'∈Q'} {c(A, r) + c(Ā, r')}
7: end function
8: function AddNewAvailable(i)
9:   for each t ∈ F_i do
10:    K_t := K_t − 1
11:    if K_t = 0 then Append t to L
12:    end if
13:  end for
14: end function
15: function RemoveNewAvailable(i)
16:  for each t ∈ F_i do
17:    if K_t = 0 thenRemove t from L
18:    end if
19:    K_t := K_t + 1
20:  end for
21: end function
22: function OnePackingSearch(q)
23:  for i := q to |L| do
24:    t := L_i, Q = {r ∈ R : c ≥ p_rt + p(A, r)}
25:    if Q ≠ ∅ then
26:      A := A ∪ {t}
27:      AddnewAvailable(t)
28:      if Cost(A) < Cost(A*) then A* := A
29:      end if
30:      OnePackingSearch(i+1)
31:      RemoveNewAvailable(t)
32:      A := A \ {t}
33:    end if
34:  end for
35: end function
36: L := ∅, A* := ∅, A := ∅                    ▷ Init main routine
37: for each t ∈ T do
38:  K_t = |P_i|
39:  if K_t = 0 thenAppend t to L
40:  end if
41: end for
42: OnePackingSearch(1)
43: return A*
```

Algorithm 1 uses a list of candidate tasks L containing all of the tasks without unassigned precedence relations, and a recursive function OnePackingSearch to generate all feasible station assignments for the first station. The list initially contains all tasks without predecessors, and new tasks are added, or removed, as tasks are added, or removed, to the station assignment $\mathcal{A}$, see functions AddNewAvailable and RemoveNewAvailable. Whenever a new station assignment is constructed, after a call to AddNewAvailable, the best cost of the assignment and of its complement are calculated, function Cost, and the best assignment $\mathcal{A}^*$ is updated if improved. The algorithm returns the best station assignment $\mathcal{A}^*$ or $\emptyset$ if no feasible solution with two stations exists.

Table 2
Optimal resolution of the 2-station problem for the example problem. For each feasible assignment $\mathcal{P}$, its corresponding station load for the first and the stations are detailed. The best robot equipment is then reported including its station load and the sum of fixed and variable costs. Finally, the total cost is reported.

Station 1			Station 2			Total
$\mathcal{A}$	Load	Cost	$\bar{\mathcal{A}}$	Load	Cost	
{1}	2	14 (r2)	{2, 3, 4, 5, 6}	infeasible	–	–
{1, 2}	8	21 (r3)	{3, 4, 5, 6}	infeasible	–	–
{1, 2, 3}	8	30 (r1)	{4, 5, 6}	9	29(r3)	59
{1, 2, 4}	10	29 (r1)	{3, 5, 6}	9	28(r3)	57
{1, 3}	8	21 (r3)	{2, 4, 5, 6}	infeasible	–	–
{2}	4	16 (r2)	{1, 3, 4, 5, 6}	infeasible	–	–
{2, 4}	6	21 (r2)	{1, 3, 5, 6}	10	32 (r1)	53
{2, 4, 6}	9	26 (r2)	{1, 3, 5}	10	27 (r3)	53

In order to illustrate the method, consider the solution of the example described in Fig. 1 and Table 1. Through the use of the enumeration algorithm, we obtain $\mathcal{P}$. The complements of some of the station assignments are also feasible for the second station, and among them the optimal solution is station assignment $\mathcal{A}^* = \{2, 4\}$ for the first station which will be manned by a robot of type 2, and station assignment $\bar{\mathcal{A}}^* = \{1, 3, 5, 6\}$ for the second station manned by a robot of type 1. The resulting configuration has a total cost of 53 units. Table 2 details the alternative station assignments and their related costs.

4. Solution procedures

As shown above, the cRALBP is strongly NP-hard, even when the number of stations is limited to two. Thus, efficient and scalable exact algorithms are impossible, unless $P = NP$. Furthermore note that in the case of a single robot $R = \{r\}$ with fixed costs $c_r^f = 1$ and variable costs $c_{rt}^v = 0$ model (1)–(7) reduces to a model for the SALBP of type 1 whose objective is to minimize the number of stations. It is well known that even strengthened versions of such models can be solved with standard IP solvers only for small instances (Ritt and Costa, 2015).

In this section we propose a genetic algorithm with a local search procedure (also called a memetic algorithm in the literature) for solving the cRALBP heuristically. A genetic algorithm evolves a population of *individuals* to improve their *fitness*. Each individual is encoded by sequence of *genes* which can assume a fixed number of values (*alleles*). New individuals are created by recombining the gene sequences of two or more individuals in the population, and may suffer a mutation in some of their genes. We obtain a memetic algorithm if a local search procedure is applied to new individuals. To establish evolutionary pressure, the selection of individuals for recombination is proportional to their fitness, and the offspring will substitute some of the less fit individuals in the population, if any. From the point of view of combinatorial optimization, individuals are solutions and their fitness is their objective value (or its negation for minimization problems). Numerous variants of genetic algorithms exist, and we refer the reader to the original publication of Holland (1975) and the more recent introduction of Sastry et al. (2014) for more details.

The algorithm proposed here is a steady-state, elitist genetic algorithm with a two-point crossover operator, and a problem-specific mutation operator. The overall algorithmic scheme is shown in Algorithm 2.

Solutions are represented by a linear order of the tasks of the cRALBP. Their fitness is determined by finding the minimal-cost assignment of robots and tasks to stations by the dynamic pro-

Algorithm 2 Memetic algorithm for the cRALBP.

Input: An instance of the cRALBP.
Output: An assignment of robots and tasks to stations.
1: generate an initial population P
2: **repeat**
3: select an individual o from P by a k-tournament
4: **with** probability q_x **do**
5: select a second, random individual p from P
6: $o := recombine(o, p)$
7: **end**
8: **with** probability q_m **do**
9: $o := mutate(o)$
10: **end**
11: if o does not change, continue with next loop iteration
12: compute the optimal station assignment for o by dynamic
13: programming
14: $o := localsearch(o)$
15: let $f(o)$ be o's fitness
16: **if** $f(o) > \min_{p\in P} f(p)$ and $\forall p \in P : f(p) \neq f(o)$ **then**
17: $P := P \setminus \{\text{argmin}_{p\in P} f(p)\} \cup \{o\}$
18: **end if**
19: **until** some stopping criterion is satisfied
20: **return** the fittest individual $\text{argmax}_{p\in P} f(p)$

gram, expression (21), (line 12 in Algorithm 2). For an optimal cost C the fitness is $M - C$ for some large constant M (e.g., $M = \sum_{r\in R, t\in T} |T| c^f_r + c^v_{rt}$). Please note that this fitness function definition was introduced in order to represent the objective as fitness maximization, as it is usual in the genetic algorithm literature.

The initial population P is created by generating $|P|$ random linear orders, and each random linear order is obtained by a topological sort choosing at each step one of the minimal elements uniformly at random. Since the algorithm operates in steady-state, each iteration of the main loop (lines 2–28 in Algorithm 2) generates a new individual, which replaces the worst individual in the population, if its fitness is better (lines 15–17 from Algorithm 2), and if its solution value is not already present in the population. The second criterion was introduced to maintain the diversity of the population. The algorithm terminates when some stopping criterion is satisfied (in our case the total running time), and returns the fittest individual found.

To generate a new individual, Algorithm 2 first selects in line 3 an existing individual o from the population using a k-tournament, i.e., the fittest of k randomly selected individuals is chosen (Miller and Goldberg, 1995). A k-tournament is equivalent to a rank-based selection, where the rth unfittest individual is selected with a probability proportional to $\Theta(r^{k-1})$. Next, with a crossover probability q_x individual o is recombined with a randomly selected individual from the population (lines 4–7 in Algorithm 2). Then, with mutation probability q_m the resulting individual suffers an additional mutation (lines 8–10 in Algorithm 2). Finally, a local search is applied (line 13 in Algorithm 2).

The local search procedure is shown in Algorithm 3. It repeatedly solves two-station subproblems to optimality, by systematically enumerating all feasible prefixes for the first station according to equation (22) and Algorithm 1, until no further improvement is possible.

To recombine two solutions we apply a two-point crossover that respects the precedence relations among the tasks. For a linear order $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ let $T(\pi) = \cup_{i\in[n]}\{\pi_i\}$ be the corresponding set of tasks, and let $\pi_{a:b} = (\pi_a, \ldots, \pi_{b-1})$ be the subsequence from positions a to $b-1$. Now suppose we have two linear orders π and ρ which respect the precedence constraints. During crossover a random non-empty segment $\pi_{a:\,b}$ of π is chosen. This

Algorithm 3 Local search procedure for the cRALBP.

Input: An assignment of robots and tasks to stations.
Output: An improved assignment.
1: **repeat**
2: **for** every pair of consecutive stations $i, i+1$, $i = 1, \ldots, |S| - 1$ **do**
3: solve the two-station subproblem $i, i+1$ optimally
4: **end for**
5: **until** a local minimum has been found
6: **return** the improved assignment

splits π into $(\pi_{1:\,a}, \pi_{a:\,b}, \pi_{b:\,n})$. Then, segment $\pi_{a:\,b}$ is substituted by a segment of the same length and containing the same tasks, but using the order given by ρ. In other words, the recombined solution is $(\pi_{1:\,a}, \tau, \pi_{b:\,n})$ where $\tau = (\tau_1, \ldots, \tau_{b-a})$, $T(\tau) = T(\pi_{a:b})$, and $\rho^{-1}(\tau_i) < \rho^{-1}(\tau_j)$ for all $i < j$, $i, j \in [b-a]$. The resulting solution is again a linear order respecting the precedence constraints.

Finally, to mutate a solution given by a linear order π, we first select randomly a non-empty, non-complete prefix $\pi_{1:\,k}$ of π (i.e., $2 \leq k \leq n$). Then all tasks whose predecessors are in $T(\pi_{1:\,k})$ are candidates for position k, one of them being π_k. If there is more than one candidate, we choose a random one, excluding π_k, otherwise π_k is chosen. Note that in the latter case π_k must be a bottleneck task (i.e., each task either precedes or succeeds π_k), and the mutation does not change the linear order. After choosing a candidate, the solution is completed with the remaining tasks in the order given by π.

5. Computational results

In this section we empirically evaluate the performance of the proposed memetic algorithm. Since the problem is new, there is no baseline from the literature to compare to. For this reason, we study four different algorithms using the components proposed here:

- The simplest is a random search (RS), which repeatedly generates a random linear order, and then applies the dynamic program, expression (21), to find the optimal assignment of the tasks and robots to the stations.
- The second algorithm is a multi-start local search (MLS). It extends RS by applying the local search of Algorithm 3 to each solution.
- The third algorithm is the genetic algorithm (GA) obtained from the memetic algorithm (Algorithm 2) by disabling the local search in line 13.
- The fourth algorithm is the full memetic algorithm (MA) proposed in the previous section.

This enables us to study the contribution of the local search and the evolutionary aspect of the MA separately. Moreover, we also compare the solution quality of the four methods to the lower bound obtained from model (16)–(18).

5.1. Test instances

The algorithms have been tested on two sets of instances. The first set uses the 32 instances proposed by Gao et al. (2009) for the robotic ALBP shown in Table 3. For each of these base instances we have generated six instances with different cost structures as follows. A cost structure is defined by: (1) the correlation ρ between the processing times and the variable costs of the tasks; and (2) the ratio σ of fixed robot costs to variable operation costs.

For given parameter ρ and task processing times p_{rt}, we first generate variable costs c^v_{rt} such that the Pearson correlation between p_{rt} and c^v_{rt}, seen as vectors of size $|T||R|$, is equal to ρ. Since

Table 3
Base instances as defined by Gao et al. (2009) of the first instance set. The table shows the number of tasks $|T|$, the number robots $|R|$, and the order strength OS.

Name	$\|T\|$	OS	$\|R\|$
Arcus 2	111	0.40	9, 13, 17, 22
Barthold 2	148	0.26	10, 14, 21, 29
Gunther	35	0.59	4, 5, 7, 12
Hahn	53	0.84	5, 7, 10, 14
Lutz 3	89	0.78	8, 12, 16, 21
Rosenberg	25	0.72	3, 4, 6, 9
Scholl	297	0.58	19, 29, 38, 50
Tonge	70	0.59	7, 10, 14, 19

the Pearson correlation is invariant under linear transformations, we normalize the values of the variable costs arbitrarily to the range [1,1000]. Note that the absolute value of the costs does not matter, and this normalization simply establishes a cost unit (see Otto et al. (2013), for a similar reasoning applied to the processing times of the tasks).

Next, we define the fixed costs of the robots as $c_r^f = \sigma \overline{c^v}_r \overline{n}$, where $\overline{c^v}_r = \sum_{t \in T} c_{rt}^v / |T|$ is the mean variable task cost of robot r, and $\overline{n} = c/\overline{p}$ is the mean number of tasks per station, for a mean processing time of $\overline{p} = \sum_{t \in T} \sum_{r \in R} p_{rt} / (|T|\,|R|)$.

In this way, we can freely adjust the correlation between time and cost, as well as the fixed-to-variable cost ratio. This experimental design allows us (1) to investigate whether structured, high-correlation instances are easier than unstructured, no correlation, instances; (2) to verify the behavior of the algorithm under different conditions; and (3) to test the impact of different cost structures on the proposed lower bound and solution methods.

For the experiments we have chosen $\rho \in \{0, 0.8\}$ (i.e., no correlation or a high time-cost correlation), and $\sigma \in \{1/3, 1, 3\}$ (i.e., a scenario dominated by the variable cost, a balanced scenario, and a scenario dominated by the fixed cost). All six combinations for the 32 base instances yield 192 test instances. In the experiments we have fixed the cycle time of these instances to $c = 300$.

The second set is based on a recently proposed set for the SALBP (Otto et al., 2013). These instances have been constructed with four experimental factors: the number of tasks, the graph structure, the order strength of the precedence graph, and the task time distribution.

We have selected the instances with a medium ($|T| = 50$) and a large ($|T| = 100$) number of tasks. The graph structure has three levels: chain (CH) graphs with at least 40% of chain tasks (i.e., tasks with a single predecessor and successor), bottleneck (BN) graphs containing bottleneck tasks with degree at least 8, and mixed (MIXED) graphs containing both chain and bottleneck tasks. The order strength (the ratio of the number of precedence relations to the maximum number of precedence relations) is 0.20 or 0.60, and for mixed graphs also 0.90 is allowed. There are three factors for the task time distribution: peak at the bottom, with task times normally distributed with mean $c/10$, peak in the middle, with task times normally distributed with mean $c/2$, and bimodal, a combination of the previous two distributions, where the distribution at $c/10$ has a smaller variance. This leads to 42 combinations of factor levels, and for each combination we select 5 out of the 25 random replications in Otto et al. (2013).

These 210 SALBP instances are extended to the RALBP as follows. We add two experimental factors, the number of robots, and the task time variation (see Blum and Miralles, 2011). The number of robots has two levels, $|T|/4$ (high number of robots) and $|T|/7$ (low number of robots). For each robot the time of task t is chosen randomly in $[p_t, \tau\, p_t]$ for a processing time of p_t in the corresponding SALBP instance. The task time variability τ has levels 2 (small variability) and 5 (high variability). For these instances the cycle time has been fixed to 1000τ, since the cycle time of the SALBP instances is 1000. This yields 840 RALBP instances. When combined with the six cost structures, as described above, we obtain 5040 instances in total. Table 4 summarizes all experimental factors and their levels.

Table 4
Experimental factors and levels of the second instance set.

Factor	Description	Levels
$\|T\|$	Number of tasks	50, 100
Gr.	Graph structure	Chain graph (CH), bottleneck graph (BN), mixed graph (MIXED)
OS	Order strength	0.2, 0.6, 0.9
TTD	Task time distribution	Peak at the bottom (PB), Peak in the middle (PM), bimodal (BM)
$\|R\|$	Number of robots	$\|T\|/7$, $\|T\|/4$
τ	Task time variability	2, 5
ρ	Time-cost correlation	0.0, 0.8
σ	Fixed-to-variable cost ratio	0.33, 0.00, 3.00

Table 5
Parameters of the GA and the MA and their initial ranges for irace (López-Ibáñez et al., 2016).

Parameter	Description	Initial range	Best	
			GA	MA
$\|P\|$	Population size	[1,1000]	867	287
$k \in \mathbb{N}$	Tournament size	[2,5]	4	4
$q_x \in \mathbb{R}$	Crossover probability	[0,1]	0.8	0.8
$q_m \in \mathbb{R}$	Mutation probability	[0,1]	0.8	0.9

Table 6
Average number of generated columns and computation time (in seconds) for the lower bounds as a function of the number of tasks, order strength and task time distribution.

$\|T\|$	OS	PB		PM		BM	
		# cols	t (s)	# cols	t (s)	# cols	t (s)
50	0.2	636.3	41.4	253.0	16.2	519.4	28.1
50	0.6	531.8	258.1	213.9	29.3	414.7	114.0
50	0.9	373.4	693.5	178.8	61.2	309.8	301.9
100	0.2	1,653.4	253.4	633.7	45.5	1,389.4	125.6
100	0.6	1,395.2	2,854.1	547.2	205.9	1,112.7	1,254.1
100	0.9	1,080.3	3,769.6	436.8	606.7	833.0	1,859.5

5.2. Parameter setting and experimental methods

Table 5 summarizes the parameters used for the GA and the MA (RS and MLS have no parameters). The table shows for each parameter an initial range that has been chosen during preliminary experiments. For the final calibration, we have applied an iterated Friedman race (as implemented in the R irace package, version 2.3.1807, López-Ibáñez et al., 2016). The budget was set to 1000 evaluations, with random sampling from the first set of instances for testing, and a time limit of 180 s. The optimal parameter settings obtained from racing are shown in columns “GA” and “MA” of Table 5. We can see that the optimal settings are very similar, preferring a high evolutionary pressure using a 4-tournament, and applying crossover and mutation with high probability. The different population sizes come from the local search which is costly and only applied in the MA. This leads to less fitness function evaluations per time, and a smaller population increases the probability of each individual getting selected.

All algorithms have been implemented in C++ and compiled with gcc 5.3.1 and maximum optimization. The experiments with

Table 7
Performance of randomized sampling (RS), multi-start local search (MLS), genetic algorithm (GA), and memetic algorithm (MA) on the first instance set. For each cost structure (ρ, σ), and for each algorithm we report the average relative deviation of the best replication, the average relative deviation over all replications, and the average time (in seconds) to find the best value.

ρ	σ	R.d.	RS			MLS			GA			MA		
			Min.	Avg.	t_{best}	Min.	Avg.	t_{best}	Min.	Avg.	t_{best}	Min.	Avg.	t_{best}
0.0	0.33	4.2	10.5	11.0	264.9	5.9	6.2	208.8	6.2	6.8	119.1	4.2	4.5	236.9
0.0	1.00	3.5	8.3	8.6	236.3	4.7	4.9	159.2	5.2	5.7	114.3	3.5	3.7	229.1
0.0	3.00	4.6	7.5	7.9	228.2	5.6	5.9	185.1	5.4	6.0	119.2	4.6	5.0	252.7
0.8	0.33	3.6	9.8	10.2	227.7	5.2	5.4	187.0	5.4	6.1	105.1	3.6	4.0	227.6
0.8	1.00	4.8	9.5	9.8	243.5	6.1	6.4	191.6	6.1	6.8	121.4	4.8	5.1	229.7
0.8	3.00	6.8	9.5	9.9	217.3	8.3	8.6	194.7	7.3	7.9	125.1	7.1	7.4	225.0

Table 8
Average utilization and percentage of fixed costs for different correlations of processing time and variable cost, and different fixed-to-variable cost factors for the instances found by MA.

Prec./ σ =	$\rho = 0.0$			$\rho = 0.8$		
	0.33	1.00	3.00	0.33	1.00	3.00
Arcus 2	92.5	97.1	97.7	90.4	94.0	93.8
Barthold 2	91.9	96.3	97.1	87.4	92.4	93.8
Gunther	81.2	96.7	97.4	83.5	89.9	89.2
Hahn	92.2	96.5	96.4	85.4	91.1	93.6
Lutz 3	89.7	95.1	97.6	87.3	92.8	94.7
Rosenberg	82.6	92.9	95.2	82.9	86.2	92.1
Scholl	90.1	95.8	96.9	86.4	92.7	93.6
Tonge	93.1	96.8	97.2	86.9	94.5	95.4
Averages	89.1	95.9	97.0	86.3	91.7	93.3
c^f(%)	31.9	49.9	72.1	27.0	49.4	73.2

Table 10
Average relative deviation from the lower bound as a function of graph structure, order strength, task time distribution, and number of tasks.

Gr.	OS	$\|T\| = 50$			$\|T\| = 100$		
		BM	PB	PM	BM	PB	PM
BN	0.2	6.3	6.4	5.8	11.9	12.7	9.7
BN	0.6	4.0	5.5	3.0	6.4	7.4	5.9
CH	0.2	6.7	7.6	6.0	11.5	12.4	10.0
CH	0.6	3.7	4.0	4.2	5.9	5.9	6.3
MIXED	0.2	6.8	8.9	5.6	11.5	12.9	9.9
MIXED	0.6	3.6	4.8	3.7	6.0	6.2	6.2
MIXED	0.9	2.4	2.0	2.0	2.3	2.3	3.0

the first set of instances (derived from Gao et al., 2009) have been done on a PC with an 8-core AMD FX-8150 processor running at 1.4 GHz and with 32 GB of main memory. For the second set of instances (derived from Otto et al., 2013) the experiments were run on a PC with 8-core Intel Xeon X5550 processors running at 2.67 GHz and with 24 GB of main memory. In both cases the experiments have been run in parallel but each using only one core. The mathematical models for the lower bounds by column generation have been solved with CPLEX 12.7.

For all heuristic methods, we report average relative deviations over 10 replications of the experiments with different random seeds. The relative deviation of a solution value UB from a lower bound LB is defined as $(UB - LB)/LB$. The detailed experimental results are available upon request to the authors.

5.3. Results

We first discuss briefly the computation of the lower bounds based on column generation. Table 6 provides summary statistics for the number of generated columns and the time to solve the models, as a function of the three main factors, namely the number of tasks $|T|$, the order strength OS, and the task time distribution, for the 5040 instances of the second set. Time and number of generated columns depend strongly on the number of tasks, and increase from a task time distribution with a peak at the bottom, over a bimodal distribution to a peak in the middle. Both can be explained by the higher number of candidate columns that must be considered. The number of columns decreases with increasing order strength, since there are less candidate columns, but the solution time increases, since the models get harder to solve. The number of columns is never more than 2200 and all models can be solved within 3.5 h.

We next analyze the results for the dataset derived from Gao et al. (2009) and in a second step we focus on the instances created according to Otto et al. (2013).

5.3.1. Instances based on Gao et al. (2009)

Table 7 presents the summarized solution quality. For each combination oftime-cost correlation ρ and fixed-to-variable cost ratio σ, it shows the average relative deviation of the best known solutions from the lower lower bound ("R.d."), and for each of the four methods the minimum and average relative deviation from the lower bound ("Min.", "Avg.") as well as the average time (in seconds) to find the best solutions ("t_{best}").

Table 9
Performance of randomized sampling (RS), multi-start local search (MLS), genetic algorithm (GA), and memetic algorithm (MA) on the second instance set. For each cost structure (ρ, σ), and for each algorithm we report the average relative deviation of the best replication, the average relative deviation over all replications, and the average time (in seconds) to find the best value.

ρ	σ	R.d.	RS			MLS			GA			MA		
			Min.	Avg.	t_{best}	Min.	Avg.	t_{best}	Min.	Avg.	t_{best}	Min.	Avg.	t_{best}
0.0	0.33	2.6	8.0	9.1	279.8	3.9	4.8	234.7	3.7	4.9	141.9	2.6	3.5	241.6
0.0	1.00	3.1	7.8	9.2	284.7	4.5	5.7	238.0	4.1	5.7	152.0	3.1	4.5	254.6
0.0	3.00	5.5	9.3	11.6	277.4	7.9	10.7	238.5	6.1	9.2	156.9	5.9	9.1	266.7
0.8	0.33	2.9	8.4	10.6	283.6	4.4	5.8	229.3	4.0	5.7	132.2	3.0	4.1	228.6
0.8	1.00	4.1	9.4	11.7	280.2	6.4	8.5	228.8	5.2	7.5	139.0	4.3	6.4	248.4
0.8	3.00	6.5	10.8	13.9	279.1	9.9	13.5	232.6	7.1	11.0	140.6	7.2	11.0	254.1

Table 11
Average relative deviation from the lower bound as a function of the number of robots $|R|$, the task time variability τ, the correlation of processing times and variables costs ρ, and the fixed-to-variable cost ratio σ.

$\|R\|$	τ/ σ =	$\rho = 0.0$			$\rho = 0.8$		
		0.33	1.00	3.00	0.33	1.00	3.00
$\|T\|/4$	2	3.7	3.9	6.4	4.0	5.2	8.2
$\|T\|/4$	5	3.7	5.3	11.6	4.6	7.6	13.8
$\|T\|/7$	2	3.1	3.6	6.6	3.6	5.0	8.4
$\|T\|/7$	5	3.2	5.2	11.7	4.3	7.6	13.6

Looking at column "R.d." we can see that the best solutions found are between 3% to 7% from the lower bound. Since the relative deviation from the unknown optimal solution values are likely to be even smaller, the best solutions are optimal or near-optimal. Of the four methods, RS performs worst, as expected. MLS and GA improve significantly over RS, and find solutions of comparable cost. This shows that the local search, as well as the evolution of solutions are effective. The solutions of MA improve again over MLS and GA, so both components are also effective when combined. Note that the best solutions of MA are almost equal to the best solutions found as shown by the differences between column "R.d." and the columns associated to the memetic algorithm. For all methods the minimum and average performance over the replications are very close, thus they are robust. The time to find the best solutions is similar for all methods, except for the GA, and never exceeds half of the time limit of 10 min. Finally, there seems to be little dependence of (relative) solution quality on the cost structure, except for a high time-cost correlation $\rho = 0.8$ in the scenario where the fixed costs dominate, $\sigma = 3$, which leads to about 2% larger gaps.

We now turn to the structure of costs and average usage of the solutions provided by the MA. Table 8 shows the average station utilization in percent for the different precedence graphs and for varying time-cost correlation ρ and fixed-to-variable cost ratio σ. The utilization is defined as the station load divided by the cycle time. The last line additionally reports the average percentage of the fixed costs over the total costs.

We can see that the utilization increases with the fixed-to-variable cost ratio σ, and decreases with the time-cost correlation ρ. This is expected, since higher fixed costs will lead to less stations and thus higher station loads. Also, if ρ is high, costs can be reduced by using more robots which execute tasks faster, at the expense of utilization. The utilization is never lower than 85% except in some cases for the two smallest instances ("Rosenberg, "Gunther") and for high fixed costs never lower than 93%, with the same two exceptions. This shows that minimizing costs does not lead to a large number of extra stations, and that if space limitations are relevant, this can be considered by increasing the fixed costs (hence reducing the number of stations).

We finally look at the fixed to total costs percentages: for a cost factor σ a value of $1-(1+\sigma)^{-1}$ would be expected, i.e., 25%, 50%, and 75% for $\sigma = 0.33, 1.00, 3.00$, respectively. The actual percentages are close to these values: when investment costs are reduced compared to operation costs, we can afford to reduce variable costs at the expense of fixed costs, and the percentage increases slightly. Similarly when the fixed costs are higher, the percentage is slightly lower.

5.3.2. Instances based on Otto et al. (2013)

In this section we focus on the analysis of the results on the instances based on systematically generated SALBP instances as proposed by Otto et al. (2013), extended to RALBP and cRALBP instances as described in Section 5.1. Table 9 show the summarized information on the solution quality for these instances. The conclusions of the previous section on the relative performance of the methods continue to hold here, and thus we focus the following analysis on the results obtained by the memetic algorithm for different factors.

We first look at the graph structure, order strength, and task time distribution factors from the SALBP instances. Table 10 shows the dependence of the average relative deviation from the lower bound on the factor levels. We find that, besides the number of tasks, the order strength has the greatest influence on solution quality, and increasing order strength leads to smaller gaps. The dependency on the graph structure, on the other hand, is weak. There is further little difference between a bimodal task time distribution, and one with a peak at the bottom, whereas a peak in the middle leads to lower gaps. We conjecture that this behavior relates to the smaller number of feasible station assignments available to the Dantzig–Wolfe reformulation which leads to stronger lower bounds, rather than to a deterioration of the memetic algorithm.

We now turn to the RALBP and cRALBP factors. Table 11 shows the dependence of the average relative deviation from the lower bound on the number of robots, the task time variability, the time-cost correlation, and the fixed-to-variable cost ratio. In this case all factors, except the number of robots, have significant influence on solution quality, with increasing gaps for higher time-cost correlation ρ and fixed-to-variable cost ratio σ, and higher task time variability, in particular for high ρ and σ.

These observations have been confirmed by an ANOVA on log-transformed average relative deviations (after checking normality and independence of residuals, and homoskedascity) and are significant with $p < 0.001$.

Finally, Tables 12 and 13 provide average absolute values for the best solutions according to the different instance factors. While these average absolute values do not provide any significant information on the quality of the proposed methods, they can be used to give insights on the results of modifying initial conditions. For instance, the results show that there are significant gains to be made by selecting among robots specially suited for the tasks on hand (high time-cost correlation) than from selecting among gen-

Table 12
Average upper bound as a function of order strength, task time variability τ, correlation of processing times and variables costs ρ, and task time distribution.

OS	τ	$\rho = 0.0$			$\rho = 0.8$		
		PB	PM	BM	PB	PM	BM
0.2	2	113,883.8	118,593.2	111,177.7	83,913.8	90,517.1	86,706.2
0.2	5	104,530.5	112,392.8	97,581.6	70,949.9	77,670.7	70,627.9
0.6	2	115,045.9	119,405.0	112,124.3	87,255.2	89,225.8	88,436.9
0.6	5	105,738.2	113,565.4	98,783.0	74,254.1	77,672.6	73,702.1
0.9	2	118,180.8	123,237.1	113,657.5	90,828.9	92,251.4	89,150.2
0.9	5	109,893.2	116,672.4	102,209.4	77,136.9	83,046.3	79,724.4

Table 13
Average upper bound as a function of the number of robots $|R|$, the task time variability τ, the correlation of processing times and variables costs ρ, and the fixed-to-variable cost ratio σ.

$\|R\|$	τ / σ =	$\rho = 0.0$			$\rho = 0.8$		
		0.33	1.00	3.00	0.33	1.00	3.00
$\|T\|/4$	2	44,510.4	71,352.3	146,262.4	33,944.3	54,138.2	112,575.8
$\|T\|/4$	5	44,572.6	67,172.2	129,700.8	31,110.6	47,255.7	94,131.0
$\|T\|/7$	2	45,201.7	72,182.4	148,490.7	34,790.2	55,460.8	115,695.2
$\|T\|/7$	5	45,216.2	68,310.6	132,988.5	31,425.0	48,133.0	96,785.6

eral robots (low time-cost correlation), see Table 12, while the expected improvements for selecting among additional numbers of robots are small, as shown by comparing the rows associated to $|T|/7$ and $|T|/4$ in Table 13.

6. Conclusions

In this work we have studied a cost-oriented version of an assembly line balancing problem with process selection features, which we denote as the cRALBP. The problem is modeled as a binary program in which both the assignment of tasks and robots to the stations is performed jointly, while minimizing fixed installation, and variable operational costs. We have shown that the optimal cost of the linear relaxation of the formulation is easily computable without relying to a linear programming solver, hence providing a cheap-to-calculate lower bound. An alternative Dantzig–Wolfe based formulation, which provides a tighter but more expensive-to-calculate lower bound, is also proposed.

In order to solve large instances, a steady-state, elitist memetic algorithm, a combination of a genetic algorithm with other optimization methods, is proposed. The memetic algorithm makes use of the results of two special cases of the cRALBP: (1) the case in which the tasks can only be performed in a given order, i.e., the precedence constraints generate a unique linear order of the tasks, and (2) the case in which the solution has only two stations.

For the first case, linearly ordered tasks, a dynamic programming approach is designed to optimally assign tasks and robots to the stations in polynomial time to the size of the problem. Consequently, the proposed memetic algorithm encodes individuals as ordered lists of tasks, which are then transformed into a feasible solution by the resolution of the dynamic programming recurrence. The use of the dynamic program simplifies the description of an individual and allows the memetic algorithm to focus the search on finding good sequences of tasks, rather than to explore both, task and robot assignment.

For the second case, the two-station subproblem, we adapt the classical single-station step of the Hoffmann heuristic (Fleszar and Hindi, 2003; Hoffmann, 1963) to optimally solve the joint assignment of tasks and robots for two consecutive stations. The method provides the basis for an effective local search procedure for the problem. As a secondary result, the theoretical analysis of the two-station case has led us to novel complexity results on the strong NP-hardness of some assembly line balancing problems. Traditional complexity results for line balancing come from the relationship with the Bin Packing Problem, in which precedence relations do not exist. In this case, the complexity of the two-station problem stems from the precedence relations among the tasks, which is (to the best of our knowledge) a novel result within the literature.

In order to test the applicability of each of the proposed algorithmic components, an experimental analysis has been conducted. The results show that the memetic algorithm is effective and robust, and both its evolutionary component as well as the local search contributes to the quality of the method. The total cost found is close to optimal, with about 6.5% above the best lower bound obtained by the Dantzig–Wolfe reformulation and it takes an average of about 4 min to find the best solution. The utilization rate of the robots is reasonably high, given that the primary objective is to minimize cost, with averages ranging from about 86% to 97%. An analysis of the different instance factors led us to conclude that the structure of the precedence graph has little influence on the performance of the memetic algorithm. Moreover, it seems to be robust and perform well, with little dependence on the cost structure, and the percentage of fixed costs is close to its expected value, while the lower bounds are tighter for those instances with lower fixed-to-variable cost ratio.

In addition to the modeling and algorithmic contributions described above, the results also provide some hints on the interplay among the main characteristics of the instances, as defined by the experimental factors. Specifically,

1. solutions show higher utilization rates when the fixed to variable costs ratio is high, while instances with low fixed to variable cost ratio have lower utilization rates. This results follows the common sense that larger investment costs impose higher utilization rates.
2. increasing the number of available processing options (robots) does not need to have a strong impact on the final production costs. We conjecture that the potential savings of introducing additional processing alternatives diminishes when the number of options is relatively large.
3. potential savings can be obtained by applying a re-engineering process on the production process in order to allow the division of tasks into smaller units (and thus, leading an instance from a peak at the middle to a peak at the bottom task time distribution), while changes associated to reducing assembly dependencies, that is smaller order strength, are not specially significant.

As a concluding remark, we want to highlight that current state-of-the-art solution procedures for assembly line balancing are able to effectively solve multifaceted problems integrating characteristics of real-life assembly systems. The literature contains multiple theoretical results that can be used as building blocks to tackle these problems. Consequently, the focus of future research should be on properly reflecting the conditions and the requirements of real life environments, as well as in the development of building blocks which can be utilized in different problems. Further exploring these building blocks in order to obtain efficient exact solution methods for the cRALBP, as well as the development of more detailed formulations are promising topics for future research. Among these more detailed formulations we would like to highlight the need to account for uncertainty within the data. As line balancing models incorporate additional tactical and strategic issues within their scope, these models need to account for the inherent uncertain nature of some of the parameters of the model, like the production throughput.

Acknowledgements

This research has been funded by the research grant “Heterogeneous assembly line balancing problems with process selection

features" number 1150306, from the Fondo Nacional de Desarrollo Científico y Tecnológico of the Ministry of Education of Chile. This research was also partially supported by the Proyecto Basal USA 1555 - Vridei 061717VP_PUBLIC Universidad de Santiago de Chile and the supercomputing infrastructure of the NLHPC (ECM-02) on which most of tests were conducted.

References

Amen, M., 2000. An exact method for cost-oriented assembly line balancing. Int. J. Prod. Econ. 64 (1–3), 187–195. doi:10.1016/S0925-5273(99)00057-2.

Amen, M., 2006. Cost-oriented assembly line balancing: model formulations, solution difficulty, upper and lower bounds. Eur. J. Oper. Res. 168 (3), 747–770. doi:10.1016/j.ejor.2004.07.026.

Araújo, F., Costa, A., Miralles, C., 2012. Two extensions for the alwabp: parallel stations and collaborative approach. Int. J. Prod. Econ. 140 (1), 483–495. doi:10.1016/j.ijpe.2012.06.032.

Bautista, J., Pereira, J., 2009. A dynamic programming based heuristic for the assembly line balancing problem. Eur. J. Oper. Res. 194 (3), 787–794. doi:10.1016/j.ejor.2008.01.016.

Bautista, J., Pereira, J., 2011. Procedures for the time and space constrained assembly line balancing problem. Eur. J. Oper. Res. 212 (3), 473–481. doi:10.1016/j.ejor.2011.01.052.

Baybards, I., 1986. A survey of exact algorithms for the simple assembly line balancing problem. Manage. Sci. 32 (8), 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. Eur. J. Oper. Res. 168 (3), 694–715. doi:10.1016/j.ejor.2004.07.023.

Blum, C., Miralles, C., 2011. On solving the assembly line worker assignment and balancing problem via beam search. Comput. Oper. Res. 38 (1), 328–339. doi:10.1016/j.cor.2010.05.008.

Borba, L., Ritt, M., 2014. Exact and heuristic methods for the assembly line worker assignment and balancing problem. Comput. Oper. Res. 45, 87–96. doi:10.1016/j.cor.2013.12.002.

Borba, L., Ritt, M., Miralles, C., 2018. Exact and heuristic methods for solving the robotic assembly line balancing problem. Eur. J. Oper. Res. forthcoming. doi:10.1016/j.ejor.2018.03.011.

Boysen, N., Fliedner, M., Scholl, A., 2007. A classification of assembly line balancing problems. Eur. J. Oper .Res. 183 (2), 674–693. doi:10.1016/j.ejor.2006.10.010.

Bukchin, J., Tzur, M., 2000. Design of flexible assembly line to minimize equipment cost. IIE Trans. 32 (7), 585–598. doi:10.1023/A:1007646714909.

Çil, Z., Mete, S., Ağpak, K., 2017. Analysis of the type ii robotic mixed-model assembly line balancing problem. Eng. Optim. 49 (6), 990–1009. doi:10.1080/0305215X.2016.1230208.

Corominas, A., Ferrer, L., Pastor, R., 2011. Assembly line balancing: general resource-constrained case. Int. J. Prod. Res. 49 (12), 3527–3542. doi:10.1080/00207543.2010.481294.

Daoud, S., Chehade, H., Yalaoui, F., Amodeo, L., 2014. Solving a robotic assembly line balancing problem using efficient hybrid methods. J. Heuristics 20 (3), 235–259. doi:10.1007/s10732-014-9239-0.

Dolgui, A., Battaïa, O., 2013. A taxonomy of line balancing problems and their solution approaches. Int. J. Prod. Econ. 142 (2), 259–277. doi:10.1016/j.ijpe.2012.10.020.

Fleszar, K., Hindi, K., 2003. An enumerative heuristic and reduction methods for the assembly line balancing problem. Eur. J. Oper. Res. 145 (3), 606–620. doi:10.1016/S0377-2217(02)00204-7.

Gadidov, R., Wilhelm, W., 2000. A cutting plane approach for the single-product assembly system design problem. Int. J. Prod. Res. 38 (8), 1731–1754.

Gao, J., Sun, L., Wang, L., Gen, M., 2009. An efficient approach for type II robotic assembly line balancing problems. Comput. Ind. Eng. 56 (3), 1065–1080. doi:10.1016/j.cie.2008.09.027.

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA.

Hazir, Ö., Delorme, X., Dolgui, A., 2015. A review of cost and profit oriented line design and balancing problems and solution approaches. Annu. Rev. Control 40, 14–24. doi:10.1016/j.arcontrol.2015.09.001.

Hoffmann, T., 1963. Assembly line balancing with a precedence matrix. Manage. Sci. 9 (4), 551–562. doi:10.1287/mnsc.9.4.551.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press.

Johnson, D.S., Niemi, K.A., 1983. On knapsacks, partitions, and a new dynamic programming technique for trees. Math. Oper. Res. 8 (1), 1–14. doi:10.1287/moor.8.1.1.

Levitin, G., Rubinovitz, J., Shnits, B., 2006. A genetic algorithm for robotic assembly line balancing. Eur. J. Oper. Res. 168 (3), 811–825. doi:10.1016/j.ejor.2004.07.030.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Stützle, T., Birattari, M., 2016. The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. 3, 43–58. doi:10.1016/j.orp.2016.09.002.

Miller, B.L., Goldberg, D.E., 1995. Genetic algorithms, tournament selection, and the effects of noise. Complex Syst. 9, 193–212.

Miralles, C., García-Sabater, J.P., Andrés, C., Cardos, M., 2007. Advantages of assembly lines in sheltered work centres for disabled. a case study. Int. J. Prod. Econ. 110 (1), 187–197. doi:10.1016/j.ijpe.2007.02.023.

Moreira, M., Cordeau, J.-F., Costa, A., Laporte, G., 2015. Robust assembly line balancing with heterogeneous workers. Comput. Ind. Eng. 88, 254–263. doi:10.1016/j.cie.2015.07.004.

Moreira, M., Miralles, C., Costa, A., 2015. Model and heuristics for the assembly line worker integration and balancing problem. Comput. Ind. Eng. 54, 64–73. doi:10.1016/j.cor.2014.08.021.

Moreira, M., Pastor, R., Costa, A., Miralles, C., 2017. The multi-objective assembly line worker integration and balancing problem of type-2. Comput. Ind. Eng. 82, 114–125. doi:10.1016/j.cor.2017.01.003.

Moreira, M., Ritt, M., Costa, A., Chaves, A., 2012. Simple heuristics for the assembly line worker assignment and balancing problem. J. Heuristics 18 (3), 505–524. doi:10.1007/s10732-012-9195-5.

Morrison, D.R., Sewell, E.C., Jacobson, S.H., 2014. An application of the branch, bound, and remember algorithm to a new simple assembly line balancing dataset. Eur. J. Oper. Res. 236 (2), 403–409. doi:10.1016/j.ejor.2013.11.033.

Mutlu, O., Polat, O., Supciller, A., 2013. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-ii. Comput. Oper. Res. 40 (1), 418–426. doi:10.1016/j.cor.2012.07.010.

Nilakantan, J., Nielsen, I., Ponnambalam, S., Venkataramanaiah, S., 2017. Differential evolution algorithm for solving ralb problem using cost- and time-based models. Int. J. Adv. Manuf. Technol. 89 (1–4), 311–332. doi:10.1007/s00170-016-9086-2.

Nilakantan, J., Ponnambalam, S., 2016. Robotic u-shaped assembly line balancing using particle swarm optimization. Eng. Optim. 48 (2), 231–252. doi:10.1080/0305215X.2014.998664.

Otto, A., Otto, C., Scholl, A., 2013. Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. Eur. J. Oper. Res. 228 (1), 33–45. doi:10.1016/j.ejor.2012.12.029.

Pape, T., 2015. Heuristics and lower bounds for the simple assembly line balancing problem type 1: overview, computational tests and improvements. Eur. J. Oper. Res. 240 (1), 32–42. doi:10.1016/j.ejor.2014.06.023.

Peeters, M., Degraeve, Z., 2006. An linear programming based lower bound for the simple assembly line balancing problem. Eur. J. Oper. Res. 168 (3), 716–731. doi:10.1016/j.ejor.2004.07.024.

Pekin, N., Azizoglu, M., 2008. Bi criteria flexible assembly line design problem with equipment decisions. Int. J. Prod. Res. 46 (22), 6323–6343. doi:10.1080/00207540701441988.

Pereira, J., 2015. Empirical evaluation of lower bounding methods for the simple assembly line balancing problem. Int. J. Prod. Res. 53 (11), 3327–3340. doi:10.1080/00207543.2014.980014.

Polat, O., Kalayci, C., Mutlu, O., Gupta, S., 2016. A two-phase variable neighbourhood search algorithm for assembly line worker assignment and balancing problem type-ii: an industrial case study. Int. J. Prod. Res. 54 (3), 722–741. doi:10.1080/00207543.2015.1055344.

Rekiek, B., Dolgui, A., Delchambre, A., Bratcu, A., 2002. State of art of optimization methods for assembly line design. Annu Rev Control 26 II, 163–174. doi:10.1016/S1367-5788(02)00027-5.

Ritt, M., Costa, A., Miralles, C., 2016. The assembly line worker assignment and balancing problem with stochastic worker availability. Int. J. Prod. Res. 54 (3), 907–922. doi:10.1080/00207543.2015.1108534.

Ritt, M., Costa, A.M., 2015. Improved integer programming models for simple assembly line balancing and related problems. International Transactions of Operations Research doi:10.1111/itor.12206.

Rubinovitz, J., Bukchin, J., Lenz, E., 1993. Ralb – a heuristic algorithm for design and balancing of robotic assembly lines. CIRP Annals - Manufacturing Technology 42 (1), 497–500. doi:10.1016/S0007-8506(07)62494-9.

Sastry, K., Goldberg, D.E., Kendall, G., 2014. Genetic Algorithms. In: Burke, E.K., Kendall, G. (Eds.), Search Methodologies. Springer, pp. 97–125.

Scholl, A., Becker, C., 2006. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur. J. Oper. Res. 168 (3), 666–693. doi:10.1016/j.ejor.2004.07.022.

Sewell, E., Jacobson, S., 2012. A branch, bound, and remember algorithm for the simple assembly line balancing problem. INFORMS J. Comput. 24 (3), 433–442. doi:10.1287/ijoc.1110.0462.

Sternatz, J., 2014. Enhanced multi-Hoffmann heuristic for efficiently solving real-world assembly line balancing problems in automotive industry. Eur. J. Oper. Res. 235 (3), 740–754. doi:10.1016/j.ejor.2013.11.005.

Tasan, S., Tunali, S., 2008. A review of the current applications of genetic algorithms in assembly line balancing. J. Intell. Manuf. 19 (1), 49–69. doi:10.1007/s10845-007-0045-5.

Triki, H., Mellouli, A., Masmoudi, F., 2017. A multi-objective genetic algorithm for assembly line resource assignment and balancing problem of type 2 (alrabp-2). J. Intell. Manuf. 28 (2), 371–385. doi:10.1007/s10845-014-0984-6.

Valério De Carvalho, J., 2002. Lp models for bin packing and cutting stock problems. Eur. J. Oper. Res. 141 (2), 253–273. doi:10.1016/S0377-2217(02)00124-8.

Fernandez de la Vega, W., Lueker, G.S., 1981. Bin packing can be solved within 1 + ϵ in linear time. Combinatorica 1 (4), 349–355. doi:10.1007/BF02579456.

Vilà, M., Pereira, J., 2014. A branch-and-bound algorithm for assembly line worker assignment and balancing problems. Comput. Ind. Eng. 44, 105–114. doi:10.1016/j.cor.2013.10.016.

Yoosefelahi, A., Aminnayeri, M., Mosadegh, H., Ardakani, H., 2012. Type ii robotic assembly line balancing problem: an evolution strategies algorithm for a multi-objective model. J. Manuf. Syst. 31 (2), 139–151. doi:10.1016/j.jmsy.2011.10.002.

Zacharia, P., Nearchou, A., 2016. A population-based algorithm for the bi-objective assembly line worker assignment and balancing problem. Eng. Appl. Artif. Intell. 49, 1–9. doi:10.1016/j.engappai.2015.11.007.