Hybrid metaheuristics for the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2)

Gema Calleja , Albert Corominas , Alberto García-Villoria , Rafael Pastor

PII: S0377-2217(15)00951-0
DOI: 10.1016/j.ejor.2015.10.025
Reference: EOR 13309

To appear in: *European Journal of Operational Research*

Received date: 29 September 2014
Revised date: 13 October 2015
Accepted date: 15 October 2015

Highlights

- An assembly line with workpieces larger than the width of the stations is tackled.
- Only restricted portions of the workpieces can be accessed from any workstation.
- Three hybrids of metaheuristics and mathematical programming are proposed.
- A novel procedure hybridizing tabu search and corridor method is provided.

Hybrid metaheuristics for the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2)

Gema Calleja, Albert Corominas, Alberto García-Villoria, Rafael Pastor

IOC Research Institute
Universitat Politècnica de Catalunya (UPC)
Av. Diagonal 647, 11th floor , 08028, Barcelona, Spain
{gema.calleja, albert.corominas, alberto.garcia-villoria, rafael.pastor}@upc.edu

Abstract

This paper addresses an assembly line balancing problem in which the length of the workpieces is larger than the width of the workstations. The problem differs from traditional variants of assembly line balancing in the sense that only a portion of the workpiece, or portions of two consecutive workpieces, can be reached from any workstation. Consequently, at any stationary stage of the cycle, each workstation can only process a portion of the tasks, namely, those which are inside the area of a workpiece that is reachable from the workstation. The objective is to find a (cyclic) movement scheme of the workpieces along the line and a task assignment to stationary stages of the production process, while minimizing the cycle time. We propose three hybrid approaches of metaheuristics and mathematical programming - one based on simulated annealing and the other two based on tabu search, relying on different neighborhood definitions. The two former approaches make use of a classical neighborhood, obtained by applying local changes to a current solution. The latter approach, in contrast, draws ideas from the corridor method to define a *corridor* around the current solution, via the imposition of exogenous constraints on the solution space of the problem. An extensive computational experiment is carried out to test the performance of the proposed approaches, improving the best results published to date.

Keywords: *Assembly line balancing with accessibility windows, hybrid metaheuristics, simulated annealing, tabu search, corridor method*

1. Introduction

As global competition and technological change accelerates, manufacturers have become increasingly interested in optimizing their production and assembly systems. In this paper, we consider a special case of assembly system that widely arises in advanced automated environments, especially in the assembly of electronic components: the assembly line with accessibility windows. The line consists of a set of workstations sequentially arranged along a transport system, which must process a number of identical workpieces. Every workstation contains a feeder with several component types and is equipped with a robot arm, which performs tasks on the workpieces. Each workstation must process a specific set of tasks on each workpiece. The tasks correspond to pick-and-place actions; picking a component type from the feeder inside the workstation and placing it on a predefined position on the workpiece (see Fig. 1).

The workpieces are fed into the assembly line starting from a reference position *x* (see Fig. 2), and are moved in *forward steps*, according to a pattern called *movement scheme*. In every halt between two forward steps, the line stands motionless and the workstations perform tasks on the workpieces. Such a halt is called a *stationary stage*. The forward steps are cyclic: after *S* forward steps, there is an identical number of workpieces lying exactly at the same positions as in the start of the cycle. The length of each forward step must be a multiple of a distance Δ called *elementary step*, which depends on the technology of the line. After each cycle, a new workpiece enters the line. At the same time, a fully assembled workpiece leaves the line.

Fig. 2 illustrates an example of a cycle with three stationary stages (thus the fourth stationary stage is identical to the first stage). Each line is a snapshot representing the positions of the workpieces in the stationary stage. The initial position of the first workpiece in the beginning of the cycle is defined by the distance *x*. The arrows on each snapshot represent the forward steps. Note that, in this example, the lengths of the forward steps are different.

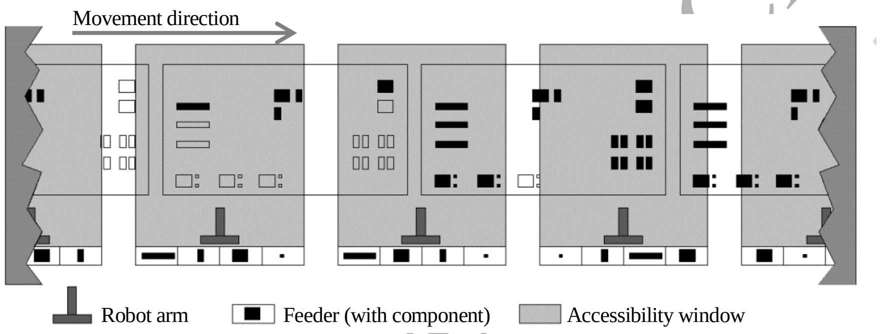

Figure 1. *An example of an assembly line with accessibility windows*

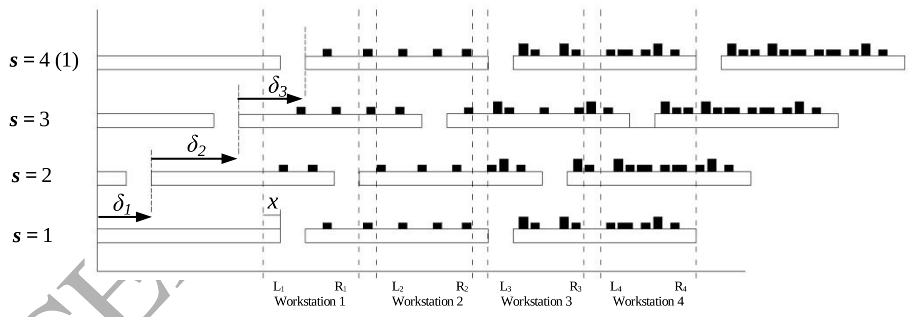

Figure 2. *Four snapshots of a cycle with three stationary stages*

Unlike common assembly lines, in this kind of line the length of the workpieces is longer than the width of the workstations. Consequently, one workpiece may be processed by several workstations at the same time, and one workstation may process portions of either one or two consecutive workpieces at the same time (recall Fig. 1). Therefore, a task can only be performed if it is situated inside the reachable interval $[L_i, R_i]$ (*accessibility window*) of the workstation *i* where it will be executed (see Fig. 2). This environment, where task positioning limits the access to restricted areas of the workpiece, motivates the so-called accessibility windows assembly line balancing problem (AWALBP) (Calleja *et al*., 2013).

A well-known classification of assembly line balancing problems is the one proposed by Baybars (1986), which differentiates between two classic problems: the Simple Assembly Line Balancing Problem (SALBP) and the General Assembly Line Balancing Problem (GALBP). The SALBP has been extensively studied in the literature thus far (Otto *et al*., 2013; Vilà and Pereira, 2013; Morrison *et al*. 2014; Pape, 2015) and is characterized by a set of restrictive assumptions (Baybars, 1986). Recently, extensive research has been done in order to address more generalized problems (GALBP), which includes problems with specific real-world restrictions (see, for example, Becker and Scholl, 2006; Capacho *et al*., 2009; Martino and Pastor, 2010; Corominas *et al*., 2011; Battaïa and Dolgui, 2012; Battaïa and Dolgui, 2013; Tuncel and Topaloglu, 2013; and Sternatz, 2014). The AWALBP is a variant of the GALBP. The optimization of AWALBP involves the solution of several *NP*-hard subproblems (Gaudlitz, 2004). With regard to the subproblems considered, the AWALBP can be tackled at four optimization levels (Calleja *et al*., 2013): the assignment of each task to one compatible workstation and stationary stage (AWALBP-L1); the initial position of the workpieces in the cycle, as well as the number and the length of the forward steps (AWALBP-L2); the component type allocation to feeders (AWALBP-L3); and the number and the type of workstations (AWALBP-L4). The objective is to minimize the cycle time. Each level addresses the optimization of its own level as well as its predecessors. For example, in AWALBP-L2 levels L1 and L2 are to be solved when solutions of L3 and L4 are given. A detailed description of AWALBP and its variants, along with a literature review has been presented in Calleja et al. (2013).

Tazari et al. (2006) studied a variant of the problem that matches an AWALBP-L1, where for each task a subset of the workstations is compatible (instead of a single workstation). Metaheuristics were applied for such variant and successful experiments on real-world instances were reported.

This paper deals with the case of AWALBP-L2 defined in Müller-Hannemann and Weihe (2006), where for each task exactly one machine is compatible. In that work, the authors describe the problem and define the conditions that a solution must fulfill. An iterative heuristic is proposed, but the corresponding enumeration procedure is not detailed. At each iteration, the proposed heuristic reduces the original problem in two steps: i) the movement scheme (which is obtained heuristically) is fixed, and ii) an algorithm is applied (for the given movement scheme) to assign each task to exactly one stationary stage of the cycle in which the location of this task on the workpiece is accessible from the workstation of this task.

The algorithm reportedly provides provably near-optimal solutions under the following assumptions: i) task lengths do not differ by orders of magnitude from each other, and ii) the number of tasks is orders of magnitude larger than the number of workstations and the number of forward steps. The authors suggest an enumeration procedure to find solutions, but do not report on computational experiments.

In contrast to the heuristic approach of Müller-Hannemann and Weihe (2006), three MILP-based approaches have been proposed in the literature. On the one hand, a variety of mathematical programming models have been presented in order to find the optimal solution. Corominas and Pastor (2009) formulated the optimization problem as a mixed-integer linear programming (MILP) model, but they do not include computational results. Based on such formulation, two enhanced MILP models were proposed by Calleja *et al*. (2013). In order to allow further research, a set of realistic benchmark instances was generated and uploaded online (https://www.ioc.upc.edu/EOLI/research/), and instances up to a certain size were solved to optimality. On the other hand, a different approach to the problem considers hybridizing heuristics and mathematical programming to solve the instances that are out of reach of the former models, which is presented in Calleja *et al*. (2014). In that work, a heuristic is used to compute good initial solutions and to derive bounds. In a second step, the obtained bounds are incorporated onto a MILP model, with the aim to reduce the model dimension and the solution space. Contrarily to the collaborative combination between a heuristic and MILP of Calleja *et al*. (2014), in this paper we propose integrative combinations between metaheuristics and MILP, where the MILP model is a subordinated element embedded inside the metaheuristic scheme.

What emerges from the computational results on AWALBP-L2 (Calleja *et al*., 2013, 2014) is that computing an optimal solution of the problem might become intractable for large size instances. This may be expected since even the simplest version of the problem, AWALBP-L1, is already NP-hard, as proven in Calleja *et al*. (2014). For this reason, metaheuristic or hybrid solution methods could be envisioned to solve this problem. In the last few years, so-called hybrid optimization approaches have become increasingly popular for tackling complex optimization problems (Blum *et al*., 2011). One of the latest trends of hybridization is the interoperation of metaheuristics with mathematical programming techniques (Boschetti *et al*., 2009). In this line, the word *matheuristic* has been coined to indicate those solution approaches that exploit the complementary strengths of exact and (meta)heuristic components (Maniezzo *et al*., 2009). Manifold possibilities of hybridization within a matheuristic arise. According to their control strategy, such hybrids can be classified into integrative (coercive) and collaborative (cooperative) combinations (Puchinger and Raidl, 2005). In integrative combinations, one technique is considered as a subordinated, embedded component of another technique, following a master-slave scheme. Collaborative algorithms, in contrast, exchange information but are not part of each other.

In this paper, we propose three hybrid metaheuristics (or matheuristics, according to the aforesaid definitions) in which mathematical programming models are used in a metaheuristic frame - one based on simulated annealing (SA) and the other two based on tabu search (TS). The proposed approaches differ in the way the neighborhood is defined. More specifically, the two former methods utilize a classical move-based neighborhood, whereas the latter one makes use of the corridor method (CM) (Sniedovich and Voβ, 2006) to draw a *corridor* around the current solution via the imposition of exogenous constraints on the problem formulation. Furthermore, combined approaches of the aforementioned hybrids with a mathematical programming model are proposed.

The remainder of the paper is organized as follows: In Sections 2 and 3, we describe the AWALBP-L2 considered in this work and introduce the proposed hybrid metaheuristics, respectively. In Sections 4, 5 and 6, we detail the proposed hybrids based on SA, TS and TS with CM, respectively. In Section 7, we present combined approaches of the aforementioned hybrids with a mathematical programming model. Comparative experimental results of the proposed hybrid metaheuristics and the best in the literature are shown in Section 8. Finally, Section 9 presents some concluding remarks.

2. Problem specification

We consider the specific case of AWALBP-L2 described in Müller-Hannemann and Weihe (2006). The considered case can be stated as follows. An assembly line is given with a number *m* of workstations. Each workstation *i* has an accessibility window to the workpieces delimited by the interval $[L_i, R_i]$ of the assembly line such that $L_1 = 0$ and $R_i > L_i > R_{i-1}$ for $i = 2,...,m$. Therefore, the accessibility windows of the workstations do not overlap. Each task can be executed only on one given workstation. On each workstation *i*, a specified set of tasks J_i must be executed for each workpiece. The total number of tasks is denoted by $N = \left|\bigcup_{i=1}^{m} J_i\right|$. For each task *j* $\left(j = 1,...,N\right)$ the triple (p_j, a_j, m_j) is known, where p_j is the processing time of task *j*, a_j is the distance from the task position to the right border of the workpiece, and m_j is the workstation that has to execute this task. Then, the solution of the problem decomposes into:

i) a movement scheme $< x : \delta_1, \delta_2, ..., \delta_S >$, which consists of:

   - the initial position *x* of the workpieces on the line.
   - the number *S* of stationary stages (which coincides with the number *S* of forward steps).
   - the values $\delta_1,\dots, \delta_S$ of the length of the forward steps, where δ_s is the number of elementary steps of the forward step *s* $\left(s = 1,...,S\right)$.

ii) for each task, an assignment to one stationary stage of the cycle where the position of the task is accessible for the station of this task.

To be feasible, a solution must hold the following conditions. First, the sum of all forward steps in a cycle must be equal to the distance *A* between two right (left) borders of two consecutive workpieces. Second, all forward steps must be a multiple of Δ (the elementary step). Finally, the third condition is that each task must be assigned to a stationary stage in which the task is accessible from its workstation.

The objective function (1) is the minimization of the cycle time (*CT*). Between two stationary stages, there is a time *T* to take into account the acceleration and deceleration of the line as well as the resetting of the robot arms. Then the total time of the cycle is equal to the sum of i) the time *T* multiplied by the number of stationary stages *S* plus ii) the time elapsed in the stationary stages constituting a cycle and iii) the time for transporting a workpiece through the assembly line at steady speed (since the latter is a constant it is not regarded for optimization purposes):

$$CT = T \cdot S + \sum_{s=1}^{S} C_s \tag{1}$$

where $\sum_{s=1}^{S} C_s$ is the total processing time corresponding to all *S* stationary stages constituting a cycle, and C_s is the completion time, for the whole line, corresponding to the stationary stage *s* ($s = 1,...,S$).

3. The proposed hybrid metaheuristics

We propose three hybrid metaheuristics that integrate mathematical programming models into metaheuristic frameworks. The proposed hybrid metaheuristics can be seen as integrative algorithms where the metaheuristic is used as the master mechanism that guides the search process and a mathematical model acts as an embedded slave.

The two first hybrid metaheuristics proposed in this work, SA and TS, rely on a mathematical programming model: the so-called *Task model* of Calleja *et al*. (2014). The *Task model* computes, for a given movement scheme, the optimal assignment of each task to one stationary stage of the cycle. This is, for a fixed movement scheme, the *Task model* allows for the computation of an optimal solution. Despite the assignment of tasks to stationary stages being an NP-hard problem (see proof in Calleja *et al*. (2014)), the *Task model* solves the problem very fast. For this reason, we define neighborhoods in the space of the movement schemes rather than around complete solutions. More specifically, in the proposed hybrid SA and TS the neighborhood is defined by applying local changes or moves to a current movement scheme.

The complete *Task model* is given next.

Data

m	number of workstations $(i=1,...,m)$
N	number of tasks $(j=1,...,N)$
J_0	set of tasks $(J_0=\{1,2,...,N\})$
J_i	set of tasks to be performed in workstation *i*, where $\bigcup_{i=1,..,m} J_i = J_0$ and $J_i \cap J_{i'} = \varnothing$ $(i=1,\dots,m; i'=1,...,m; i \neq i')$
p_j	processing time of task *j* $(j=1,...,N)$
S	number of forward steps in a cycle (therefore, also the number of stationary stages)
Π_j	set of stationary stages where task *j* is accessible from the workstation where it can be performed $(j=1,...,N)$.

Variables

$y_{js} \in \{0,1\}$	$y_{js}=1$ *iff* task *j* is performed in stationary stage *s* $(j=1,..,N; s \in \Pi_j)$
C_s	completion time corresponding to the stationary stage $s\,(s=1,...,S)$

Model

$$[MIN]\; z=\sum_{s=1}^{S} C_s \tag{2}$$

$$\sum_{s\in\Pi_j} y_{js}=1 \qquad j=1,...,N \tag{3}$$

$$\sum_{j\in J_i \,|\, s\in\Pi_j} p_j \cdot y_{js} \le C_s \qquad i=1,...,m;\; s=1,...,S \tag{4}$$

The objective (2) is the minimization of the completion time of the stationary stages. Constraints (3) impose that each task is assigned to one, and only one, stationary stage, and (4) ensure that the time corresponding to the stationary stages is not less than the processing time at any workstation.

In the third proposed metaheuristic (TS-CM), in contrast, the neighborhood is defined by building a *corridor* around a current movement scheme in order to iteratively solve smaller portions of the target problem. More specifically, exogenous constraints are imposed on the original formulation of the problem and, subsequently, the constrained problem (denoted *Solve-corridor model)* is solved.

4. Hybrid simulated annealing metaheuristic

Simulated annealing (SA) is a probabilistic optimization method which since its first introduction by Kirkpatrick *et al*. (1983), has been recognized as a simple yet powerful metaheuristic that provides excellent solutions to a wide variety of hard combinatorial optimization problems (Suman and Kumar, 2006).

Basically, SA is a local search procedure that tries to avoid being trapped in local optima by allowing probabilistically moves to worse solutions. The algorithm starts from an initial solution, which is initially the current solution *y*, and by initializing the value of a parameter *t* called *temperature*. Then, at each iteration, a solution *y'* from the neighborhood of the current solution *N(y)* is randomly selected. If the neighbor is not worse than the current solution, then the neighbor is accepted and replaces the current solution. In the case that it is worse, the neighbor can also be accepted, with a probability that depends on i) how much worse is the neighbor, and ii) the value of the temperature *t*. Initially, the algorithm starts at a high temperature *t* (that is, the probability of accepting deteriorating moves is high), which then gradually decreases and approaches zero. The number of iterations for which the temperature remains constant before being reduced is *itt*. The SA algorithm is presented in Fig. 3.

The proposed hybrid combines the general scheme of SA (Fig. 3) with the *Task model*. As mentioned in Section 3, the search is performed in the space of the movement schemes. In each iteration, the *Task model* is employed to compute the optimal cycle time for the current neighbor movement scheme, which provides a complete current solution for the problem. The obtained cycle time value determines whether the candidate movement scheme (along with its optimal task assignment) will be accepted or rejected as the new current solution in the SA local search.

SA

```
Let $f(y)$ be the objective function to be minimized of the solution $y$
Let $N(y)$ be the neighborhood of the solution $y$
Let $A(t)$ be a new temperature value obtained from the temperature $t$

 1.  Initialize the parameters:
         $t_0$ (initial temperature)
         $itt$ (number of iterations during which the temperature remains constant)
 2.  $t := t_0$
 3.  $y$:= Generation of the initial solution
 4.  while the stopping criterion is not satisfied do
 5.      for ($i := 0$; $i < itt$; $i := i + 1$):
 6.          $y'$:= randomly select $y'$ from $N(y)$
 7.          if $f(y') \le f(y)$ then $y := y'$
 8.          else $y := y'$ with a probability $\exp(-(f(y')-f(y))/t)$
 9.          end
10.      end
11.      $t := A(t)$
12.  end
13.  return the best solution found
```

***Figure 3.** General scheme of simulated annealing*

The efficiency of the general scheme of SA depends on some key decisions. Some of these decisions are problem-specific, whereas some others are generic to SA. Specific decisions for the AWALBP-L2 include the definition of neighborhood of a solution ($N(y)$), and the generation of the initial solution. General decisions are the cooling schedule to decrease the temperature $A(t)$ and the stopping criterion of the algorithm. In the following we outline such decisions.

4.1 Neighborhood of movement schemes

The proposed SA hybrid makes use of three neighborhood structures, N_1, N_2, and N_3, as follows. N_1 consists in transferring one elementary step from a forward step to another forward step. N_2 consists in inserting a new forward step by transferring one elementary step from an existing forward step to a new one. Finally, N_3 considers the neighbors obtained by varying the value of the initial position x in the interval $0 \le x \le \min\left(R_1 + a^1_{\min}, A - \Delta\right)$, where $a^1_{\min} = \min_{j \in J_1} a_j$. Note that in the two first neighborhood types, a forward step with only one elementary step may achieve length zero if its only elementary step is transferred (and thus such forward step disappears from the movement scheme). Therefore, the number of forward steps may vary. More specifically, it can remain equal or decrease in N_1, and it can remain equal or increase in N_2. Feasibility loss following transference or insertion of elementary steps can occur if the resulting movement scheme contains some tasks whose position is not accessible at any stationary stage. In any case, we consider only those neighbors which are feasible. At each iteration of the SA algorithm, it is selected at random from which of the three neighborhoods a neighbor of the current movement scheme will be obtained. The values of the probabilities associated to the neighborhood selection are to be fine-tuned (see Section 8).

4.2 Initial solution

An initial solution is obtained by using the *Initial solution matheuristic* proposed in Calleja *et al*. (2014). It consists of i) an algorithm to generate, for a given value of *x*, a feasible movement scheme and ii) a mathematical model (the *Task model*), to compute the optimal assignment of tasks to stationary stages (for the generated movement scheme). The initial solutions obtained with this procedure appear to be of good quality and the necessary computational time is on average as small as a few milliseconds. Among the obtained solutions, computed for all values of *x* multiples of Δ, a solution with the minimum cycle time is identified. In case of having several solutions with the minimum cycle time, a solution with the minimum number of stationary stages is selected. The information given by the obtained initial solution is used to compute a lower bound on the value of the cycle time, $LB1^{CT}$, which is used to certificate whether a current solution is optimal (see Section 4.4). The bound $LB1^{CT}$ is computed as follows. Since the *Initial solution matheuristic* has been proven to provide solutions with the minimum number of stationary stages (see proof in Calleja *et al*. 2014), the solution with the minimum number of stationary stages among all those obtained with the mentioned matheuristic, Sol^S, gives a lower bound on the number of stationary stages, $LB1^S$. Then, we derive a lower bound on the cycle time, $LB1^{CT}$, by summing lower bounds on the two terms that compose the objective function (see Eq. (1)): (i) $T \cdot LB1^S$ plus (ii) a lower bound on the completion time of the stationary stages, which we name W_{max}, corresponding to the processing time of the most loaded workstation on the line. This is, $W_{max} = \max_{i=1,\dots,m} \sum_{j \in J_i} p_j$.

4.3 Cooling schedule

The cooling schedule specifies how the temperature of the SA algorithm is decreased as the search progresses. We use geometric cooling, one of the most popular schedules used in the literature, that is, $A(t) = \alpha \cdot t$, where $0 < \alpha < 1$ (Downsland and Adenso-Díaz, 2003, Henderson *et al*., 2003). The value of the α parameter, as well as the initial temperature t_0 and the number of iterations during the temperature remains constant, *itt*, are to be fine-tuned, as explained in Section 8.

4.4 Stopping criterion

The algorithm stops when one of the following conditions is reached: i) a specified maximum time has elapsed, or ii) the objective function value of a solution coincides with $LB1^{CT}$ and thus the solution is proven optimal.

5. Hybrid tabu search metaheuristic

Tabu search (TS) is a metaheuristic originally proposed by Glover (1986) that has been successfully applied in many difficult combinatorial optimization problems (Glover, 1997, Lapierre *et al*., 2006, Pedersen *et al*., 2009). Like SA, TS can be seen as a local search that allows non-improving moves. The innovative idea of TS is the explicit use of memory structures, that record not only information about the current solution, but also information about the recent search trajectory followed to reach the current solution. Essentially, a TS algorithm moves at each iteration from a solution *y* to a solution in its neighborhood *N*(*y*), and may accept worse neighbors than the current solution. To prevent endless cycling and guide the search into unexplored areas, some formerly visited solutions, or attributes of them, are temporarily declared tabu or prohibited. The number of iterations that an attribute remains tabu is called its tabu tenure. The tabu status of a solution, though, can be overridden if a specified aspiration criterion is met; for example, if a tabu solution is better than the best solution found so far. The general TS algorithm is presented in Fig. 4. For a thorough presentation of the method, we refer the interested readers to Glover (1989, 1990) and Gendreau (2003).

TS

```
1. Define the neighborhood $N(y)$
2. Let $y$ be an initial solution and $y^*:=y$
3. while the stopping criterion is not satisfied do
4.     Let $y'$ be the best solution from $N(y)$ which is allowed by aspiration or is not tabu
5.     if $y'$ is better than $y^*$, then $y^* := y'$ end
6.     Add the current move to the tabu list (removing its oldest move if it is full)
7.     $y := y'$
8. end
9. return $y^*$
```

Figure 4. *General scheme of tabu search*

The proposed hybrid TS relies on the general TS guidelines presented by Glover (1989, 1990), as shown in Fig. 4. As in the proposed SA-hybrid, we build the neighborhood around the movement schemes. Subsequently, the *Task model* is used to find an optimal assignment of tasks to stationary stages of the current movement scheme, which provides a complete current solution. A similar approach embedding a LP model in a probabilistic tabu search to solve a facility layout problem with unequal area departments has been proposed in Kulturel-Konak (2012). As in our paper, a mathematical programming model is used to evaluate the non-tabu solutions of the neighborhood of the current solution with the difference that instead of evaluating each and every element of the neighborhood, it considers only evaluating a random sample to reduce computational effort.

In our approach we consider the same initial solution generation, neighborhood structures and stopping criterion as in the proposed SA. The remainder elements of the proposed TS-based hybrid, i.e., the tabu lists, tabu attributes and aspiration criterion, are defined in the following subsections.

5.1 Tabu lists and tabu attributes

The tabu list is directly related to the neighborhood structure used to solve the problem. We consider the three neighborhoods N_1, N_2, and N_3 proposed in Section 4.1. In each iteration, the best neighbor movement scheme is searched within the three neighborhoods. Neighborhoods N_1 and N_2 are similar structures since they are both generated by transferring one elementary step to an existing or a new forward step. Neighborhood N_3, though, is a different structure based on the value of the initial position x. Therefore, we consider two different tabu lists, a first tabu list for the neighbors selected from N_1 or N_2, and a second tabu list for those selected from N_3, as follows. We call *transmitter forward step* the forward step which transfers one elementary step. Similarly, a *receiver forward step* is the one which receives an elementary step. Then, the first tabu list, T_1, contains attributes consisting of four elements: i) initial position value, ii) the number *S* of forward steps iii) the transmitter forward step *s* and its length δ_s, and iv) the receiver forward step *s'* and its length $\delta_{s'}$. The second tabu list (T_2), though, contains only the two first aforementioned elements.

A numerical example is shown in Table 1. Let $< 5 : 2, 3, 6, 2 >$ be a current movement scheme (column 1), with an initial position $x = 5$ and four forward steps with 2, 3, 6 and 2 elementary steps, respectively. Column 2 states the neighborhood type from which a neighbor will be generated. In the case of neighborhood N_2, two subtypes are distinguished: N_2(a), where an elementary step is inserted *between* the forward steps of the current movement scheme, and N_2(b), where the elementary step is inserted *after the last* forward step of the current movement scheme. Column 3 gives an example of a neighbor movement scheme obtained from each neighborhood type. Column 4 indicates in which tabu list, T_1 or T_2, the attribute will be recorded. Finally, Column 5 details the attribute to be stored in the tabu list when the neighbor movement scheme is set tabu. For example, if the best neighbor is < 5: 1, 4, 6, 2 >, which has been obtained from N_1, then the tabu attribute is $< x = 5, S = 4, \delta_1 = 2, \delta_2 = 3 >$ and it is added to the T_1 tabu list,. In the case that the best neighbor belongs to N_2, the same tabu attribute and tabu list are considered. Note that, in N_2(b), the receiver forward step does not exist in the current movement scheme and thus the last element of the tabu attribute is considered as zero $(\delta_5 = 0)$. Finally, if the best neighbor is the one obtained from N_3 then the tabu attribute is $< x = 5, S = 4 >$ and it is added to the T_2 tabu list.

Table 1. An example of the different neighbor movement schemes and their tabu attributes

Current movement scheme	Neighborhood	Neighbor movement scheme	Tabu list type	Tabu attribute
< 5: 2, 3, 6, 2 >	N_1	< 5: 1, 4, 6, 2 >	T_1	$< x = 5, S= 4, \delta_1 = 2, \delta_2 = 3 >$
	N_2(a)	< 5: 2, 2, 1, 6, 2 >	T_1	$< x = 5, S = 4, \delta_2 = 3, \delta_3 = 6 >$
	N_2(b)	< 5: 2, 2, 6, 2, 1 >	T_1	$< x = 5, S = 4, \delta_2 = 3, \delta_5 = 0 >$
	N_3	< 7: 2, 3, 6, 2 >	T_2	$< x = 5, S = 4 >$

The lengths of tabu lists are to be fine-tuned, as explained in Section 8.

5.2 Aspiration criterion

We use the most commonly used aspiration criterion in the TS literature (Gendreau and Potvin, 2005) which allows a tabu move when it results in a solution better than the current best-known solution.

6. Hybrid tabu search - corridor method metaheuristic

A hybrid approach combining TS and a Corridor Method (CM) is presented next. The CM is a matheuristic introduced by Sniedovich and Voβ (2006), which intertwines mathematical programming techniques with metaheuristic features. The central idea of the CM relies on the iterative use of an exact method to solve optimally restricted portions of the solution space of a given problem. Such portions of the original space are defined by building a *corridor* around a current solution via the imposition of exogenous constraints.

The proposed hybrid TS-CM follows the scheme of the TS metaheuristic proposed by Glover (1989, 1990) (recall Fig. 4), as described in Section 5. However, the proposed hybrid TS-CM differs from the aforementioned one in the sense that now the neighborhoods are not defined via local changes or moves, but constructed by adding exogenous constraints onto an embedded MILP model. Such MILP model is subsequently used to solve the resulting portion of the problem space. Furthermore, additional constraints are also imposed in order to model the tabu lists and the aspiration criterion. An approach making use of TS as a master strategy and a branch and bound solver as an embedded mechanism for solving relaxed instances of the generalized assignment problem has been proposed by Woodcock and Wilson (2010). As in our paper, the authors use mathematical programming techniques to move from a current solution to a new one. However, to the best of our knowledge, we are not aware of any work in the literature hybridizing TS with CM in the way presented here.

The proposed procedure starts from an initial solution obtained with the *Initial solution matheuristic* of Calleja *et al*. (2014) and explores the neighboring solution space in search of an improving solution.

At each iteration, the MILP model receives a current solution as input. Based on this solution, bounds on the cycle time and the number of forward steps are computed and incorporated to the model. Next, the constrained version of the problem defined by the corridor is solved by using the MILP model within a limited computational time.

The overall algorithm terminates when one of the following criteria is reached: i) a maximum running time, or ii) the problem is solved to proven optimality since a solution is obtained whose objective function value coincides with the lower bound of the cycle time, $LB1^{CT}$.

6.1 *Tabu list and aspiration criterion*

A short-term memory structure is used as a tabu list, which stores the attributes of the movement schemes recently visited. More specifically, we propose a composite attribute for the tabu list, $< Tabu^{S}, Tabu^{x} >$, where $Tabu^{S}$ and $Tabu^{x}$ express, respectively, the number $\hat{S}$ of stationary stages and the initial position $\hat{x}$ of the movement scheme of the current solution. Then, the set of tabu attributes in a tabu list is defined by $\left\{< Tabu_h^S, Tabu_h^x >: h = 1,...,TT\right\}$, where TT is the number of attributes contained in the tabu list.

The proposed TS-CM uses the aspiration criterion described in Section 5.2, which overrides the tabu status of a solution if its objective function is better than that of the best-known solution so far.

6.2 The embedded corridor method

The basic elements of the CM are: a problem *P* generally belonging to the class of *NP*-hard problems, a very large feasible solution space *Y* and an exact method *M* capable of solving *P* to optimality if the size of the solution space is not too large. The CM imposes exogenous constraints on the original formulation of the problem in such a way that smaller manageable portions of the solution space are identified. These exogenous constraints define a *corridor*, i.e., a set of solutions, around a given current solution $\hat{y} \in Y$. The nature of the imposed constraints should be such that they are compatible with both the structure of the problem *P* and the method *M* used to solve them.

Let us assume that method *M* is a MILP model. One way to identify smaller manageable portions of the solution space to be explored with the model is to constrain the domains of the variables that are present in a current solution. In the following we outline how a CM can be applied to a MILP model. Let us suppose that we are given a current solution $\hat{y}$ with a number Ψ of decision variables $\left(\hat{y}_1, \hat{y}_2, ..., \hat{y}_\Psi\right)$. In order to impose constraints on the variable domains, we need to limit the distance between the value of a variable y_n and its current value $\hat{y}_n$. Therefore, a neighborhood around a current solution can be generated by drawing corridors as follows:

$$N\left(\hat{y}\right) := \left\{\left(y_1, y_2, ..., y_\Psi\right) \in Y : \left(\hat{y}_n - R_n\right) \leq y_n \leq \left(\hat{y}_n + R_n\right), n = 1, ..., \Psi\right\} \qquad (5)$$

where $R_n \ \left(n = 1, ..., \Psi\right)$ is a parameter used to define the corridor width.

Equation (5) limits the solution space only to those solutions whose distance from the current solution, for each variable y_n, is not greater than a given maximum value R_n.

Finally, in order to incorporate the neighborhood definition to the original MILP formulation of the problem, the following constraints are imposed:

$$y_n \geq \hat{y}_n - R_n \qquad (n = 1, ..., \Psi) \qquad (6)$$
$$y_n \leq \hat{y}_n + R_n \qquad (n = 1, ..., \Psi) \qquad (7)$$

At each iteration, constraints (6)-(7) are therefore imposed onto the original model, which is introduced in Section 6.4, and the new constrained version is solved by applying a suitable algorithm.

In the following we introduce the notation required to formulate a fitting model for the CM. Let us consider the movement scheme of the current solution with an initial position $\hat{x}$ and $\hat{S}$ forward steps of lengths $< \hat{\delta}_1, \hat{\delta}_2, ..., \hat{\delta}_{\hat{S}} >$. Let us suppose that we generate a corridor of width R^δ around each variable δ_s, such that $\delta_s \in \left[\hat{\delta}_s - R^\delta, \hat{\delta}_s + R^\delta\right]$. As a result of the transference of elementary steps in the generation of neighbor movement schemes, there either may be some forward steps which become empty (i.e., their length is zero and thus disappear) or some whose length is necessarily greater than zero (i.e., it is known a priori that they will exist in any neighbor movement scheme). Therefore, the actual number of variables δ_s that a neighbor solution will have is not known in advance. In order to model the number of variables needed, we define the following additional data.

Let *US* be the upper bound on the number of forward steps, empty or not, that may be generated for the feasible solutions contained in the corridor, while respecting the corridor width:

$$US = \min\left(A/\Delta,\, \hat{S} + \sum_{s=1}^{\hat{S}} \min\left(R^{\delta}, \hat{\delta}_s \right) \right) \tag{8}$$

In the proposed model, *US* is used to upper bound the number of variables δ_s, such that $s = 1,\ldots,US$.

By *ES* we denote the set of forward steps whose existence *can* be assured a priori in all the feasible solutions of the space delimited by the corridor:

$$ES = \left\{ s = 1,\ldots,\hat{S} : \hat{\delta}_s - R^{\delta} \geq 1 \right\} \tag{9}$$

Let *NES* then be the set of forward steps whose existence *cannot* be assured a priori in all the feasible solutions of the space delimited by the corridor:

$$NES = \left\{ 1,\ldots,US \right\} \setminus ES \tag{10}$$

Finally, we derive an upper bound on the number of non-zero forward steps, UB^S, that a neighbor movement scheme may have, by the sum of (i) the current number of forward steps, (ii) the total number of elementary steps that can be transferred by the forward steps of the current movement scheme whose values are greater than R^{δ}, and (iii) the total number of elementary steps that can be transferred by the forward steps of the current movement scheme whose values are greater than one but equal to or smaller than R^{δ}:

$$UB^S = \hat{S} + \left| ES \right| \cdot R^{\delta} + \sum_{s=1,\ldots,\hat{S} \mid \hat{\delta}_s \leq R^{\delta}} \left(\hat{\delta}_s - 1 \right) \tag{11}$$

Fig. 5 depicts a numerical example for the computation of UB^S. Let us assume that $< 0 : 6, 7, 2, 1 >$ is the movement scheme of the current solution $\left(\hat{x} = 0, \hat{S} = 4, \hat{\delta}_1 = 6, \hat{\delta}_2 = 7, \hat{\delta}_3 = 2, \hat{\delta}_4 = 1 \right)$ and that a corridor of width $R^{\delta} = 3$ is built around the values of the forward steps. The idea is to construct a neighbor movement scheme in such a way that the maximum number of non-zero forward steps is obtained. This can be done by transferring as many elementary steps as possible from each forward step in such a way that the latter keeps at least one elementary step. In the example of Fig. 5, such transfers generate a neighbor movement scheme with 11 forward steps, which gives the value for UB^S.

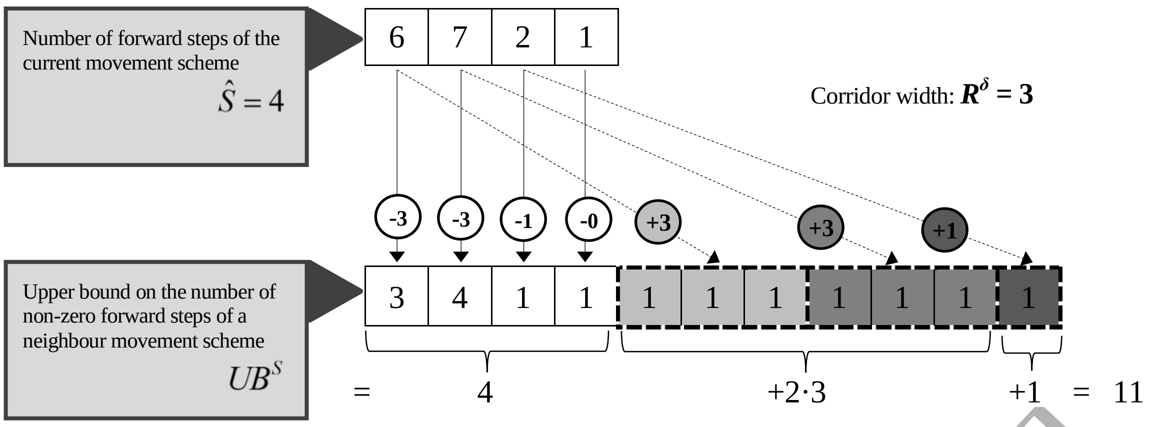
Number of forward steps of the current movement scheme
$\hat{S} = 4$
6 7 2 1
Corridor width: $R^{\delta} = 3$
-3 -3 -1 -0 +3 +3 +1
3 4 1 1 1 1 1 1 1 1 1
Upper bound on the number of non-zero forward steps of a neighbour movement scheme
UB^S
= 4 +2·3 +1 = 11

Figure 5. *Example of the computation of parameter UB^S*

In order to further illustrate the required notation let us consider a numerical example of a line with $\Delta = 1$ and $A = \sum_{s=1}^{\hat{S}} \Delta \cdot \hat{\delta}_s = 5$ (Table 2). Let us suppose that $<0:4,1>$ is the movement scheme of the current solution $\left(\hat{x} = 0, \hat{S} = 2, \hat{\delta}_1 = 4, \hat{\delta}_2 = 1\right)$. Let us assume that we build a corridor of width R^{δ}=1 around the forward step values of the movement scheme. The neighbor movement schemes admissible in this corridor are given in the second column of Table 2. Then, the neighbor movement schemes that can be generated inside a corridor of width R^{δ}=1 around the current forward steps may contain at most four forward steps (empty or not) and thus $US = 4$. As can be seen from the generated neighbors, there may be some forward steps whose length is necessarily greater than zero if the current length of the forward step is greater than the corridor width. In this example, the set of non-zero forward steps is $ES = \{1\}$. Conversely, the rest of forward steps of a neighbor movement scheme may be of length zero and thus $NES = \{2,3,4\}$. Finally, the upper bound on non-zero forward steps is $UB^S = 3$.

Table 2. A numerical example illustrating the values of parameters ES, NES, US and UB^S

Current movement scheme	Neighbors within a corridor of width R^{δ} = 1	Data values
<0: 4, 1 >	< 0: 5, 0, 0, 0 > < 0: 3, 2, 0, 0 > < 0: 3, 1, 1, 0 > < 0: 4, 0, 1, 0 > < 0: 4, 0, 0, 1 > < 0: 3, 0, 1, 1 > < 0: 3, 1, 0, 1 >	$US = 4$ $ES = \{1\}$ $NES = \{2,3,4\}$ $UB^S = 3$

6.3 Definition of corridors

In order to apply a corridor around a current solution we need to select which variables of the current solution will be restricted. The width of such corridor will allow only for the exploration of those solutions that are at a maximum distance from a current one. More specifically, we apply corridors to some of the variables that define the movement scheme of a current solution. Several possibilities for the construction of corridors arise, depending on which variables are selected. Specifically, we consider three alternate corridors, denoted *C1*, *C2* and *C3*, which are explained next.

6.3.1. Corridor C1

Given a current movement scheme, corridor *C1* constructs a neighborhood around the variable $\delta_S \left(s=1,\ldots,US\right)$ by including all movement schemes whose forward steps have a length within a distance R^{δ} from the current lengths.

In the corridor, the number of forward steps is lower bounded by LB^S and thus the maximum number of elementary steps that a forward step may achieve is given by the expression $\left(A/\Delta - LB^S + 1\right)$.

6.3.2 Corridor C2

The second type of corridor, *C2*, builds a neighborhood around $\delta_S \left(s=1,\ldots,US\right)$ and around the number of forward steps S .

An additional parameter, R^S , is added to express the corridor around the current number of forward steps $\hat{S}$. Consequently, the following data are modified:

$$LB^S = \max\left(\hat{S} - R^S, \left|ES\right|, LB1^S\right) \tag{12}$$

$$UB^S = \min\left(\hat{S} + R^S, \hat{S} + \left|ES\right| \cdot R^{\delta} + \sum_{s=1,\ldots,\hat{S} \left| \hat{\delta}_s \le R^{\delta}\right.} \left(\hat{\delta}_s - 1\right)\right) \tag{13}$$

where (12)-(13) define the corridor and then the expression for *US* is modified as follows:

$$US = \min\left(A/\Delta, \hat{S} + \left(\hat{S} - \left|ES\right|\right) + \min\left(R^S, \sum_{s=1}^{\hat{S}} \min\left(R^{\delta}, \hat{\delta}_s - 1\right)\right)\right) \tag{14}$$

6.3.3 Corridor C3

The third corridor structure considers the construction of a neighborhood around the lengths of the forward steps δ_s, the number of forward steps *S* and the initial shift *x*. Again, we consider an additional parameter, R^x, to express the width corridor around the variable *x*. The following parameters arise:

$$X^- = \left(\hat{x} - R^x\right) \bmod \left(\min\left(R_1 + a^1_{\min}, A - \Delta\right) + 1\right) \tag{15}$$

$$X^+ = \left(\hat{x} + R^x\right) \bmod \left(\min\left(R_1 + a^1_{\min}, A - \Delta\right) + 1\right) \tag{16}$$

Once the values X^- and X^+ have been defined, two cases may arise:

a) $X^- \le X^+$. The corridor around *x* is defined as: $X^- \le x \le X^+$

b) $X^- > X^+$. In this case, the corridor is a wrap-around interval where the values admitted for the variable *x* are:

$$x \in \left\{ X^+, X^+ + 1, X^+ + 2, ..., \min\left(R_1 + a^1_{\min}, A - \Delta\right), 1, 2, ..., X^- \right\}.$$

To define the corridor, we introduce the variable $r \in \{0,1\}$ and the following constraints:

$$x \geq X^- - MR^x_1 \cdot r \qquad (17)$$

$$x \leq X^+ + MR^x_2 \cdot (1 - r) \qquad (18)$$

where:

$$MR^x_1 = X^-$$

$$MR^x_2 = \min\left(R_1 + a^1_{\min}, A - \Delta\right) - X^+$$

6.4 *A MILP model for the corridor method*

In this section we present a MILP model, denoted *Solve-corridor*, to be used at each iteration of the proposed hybrid TS-CM. Such MILP model is used to define a neighborhood (incorporating corridors), and to obtain the best neighbor that is not tabu or fulfills the aspiration criterion. The model is inspired on the *Solve model* of Calleja *et al*. (2014). The new contributions to the formulation correspond to the imposition of exogenous constraints to define the corridor, and the addition of the tabu lists and the aspiration criterion.

In the following, we present the *Solve-corridor* MILP model, which includes a corridor of type *C1* around the variables δ_s $(s = 1, ..., US)$. The proposed model can be easily adapted to include corridors of types *C2* and *C3*, by incorporating the modifications stated in sections 6.3.2 and 6.3.3, respectively.

Data

N	number of tasks $(j = 1, ..., N)$
m	number of workstations $(i = 1, ..., m)$
m_j	workstation where task *j* has to be executed $(j = 1, ..., N)$
$[L_i, R_i]$	accessibility window of workstation *i* $(i = 1, ..., m)$, where $L_1 = 0$ and $R_i > L_i > R_{i-1}, (i = 2, ..., m)$
A_0	workpiece's length
A	distance between the right borders of two successive workpieces of the assembly line ($A > A_0$)
T	time to take into account acceleration and deceleration between two consecutive stationary stages
Δ	length of an elementary step
p_j	processing time of task *j* $(j = 1, ..., N)$

a_j ($0 \le a_j \le A_0$), distance to the right border of the workpiece corresponding to the task j $(j = 1, ..., N)$

J_0 set of tasks $(J_0 = \{1, 2, ..., N\})$

J_i set of tasks to be performed on workstation i, $J_i = \{j \in J_0 : m_j = i\}$, $(i = 1, ..., m)$ where $\bigcup_{i=1,...,m} J_i = J_0$, $J_i \cap J_{i'} = \varnothing$, $\forall\, i, i' = 1, \ldots, m \mid i \ne i'$

R^δ corridor width

$\hat{S}$ number of stationary stages of the TS current solution

$\hat{x}$ initial position of the workpiece with respect to the left limit of workstation 1 of the TS current solution

$\hat{\delta}_s$ number of elementary steps of the forward step s $\left(s = 1, ..., \hat{S}\right)$ of the TS current solution

$$US = \min\left(A/\Delta, \hat{S} + \sum_{s=1}^{\hat{S}} \min\left(R^\delta, \hat{\delta}_s\right)\right)$$

$$ES = \left\{s = 1, ..., \hat{S} : \hat{\delta}_s - R^\delta \ge 1\right\}$$

$$NES = \{1, ..., US\} \setminus ES$$

LB^S lower bound on the number of stationary stages $\left(LB^S = \max\left(|ES|, LB1^S\right)\right)$, where $LB1^S$ is a lower bound obtained as described in Section 4.2

UB^S upper bound on the number of forward steps

$$UB^S = \hat{S} + |ES| \cdot R^\delta + \sum_{s=1..\hat{S} \mid \hat{\delta}_s \le R^\delta} \left(\hat{\delta}_s - 1\right)$$

LB^{CT} lower bound on the cycle time $\left(LB^{CT} = T \cdot LB^S + W_{max}\right)$, where $W_{max} = \max_{i=1,...,m} \sum_{j \in J_i} p_j$

UB^{CT} upper bound on the cycle time, $UB^{CT} = T \cdot UB^S + \sum_{j=1}^{N} p_j$

CT^* cycle time of the best solution found so far within the TS

TT tenure or size of the tabu list

$Tabu_h^S, Tabu_h^x \left(h = 1, ..., TT\right)$ define the list of tabu attributes (see Section 6.1): $\left\{< Tabu_h^S, Tabu_h^x >: h = 1, ..., TT\right\}$

$kmin_j$ minimum number of times that a workpiece should be moved forward by A elementary steps such that task j is accessible in its workstation $\left(j = 1, ..., N\right)$, where $kmin_j = \left\lceil \frac{L_{m_j} + a_j - \min(R_1 + a^1_{\min}, A-\Delta) - A + \Delta}{A} \right\rceil$, being $a^1_{\min}$ the closest distance of a task position j $(j \in J_1)$ to the right border of the workpiece ($a^1_{\min} = \min_{j \in J_1} a_j$)

$kmax_j$ maximum number of times that a workpiece should be moved forward by A elementary steps such that task j is accessible in its workstation $(j = 1,...,N)$, where $i | j \in J_i$, $kmax_j = \left\lfloor \frac{R_i + a_j}{A} \right\rfloor$

Variables

$x \in \mathbb{Z}^+$	initial position of the workpiece with respect to the left limit of workstation 1, where $0 \le x \le \min\left(R_1 + a^1_{\min}, A - \Delta\right)$
$\delta_s \in \mathbb{Z}^+$	number of elementary steps of the forward step s $(s = 1,...,US)$
$\eta_s \in \{0,1\}$	$\eta_s = 1$ *iff* the forward step s exists, $s \in NES$
$b_{jsk} \in \{0,1\}$	$b_{jsk} = 1$ *iff* task j is performed during stationary stage s after the workpiece has been moved forward k times by A elementary steps, $(j = 1,...,N; s = 1,...,US; k = kmin_j,...,kmax_j)$
C_s	completion time, for the whole line, corresponding to the stationary stage s $(s = 1,...,US)$, where $\min_{j=1}^{N} p_j \le C_s \le W_{max}$ $(s \in ES)$ and $0 \le C_s \le W_{max}, s \in NES$
$y_{hg} \in \{0,1\}$	auxiliary variables $(h = 1,...,TT; g = 1,...,4)$. If the solution has the h-th tabu attribute in the tabu list, then the four variables y_{h1} , ..., y_{h4} take value 1.
$w \in \{0,1\}$	1 *iff* the new solution fulfills the aspiration criterion
$u_1, u_2, v_{s1}, v_{s2} \in \{0,1\}$	auxiliary variables that are used to remove the current TS solution from the solution space $\left(s = 1,...,\hat{S}\right)$

Model

$$[MIN]\, z = T \cdot \left(|ES| + \sum_{s \in NES} \eta_s \right) + \sum_{s=1}^{US} C_s \tag{19}$$

$$LB^{CT} \le T \cdot \left(|ES| + \sum_{s \in NES} \eta_s \right) + \sum_{s=1}^{US} C_s \tag{20}$$

$$LB^{S} \le |ES| + \sum_{s \in NES} \eta_s \tag{21}$$

$$|ES| + \sum_{s \in NES} \eta_s \le UB^S \qquad (22)$$

(a) movement scheme constraints

$$\hat{\delta}_s - R^{\delta} \le \delta_s \qquad s = 1,\dots,\hat{S} \qquad (23)$$

$$\delta_s \le \min\left(\hat{\delta}_s + R^{\delta}, A/\Delta - LB^S + 1\right) \qquad s \in ES \qquad (24)$$

$$\delta_s \le \min\left(\hat{\delta}_s + R^{\delta}, A/\Delta - LB^S + 1\right) \cdot \eta_s \qquad s = 1,\dots,\hat{S} : s \in NES \qquad (25)$$

$$\delta_s \le \min\left(R^{\delta}, A/\Delta - LB^S + 1\right) \cdot \eta_s \qquad s = \hat{S}+1,\dots,US \qquad (26)$$

$$\Delta \cdot \sum_{s=1}^{US} \delta_s = A \qquad (27)$$

$$\delta_s \ge \eta_s \qquad s \in NES \qquad (28)$$

(b) accessibility constraints

$$A \cdot k - a_j + x + \Delta \cdot \sum_{l=1}^{s-1} \delta_l \ge L_i - M_{js} \cdot (1 - b_{jsk})$$
$$s = 1,\dots,US;\ j = 1,\dots,N;\ k = kmin_j,\dots,kmax_j \qquad (29)$$

$$A \cdot k - a_j + x + \Delta \cdot \sum_{l=1}^{s-1} \delta_l \le R_i - M'_{js} \cdot (1 - b_{jsk})$$
$$s = 1,\dots,US;\ j = 1,\dots,N;\ k = kmin_j,\dots,kmax_j \qquad (30)$$

where:

$$M_{js} = L_{m_j} - A \cdot kmin_j + a_j - \Delta \cdot (s-1)$$

$$M'_{js} = A \cdot \left(kmax_j + 1\right) + \min(R_1 + a^1_{\min}, A - \Delta) - R_{m_j} - a_j$$

(c) task assignment constraints

$$\sum_{s=1}^{US} \sum_{k=kmin_j}^{kmax_j} b_{jsk} = 1 \qquad j = 1,\dots,N \qquad (31)$$

$$\sum_{j \in J_i} p_j \cdot \sum_{k=kmin_j}^{kmax_j} b_{jsk} \le C_s \qquad i = 1,\dots,m;\ s = 1,\dots,US \qquad (32)$$

$$\sum_{j \in J_i} \sum_{k=kmin_j}^{kmax_j} b_{jsk} \le |J_i| \cdot \eta_s \qquad i = 1,\dots,m;\ s \in NES \qquad (33)$$

$$\sum_{j=1}^{N} \sum_{k=kmin_j}^{kmax_j} b_{jsk} \ge \eta_s \qquad s \in NES \qquad (34)$$

$$\sum_{j=1}^{N}\sum_{k=kmin_j}^{kmax_j} b_{jsk} \geq 1 \qquad s \in ES \qquad (35)$$

(d) TS constraints

$$|ES| + \sum_{s\in NES} \eta_s \leq Tabu_h^S - 1 + M_{h1}\cdot y_{h1} \qquad h=1,\dots,TT \qquad (36)$$

$$|ES| + \sum_{s\in NES} \eta_s \geq Tabu_h^S + 1 - M_{h2}\cdot y_{h2} \qquad h=1,\dots,TT \qquad (37)$$

$$x \leq Tabu_h^x - 1 + M_{h3}\cdot y_{h3} \qquad h=1,\dots,TT \qquad (38)$$

$$x \geq Tabu_h^x + 1 - M_{h4}\cdot y_{h4} \qquad h=1,\dots,TT \qquad (39)$$

$$\sum_{g=1}^{4} y_{hg} \leq 3 + w \qquad h=1,\dots,TT \qquad (40)$$

$$T\cdot\left(|ES| + \sum_{s\in NES}\eta_s\right) + \sum_{s=1}^{US} C_s + 1 \leq CT^* + M_1^A\cdot(1-w) \qquad (41)$$

$$T\cdot\left(|ES| + \sum_{s\in NES}\eta_s\right) + \sum_{s=1}^{US} C_s \geq CT^* - M_2^A\cdot w \qquad (42)$$

where $(h=1,\dots,TT)$:

$$M_{h1} = UB^S - Tabu_h^S + 1$$

$$M_{h2} = Tabu_h^S + 1 - LB^S$$

$$M_{h3} = \min\left(R_1 + a_1^{\min}, A - \Delta\right) - Tabu_h^x + 1$$

$$M_{h4} = Tabu_h^x + 1$$

$$M_1^A = UB^{CT} + 1 - LB^{CT}$$

$$M_2^A = CT^* - LB^{CT}$$

$$\delta_s \leq \hat{\delta}_s - 1 + M_{s1}^{\delta}\cdot v_{s1} \qquad s=1,\dots,\hat{S} \qquad (43)$$

$$\delta_s \geq \hat{\delta}_s + 1 - M_{s2}^{\delta}\cdot v_{s2} \qquad s=1,\dots,\hat{S} \qquad (44)$$

$$x \leq \hat{x} - 1 + M_1^x\cdot u_1 \qquad (45)$$

$$x \geq \hat{x} + 1 - M_2^x\cdot u_2 \qquad (46)$$

$$u_1 + u_2 + \sum_{s=1}^{\hat{S}}\left(v_{s1} + v_{s2}\right) \leq 2\cdot\hat{S} + 1 \qquad (47)$$

where $(s=1,\dots,\hat{S})$:

$$M_{s1}^{\delta} = \min\left(\hat{\delta}_s + R^{\delta}, (A/\Delta) - LB^S + 1\right) - \hat{\delta}_s + 1$$

$$M_{s2}^{\delta} = \hat{\delta}_s + 1 - \max\left(\hat{\delta}_s - R^{\delta}, 0\right)$$

$$M_1^x = \min\left(R_1 + a_{\min}^1, A - \Delta\right) - \hat{x} + 1$$

$$M_2^x = \hat{x} + 1$$

The model captures the following features: The objective (19) is the minimization of the cycle time. Constraint (20) introduces a lower bound on the value of the objective function; (21) and (22) lower and upper bound, respectively, the number of the existing forward steps; (23-26) define the corridor, (27) states that the distance covered in the forward steps of a cycle corresponds to the distance between the right borders of two consecutive workpieces on the line; (28) forbid null forward steps by imposing that, if the number of elementary steps is zero, then the associated forward step *s* does not exist; constraints (29)-(30) guarantee that each task is accessible from the only station that is able to perform it, during the stationary stage in which the task will be executed; (31) impose that each task is assigned to one, and only one, stationary stage; (32), that the time corresponding to the stationary stages is not less than the processing time at any station; (33) avoid assigning a task to a non-existing stationary stage; (34)-(35) force that at least one task has to be assigned to each stationary stage; Constraints (36)-(40) represent the tabu constraints, in such a way that if the current movement scheme has the *h*-th tabu attribute, all the associated binary variables y_{hg} will have value 1. Such constraints prevent moving to a solution that is marked tabu and is not allowed by the aspiration level; (41)-(42) express the aspiration criterion, so that the binary variable *w* has value 1 if and only if the solution fulfills the aspiration criterion; finally, constraints (43)-(47) remove the current movement scheme from the solution space of the mathematical model.

7. Combinations of hybrid metaheuristics and MILP

To further improve the quality of the solution of the AWALBP-L2, we propose combining the use of the MILP *Solve model* of Calleja *et al*. (2014) with the afore-presented hybrid metaheuristics. This model requires an initial solution in order to compute bounds. We generate an initial solution by using the *Initial solution matheuristic*, and next we search for an improving solution by combining the use of the *Solve model* and the hybrid metaheuristic in two alternative ways: i) using the *Solve model* with the initial solution obtained with the *Initial solution matheuristic* and then trying to improve the solution obtained by the model using one of the proposed hybrids. Or ii) executing one hybrid and then using the obtained solution as the initial solution for the *Solve model*.

Table 3. Combinations of the proposed hybrids and MILP.

Initial solution matheuristic	+	*Solve model* 900 s 1800 s 2700 s	+	SA-, TS-, or TS-CM- hybrid
		SA-, TS-, or TS-CM- hybrid	+	*Solve model* 900 s 1800 s 2700 s

Table 3 illustrates the considered combinations of hybrids and MILP. The two rows show, respectively, the two combinations types considered. In the first row MILP is applied before the hybrid, whereas in the second row MILP is applied after. In each combination type, we consider limiting the running time of the model to 900, 1800 or

2700 s, whereas the *Initial solution matheuristic* and the hybrid are executed in the remaining run time until a 3600 s total limit is reached. In any case, the proposed combined approach stops before the limit time if a solution with an objective function value equal to the lower bound on the cycle time, $LB1^{CT}$, is found.

8. Computational results

We present comparative results for the proposed hybrid metaheuristics and the existing results in the literature, namely the approach using the *Initial solution matheuristic* and the *Solve model* of Calleja *et al*. (2014). We aim to examine, in particular, the effectiveness of the proposed hybrids in finding high-quality solutions for large instances of the problem.

The hybrid procedures were implemented in Java and the mathematical models were solved using IBM ILOG CPLEX 12.2. The absolute optimality gap was set to $1\text{-}10^{-6}$ since, without loss of generality, all data are integers and thus the objective function value is also an integer. Experiments were performed in Intel Core 3.33 GHz workstations with 4 GB of RAM operating under Windows-7 (64 bits).

The performance of the proposed hybrids was tested on the same set of 1200 problem instances as in Calleja *et al*. (2013, 2014). These benchmarking instances can be downloaded from https://www.ioc.upc.edu/EOLI/research/. The instances are based on the features arising from industrial applications. According to the description of real-world test cases given by Gaudlitz (2004) and Tazari (2006) and to the technical specifications of this type of line (http://www.icco.ro/files/icco-emt/328.pdf), the following features are commonly found. The workpiece length is usually up to 2.5 times larger than the workstation, the number of workstations may range from 7 to 20 and the number of tasks may vary from 100 to 940. In our set of instances the lengths of the workpieces are from 1.1 to 4 times larger than the width of the workstations, the number of workstations ranges from 5 to 40, and the number of tasks ranges from 50 to 1000.

With respect to the difficulty in solving the problem, the most influential parameters are A_0, m and N, where A_0 is the workpiece length, m is the number of workstations in the assembly line and N is the number of tasks. For this reason, in the considered instances the mentioned parameters are distributed along the following ranges: A_0={11-15, 16-20, 21-25, 26-30, 31-35, 36-40}, m = {5-10, 11-20, 21-30, 31-40} and N={50-200, 201-400, 401-600, 601-800, 801-1000}. Additionally, the instances have the following characteristics. The width of the accessibility windows is 10 length units (lu) and the length of the elementary step Δ is 1 lu. The time T is 200 time units (tu). The processing time of tasks was randomly generated between 100 and 150 tu. The positions of tasks were also randomly generated along the workpiece length A_0. The distance between two consecutive workpieces in the line is 1 lu and thus $A = A_0 + 1$.

In order to test the quality of the proposed approaches, we carried out the following experiments. Firstly, each hybrid metaheuristic was tested alone. In the remaining of this section, we denote by *SA*, *TS* and *TS-CM* the hybrid metaheuristics based on simulated annealing, tabu search and tabu search with corridor method, respectively. Secondly, the combinations of hybrids and the MILP *Solve model* were also tested. By $MILP_{\text{time}}$+*Hybrid* we denote the combination type where the *Solve model* is executed before the hybrid, with $MILP_{\text{time}} \in \{900, 1800, 2700\}$ seconds and $Hybrid \in \{SA, TS, TS\text{-}CM\}$. Accordingly, *Hybrid*+$MILP_{\text{time}}$ denotes the combination where the model is executed after the hybrid. All experiments are compared with respect to be best existing approach in the literature obtained in Calleja *et al*. (2014).

The values of the algorithmic parameters used in the implementation of the hybrids *SA* and *TS* were set based on computational experiments applying CALIBRA (Adenso-Díaz and Laguna, 2006), a systematic procedure used in the literature to find the best parameter values associated with heuristic or metaheuristic algorithms. Since the small to medium-size instances of AWALBP-L2 are not very sensitive to values of algorithmic parameters, we generated a training set of 48 large to very large-scale instances. The set was created by generating 2 instances for each of the 24 combinations of the following ranges: A_0 = {31-35, 36-40}, m = {5-10, 11-20, 21-30, 31-40}, and N= {401-600, 601-800, 801-1000}. The obtained parameters values are the following. As for *SA*, the values of the α, *itt* and t_0 parameters are 0.9875, 1400 and 115, respectively, and the probabilities associated to selection from neighborhoods N_1, N_2 and N_3 are 0.75, 0.1 and 0.15, respectively. As for *TS*, the tabu tenures are 24 for the tabu list associated to neighborhoods N_1 or N_2, and 8 for the tabu list associated to neighborhood N_3.

A preliminary test was carried out to examine the influence of the parameters of *TS-CM* on different instances sizes of the data set, being the very large instances the most sensitive. Therefore, a set containing the 20 largest instances of the problem was used to test the performance of the three corridor structures *C1*, *C2* and *C3* proposed in Section 6.4, for different values of the tabu tenure (5, 10, 20) and corridor widths $\left(R^{\delta} = 1, 3, 5;\ R^{S} = 1, 2;\ R^{x} = 1, 2\right)$. Thus 9 combinations of values were tested for *C1*, 18 for *C2* and 36 for *C3*, and *C3* provided the best performance in terms of the improvement of the objective function with respect to the initial solution. We therefore select the combination of *C3* that yielded the best results to be used in the *TS-CM* hybrid. Such combination has the following parameters values: the tabu tenure is 5 and the corridor widths around the forward steps, the number of stationary stages and the initial shift are, respectively, $R^{\delta} = 3$, $R^{S} = 1$ and $R^{x} = 1$. Finally, we set the run time limit of a *TS-CM* iteration to 300 s.

Among the 1200 initial solutions obtained with the method proposed in Calleja *et al*. (2014), 457 initial solutions (38.08%) yielded an objective function value coincident with the computed lower bound on the cycle time, ($LB1^{CT}$), and thus were certified as optimal solutions. We therefore focus on the comparative results for the remaining 743 instances.

Table 4 displays comparative results for the proposed hybrids with respect to Calleja *et al*. (2014) on the 743 instances considered. Among these 743 instances, we know 519 optimal solutions obtained with all the methods tested so far (including the ones presented in this paper), which are used for optimality verification in the proposed methods. In the table, the results are grouped in four main rows. The first row shows the results corresponding to the best existing method of the literature (the *Solve model* of Calleja *et al*. (2014)) and the remaining rows, the results for *SA*, *TS*, *TS-CM* and their combinations with the *Solve model*. For each experiment, in the first column (*% optima/Method*) we provide the percentage of instances that were certified optimal by the method. The second column (*% optima/Known optimal solutions*) provides the percentage of instances that were certified optimal by comparing the obtained solution with the known optimal solutions. Finally, the third column (*% ave. GAP*) gives the average relative gap of the 743 instances considered, with respect to the best lower bound available. The relative gap is defined to be $((|BF|-|BB|)/|BF|)\cdot 100$, where the best found value *BF* is the objective function value of the solution found by the procedure and the best bound value *BB* is the best bound known on the instance's solution. The value of *BB* is the maximum value among the following: i) the best bound computed by CPLEX among the models of (Calleja *et al*. 2013, 2014), ii) the theoretical lower bound on the cycle time, $LB1^{CT}$, proposed in (Calleja *et al*. 2014) and iii) the best bound computed by CPLEX among the *Hybrid+MILP*$_{\text{time}}$ experiments.

What can easily be inferred from these results is that, in terms of percentage of optimal solutions, a better performance is obtained when the proposed hybrids are combined with MILP than when executed alone (in all cases for *SA* and *TS*, and in 10 out of 12 cases for *TS-CM*). Specifically, the overall best optimality percentage was found using the combination *TS+MILP*$_{2700}$ (67.03% for optima certified by the method itself) and *MILP*$_{2700}$*+SA* (69.45% for optima certified by comparison with the known optima). On the other hand, a better relative gap percentage is obtained for six different procedures (1.83%; among these three procedures *MILP*$_{900}$+ *TS-CM* provided the best result - 69.31- in terms of % of optimal solutions).

Table 4. Average results for the proposed experiments.

	% optima		*% ave. GAP*
	Method	***Known optimal solutions***	
Calleja et al. (2014)	63.66	65.68	2.56
SA	31.22	56.53	2.99
SA + MILP$_{900}$	62.72	66.89	2.31
SA + MILP$_{1800}$	64.47	67.70	2.25
SA + MILP$_{2700}$	66.22	67.83	2.20
MILP$_{900}$ ***+ SA***	61.78	68.37	2.12
MILP$_{1800}$ ***+ SA***	63.80	68.64	2.05
MILP$_{2700}$ ***+ SA***	65.41	69.45	1.99
TS	30.82	67.29	1.88
TS + MILP$_{900}$	64.74	68.78	1.83
TS + MILP$_{1800}$	65.81	68.78	1.83
TS + MILP$_{2700}$	67.03	68.91	1.83
MILP$_{900}$ ***+ TS***	61.78	68.78	1.87
MILP$_{1800}$ ***+ TS***	63.93	69.04	1.88
MILP$_{2700}$ ***+ TS***	65.01	68.78	1.92
TS-CM	30.96	68.78	1.83
TS-CM + MILP$_{900}$	64.60	69.18	1.83
TS-CM + MILP$_{1800}$	65.55	69.18	1.86
TS-CM + MILP$_{2700}$	65.95	68.64	1.98
MILP$_{900}$ ***+ TS-CM***	62.05	69.31	1.83
MILP$_{1800}$ ***+ TS-CM***	63.93	68.91	1.88
MILP$_{2700}$ ***+ TS-CM***	64.47	68.24	2.01

We focus on the best results obtained in terms of percentage of optimal solutions certified with known optima (*MILP*$_{2700}$+*SA*) and average relative gap (*MILP*$_{900}$+*TS-CM*). Table 5 summarizes the most relevant results for *MILP*$_{2700}$+*SA* (column 1) and for *MILP*$_{900}$+*TS-CM* (column 2) compared to the best existing results of Calleja *et al*. (2014) (column 3). The first row (*% equal CT*) shows the percentage of solutions which provided the same objective function value as in Calleja *et al*. (2014). Row 2 (*% improvement*) shows the percentage of instances that outperform the solution of Calleja *et al*. (2014). Rows 3 (*ave.*) and 4 (*max.*) show, respectively, the average and the maximum improvement among such instances. Conversely, the percentage of solutions that worsen the objective is given in row 5 *(% decrease*), and its average and maximum worsening values are shown in rows 6 (*ave.*) and 7 (*max.*), respectively. In both experiments, results show a high percentage of instances that equal (around 75%) or improve (around 25%) the objective function value of Calleja *et al*. (2014), whereas the percentage of instances that worsen the objective function value is kept low (between 0.67% and 1.07% for *MILP*$_{2700}$+*SA* and for *MILP*$_{900}$+*TS-CM*, respectively). In row 8 we examine the average gap (*% ave. GAP*) of the 743 instances considered with respect to the best lower bound available, which decreased from 2.56% to 1.99% in *MILP*$_{2700}$+*SA* and to 1.83% in *MILP*$_{900}$+*TS-CM*. Additionally, in row 9 we examine the maximum gap (*% max. GAP*), which remained equal in *MILP*$_{2700}$+*SA* but decreased to 22.05% in *MILP*$_{900}$+*TS-CM*. Finally, the percentage of optimal solutions, obtained by comparison with known optima (within the 743 instances considered) is shown in row 10 (*% total optima*), which increased from 65.68% to 69.31% in *MILP*$_{900}$+*TS-CM* and rose to 69.45% in *MILP*$_{2700}$+*SA*.

Table 5. Computational results for MILP$_{2700}$ +SA and MILP$_{900}$+TS-CM with respect to Calleja et al. (2014).

	MILP*$_{2700}$ *+ SA	***MILP*$_{900}$*+TS-CM***	***Calleja et al. (2014)***
% equal CT	76.18	73.76	-
% improvement	23.15	25.17	-
ave.	2.69	3.15	-
max.	10.41	11.81	-
% decrease	0.67	1.07	-
ave.	1.87	1.39	-
max.	3.51	4.03	-
% ave. GAP	1.99	1.83	2.56
% max. GAP	23.12	22.05	23.12
% total optima	69.45	69.31	65.68

In order to assess the overall solution of the AWALBP-L2, we compare the results of the complete set of 1200 instances with respect to those obtained in Calleja *et al*. (2014). Specifically, the percentage of optimal solutions rose from 78.75% to 81.08% in *MILP*$_{2700}$+*SA*, and to 81.00% in *MILP*$_{900}$+*TS-CM*.

If we consider the optima obtained among all the proposed methods to date, we obtain that, the problem has been solved optimally for 81.33% of the instances.

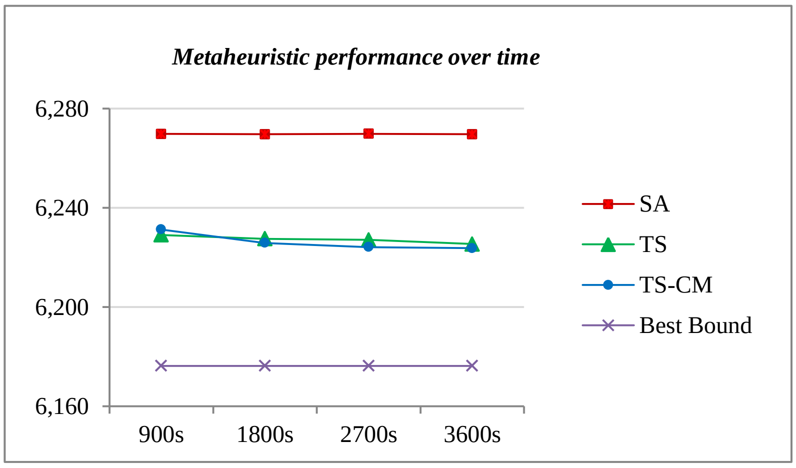

Figure 6. *Objective function values according to increasing computational time-limit*

Finally, we study the performance of the proposed metaheuristics over time. Fig. 6 depicts the obtained values of the objective function for the proposed metaheuristics (*SA*, *TS* and *TS-CM*) over increasing time-limit (900 s, 1800 s, 2700 s and 3600 s). Results show that all heuristics converge very fast to a certain suboptimal value or, more precisely, that all heuristics improve only by a negligible amount after 900 s. More specifically, the gap is reduced over time by 0.002, 0.06 and 0.12% for *SA*, *TS* and *TS-CM*, respectively. Consequently, it seems reasonable to limit the computational time for the metaheuristic to a few minutes and dedicate the remaining time to the MILP model.

9. Conclusions and perspectives

In this paper, we have presented three hybrid metaheuristics, based on simulated annealing, tabu search and tabu search with corridor method, to solve the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2) for the case where each task can only be performed in one workstation.

The proposed hybrids use a mathematical model in a metaheuristic frame. More precisely, the proposed hybrids follow a metaheuristic mechanism to guide the search and iteratively use an embedded mathematical model. While the hybrid SA and TS metaheuristics deploy move-based neighborhoods, the hybrid TS-CM features neighborhoods that are constructed within the mathematical model used to explore them. We have presented a hybrid metaheuristic where a tabu search is used to guide a MILP model over reduced portions of the original solution space. Borrowing the basic idea of the Corridor Method, such portions are defined by building corridors around a current solution, via the imposition of exogenous constraints. The resulting constrained version of the problem is then solved with the MILP model. To the best of our knowledge, this is the first time in the literature that such TS-CM hybridization is presented.

The performance of the proposed hybrids has been tested in an extensive computational experiment. They have been tested alone and in combination with a bounded mathematical programming model. The best result, in terms of percentage solutions certified with known optima, was obtained for a combination where the model is executed first and then the obtained solution is tried to be improved by a SA. Such alternative currently stands as the best method proposed for the AWALBP-L2.

The fundamental ideas on which the proposed hybrids are inspired are open in nature and extend interesting perspectives in combining mathematical programming with a metaheuristic framework, either for improving the solutions of the problem presented here or for tackling other combinatorial problems.

References

Adenso-Díaz, B, Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54 (1), 99-114.

Battaïa, O., Dolgui, A. (2012). Reduction approaches for a generalized assembly line balancing problem. *Computers and Operations Research*, 39 (10), 2337-2345.

Battaïa, O., Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, 142 (2), 259-277.

Baybars, I. (1986). A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, 32 (8), 909-932.

Blum, C., Puchinger, J., Raidl, G., Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11 (6), 4135-4151.

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

Boschetti, M., Maniezzo, V., Roffilli, M., Bolufé, A. (2009). Matheuristics: Optimization, Simulation and Control. In: *Lecture Notes in Computer Science*, 5818, 171-177. *Hybrid Metaheuristics - 6th International Workshop, HM 2009, Proceedings*.

Calleja, G., Corominas, A., García-Villoria, A., Pastor, R. (2013). A MILP model for the Accessibility Windows Assembly Line Balancing Problem (AWALBP). *International Journal of Production Research*, 51 (12), 3549-3560.

Calleja, G., Corominas, A., García-Villoria, A., Pastor, R. (2014). Combining matheuristics and MILP for the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2). *Computers and Operations Research*, 48, 113-123.

Capacho, L., Pastor, R., Dolgui, A., Guschinskaya, O. (2009). An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem. *Journal of Heuristics*, 1 (2), 109-132.

Corominas, A., Pastor, R. (2009). A MILP model for the Visibility Windows Assembly Line Balancing Problem (VWALBP): the case of the Müller-Hannemann & Weihe problem. Technical report. Universitat Politècnica de Catalunya. Available from: http://upcommons.upc.edu/e-prints/bitstream/2117/7047/1/IOC-DT-P-2009-09.pdf.

Corominas, A., Ferrer, L., Pastor, R. (2011). Assembly line balancing: general resource-constrained case. *International Journal of Production Research*, 49 (12), 3527-3542.

Downsland, K.A., Adenso-Díaz, B. (2003). Heuristic design and fundamentals of the Simulated Annealing. *Inteligencia Artificial*, 19, 93-102.

Gaudlitz, R. (2004). Optimization algorithms for complex mounting machines in PC board manufacturing. Technical University of Darmstadt, Germany. Diploma thesis.

Gendreau, M. (2003). An introduction to Tabu Search. Chapter 2 in Handbook of metaheuristics, Eds. Glover & Kochenberger, Kluwer Academic Publishers, 37-54.

Gendreau, M., Potvin, J. Y. (2005). *Tabu Search.* Chapter 6 in *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques*. Eds. Burke and Kendall, Kluwer Academic Publishers, 165-186.

Glover, F. (1986). Future paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, 13 (5), 533-549.

Glover, F. (1989). Tabu Search – part I. *ORSA Journal on Computing*, 1 (3), 190-206.

Glover, F. (1990). Tabu search – part II. *ORSA Journal on Computing*, 2 (1), 4-32.

Glover, F. (1997). Tabu Search and Adaptive Memory Programming – Advances, Applications and Challenges. Chapter 1 in Interfaces in Computer Science and Operations Research. Eds. R.S.Barr, R.V.Helgason, and J.L. Kennington, Kluwer, 1-75.

Henderson, D., Jacobson, S.H., Johnson, A.W. (2003). The Theory and Practice od Simulated Annealing. Chapter 10 in *Handbook of Metaheuristics*, Eds. Glover and Kochenberger, Kluwer Academic Publishers, 287-319.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. (1983). Optimization by Simulated Annealing. *Science*, 220 (4598), 671-680.

Kulturel-Konak, S. (2012). A linear programming embedded probabilistic tabu search for the unequal-area facility layout problem with flexible bays. *European Journal of Operational Research*, 223 (3), 614-625.

Lapierre, S.D., Ruiz, A., Soriano, P. (2006). Balancing assembly lines with tabu search. *European Journal of Operational Research*, 168 (3), 826-837.

Maniezzo, V., Stützle, T., Voβ, S. (2009). Matheuristics: Hybridizing metaheuristics and mathematical programming. *Annals of Information Systems*, 10, Springer.

Martino, L., Pastor, R. (2010). Heuristic procedures for solving the general assembly line balancing problem with setups. *International Journal of Production Research*, 48 (6), 1787-1804.

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. *European Journal of Operational Research*, 236 (2), 403-409.

Müller-Hannemann, M., Weihe, K. (2006). Moving policies in cyclic assembly line scheduling. *Theoretical Computer Science*, 351 (3), 425-436.

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

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 (1), 32-42.

Pedersen, M.B., Crainic, T.G., Madsen, O.B.G. (2009). Models and Tabu Search Metaheuristics for Service Network Design with Asset-Balance Requirements. *Transportation Science*, 43 (2), 158-177.

Puchinger, J., Raidl, G. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. *Artificial Intelligence and*

Knowledge Engineering Applications: A Bionspired Approach in Lecture Notes in Computer Science, 3562, 41-53.

Sniedovich, M., Voβ, S. (2006). The corridor method: a dynamic programming inspired metaheuristic. *Control and Cybernetics*, 35 (3), 551-578.

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 (3), 740-754.

Suman, B., Kumar, P. (2006). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, 57 (10), 1143-1160.

Tazari, S., Müller-Hannemann, M., Weihe, K. (2006). Workload Balancing in Multi-stage Production Processes, *WEA 2006*, LNCS 4007, 49-60, Springer.

Tuncel, G., Topaloglu, S. (2013). Assembly line balancing with positional constraints, task assignment restrictions and station paralleling: A case in an electronics company. *Computers & Industrial Engineering*, 64 (2), 602-609.

Vilà, M., Pereira, J. An enumeration procedure for the assembly line balancing problem based on branching by non-decreasing idle time. *European Journal of Operational Research*, 229 (1), 106-113, 2013.

Woodcock, A.J., Wilson, J.M. (2010). A hybrid tabu search/branch & bound approach to solving the generalized assignment problem. *European Journal of Operational Research*, 207 (2), 566-578.