Computers & Operations Research 54 (2015) 64–73

Contents lists available at ScienceDirect

Computers & Operations Research

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

Model and heuristics for the Assembly Line Worker Integration and Balancing Problem

Mayron César O. Moreira [a,*], Cristóbal Miralles [b], Alysson M. Costa [c]

[a] Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Brazil
[b] ROGLE - Departamento Organización de Empresas, Universitat Politècnica de València, Spain
[c] Department of Mathematics and Statistics, University of Melbourne, Australia

ARTICLE INFO

Available online 6 September 2014

Keywords:
Assembly line balancing
Disabled workers
Mathematical modeling

ABSTRACT

We propose the Assembly Line Worker Integration and Balancing Problem (ALWIBP), a new assembly line balancing problem arising in lines with conventional and disabled workers. The goal of this problem is to maintain high productivity levels by minimizing the number of workstations needed to reach a given output, while integrating in the assembly line a number of disabled workers. Being able to efficiently manage a heterogeneous workforce is especially important in the current social context where companies are urged to integrate workers with different profiles. In this paper we present mathematical models and heuristic methodologies that can help assembly line managers to cope with this additional complexity. We demonstrate by means of a robust benchmark how this integration can be done with losses of productivity that are much lower than expected.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

According to the International Labour Organization (ILO), people with disabilities represent an estimated 10% of the world's population, including approximately 500 million of working age; being apparent that in the unemployment rates of the disabled are much higher than the average.

Employment is the main path for social inclusion and participation in modern societies. Having a job is not only the basis for the survival and stability for many individuals, but also a key way of accessing many rights as citizens. Therefore the welfare and the social inclusion of the disabled depend very much on the degree of labor integration they are able to achieve. Different active policies to fight against discrimination have been set during the last few decades, following models that are more/less inclusive depending on the local culture. Across specific national legislations, a general common formula is to reserve a share of workplaces in ordinary companies for people with disabilities. This share normally increases with the size of the company and, depending on the country legislation, usually goes from 2% to 5% of the jobs.

Unfortunately, it is also a common phenomenon in many countries that this share is not always respected, indicating that the solution should come not only by legal imposition, but mainly by overcoming the prejudices about the capabilities of the disabled, and by the genuine commitment of ordinary companies to include integration programs in their strategies. The aim of this paper is to contribute in making this commitment easier: (1) by providing the production managers with practical approaches that ease the integration of disabled workers in the production lines; (2) by demonstrating that, through the approaches proposed, the productivity of production systems suffers little (and often none) decrease.

Once stated the great importance of integrating Disabled into the workforce of ordinary companies, we should make a brief introduction on some previous work inspired on the specific scenario of the so-called “Sheltered Work Centers for Disabled” (henceforth SWDs). SWDs are a special work formula legislated in many countries (with different variants) whose only difference from an ordinary company is that most of its workers must be disabled, and therefore they receive some institutional help in order to be able to compete in real markets. This labor integration formula has been successful in decreasing the former high unemployment rates of countries like Spain, and one of the strategies used by SWDs to facilitate the labor integration has been the adoption of assembly lines. In this sense, Miralles et al. [9] were the first to evidence how the integration of disabled workers in the productive systems can be done without losing, even gaining, productive efficiency through the use of assembly lines. This pioneer reference defined the so-called Assembly Line Worker Assignment and Balancing Problem (ALWABP) and demonstrated how the division of worker into single

* Corresponding author.
E-mail addresses: mayron@icmc.usp.br (M.C.O. Moreira), cmiralles@omp.upv.es (C. Miralles), alysson.costa@unimelb.edu.au (A.M. Costa).

http://dx.doi.org/10.1016/j.cor.2014.08.021
0305-0548/© 2014 Elsevier Ltd. All rights reserved.

tasks becomes a powerful tool for making certain workers disabilities invisible.

1.1. Contribution and outline of this work

ALWABP was inspired in the SWDs reality, where the very high diversity of most of the workers and their limitations are the main characteristics. This scenario is quite different to that one of an ordinary company, where the aim is to efficiently integrate in the workforce just some workers, often to cope with the 2–5% of disabled workers legislation requirements. In this case the problem supposes much less diversity in the input data, and can also be stated with very different approaches with respect to the ALWABP, regarding the objective function, the hypothesis and model defined, and the kind of appropriate solution procedures.

The aim of this paper is to introduce and analyze this new problem that has been named "Assembly Line Worker Integration and Balancing Problem" (ALWIBP). Our study aims to answer specific requirements that normally arise in assembly lines of ordinary companies, where only few disabled workers have to be integrated, providing the production managers with practical tools that ease the integration of disabled workers in the most efficient manner. We propose new mathematical models for the problem as well as a constructive heuristic based on the similarities between the proposed problem and the so-called Simple Assembly Line Balancing Problem (SALBP).

The remainder of this paper is structured as follows: in Section 2, we state a formal codification of the new problem and some extensions, analyzing their practical implications and reviewing references of the literature with useful related approaches. Section 3 then presents the corresponding IP models for the proposed versions of the ALWIBP while Section 4 describes a fast heuristic that has been developed to solve the problem. A experimental study in order to analyze the effectiveness of the proposed models and algorithms is conducted in Section 5. General conclusions end this paper.

2. The Assembly Line Worker Integration and Balancing Problem

2.1. Introduction: SALBP vs ALWABP

The SALBP was initially reviewed by Baybars [1] and consists of an assembly line balancing problem with several well-known simplifying hypotheses. This classical single-model problem which aims at finding the best feasible assignment of tasks to stations so that certain precedence constraints are fulfilled, has been the reference problem in the literature in its two basic versions: when the cycle time C is given, and the objective is to optimize the number of necessary workstations, the problem is called SALBP-1. Whereas when there is a given number m of workstations, and the goal is to minimize the cycle time C the literature knows this second version as SALBP-2 [17].

A trend in Assembly Line research in the last decade has been to narrow the gap between the theoretical proposals and the industrial reality, which faces multiple specific configurations such as multi-manned workstations [6,7], two sided assembly lines [8,14], or operator allocation in job sharing and operator revisiting lines [20], among many others. As part of this trend, Miralles et al. [9] properly defined the ALWABP, a generalization of the SALBP where, in addition to the assignment of tasks to stations, a set of heterogeneous workers also has to be assigned to stations. In this scenario each task has a worker-dependent processing time, which allows taking into account the limitations and specific production rates of each worker. Moreover, when the time to execute a task for certain worker is very high, this assignment is considered infeasible in the input data matrix.

Since Miralles et al. [9], many other references have contributed to give ALWABP visibility throughout academia, proposing different methods to solve the problem. The same authors have later developed a branch-and-bound algorithm for the problem, obtaining the exact solution of small-sized instances [10]. Because of the problem complexity and the need to solve larger instances, the literature has since then shifted its efforts to heuristic methods. The current state-of-the-art methods for solving the ALWABP are the iterated beam search (IBS) metaheuristic of Blum and Miralles [2], the biased random-key genetic algorithm of Moreira et al. [11], the iterative genetic algorithm of Mutlu et al. [12], the heuristic and the branch-and-bound algorithms of Borba and Ritt [3] and the branch-and-bound algorithm of Vilà and Pereira [19].

2.2. ALWIBP

The ALWABP problem was inspired in the SWDs reality with most workers presenting a high diversity of operation times; whereas the ALWIBP scenario introduced in Section 1 intends to simulate the more inclusive situation in which disabled workers relative (in a small number) are integrated in a conventional assembly line. It has to be noted that the main (and only studied) problem focusing on disabled integration in assembly lines has been the ALWABP-2 [10,11] e.g., since the typical objective at SWD is to be as efficient as possible with the (diverse) available workforce.

In the scenario associated with the ALWIBP, it makes sense to deal with the type 1 problem, since a reasonable aim of a production manager can be to integrate the given disabled workers (in some cases some 2 or 5% of workers, or even more whether some compensation is needed due to low shares in other factory sections) while minimizing the number of additional workstations needed for doing so. This problem is named ALWIBP-1, by analogy with the SALBP case.

In addition to this basic objective, once inside the solution subspace with minimal number of workstations, the manager may aim to find those assignments in which the idle time in stations with disabled workers is minimum, in order to increase their participation in the production process. We call this extension ALWIBP-1S_{min}. If, according to Boysen et al. [4] classification, ALWABP-2 was stated as $[pa, link, cum|equip|c]$, in this case we can define ALWIBP-1 as $[pa, link, cum|equip|m]$, while the ALWIBP-1S_{min} can be stated as $[pa, link, cum|equip|m, SLL^{stat}]$ using the same codification scheme.

In the following, we propose integer linear models for the basic ALWIBP-1 situation and also for the extension proposed.

3. Mathematical models

In this section, we present a mathematical model for the ALWIBP-1 defined earlier, and further extend it to cope with ALWIBP1-S_{min} extra objective with the use of the following notation:

N	set of tasks to be assigned;
S	set of workstations;
W	set of disabled workers, $\|W\| \leq \|S\|$;
t_i	execution time of task i when assigned to a "conventional" worker;
t_{wi}	execution time of task i when assigned to disabled worker $w \in W$;
$I_w \subseteq N$	set of unfeasible tasks for worker $w \in W$;
F_i	set of immediate successors of task i;
F_i^*	set of all successors of task i.

3.1. Model for ALWIBP-1

The proposed formulation follows the idea used by Petterson and Albracht [15] when modeling the SALBP-1. Let q be an artificial task and $D_q = \{i \in N | F_i = \varnothing\}$ be the set of tasks that have no followers. We define a new precedence graph in which $N' = N \cup \{q\}$ and all tasks in D_q precede task q. The execution time of task q is always 0, i.e., $t_q = t_{wq} = 0, \forall w \in W$. The ALWIBP-1 can thus be modelled as

$$\text{Min} \sum_{s \in S} s x_{sq} \tag{1}$$

subject to

$$\sum_{s \in S} x_{si} = 1, \quad \forall i \in N', \tag{2}$$

$$\sum_{s \in S} y_{sw} = 1, \quad \forall w \in W, \tag{3}$$

$$\sum_{w \in W} y_{sw} \leq 1, \quad \forall s \in S, \tag{4}$$

$$\sum_{s \in S | s \geq k} x_{si} \leq \sum_{s \in S | s \geq k} x_{sj}, \quad \forall i,j \in N', j \in F_i, k \in S, k \neq 1, \tag{5}$$

$$\sum_{i \in N'} t_i x_{si} \leq \overline{C}, \quad \forall s \in S, \tag{6}$$

$$\sum_{i \in N' \setminus I_w} t_{wi} x_{si} \leq \overline{C} + L_w(1 - y_{sw}), \quad \forall s \in S, \forall w \in W, \tag{7}$$

$$y_{sw} \leq 1 - x_{si}, \quad \forall s \in S, \forall w \in W, \forall i \in I_w, \tag{8}$$

$$\sum_{s \in S | s \geq k} y_{sw} \leq \sum_{s \in S | s \geq k} x_{sq}, \quad \forall w \in W, \forall k \in S, k \neq 1, \tag{9}$$

$$x_{si} \in \{0,1\}, \quad \forall s \in S, \forall i \in N', \tag{10}$$

$$y_{sw} \in \{0,1\}, \quad \forall s \in S, \forall w \in W. \tag{11}$$

where:

x_{si}	binary variable equals to one if task $i \in N'$ is assigned to workstation $s \in S$,
y_{sw}	binary variable equals to one if a disabled worker $w \in W$ is assigned to workstation $s \in S$,
L_w	large constant, $w \in W$.

The objective function minimizes the index associated with the last station (the one that executes the fictitious last task q). In association with constraints (3) which state that all disabled workers are assigned, this objective function minimizes the number of "conventional" workers used in the line. Constraints (4) guarantee that each workstation receives at most one (disabled) worker. This is a fair assumption in many practical situations even for lines solely with "conventional" workers. In the case of workers with disabilities, this fact gains in importance since these workers might have special constraints which might force that the workstation be modified accordingly. Constraints (2) ensure that all tasks are assigned, while constraints (5) guarantee that the precedence relations are respected. These inequalities were proposed by Ritt and Costa [16] which analyzed several versions of precedence constraints and concluded that constraints (5) presented better theoretical and practical results.

Constraints (6) and (7) ensure that the cycle time is respected at stations without and with disabled workers, respectively. Constant L_w must be sufficiently large to deactivate these last constraints if $y_{sw} = 0$. Therefore, we take $L_w = \sum_{i \in N \setminus I_w} |t_{wi} - t_i|$. This expression assumes the maximum additional time that a disabled worker w spends at a station in comparison to a conventional worker (this would be the additional time needed for the execution of all worker-feasible tasks). Also, we highlight that even though a workstation must have only a single worker, this set of constraints allow that more than one task can be assigned to the same station.

Finally, constraints (8) and (9) guarantee that disabled workers are not assigned to tasks which they are not able to execute and that they execute at least one task, respectively.

3.2. ALWIBP-1S_{min} model extension

In order to extend this formulation to the ALWIBP-1S_{min} case, one can simply note that this new problem is characterized by the addition of another term in the objective function related to the idle time of the disabled workers. The new goal is to hierarchically minimize the number of stations (with higher priority) and the idle time of the stations with disabled workers. Thereby, this version of the problem aims to obtain more balanced solutions that increase the participation of these workers.

To model this situation, we use non-negative real variables $\delta_w, w \in W$, to measure the idle time of each disabled worker w. The values of these new variables are obtained with the aid of slack variables $l_{sw}, \forall s \in S, \forall w \in W$ associated with constraints (7), which are rewritten as

$$\sum_{i \in N' \setminus I_w} t_{wi} \cdot x_{si} + l_{sw} = \overline{C} + L_w(1 - y_{sw}), \quad \forall s \in S, \forall w \in W, \tag{12}$$

and with new constraints which are added to establish the correct relations between δ_w and the slack variables:

$$\delta_w \geq l_{sw} - (1 - y_{sw}) \cdot \left(L_w + \sum_{i \in N'} t_i \right), \quad \forall s \in S, \forall w \in W. \tag{13}$$

Variables l_{sw} measure the slack for constraints (12) obtained with each pair $s \in S, w \in W$, while constraints (13) define δ_w as the idle time of worker w, which is obtained with the value of l_{sw} in the constraint corresponding to the actual assignment y_{sw}.

The objective function can now include terms associated with these idle times, and the ALWIBP-1S_{min} can be written as

$$\text{Min} \sum_{s \in S} s x_{sq} + \sum_{w \in W} \frac{\delta_w}{\overline{C} |W|} \tag{14}$$

subject to

(2)–(6), (8)–(13),

$$l_{sw} \in \mathbb{R}_+, \quad \forall s \in S, \forall w \in W, \tag{15}$$

$$\delta_{sw} \in \mathbb{R}_+, \quad \forall s \in S, \forall w \in W. \tag{16}$$

The constant term multiplying the idle time variables imposes a hierarchical characteristic in the objective function, giving priority to the minimization of stations and using the idle times as a secondary objective.

4. Constructive insertion heuristic

We propose a Constructive Insertion Heuristic (CIH) for the ALWIBP-1. This heuristic relies on the similarities between the ALWIBP and the classical SALBP. Indeed, during the heuristic procedures, a SALBP-1 solution is found and then iteratively adapted to incorporate the heterogeneous workers available in the ALWIBP-1. In the following section, this heuristic is presented in general terms. A formal description of the algorithm is presented in Section 4.2. Finally, some variants of the procedure and two post-optimization routines are described in (Sections (4.3) and 4.4), respectively.

4.1. General structure of the proposed heuristic

The main steps of the proposed heuristic are described in Algorithm 1.

Algorithm 1. Constructive insertion heuristic.

1: Ignore heterogeneous workers and obtain a SALBP-1 solution;
2: Divide *remaining* line in segments (containing a certain number of stations each);
3: Try all available heterogeneous workers in each station of the first line segment;
4: Select best assignment and fix the solution on stations prior to the selected one.
5: If there are still workers to assign, go to step 2; Otherwise, end.

In the following subsections, each step of the algorithm is explained in detail.

4.1.1. Step 1: obtain a SALBP-1 solution

A feasible solution for the SALBP-1, x_{ref}, is used as a starting point of the algorithm. This solution can be obtained with any heuristic approach. In this study, we use the best solution found by CPLEX 12.4 in 10 min of running time on a single thread of a 2.3 GHz Intel machine with 6 GB of memory.

4.1.2. Step 2: divide line into segments

The part of the line that has not yet been fixed is divided into $|W_a|$ segments, where W_a is the set of disabled workers which have not yet been assigned. If a single worker is available, all stations in the remaining line are considered, otherwise, we consider stations in the current segment, S_c, defined as $S_c \leftarrow \{s_b+1, \ldots, s_b+1+\lfloor (m_c-s_b)/|W_a| \rfloor\}$, where s_b is the last fixed station and m_c is the total number of stations in the current solution.

This choice of line segments follows two main rationales: the first is to have a sufficient number of stations for assigning the remaining workers with disabilities after each of these workers is assigned. The second is to somehow impose an even distribution of such workers along the line, with the goal of allowing a deeper integration between workers with and without disabilities – with the more experienced workers serving as 'monitors' for the newcomers.

4.1.3. Step 3: try all possible assignments in current segment

In this step, we try to obtain a solution by considering all assignments of workers in W_a to stations in the current analyzed line segment S_c, one at a time. When a station $s \in S_c$ is considered for the assignment of worker $w \in W_a$, the following solution is obtained: in stations prior to s, the current solution, x_{tmp}, remains fixed. From station s and on, the solution is rebuild using a constructive heuristic based on that of Scholl and Voß [18]. We call this procedure *salbp-1* (x^{tmp}, w, s). The following heuristic criteria were used to prioritize the assignment of tasks during the procedure (refer Scholl and Voß [18] for more details):

- *MaxTime*: Ascending order of task execution times, t_i;
- *MaxPW*: Descending order of positional weights, $pw_i = t_i + \sum_{j \in F_i^*} t_j$;
- *MaxIF*: Descending order of immediate followers, F_i;
- *MaxF*: Descending order of followers, F_i^*;

In case a disabled worker is being considered, the appropriate parameters t_{wi} and t_{wj} are considered in criteria MaxTime and MaxPw.

4.1.4. Step 4: select best assignment

In this step, the best solution x^* among those obtained for each assignment (w,s), with $w \in W_a$ and $s \in S_c$ is chosen. A solution x^1 obtained with an assignment of worker w_1 to station s_1 is considered better than a solution x^2 $(x^1 < x^2)$ if the number of final stations of x^1 (designed by $stations(x^1)$) is smaller than $stations(x^2)$. If a tie happens, we consider the solution with larger idle time in the last station to be the best one.

4.2. Formal structure of the proposed algorithm

A formal description of the whole implemented heuristic is presented in Algorithm 2. It uses a given SALBP-1 solution as input as described in Section 4.1.1. Line division into segments described in Section 4.1.2 is effected in lines 1–2 and in lines 20–21. The main loop of the algorithm occurs in lines 5–23: while there are workers yet to be assigned, each pair $(w \in W_a, s \in S_c)$ is tested by obtaining a complete solution with the procedure described in Section 4.1.3 (line 11) and the best one (according to the criterion presented in Section 4.1.4) is selected (lines 12–15).

Algorithm 2. Constructive Insertion Heuristic.

Require: x^{ref} (SALBP-1 reference solution);
1: $m_c \leftarrow stations(x^{ref})$;
2: $S_c \leftarrow \left\{1, \ldots, \lceil \frac{m_c}{|W_a|} \rceil\right\}$;
3: $W_a \leftarrow W$;
4: Let x^* be the incumbent solution (initially, $stations(x^*) = \infty$);
5: **while** $W_a \neq \varnothing$ **do**
6: $w_b \leftarrow \infty$;
7: $s_b \leftarrow \infty$;
8: **for all** $w \in W_a$ **do**
9: **for all** $s \in S_c$ **do**
10: Fix solutions of x^{ref} on stations prior to s in x^{tmp};
11: $x^c \leftarrow proc_salbp{-}1(x^{tmp}, w, s)$;
12: **if** $x^c < x^*$ **then**
13: $w_b \leftarrow w$;
14: $s_b \leftarrow s$;
15: $x^* \leftarrow x^c$;
16: **end if**
17: **end for**
18: **end for**
19: $W_a \leftarrow W_a \setminus \{w_b\}$;
20: $m_c \leftarrow stations(x^*)$;
21: $S_c \leftarrow \left\{s_b+1, \ldots, s_b+1+\left\lfloor \frac{m_c-s_b}{|W_a|} \right\rfloor\right\}$;
22: $x^{ref} \leftarrow x^*$;
23: **end while**
Ensure: x^* (best solution found).

4.3. Algorithmic variants

The proposed algorithm tries to assign workers from the first ordered station to the last one. A variant of this strategy might start at the last stations and proceed backwards. In this case, the line division into segments presented in Section 4.1.2 needs to be modified. Equations in line 2 and 21 of Algorithm 2 are modified for $S_c \leftarrow \{\lceil m_c/|W_a| \rceil, \ldots, m_c\}$ and $S_c \leftarrow \{\lfloor (s_b-1)/|W_a| \rfloor, \ldots, s_b-1\}$, respectively. Since in this backward approach the tasks and workers assigned in the end of the line are kept fixed, the algorithm may need to insert intermediate stations. Therefore, the tie-breaker for the procedure described in Section 4.1.4 is no

longer the idle time in the last station but the idle time in the last non-fixed station in the current solution.

In addition to changing the order the ordered stations are visited for worker assignments, one can also change the order the tasks are assigned in the procedure described in Section 4.1.3. Indeed, as usual, the precedence graph can be considered in its normal or reversed form, yielding four algorithmic variants (two worker assignment strategies combined with two task assignment strategies). Since the algorithm proposed is very fast, all four strategies are used at each run of the algorithm and the best result is kept; this increases the robustness of the algorithm with respect to the precedence graph structure.

4.4. Post-optimization routines useful for ALWIBP-1S_{min}

The solution obtained by the CIH is improved by means of a mixed-integer programming neighborhood based on the original model (1)–(11). The main idea is to reduce the computational burden needed to solve the model by fixing the worker assignment variables and allowing changes only in task assignments variables. This can be done by solving the original model (1)–(11) with the addition of the following constraints:

$$y_{s(w),w} = 1, \qquad \forall w \in W_a, \tag{17}$$

where $s(w)$ indicates the station to which worker w is assigned in the solution of the CIH. Clearly, the model can be adjusted accordingly, to eliminate useless constraints and variables.

In order to further reduce the computational effort needed to solve the model, one can reduce the freedom of task assignment variables in the MIP neighborhood, by allowing tasks to be assigned only to the same station it was assigned in the CIH solution or to neighboring stations. This can be easily done with the addition of the following constraints:

$$x_{s(i),i} + x_{s(i)-1,i} + x_{s(i)+1,i} = 1, \quad \forall i \in N, s(i) \in S \setminus \{1, m\}, \tag{18}$$

$$x_{1i} + x_{2i} = 1, \quad \forall i \in N, s(i) = 1, \tag{19}$$

$$x_{mi} + x_{m-1,i} = 1, \quad \forall i \in N, s(i) = m. \tag{20}$$

where $s(i)$ indicates the station to which task i was assigned in the CIH solution.

Additionally, the objective function of this post-optimization program can be changed to incorporate the characteristics of the ALWABP-1$S_{\min}$, in which one prioritizes solutions with low idle times for the disabled workers. In this case, it suffices to use (14) as objective function and add constraints:

$$\sum_{i \in N} p_{s(w),i} \cdot x_{si} + \delta_w = C, \quad \forall w \in W, \tag{21}$$

Note that these simpler constraints replace constraints (12)–(13) in the original ALWIBP-1$S_{\min}$ problem, since the worker assignments are already known in the post-optimization phase.

5. Experimental study

5.1. Justification of a new ALWIBP benchmark

As discussed in Section 2, the ALWABP was inspired by the situation found at SWDs where the very high diversity of workers and their limitations are the main characteristics; whereas the ALWIBP scenario simulates the "desirable" situation of only a small percentage of disabled workers being integrated in conventional assembly lines. Moreover, as stated earlier, the main (and only studied) approach in this scenario has been ALWABP-2, since the typical objective in SWD is to be as efficient as possible with the (diverse) available workforce. Instead, in this new scenario, the assembly line balancing of type 1 (minimization of the number of stations given the desired cycle time) approach becomes realistic, since the basic aim of a production manager can be to integrate the normative (common in most countries) 2–5% of disabled workers into the assembly line, while maintaining a given productivity. Many previous proposals for the ALWABP-2 were evaluated with the set of 320 benchmark instances first proposed by Chaves et al. [5]. Once stated the completely different scenario where the ALWIBP arises, it is clear that this classical ALWABP benchmark is not useful here since, as explained above: (1) only a little share of the workers are disabled; and (2) the basic aim is now to minimize the number of workstations with non-disabled workers (ALWIBP-1 perspective).

5.2. ALWIBP benchmark scheme

As many other ALB approaches, the ALWABP benchmark was constructed from the only SALBP reference (the Scholl data collection of http://www.assembly-line-balance.de), that was considered robust enough and has been extensively used to test most proposals in the literature so far. Nevertheless, as recently demonstrated by Otto et al. [13], this framework does not seem rigorous enough. The problems were collected from different empirical and not empirical sources, and are based on only 25 precedence graphs; where just 18 distinct graphs have more than 25 tasks and thus are meaningful for comparing solution methods. Moreover, Otto et al. [13] also point out the triviality of many classical instances. Therefore, they propose a SALBP generator and a new very robust challenging benchmark whose graphs morphologies include a sufficient variety of chains, bottlenecks and modules. Basically, it has different cells of data sets (with 25 different instances per cell) following a full-factorial design for the following parameters:

1. Type of the graph: precedence graphs containing more chains, more bottlenecks, or a mix of both.
2. Order Strength: "low", "medium" and "high".
3. Distribution of task times: "peak at the bottom", "bimodal" and "peak in the middle".
4. Number of tasks: "small", "medium", "large" and "very large".

Thus, we selected as basis the following collection of subsets from the Otto et al. [13] benchmark:

1. We consider that diversity of graphs is sufficiently ensured selecting only the "mixed" instances (that have both chains and bottlenecks).
2. From them, we only select the data subsets with "low" and "high" Order Strength (that would give us clearer correlations if needed).
3. And from them, regarding distribution of task times, we select the subsets with "peak at the bottom" and "bimodal" distribution (we discard the "peak in the middle" subset because the optimal SALBP solution is unknown for many of the instances).
4. Finally, we take the resulting 100 robust instances of the "medium" (with $n=50$ tasks), "large" (with $n=100$ tasks), and "very large" (with $n=1000$ tasks) subsets, and we generate three corresponding benchmarks that we use separately in our experimentation (we discard the "small" subset since it is advised for simple testing only).

In all three cases we generate the benchmark using the same procedure: for each original instance we respect the precedence network and the conventional task time, and then we generate four different instances by adding one disabled worker with: high or low variability of task time respect to the original ones, and high

Table 1
Computational results: ALWIBP-1 and ALWIBP-1S_{min} models ($|N| = 50$).

$\|W\|$	Var	Inc (%)	Δ	t(s)	$m_\uparrow \pm \sigma_{m_\uparrow}$	$m_\uparrow \pm \sigma_{m_\uparrow}$(%)	τ	$\tau_{S_{min}}$	η(%)	$\eta_{S_{min}}$(%)	$\beta \pm \sigma_\beta$(%)	θ
1	U[t, 2t]	10	95	104.0	0.2 ± 0.4	2.8 ± 5.5	149.9	1.6	66.2	87.6	10.9 ± 2.7	78
		20	95	90.6	0.2 ± 0.4	2.9 ± 5.6	131.4	3.1	62.8	85.6	10.9 ± 2.7	78
	U[t, 5t]	10	95	125.0	0.4 ± 0.5	5.0 ± 6.3	231.6	9.6	43.7	63.4	10.6 ± 2.6	59
		20	95	120.4	0.4 ± 0.5	5.4 ± 6.5	244.1	10.2	39.9	58.9	10.6 ± 2.5	57
2	U[t,2t]	10	92	177.7	0.4 ± 0.5	4.7 ± 6.5	105.9	1.9	67.9	81.6	21.3 ± 5.0	63
		20	96	85.8	0.4 ± 0.5	4.7 ± 6.3	101.3	3.5	63.0	79.9	21.3 ± 5.2	62
	U[t, 5t]	10	93	166.3	0.7 ± 0.5	8.5 ± 6.8	127.1	7.4	51.6	64.3	20.5 ± 4.8	30
		20	93	152.7	0.7 ± 0.5	8.6 ± 7.1	140.1	8.1	47.7	60.6	20.5 ± 4.8	30
3	U[t,2t]	10	91	195.7	0.5 ± 0.5	6.4 ± 6.5	72.8	2.7	73.0	82.0	31.4 ± 7.3	48
		20	89	213.6	0.6 ± 0.5	6.9 ± 6.4	76.5	3.1	69.0	79.2	31.3 ± 7.3	43
	U[t, 5t]	10	95	109.6	1.1 ± 0.5	13.3 ± 8.1	114.0	7.4	51.3	62.0	29.4 ± 6.5	8
		20	90	217.0	1.2 ± 0.5	14.1 ± 8.6	126.5	11.0	51.1	57.9	29.2 ± 6.3	6
4	U[t,2t]	10	89	222.7	0.7 ± 0.5	8.3 ± 6.8	56.5	3.9	75.9	81.3	41.2 ± 9.5	32
		20	86	301.2	0.8 ± 0.5	9.2 ± 7.5	68.0	5.0	70.5	77.7	40.8 ± 9.3	28
	U[t, 5t]	10	89	278.8	1.5 ± 0.6	17.0 ± 9.7	109.6	11.7	53.7	59.7	37.9 ± 7.8	1
		20	84	362.9	1.6 ± 0.6	18.1 ± 9.9	120.5	11.7	50.7	58.2	37.6 ± 7.9	0

or low percentage of incompatibilities. The two levels defined for the task times variability used the distributions $U[t_i, 2t_i]$ and $U[t_i, 5t_i]$ for low and high variability, and the low and high percentage of incompatibilities in the tasks-workers matrix was set to 10% and 20% approximately. Following the same scheme we created 400 additional instances with two workers, then three workers, and finally four workers.

Following this outline, we finally obtain three reliable benchmarks with the structure described, one with 1600 "medium" instances, a second one with 1600 "large" instances, and a third one with 1600 "very large" instances. It is important to note that in our source benchmark the instances are classified from "less tricky" to "extremely tricky" and it happens that, the four cells finally selected as base have a quite symmetric composition regarding this "triviality" measure.

Finally, it is also important to note that the following ALWIBP computational study has always used an input cycle time of 1000 time units simply because it is the value used by Otto et al. [13], making the results of both cases comparable.

5.3. ALWIBP computational study

Our computational study consists of three parts: in the first experiment we use the "medium" and "large" benchmarks, obtaining exact solutions for almost all the instances with the model (and extension) of Section 3. In the second experiment we use the same two benchmarks to compare the exact results with the ones obtained by the heuristic procedures. In the third experiment we use the "very large" benchmark to get further evidence of the good behavior of the heuristic against large problems that are difficult to solve exactly. Furthermore, we validate the obtained results by comparing them to those of another heuristic procedure.

5.3.1. Experiment 1: exact results for ALWIBP-1 and ALWIBP-1S_{min}

One basic aim of every company should be to integrate at least the normative percentage of disabled workers into the workforce. In this first experiment, we aim to demonstrate that the proposed methods enable the inclusion of higher percentages of disabled workers in the line without important losses in productivity. It has to be noted that productivity always means somehow (output result/input resources involved), and in assembly lines productivity can be defined as (throughput rate/number of workstations). Therefore (considering that in this experiment the Throughput rate is fixed), an increase on the number of workstations means, in general, a decrease of productivity.

Having this in mind, this first experiment was devoted to check what would be the shape of this expected loss of productivity. For the medium and large instances, both ALWIBP-1 and ALWIBP-1S_{min} models could be solved in reasonable computation times using the commercial package CPLEX 12.4 (1 *thread*, limit time of 1800 s, and 6GB as tree limit) with Intel Core i7 3.4 GHz, 16GB RAM.

In Tables 1 and 2, the columns indicate:

- **Δ**: number of instances solved to optimality;
- **t(s)**: computational time (on average);
- **$m_\uparrow$** and **$\sigma_{m_\uparrow}$**: number of additional workstations needed in the ALWIBP-1 solution, with respect to the best know solution of SALBP-1 (average and deviation);
- **$m_\uparrow$** and **$\sigma_{m_\uparrow}$(%)**: percentage of additional workstations needed in the ALWIBP-1 solution, with respect to the best know solution of SALBP-1 (average and deviation);
- **τ**: *idle time* of stations with disabled workers (on average) in ALWIBP-1;
- **τ_{Smin}**: *idle time* of stations with disabled workers (on average) in ALWIBP-1S_{min};
- **η(%)**: percentage of tasks performed by disabled workers with respect to the mean number of tasks assigned to ordinary workers in ALWIBP-1;
- **η_{Smin}(%)**: percentage of tasks performed by disabled workers with respect to the mean number of tasks assigned to ordinary workers in ALWIBP-1S_{min}.
- **β** and **σ_β(%)**: percentage of the number of disabled in the assembly line ($|W|/m$), where m is the number of stations in the ALWIBP-1 solution (average and deviation);
- **θ**: number of instances which there is no increase of the number of stations with respect to the SALBP-1 solution; no increase in the number of stations with respect to the SALBP-1 solution (average and deviation).

We analyze first the results for "middle" instances. As expected, the increase in the number of stations grows with the number of disabled workers to be integrated and with the variability of the task times. Nevertheless, it can be observed that even in the most constrained case (4 disabled workers with execution times of up to 5 times the conventional time and 20% incompatibility), an average of only 1.6 new stations had to be added to integrate the workers.

This result is even more remarkable when the size of lines is taken into consideration. Indeed, in column 12, the β metric shows us that an average increase of 0.4 workstations imply in approximately 20% of the workforce composed of disabled workers. This is, in fact, relevant in real contexts, once the legislation of many countries requires a rate of disabled workers ranging from 2% to 5% in industries.

Concerning the θ metric, we see that in 38% of the cases we obtain a line balancing for the ALWIBP-1 case with the same number of workstations of the SALBP-1 solution, resulting in the best of possible scenarios (integrate all disabled workers without increasing production costs). Furthermore, the percentage of disabled workers that take part in the workforce keep at least in 5% in these cases.

As the sub-space of possible ALWIBP-1 optimal solutions is large, we can combine this primary aim of minimizing conventional workstations with the (important) secondary objective of minimizing the idle time of disabled workers. After applying the extension ALWIBP-1S_{min} to all the instances the results show that we maintain in all cases the same rough productivity (no increase of the number of workstations), while we reach a very little mean idle time (see column $\tau_{S_{min}}$). We note that in the column related to the $\eta(\%)$ and $\eta_{Smin}(\%)$ criteria, since the increase of these values follows the reduction of idle time in all scenarios studied.

Concerning the ALWIBP models in the "large" instances (with 100 tasks), Table 2 has shown that the obtention of optimal solutions becomes difficult when 3 and 4 disabled workers must be inserted in assembly lines (column Δ). However, the number of additional workstations is smaller than we notice in the experiments of "middle" instances, which implies that its approach may be useful in more complex assembly lines. Furthermore, the percentage of disabled workforce integrated continues in at least 5%, as it is desired.

5.3.2. Experiment 2: heuristics comparison

For validation purposes, we solved the same 1600 "medium" and "large" instances with the CIH procedure and its variations with post-optimization phase (CIH-LS1 and CIH-LS2) presented in section 4. The MIP-local search routines were run with the package CPLEX 12.4 with the same parameter settings, except for the computational time limit, which was set to 60 s.

Table 2
Computational results: ALWIBP-1 and ALWIBP-1S_{min} models ($|N| = 100$).

|W|	Var	Inc (%)	Δ	t(s)	$m_{\uparrow} \pm \sigma_{m_{\uparrow}}$	$m_{\uparrow} \pm \sigma_{m_{\uparrow}}(\%)$	τ	$\tau_{S_{min}}$	$\eta(\%)$	$\eta_{S_{min}}(\%)$	$\beta \pm \sigma_{\beta}(\%)$	θ
1	**U[t,2t]**	**10**	86	309.1	0.2 ± 0.4	1.0 ± 2.3	68.9	1.1	74.4	87.0	5.6 ± 1.4	83
		20	85	330.6	0.2 ± 0.4	1.0 ± 2.3	85.6	1.7	67.0	85.7	5.6 ± 1.4	82
	U[t, 5t]	**10**	81	435.2	0.3 ± 0.5	2.0 ± 2.9	109.1	3.4	56.8	66.0	5.5 ± 1.3	66
		20	74	521.9	0.4 ± 0.5	2.4 ± 3.1	125.1	5.3	49.7	67.0	5.5 ± 1.3	59
2	**U[t,2t]**	**10**	67	675.8	0.4 ± 0.5	2.3 ± 3.1	70.4	1.5	71.4	88.5	11.0 ± 2.7	61
		20	65	679.5	0.4 ± 0.5	2.4 ± 3.1	72.7	1.7	71.3	85.5	11.0 ± 2.7	59
	U[t, 5t]	**10**	53	971.1	0.7 ± 0.5	4.2 ± 2.9	95.9	5.5	57.1	65.9	10.8 ± 2.6	28
		20	49	1045.3	0.8 ± 0.4	4.5 ± 2.7	105.5	4.8	55.4	66.2	10.8 ± 2.6	21
3	**U[t,2t]**	**10**	47	1070.7	0.6 ± 0.5	3.3 ± 3.1	48.8	1.7	74.6	85.2	16.4 ± 3.9	43
		20	47	1099.7	0.6 ± 0.5	3.5 ± 3.0	55.0	2.8	73.4	83.8	16.4 ± 3.9	39
	U[t, 5t]	**10**	28	1397.5	1.1 ± 0.5	6.4 ± 3.2	86.7	5.5	57.4	65.1	15.9 ± 3.7	5
		20	28	1407.1	1.1 ± 0.5	6.5 ± 3.2	87.4	4.1	58.0	63.3	15.9 ± 3.7	4
4	**U[t,2t]**	**10**	34	1300.6	0.8 ± 0.5	4.4 ± 2.9	55.2	2.7	74.1	82.6	21.6 ± 5.1	26
		20	32	1362.7	0.8 ± 0.5	4.6 ± 2.9	55.5	3.4	72.4	80.8	21.6 ± 5.1	22
	U[t, 5t]	**10**	8	1707.1	1.5 ± 0.5	8.4 ± 4.0	89.3	7.4	57.7	61.4	20.8 ± 4.7	2
		20	12	1724.5	1.5 ± 0.5	8.5 ± 3.7	109.6	8.2	57.6	58.5	20.8 ± 4.8	1

Table 3
Computational results: CIH approaches ($|N| = 50$).

|W|	Var	Inc %	CIH				CIH-LS1				CIH-LS2			
			$m_{h_{\uparrow}} \pm \sigma_{m_{h_{\uparrow}}}$	$m_{h_{\uparrow}} \pm \sigma_{m_{h_{\uparrow}}}(\%)$	$t_h(s)$	Ties	$m_{h_{\uparrow}} \pm \sigma_{m_{h_{\uparrow}}}$	$m_{h_{\uparrow}} \pm \sigma_{m_{h_{\uparrow}}}(\%)$	$t_h(s)$	Ties	$m_{h_{\uparrow}} \pm \sigma_{m_{h_{\uparrow}}}$	$m_{h_{\uparrow}} \pm \sigma_{m_{h_{\uparrow}}}(\%)$	$t_h(s)$	Ties
1	**U[t,2t]**	**10**	0.2 ± 0.4	1.8 ± 4.3	0.0	84	0.1 ± 0.2	0.6 ± 2.7	10.5	95	0.1 ± 0.3	1.3 ± 3.6	0.1	88
		20	0.1 ± 0.3	1.3 ± 3.6	0.0	88	0.0 ± 0.1	0.2 ± 1.7	10.8	98	0.1 ± 0.3	1.1 ± 3.4	0.1	89
	U[t, 5t]	**10**	0.2 ± 0.4	2.1 ± 4.5	0.0	81	0.1 ± 0.3	0.6 ± 2.6	10.8	92	0.2 ± 0.4	1.8 ± 4.2	0.1	83
		20	0.2 ± 0.4	2.3 ± 4.7	0.0	79	0.0 ± 0.2	0.4 ± 1.8	9.9	96	0.2 ± 0.4	2.0 ± 4.4	0.0	81
2	**U[t,2t]**	**10**	0.2 ± 0.4	2.3 ± 4.6	0.0	79	0.1 ± 0.3	0.9 ± 3.1	12.2	91	0.2 ± 0.4	2.0 ± 4.4	0.1	82
		20	0.2 ± 0.4	2.4 ± 4.7	0.0	78	0.1 ± 0.3	1.0 ± 3.4	10.7	91	0.2 ± 0.4	2.4 ± 4.7	0.1	78
	U[t, 5t]	**10**	0.3 ± 0.5	3.2 ± 5.1	0.0	70	0.2 ± 0.4	1.7 ± 4.1	13.4	85	0.3 ± 0.5	3.0 ± 5.0	0.1	72
		20	0.3 ± 0.5	3.4 ± 5.2	0.0	69	0.2 ± 0.4	2.1 ± 4.4	12.1	81	0.3 ± 0.5	3.4 ± 5.2	0.1	69
3	**U[t,2t]**	**10**	0.3 ± 0.5	3.3 ± 5.2	0.0	69	0.1 ± 0.3	1.3 ± 3.4	16.4	86	0.3 ± 0.5	3.1 ± 5.0	0.1	71
		20	0.3 ± 0.4	2.8 ± 5.0	0.0	74	0.1 ± 0.3	0.9 ± 3.0	14.1	91	0.2 ± 0.4	2.4 ± 4.7	0.1	78
	U[t, 5t]	**10**	0.4 ± 0.5	3.9 ± 5.2	0.0	62	0.2 ± 0.4	1.4 ± 3.5	17.0	85	0.4 ± 0.5	3.9 ± 5.2	0.0	62
		20	0.4 ± 0.5	3.9 ± 5.1	0.0	61	0.2 ± 0.4	1.6 ± 3.5	14.4	83	0.4 ± 0.5	3.9 ± 5.1	0.0	61
4	**U[t,2t]**	**10**	0.4 ± 0.5	3.9 ± 5.4	0.0	63	0.2 ± 0.4	2.0 ± 4.5	16.6	82	0.4 ± 0.5	3.7 ± 5.3	0.1	65
		20	0.3 ± 0.5	3.5 ± 5.1	0.0	66	0.1 ± 0.3	1.3 ± 3.7	13.4	84	0.3 ± 0.5	3.4 ± 5.1	0.0	67
	U[t, 5t]	**10**	0.5 ± 0.5	5.2 ± 5.1	0.0	46	0.3 ± 0.4	2.3 ± 4.2	18.7	75	0.5 ± 0.5	5.1 ± 5.1	0.0	47
		20	0.6 ± 0.5	5.2 ± 5.1	0.0	46	0.3 ± 0.5	2.6 ± 4.3	14.6	72	0.5 ± 0.5	5.2 ± 5.0	0.0	46

Table 4
Computational results: CIH approaches ($|N| = 100$).

$\|W\|$	Var	Inc (%)	CIH				CIH-LS1				CIH-LS2			
			$m_{h_\uparrow} \pm \sigma_{m_{h_\uparrow}}$	$m_{h_\uparrow} \pm \sigma_{m_{h_\uparrow}}$ (%)	$t_h(s)$	Ties	$m_{h_\uparrow} \pm \sigma_{m_{h_\uparrow}}$	$m_{h_\uparrow} \pm \sigma_{m_{h_\uparrow}}$ (%)	$t_h(s)$	Ties	$m_{h_\uparrow} \pm \sigma_{m_{h_\uparrow}}$	$m_{h_\uparrow} \pm \sigma_{m_{h_\uparrow}}$ (%)	$t_h(s)$	Ties
1	U[t,2t]	10	0.3 ± 0.4	1.5 ± 2.6	0.0	73	0.1 ± 0.3	0.5 ± 1.6	47.5	89	0.2 ± 0.4	1.1 ± 2.3	5.2	79
		20	0.3 ± 0.5	1.8 ± 2.9	0.0	71	0.2 ± 0.4	0.9 ± 2.3	46.9	82	0.2 ± 0.4	1.3 ± 2.6	6.6	77
	U[t, 5t]	10	0.4 ± 0.5	1.9 ± 2.7	0.0	64	0.2 ± 0.4	0.8 ± 2.0	43.4	81	0.4 ± 0.5	1.9 ± 2.7	2.7	65
		20	0.3 ± 0.4	1.4 ± 2.5	0.0	73	0.1 ± 0.3	0.4 ± 1.7	40.2	87	0.3 ± 0.4	1.3 ± 2.4	2.8	75
2	U[t,2t]	10	0.3 ± 0.4	1.3 ± 2.4	0.0	70	0.2 ± 0.4	0.8 ± 1.9	48.1	80	0.2 ± 0.4	1.0 ± 2.2	2.3	76
		20	0.3 ± 0.4	1.3 ± 2.4	0.0	75	0.2 ± 0.4	0.8 ± 2.0	46.6	82	0.2 ± 0.4	1.2 ± 2.3	2.1	77
	U[t, 5t]	10	0.3 ± 0.5	1.7 ± 2.7	0.1	68	0.2 ± 0.4	0.9 ± 2.2	46.3	74	0.3 ± 0.5	1.6 ± 2.6	3.0	70
		20	0.3 ± 0.5	1.6 ± 2.6	0.0	72	0.2 ± 0.4	0.9 ± 2.2	43.9	78	0.3 ± 0.5	1.6 ± 2.6	1.3	72
3	U[t,2t]	10	0.3 ± 0.5	1.5 ± 2.6	0.1	70	0.2 ± 0.4	0.9 ± 2.1	49.6	78	0.3 ± 0.4	1.4 ± 2.5	3.1	72
		20	0.3 ± 0.5	1.6 ± 2.6	0.1	71	0.2 ± 0.4	0.6 ± 1.9	46.9	82	0.3 ± 0.4	1.4 ± 2.4	2.5	74
	U[t, 5t]	10	0.4 ± 0.5	2.1 ± 2.9	0.1	63	0.2 ± 0.4	0.9 ± 2.1	53.4	79	0.4 ± 0.5	2.1 ± 2.9	0.7	63
		20	0.4 ± 0.5	2.2 ± 2.8	0.1	57	0.2 ± 0.4	0.9 ± 2.2	46.9	75	0.4 ± 0.5	2.2 ± 2.8	1.2	58
4	U[t,2t]	10	0.3 ± 0.5	1.6 ± 2.5	0.1	68	0.3 ± 0.4	1.0 ± 2.4	48.3	68	0.3 ± 0.5	1.4 ± 2.4	1.7	70
		20	0.4 ± 0.5	1.8 ± 2.8	0.1	63	0.2 ± 0.4	0.9 ± 2.2	48.9	74	0.3 ± 0.5	1.7 ± 2.7	2.0	66
	U[t, 5t]	10	0.4 ± 0.5	2.3 ± 2.8	0.1	56	0.2 ± 0.4	0.7 ± 2.2	50.5	72	0.4 ± 0.5	2.2 ± 2.8	2.3	57
		20	0.5 ± 0.5	2.7 ± 2.9	0.1	52	0.3 ± 0.4	1.3 ± 2.5	49.6	70	0.5 ± 0.5	2.5 ± 2.9	0.9	55

In Tables 3 and 4, the columns indicate:

- $\mathbf{m_{h_\uparrow}}$ and $\boldsymbol{\sigma_{m_{h_\uparrow}}}$: number of additional workstations in the CIH solution, with respect to the solution obtained when solving the ALWIBP-1 model (average and deviation);
- $\mathbf{m_{h_\uparrow}}$ and $\boldsymbol{\sigma_{m_{h_\uparrow}}}$ (%): percentage of additional workstations in the CIH solution, with respect to the solution obtained when solving the ALWIBP-1 model (average and deviation);
- $\mathbf{t_h}(s)$: Computational time used by the heuristic (on average);
- **Ties**: Number of instances in which the CIH solution had the same number of workstations (no increase) as the ALWIBP-1 model solution;
- $\mathbf{Ties}_{sp\ 1}$: Number of instances in which the CIH solution had the same number of workstations (no increase) as the SALBP-1 model solution;

The results of Table 3 show the robust behavior of the proposed heuristic. Indeed, even in the most difficult situations (U[t,5t], Inc 20%) the increase of stations is 5.2% in average and the number of ties is 46. The computational times are practically null, which shows the computational efficiency of the heuristic. Concerning the CIH approaches with post-optimization routines, we see that the greater flexibility of the LS1 causes significant improvement in the results of CIH, increasing the number of ties compared with the solutions of ALWIBP-1 model in 24% (272 instances). We also outline that, in some cases, the average number of additional stations in the assembly lines is reduced by half. Despite the fact that LS2 improves the quality of solutions in only a few instances, it is very computationally efficient, running in very little time (0.1 s in average).

Table 4 shows the results obtained for the large instances. We notice that the number of Ties (66%) is practically the same as observed in the previous case, but the average number of additional workstations is smaller. This states that the good results of this scenario is powered by the flexibility of assembly lines with many tasks.

The computational times of the three approaches of the CIH show that the method is scalable. Moreover, the LS1 increases the number of Ties related to the solutions of the ALWIBP-1 model, spending up to 53 s on average, a very encouraging result if we consider that the CPLEX exceeds the computational time limit in most of cases.

Table 5
Computational results: CIH ($|N| = 1000$).

$\|W\|$	Var	Inc (%)	CIH			
			$m_{h_\uparrow} \pm \sigma_{m_{h_\uparrow}}$	$m_{h_\uparrow} \pm \sigma_{m_{h_\uparrow}}$ (%)	$t_h(s)$	$\mathbf{Ties}_{sp1}$
1	U[t,2t]	10	0.8 ± 0.9	0.5 ± 0.7	50.1	34
		20	0.9 ± 0.9	0.5 ± 0.7	50.8	31
	U[t, 5t]	10	0.9 ± 0.9	0.6 ± 0.7	50.4	31
		20	1.0 ± 0.9	0.6 ± 0.7	50.2	29
2	U[t,2t]	10	1.1 ± 0.9	0.7 ± 0.7	106.6	25
		20	1.1 ± 0.9	0.7 ± 0.7	105.9	24
	U[t, 5t]	10	1.5 ± 1.0	1.0 ± 0.8	105.9	16
		20	1.5 ± 1.1	1.0 ± 0.8	105.9	17
3	U[t,2t]	10	1.2 ± 1.0	0.8 ± 0.8	167.1	21
		20	1.3 ± 1.0	0.8 ± 0.7	167.3	22
	U[t, 5t]	10	1.8 ± 1.1	1.1 ± 0.8	164.4	14
		20	1.9 ± 1.0	1.2 ± 0.8	163.6	8
4	U[t,2t]	10	1.4 ± 1.0	0.9 ± 0.8	223.4	21
		20	1.4 ± 1.0	0.9 ± 0.8	225.4	21
	U[t, 5t]	10	2.1 ± 1.2	1.4 ± 0.9	224.1	10
		20	2.1 ± 1.2	1.4 ± 0.9	222.0	9

5.3.3. Experiment 3: Validation of heuristics

In this section, we present results of the CIH taking into account the largest instances of the ALWIBP benchmark. Also, we validate our method by comparing it with a more straightforward substitution heuristic.

CPLEX fails in obtaining solutions for the "very large" instances described earlier. Although these instances are more theoretical than practical, computational experiments with this scenario are important in order to put the scalability of the solving methods to proof. Thus, we highlight the ability of the CIH to solve problems of these dimensions in a reasonable computational times (3 min on average), as shown in Table 5. Additionally it is remarkable that in 21% of the instances, the number of stations of a conventional assembly line did not change with the insertion of disabled workers.

In order to study the effectiveness of our algorithm in comparison to more straightforward strategies, we have implemented a simple substitution heuristic (denoted by SH) and evaluated the obtained results. The idea behind the SH is simply to substitute conventional workers for disabled ones, maintaining the task assignments. Our goal with this experiment is to show that simple strategies such as this one, which lack the ability to perform more structural changes in the line structure are ineffective. The SH

works as follows: given a SALBP-1 solution the algorithm searches for the most suitable stations in which to insert the disabled workers without changing task assignment and respecting cycle time and task/worker feasibilities constraints. Algorithm 3 presents the pseudo-code of this heuristic. Note that the *LD* formulation, modelled by Eqs. (22)–(28) is part of the kernel of the SH.

Algorithm 3. Simple Heuristic.

Require x^{ref} (SALBP-1 reference solution);
1: $\overline{m} \leftarrow stations(x^{ref})$;
2: Let M be an integer matrix with $\overline{m}$ rows and $|W|$ columns;
3: **for** $s = 1, \ldots, \overline{m}$ **do**
4: **for all** $w \in W$ **do**
5: $M_{sw} \leftarrow load\ (s, w, x^{ref})$;
6: **end for**
7: **end for**
8: $(l^*, y^*) \leftarrow solveLD(M)$;
9: **if** $l^* = \overline{C}$ **then**
10: **return** y^*;
11: **else**
12: *there is no feasible solution*;
13: **end if**

$$\text{Min} l \tag{22}$$

subject to

$$l \geq \overline{C}, \tag{23}$$

$$l \geq M_{sw} y_{sw}, \quad s = 1, \ldots, \overline{m}, \forall w \in W, \tag{24}$$

$$\sum_{s=1}^{\overline{m}} y_{sw} = 1, \quad \forall w \in W, \tag{25}$$

$$\sum_{w \in W} y_{sw} \leq 1, \quad s = 1, \ldots, \overline{m}, \tag{26}$$

$$y_{sw} \in \{0, 1\}, \quad s = 1, \ldots, \overline{m}, \forall w \in W, \tag{27}$$

$$l \in \mathbb{Z}. \tag{28}$$

In lines 1–6, we compute the elements of the matrix M as the load of station s if the "conventional" worker in that station is replaced by disabled worker w. To do that, we consider the *load* function (line 5). Next, in line 7, we solve the *LD* formulation. Lines 9–13 test the effectiveness of the SH. Thus, if the resulted solution is feasible, that is, the best value found by *LD* model is equal to the desired cycle time, the algorithm returns the variables y which indicate the placement of each worker along the assembly line.

Computational experiments were conducted using the same input solutions as those used for the CIH, as well as the same machine settings and CPLEX version. The SH could only find feasible solutions for about 4% of the instances (3%, 2% and 7% for the medium, large and very large instances, respectively), confirming our hypothesis that a more elaborate strategy such as the proposed CIH is necessary when dealing with the problem.

6. Conclusions

We propose the Assembly Line Worker Integration and Balancing Problem (ALWIBP), a new assembly line balancing problem arising in lines with conventional and disabled workers. This problem is relevant in a context where companies are urged to integrate disabled workers in their conventional productive schemes in order to cope with legislation issues or to include corporate social responsibility goals in the production planning process.

To solve this problem, we first develop an integer linear model that minimizes the number of stations while ensuring the presence of all disabled workers in the assembly line. From this model, one variant that reduces the idle time of stations with disabled workers was proposed. We implement a heuristic approach called CIH that starts with a simple assembly line balancing situation and inserts the available disabled workers while reducing additional stations. Furthermore, aiming the improvement of CIH solutions, two post-optimizations procedures based on MIP neighborhoods were designed. Finally, to validate heuristic results, we also implemented a simple substitution heuristic (SH) that tries to insert disabled workers in a conventional assembly line without changing its original task settings.

Results of an experimental study on an extensive number of instances indicate the efficiency of the proposed heuristics, leading to the conclusion that not only disabled workers can be included in the assembly lines with little productivity loss, but also that other planning goals can be simultaneously considered. Further work on this topic includes the proposal of new adjacent objectives, implementation of more sophisticated methods, and extensions that cope with job rotation schemes, U-shaped assembly lines and some lexicographic objectives.

Acknowledgments

This research was supported by CAPES-Brazil and MEC-Spain (coordinated project CAPES DGU 258-12/PHB2011-0012-PC) and by FAPESP-Brazil. We also thank the project "CORSARI MAGIC DPI 2010-18243" of the Ministerio de Ciencia e Innovación del Gobierno de España within the Program "Proyectos de Investigación Fundamental No Orientada". The authors also thank two anonymous reviewers and the editor for their useful comments.

References

[1] Baybars I. A survey of exact algorithms for the simple assembly line balancing problem. Manag Sci 1986;32:909–32.
[2] Blum C, Miralles C. On solving the assembly line worker assignment and balancing problem via beam search. Comput Oper Res 2011;38:328–39.
[3] Borba L, Ritt M. A heuristic and a branch-and-bound algorithm for the Assembly Line Worker Assignment and Balancing Problem. Comput Oper Res 2014;45:87–96.
[4] Boysen N, Fliedner M, Scholl A. A classification of assembly line balancing problems. Eur J Oper Res 2007;183:674–93.
[5] Chaves AA, Lorena LAN, Miralles C. Hybrid metaheuristic for the assembly line worker assignment and balancing problem. Lecture notes on computer science, vol. 5818; 2009, p. 1–14.
[6] Dimitriadis SG. Assembly line balancing and group working: a heuristic procedure for workers' groups operating on the same product and workstation. Comput Oper Res 2006;33:2757–74.
[7] Kellegöz T, Toklu B. An efficient branch and bound algorithm for assembly line balancing problems with parallel multi-manned workstations. Comput Oper Res 2012;39:3344–60.
[8] Kim YK, Song WS, Kim JH. A mathematical model and a genetic algorithm for two-sided assembly line balancing. Comput Oper Res 2009;36:853–65.
[9] Miralles C, Garcia-Sabater JP, Andres C, Cardos M. Advantages of assembly lines in Sheltered Work Centres for Disabled: a case study. Int J Prod Econ 2007;110:187–97.
[10] Miralles C, Garcia-Sabater JP, Andres C, Cardos M. Branch and bound procedures for solving the Assembly Line Worker Assignment and Balancing Problem: application to Sheltered Work Centres for Disabled. Discret Appl Math 2008;156:352–67.
[11] Moreira MCO, Ritt M, Costa AM, Chaves AA. Simple heuristics for the assembly line worker assignment and balancing problem. J Heuristics 2012;18:505–24.
[12] Mutlu O, Polat O, Supciller A. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-II. Comput Oper Res 2013;40:418–26.
[13] Otto A, Otto C, Scholl A. Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. Eur J Oper Res 2014;228:33–45.
[14] Özcan U, Toklu B. Multiple-criteria decision-making in two-sided assembly line balancing: a goal programming and a fuzzy goal programming models. Comput Oper Res 2009;36:1955–65.

[15] Petterson J, Albracht J. Assembly-line balancing: zero-one programming with fibonacci search. Oper Res 1975;23:166–72.
[16] Ritt M, Costa AM. A comparison of formulations for the simple assembly line balancing problem. arXiv:1111.0934 [cs.DM].
[17] Scholl A. Balancing and sequencing assembly lines. 2nd ed. Heidelberg: Physica-Verlag; 1999.
[18] Scholl A, Voß S. Simple assembly line balancing—heuristic approaches. J Heuristics 1996;2:217–44.
[19] Vilà M, Pereira J. A branch-and-bound algorithm for assembly line worker assignment and balancing problems. Comput Oper Res 2014;44:105–14.
[20] Zeng X, Wong W-K, Leung SY-S. An operator allocation optimization model for balancing control of the hybrid assembly lines using pareto utility discrete differential evolution algorithm. Comput Oper Res 2012;39:1145–59.