LB-ALBP: The Lexicographic Bottleneck Assembly Line Balancing Problem

Rafael Pastor

LB-ALBP: The Lexicographic Bottleneck Assembly Line Balancing Problem

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2009-IJPR-0925.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	26-Jan-2010
Complete List of Authors:	Pastor, Rafael; Universidad Politécnica de Cataluña, Instituto de Organización y Control de Sistemas Industriales
Keywords:	ASSEMBLY LINE BALANCING, ASSEMBLY LINES
Keywords (user):	

LB-ALBP: The *Lexicographic Bottleneck Assembly Line Balancing Problem*[†]

Rafael Pastor[*]
IOC Research Institute
Technical University of Catalonia
Av. Diagonal 647, 11th floor, 08028, Barcelona, Spain
rafael.pastor@upc.edu

Abstract

The classic assembly line balancing problem (ALBP) basically consists of assigning a set of tasks to a group of workstations while maintaining the tasks' precedence relations. When the objective is to minimize the number of workstations m for a given cycle time CT, the problem is referred to as ALBP-1; if the objective is to minimize CT given m, then the problem is called ALBP-2. The only objective in ALBP-2 is to minimize CT, i.e., the workload of the most heavily loaded workstation (the bottleneck). However, considering the second-biggest, third-biggest, etc. workloads, can be important. Distributing a workload among six workstations as 10, 10, 10, 4, 3, 3, is not the same as distributing it as 10, 6, 6, 6, 6, 6. The CT value is the same, but the second distribution is beyond question more reliable and balanced. In this paper, we present and formalize a new assembly line balancing problem: the *Lexicographic Bottleneck Assembly Line Balancing Problem* (LB-ALBP). The LB-ALBP hierarchically minimizes the workload of the most heavily loaded workstation (CT), followed by the workload of the second most heavily loaded workstation, followed by the workload of the third most heavily loaded workstation, and so on. We present two mixed-integer linear programming (MILP) models designed to solve the LB-ALBP optimally, together with three heuristic procedures based on these MILPs.

***Keywords*:** assembly line balancing.

1. Introduction

Assembly lines are components of many production systems, such as those used in the automotive and household appliance industries. The problem of designing and balancing assembly lines is very difficult to solve due to its combinatorial nature –it is NP-hard (see, e.g., Wee and Magazine, 1982)– and the large number of tasks and constraints that occur in real-life situations.

The classic assembly line balancing problem (ALBP) basically consists of assigning a set of tasks (each of them characterized by its processing time) to an ordered sequence of workstations, in such a way that the precedence constraints between the tasks are maintained and a given efficiency measure is optimized. Regarding the conventional terminology (e.g., Baybars, 1986), when the objective is to minimize the number of

[†] Supported by the Spanish Ministry of Education and Science, projects DPI2004-03472 and DPI2007-61905, co-financed by FEDER.
[*] Corresponding author: Rafael Pastor, IOC Research Institute, Av. Diagonal, 647 (edif. ETSEIB), p. 11, 08028 Barcelona, Spain; Tel. + 34 93 401 17 01; fax + 34 93 401 66 05; e-mail: rafael.pastor@upc.edu.

workstations for a given upper bound on the cycle time, the problem is known as ALBP-1; if the objective is to minimize the cycle time given a number of workstations, then the problem is called ALBP-2.

The problem of designing and balancing assembly lines has been extensively examined in the literature. A number of synthesis studies has been published, including Baybars (1986), Ghosh and Gagnon (1989), Erel and Sarin (1998), Scholl (1999), Rekiek et al. (2002b), Becker and Scholl (2006), Scholl and Becker (2006) and Boysen et al. (2007, 2008); however, most papers focus on the simple case. This problem has been approached using heuristic procedures –e.g., Talbot et al. (1986) and Ponnambalam et al. (1999)–, exact procedures based on binary linear programming –e.g., White (1961) and Pastor and Ferrer (2009)–, integer linear programming –e.g., Talbot and Patterson (1984)–, dynamic programming –e.g., Kao and Queyranne (1982)–, and branch and bound –e.g., FABLE by Johnson (1988), EUREKA by Hoffman (1992) and SALOME by Scholl and Klein (1997)–. A significant variety of complex cases has been examined, including problems that consider lines with parallel workstations or parallel tasks; mixed or multi-models; multiple products; U-shaped, two-sided, buffered or parallel lines; incompatibility between tasks; stochastic processing times; and equipment selection –e.g., Park et al. (1997), Amen (2001, 2006), Ağpak and Gökçen (2005), Ding et al. (2006), Gamberini et al. (2006), Gökçen et al. (2006), Andrés et al. (2008), Capacho and Pastor (2008), Corominas et al. (2008), Capacho et al. (2009), Corominas and Pastor (2009), Cortés et al. (2009), Martino and Pastor (2010) and Pastor et al. (2010)–. As a result, generalized problems are becoming a widespread subject.

During the last years, several artificial intelligence techniques have been used, including genetic algorithms, as employed in Rubinovitz and Levitin (1995); simulated annealing, as applied in Suresh and Sahu (1994); tabu search, as used in Pastor at al. (2002); ant colony, as employed in Baykasoglu et al. (2003); or the knowledge-based approach developed in Malakooti and Kumar (1996).

The ALBP-2 is a problem with a *min-max* objective, since it tries to minimize the workload of the most heavily loaded workstation (the bottleneck). The optimal solution of problems with this type of objective guarantees that the maximum value is as small as possible; however, the remaining values are not considered, even though they may also be important. In ALBPs, it is important to consider the second-biggest, third-biggest, etc. workloads: the reliability of the line is improved; the workload is prone to be uniformly distributed among all of the workstations; and, moreover, as it is exposed in Boysen et al. (2006), the quality defects caused by stations with disproportionately large station times are avoided.

The main objective of this paper is to present a new assembly line balancing problem: the *Lexicographic Bottleneck Assembly Line Balancing Problem* (LB-ALBP). The LB-ALBP hierarchically minimizes the workload of the most heavily loaded workstation, followed by the workload of the second most heavily loaded workstation, followed by the workload of the third most heavily loaded workstation, and so on. The LB-ALBP is presented and formalized, and two mixed-integer linear programming models are designed to solve it optimally, together with three heuristic procedures based on these mathematical programs.

The remainder of the paper is organized as follows. Section 2 describes and characterizes the LB-ALBP. Section 3 presents two mixed-integer linear programming (MILP) models designed to solve the problem optimally. Section 4 describes three heuristic procedures based on these MILP models. Finally, Section 5 offers conclusions and ideas for further research.

2. The *Lexicographic Bottleneck Assembly Line Balancing Problem* (LB-ALBP)

The objective of the type 2 assembly line balancing problem (ALBP-2) is to minimize the workload of the most heavily loaded workstation (the bottleneck) and, therefore, it is a problem with a *min-max* objective. The scientific literature includes several problems with *min-max* objectives: location, production according to an ideal value, etc. The optimal solution of a problem with this kind of objective guarantees that the maximum value is as small as possible. However, the remaining values are not considered, and they can also be important because, even though they do not exceed the maximum value (the bottleneck), their distribution can be highly irregular and therefore inappropriate.

Indeed, in the ALBP, it is important to consider the second-largest workload, the third-largest workload, etc., because this allows us to reduce the *criticalness* of the workstations. That is, the smaller the difference between the processing time (workload) of a station and the cycle time, the more *critical* the workstation. By reducing criticalness, the reliability of the line is improved. Figure 1a shows that distributing a workload among six workstations as 10, 10, 10, 4, 3, 3 is not the same as distributing it as 10, 6, 6, 6, 6, 6. The cycle time value is the same, but the second distribution is beyond question more reliable than the first one because it reduces the criticalness of the workstations. It is normally assumed that the processing time of the tasks is known and constant; however, in industrial environments, these times often change from one cycle to the next one, especially in manual lines. Stochastic processing times increase the probability that the cycle time will be greater than a given value. For example, let ω be the probability that a workstation with a workload equal to the cycle time CT will be greater than CT. For the first workload distribution shown in Figure 1a, the probability of one workstation having a workload greater than CT is equal to $3\omega - 3\omega^2 + \omega^3 \left(1-(1-\omega)^3\right)$. On the other hand, for the second distribution shown in Figure 1a, the probability of one workstation having a workload greater than CT is equal to just ω (only fully loaded stations have been considered to calculate the probability of station overloads, since the probability that a non fully loaded station has a workload greater than CT is very inferior to the probability that a fully loaded station has a workload greater than CT).

Insert Figure 1

Another aspect to be considered could be the distribution of a line's most critical workstations. Figure 1b shows that a distribution of 10, 4, 10, 3, 10, 3 has the same criticalness as a distribution of 10, 10, 10, 4, 3, 3. However, according to the aforementioned definition of criticalness, the first distribution seems preferable to the

second one, since any problem that may occur at a very critical workstation could be sorted out if the next workstation is not critical. This aspect is not taken into account and is an open field of future research.

We define and formalize a new assembly line balancing problem, referred to by the authors as the *Lexicographic Bottleneck Assembly Line Balancing Problem* (LB-ALBP). The LB-ALBP minimizes the workload of the most heavily loaded workstation (cycle time) and then goes on minimizing the workload of the second most heavily loaded workstation, followed by the workload of the third most heavily loaded workstation, and so on. It is therefore a multi-objective optimization problem with a hierarchical structure (Krajewski and Ritzman (1979), Starr (1979) and Cortés et al. (2001, 2006)). In general, the term "lexicographic optimization" is applied when different objectives must be optimized, but there is also literature that applies this term when the same objective is being optimized (e.g. Pentico, 2007). For more details about the *lexicographic bottleneck* objective, see, for example, Burkard and Rendl (1991) and Sokkalingam and Aneja (1998).

Several papers have already dealt with multi-objective ALBPs, but the approach proposed here is different. For example, Miyazaki and Ohta (1987) optimally solve the line balancing problem in a way that minimizes the number of workstations and the cycle time. Malakooti and Kumar (1996) use a multi-criteria decision-aid method for ALBPs where the objectives are the number of stations, the cycle time, the buffer size and the total cost of the operations. Kim et al. (1996) present genetic algorithms to solve ALBPs with various objectives: minimizing the number of workstations, minimizing cycle time, maximizing workload smoothness and maximizing work-relatedness. Rekiek et al. (2002a) also take multiple objectives into account: total cost, imbalance, reliability and congestion. In our comprehensive review of the literature on ALBPs, this type of problem has not previously been addressed.

On the other hand, the *lexicographic bottleneck* objective also tends to homogeneously distribute the workload among all workstations, which avoids large differences in the workers' workloads and, as it is exposed in Boysen et al. (2006), prevents quality defects caused by stations with disproportionately large station times. Other objective functions also tend to homogenize the workload distribution, such as: the "smoothness index" (Moodie and Young, 1965), $SI = \sqrt{\sum_{j=1}^{m} \left(ts_{\max} - ts_j \right)^2}$, where m is the number of workstations, $ts_{\max}$ is the maximum station time and ts_j is the time of workstation j; the "workload range", $ts_{\max} - ts_{\min}$, where $ts_{\max}$ and $ts_{\min}$ are the maximum and minimum workstation times, respectively; and the imposition of a time interval on the workload at any workstation (Pastor and Corominas, 2000), $\left[\overline{ts} - \sigma, \overline{ts} + \sigma \right]$, where $\overline{ts}$ is the average workload of the stations and σ is the allowed tolerance. Note that the *lexicographic bottleneck* objective is different –as it has been exposed, for assignment problems, by Pentico (2007)–.

The following example proves that the optimal solution according to the *lexicographic bottleneck* objective, LB, may be different from the optimal solution according to,

for example, the "smoothness index" objective, SI. Figure 2 shows the precedence graph for an instance with 10 tasks, with the processing time of each task over the vertex that represents the task. Table 1 shows the two optimal solutions with five workstations. For each optimal solution, the following information is given: the tasks assigned to each workstation (and, in parentheses, the total workload of the station), Sta_x, and the value of the solution according to the objectives LB and SI.

Insert Figure 2

Insert Table 1

3. Mathematical programming models of the LB-ALBP

The LB-ALBP is a multi-objective optimization problem in which the order of the objectives is the main point and there is no interest in a continuous trade-off among the functions. As Pentico (2007) shows, there are two basic methods to incorporate multiple criteria in decision models: (1) by combining them in a single criterion, and (2) by considering them separately and sequentially. To solve the LB-ALBP optimally, we have designed two mathematical programming models, one for each method:

a) Combining multiple criteria in one criterion: The Global Hierarchical Model (GHM), which solves the problem in a way that minimizes a weighted sum of functions, with enough different weights to preserve the hierarchy of the proposed objectives.

b) Considering multiple criteria sequentially: The Successive Hierarchical Model (SHM), which carries out a preemptive goal programming. This method is an extension of linear programming that considers multiple objectives separately and sequentially. It observes and sequentially minimizes the objectives once the optimal values of the highest-priority objectives have been obtained.

Specifically, the ALBP that is modeled and solved as an LB-ALBP is the simplest case (the Simple Assembly Line Balancing Problem, SALBP), characterized as follows (Baybars, 1986): serial (straight) assembly lines processing a single model of a single product are considered; all input parameters are known with certainty; the task processing times are independent of the workstation at which they are performed and of the preceding and following tasks; all workstations are equipped and manned to process any of the tasks, and any task can be processed at any workstation; a task cannot be split among two or more workstations; tasks cannot be processed in arbitrary sequences due to technological precedence requirements; all tasks must be processed; and no assignment restrictions apart from precedence constraints are considered.

3.1. Preprocess

An initial preprocess stage is carried out: an upper bound on the cycle time is calculated and it is used to calculate the range of workstations to which any task i can be assigned.

The upper bound on the cycle time, CT, is calculated using the Helgeson and Birnie (1961) heuristic to solve the SALBP-1 iteratively. The iterative process produces one-unit increases above a lower bound on the cycle time until a solution is obtained for the specific number of workstations. It starts with the maximum among, on the one hand, the duration of the longest task and, on the other hand, the minimum integer number equal to or greater than the value of the quotient of the total time of the tasks and the total number of workstations (see, e.g., Scholl, 1999).

Once CT is evaluated, we use the well-known concept of earliest and latest station (E_i and L_i, respectively) to which a task i can be assigned: before assigning a task we must assign the total time of the preceding tasks and, likewise, after assigning a task we need to assign the total time of the tasks that follow it. As a result, the range of workstations $[E_i, L_i]$ to which task i can be assigned (see, e.g., Scholl, 1999) is obtained.

3.2. Global Hierarchical Model (GHM)

The Global Hierarchical Model (GHM) solves the problem by minimizing a weighted sum of functions with enough different weights to preserve the hierarchy of the proposed objectives. Therefore, a single mixed-integer linear programming (MILP) model is solved.

Data:

i, p	Index of tasks.
j	Index of workstations.
n	Number of tasks $(i = 1,...,n)$.
m	Number of workstations $(j = 1,...,m)$.
t_i	Processing time of task i.
CT	Upper bound on the cycle time.
E_i, L_i	Earliest and latest workstations, respectively, to which task i can be assigned, given a value of CT.
P	Set of ordered pairs of tasks (i, p) such that there is an immediate precedence relation between them.
α_k	Parameters to weigh the components of the objective function $(k = 1,...,m)$. $\alpha_k >> \alpha_{k+1}$ guaranteeing that the hierarchy of the proposed objectives is preserved.

Variables:

$x_{ij} \in \{0,1\}$	1, if and only if task i is assigned to workstation j $(\forall i; j = E_i,...,L_i)$.

S_j	Workload assigned to workstation j (i.e., the sum of the processing times of the tasks assigned to the workstation j).
T_k	Workload of the k-th most heavily loaded workstation. T_1 is the workload of the most heavily loaded station (T_1 is therefore the cycle time); T_2 is the workload of the second most heavily loaded station, etc.
$y_{jk} \in \{0,1\}$	1, if and only if station j has the k-th highest workload, T_k.

Model:

$$[MIN]\, z = \sum_{k=1}^{m} \alpha_k \cdot T_k \tag{1}$$

$$\sum_{j=E_i}^{L_i} x_{ij} = 1 \qquad \forall i \tag{2}$$

$$\sum_{\forall i \mid j \in [E_i, L_i]} t_i \cdot x_{ij} \le S_j \qquad \forall j \tag{3}$$

$$\sum_{k=1}^{m} y_{jk} = 1 \qquad \forall j \tag{4}$$

$$\sum_{j=1}^{m} y_{jk} = 1 \qquad \forall k \tag{5}$$

$$S_j \le T_k + M \cdot \sum_{s=1}^{k-1} y_{js} \qquad \forall k; \forall j \tag{6}$$

$$\sum_{j=E_i}^{L_i} j \cdot x_{ij} \le \sum_{j=E_p}^{L_p} j \cdot x_{pj} \qquad \forall (i,p) \in P \tag{7}$$

with $M = \min\left\{\sum_{i=1}^{n} t_i,\ CT - \min(t_i),\ CT - \max\left(0,\ \sum_{i=1}^{n} t_i - (m-1)\cdot CT\right)\right\}$

The objective function (1) minimizes a weighted sum of the workloads of the workstations; constraint set (2) implies that each task i is assigned to one and only one workstation; constraints (3) calculate the processing time (workload) assigned to each workstation; constraints (4) impose that each workstation is assigned to only one position in the list of workstations (ordered by workload); constraint set (5) imposes that each position of the workstation list, ordered by workload, is assigned to only one workstation; constraints (6) impose that if workstation j is the k-th most heavily loaded workstation, its workload will not be greater than the workload of the first k most heavily loaded workstations; and constraint set (7) imposes the technological precedence conditions.

When m is high, the model requires very high values of the parameters α_k; this situation can negatively affect numerical precision when the model is solved. It is

known that the conversion of a multi-objective hierarchical problem in a scalar optimization problem leads to artificial cost functions and to a difficult tuning of the weight that the designer wishes to associate with each criterion (Rekiek et al., 2002a).

3.3. Successive Hierarchical Model (SHM)

The Successive Hierarchical Model (SHM) carries out a preemptive goal programming; i.e., it sequentially solves $m-1$ mixed-integer linear programming (MILP) models by observing and sequentially minimizing the objectives once the optimal values of the highest-priority objectives have been obtained.

Next, we show the first model to be solved, in which the workload of the most heavily loaded workstation, T_1, is minimized. In this submodel, as in the following ones, constraints (2), (3) and (7) must be added.

$$[MIN]\, z = T_1 \tag{8}$$

$$S_j \le T_1 \qquad \forall j \tag{9}$$

(8) minimizes the workload of the most heavily loaded workstation and (9) imposes that all workstations have assigned a workload that is not greater than the workload of the most heavily loaded workstation.

The result of this model is T_1, which is fixed and becomes a datum. The second model to be solved, to minimize the workload of the second most heavily loaded workstation, T_2, is shown below.

$$[MIN]\, z = T_2 \tag{10}$$

$$\sum_{j=1}^{m} y_{j1} = 1 \tag{11}$$

$$S_j \le T_1 \qquad \forall j \tag{12}$$

$$S_j \le T_2 + M_1^{'} \cdot y_{j1} \qquad \forall j \tag{13}$$

with $M_1^{'} = T_1 - \dfrac{\sum_{i=1}^{n} t_i - T_1}{m-1}$

(10) minimizes the workload of the second most heavily loaded workstation; (11) imposes that only one workstation is defined as the most heavily loaded; (12) implies that all workstations have assigned a workload that is not greater than the workload of the most heavily loaded station; and (13) imposes that all the workstations, except the workstation that has assigned the workload T_1, have a workload smaller than or equal to the workload of the second most heavily loaded station. The result of this model is T_2.

Finally, we show the q-th model to be solved $(q = 3,...,m-1)$; i.e., the model to be solved once the workloads of the $q-1$ most heavily loaded stations, T_1, …, T_{q-1}, have been determined. This model minimizes the workload of the q-th most heavily loaded station, T_q. In this case, T_1,…, T_{q-1} are known parameters and T_k is the variable that indicates the workload of the k-th most heavily loaded workstation $(k = q,...,m-1)$.

$$[MIN]\, z = T_q \tag{14}$$

$$\sum_{j=1}^{m} y_{jk} = 1 \qquad k = 1,...,q-1 \tag{15}$$

$$\sum_{k=1}^{q-1} y_{jk} \le 1 \qquad \forall j \tag{16}$$

$$S_j \le \sum_{k=1}^{q-2} T_k \cdot y_{jk} + T_{q-1} \cdot \left(1 - \sum_{k=1}^{q-2} y_{jk}\right) \qquad \forall j \tag{17}$$

$$S_j \le T_q + \sum_{k=1}^{q-1} M_k^{'} \cdot y_{jk} \qquad \forall j \tag{18}$$

with $M_k^{'} = T_k - \dfrac{\sum_{i=1}^{n} t_i - \sum_{s=1}^{q-1} T_s}{m-(q-1)}$

(14) minimizes the workload of the q-th most heavily loaded workstation; (15) imposes that only one station is defined as the k-th most heavily loaded workstation $(k = 1,...,q-1)$; (16) implies that a workstation can either be one of the $q-1$ most heavily loaded stations or not have a defined workload –and, thus, a defined position in the list of workstations (ordered by workload)–; (17) imposes that the workload of any station that is not one of the $q-1$ most heavily loaded stations may not be greater than T_{q-1} and also imposes that the workload of the k-th most heavily loaded station $(k = 1,...,q-1)$ is T_k; and (18) imposes that the workload of a workstation that is not any of the $q-1$ most heavily loaded stations is smaller than or equal to the workload of the q-th most heavily loaded station. The result of this model is T_q.

The value T_m is forced when the other values T_k $(k = 1,...,m-1)$ are known.

3.4. A measurement of solution quality

We have designed a parameter, δ, to quantitatively compare two solutions of an LB-ALBP instance. δ considers, between both solutions, the difference in workload of each pair of workstations according to the list in descending order of workload:

$$\delta = \frac{\Delta_1 \cdot \beta^m + \Delta_2 \cdot \beta^{m-1} + ... + \Delta_m \cdot \beta}{CT_{best} \cdot \beta^{m-1}}$$

where Δ_j is the positive, null or negative workload difference in the j-th most heavily loaded station between the worst and best solutions which are compared (i.e., Δ_j is the workload, in the j-th most heavily loaded station, that the worst solution makes worse, or improves, with regard to the best one); m is the number of workstations; CT_{best} is the best cycle time of the two solutions compared; and β is a parameter whose value must guarantee the hierarchy of the objectives. In this paper, β has been set to 100, and we have checked that the hierarchy of the objectives is guaranteed.

For example, let us consider an instance with four workstations and the solutions Sol_1 and Sol_2, which have workloads, in descending order, of 50-48-46-44 and 50-47-46-45 time units, respectively. Sol_2 is better than Sol_1 because, although the cycle times (T_1) are equal, Sol_2 has a smaller workload in the second most heavily loaded station (T_2). The value δ associated with Sol_1, with values of Δ_j equal to 0 (50-50), 1 (48-47), 0 (46-46) and -1 (44-45), is obtained as follows: $\delta = \frac{0 \cdot 100^4 + 1 \cdot 100^3 + 0 \cdot 100^2 + (-1) \cdot 100}{50 \cdot 100^3} = 0.02$

The parameter δ allows us to make comparisons between solutions and with the optimum solution (if it is available); and the worse a solution is with regard to the reference solution, the larger the value of δ is.

δ also allows us to compare the obtained solutions with a lower bound of the optimum solution. A simple lower bound can be obtained taking into account n (number of tasks), m (number of workstations), t_i (processing time of task i), $t_{\max}$ (maximum processing time) and a lower bound on the cycle time $CT_{\min} = \max\left(t_{\max}; \left\lceil \sum_{i=1}^{n} t_i \Big/ m \right\rceil\right)$.

For example, let us consider an instance with $n = 11$, $m = 6$, $\sum_{i=1}^{n} t_i = 122$, $t_{\max} = 26$ and $CT_{\min} = \max(26; 21) = 26$. Then the workload in the most heavily loaded station is 26 and 96 (122-26) time units must be homogeneously distributed into 5 workstations. A lower bound of the optimum solution, which has workloads, in descending order, of 26-20-19-19-19-19 time units, respectively, can be obtained.

3.5. Computational experiment

The effectiveness of the two proposed models (GHM and SHM) was evaluated by solving all of the 302 well-known SALBP-2 instances available on Scholl and Klein's homepage for assembly line balancing research (www.assembly-line-balancing.de).

The MILP models were solved using the ILOG CPLEX 9.0 optimization software on a 3.00 GHz Pentium IV PC with 1 GB RAM; the relative MIP gap tolerance was set to 0.01. In order to compare the two MILP models designed to solve the LB-ALBP optimally, a huge calculation time would be necessary (the time that the optimization software needs to solve the mathematical models); thus, an artificial limit of 18,000 seconds was used and only the number of optimal solutions was considered.

The values of the parameters α_k $(k = 1,...,m)$ were defined as follows. Let ND be the number of digits of the value of the upper bound on the cycle time, CT; let v be the integer part of $m/2$; thus, $\alpha_k = 10^{ND\cdot(v-(k-1))}$ $(k = 1,...,m)$.

Table 2 shows the results obtained for each model (GHM and SHM): the number of instances with a proved optimal solution $(N°Opt)$; and the minimum $(t_{\min})$, average $(\bar{t})$ and maximum $(t_{\max})$ computing times, in seconds, required for the instances that are solved optimally by both models.

Insert Table 2

Table 2 shows that SHM has a better behaviour, since it is able to optimally solve 25 more instances than GHM (57 and 32 instances, respectively). For the 27 instances that are optimally solved by both models, the average computing time $(\bar{t})$ of GHM is slightly shorter than the average computing time of SHM.

It is noticed that GHM also labels a further 24 instances as optimal (although they are not). These incoherent solutions are due to the numerical precision problems that appear when the values of the artificial parameters α_k are fixed. For 3 of these 24 instances, negative workloads are assigned to workstations. In the 21 remaining instances, although the optimization software guarantees the optimal solution, the GHM does not reach the optimum; i.e., the value δ, which is obtained by comparing the GHM solution with the SHM optimal solution, is greater than 0 (the average value of δ is 49.46, and the maximum value is 251.42).

For one instance of the SHM the optimization software guarantees the optimal solution, although this solution is not optimal (although it is feasible). In this case, the value of δ, which is obtained by comparing the SHM solution with the GHM solution, is 0.0033. This difference is due to the allowed relative MIP gap tolerance, since the cycle time in this instance is very high.

4. MILP-based heuristic procedures for solving the LB-ALBP

As shown above, the best of the two proposed MILP models can optimally solve only 57 of the 302 instances. Therefore, and although formidable progress has been made in recent years in terms of both computational power and computational technology, heuristic procedures must be proposed.

Section 4 presents three heuristic procedures based on the two MILP models introduced in Section 3. These heuristics consist of running the mathematical programming models for a limited computing time.

4.1. Heuristic derived from the Global Hierarchical Model (H_GHM)

The heuristic derived from the Global Hierarchical Model (H_GHM) consists of running the GHM for 18,000 seconds and preserving the best solution obtained.

4.2. Heuristics derived from the Successive Hierarchical Model (Hx_SHM)

The heuristics derived from the Successive Hierarchical Model (Hx_SHM) consist of running SHM for 18,000 seconds and preserving the best solution obtained. In this case, we must specify how this computing time is distributed, since it is a preemptive goal programming method in which $m-1$ mixed-integer linear programming models are sequentially solved.

The heuristics H1_SHM and H2_SHM were designed according to two ways of distributing the 18,000 seconds among the various submodels to be solved.

H1_SHM consists of assigning the available computing time to the submodel to be solved, once the previous submodels have been optimally solved. The first submodel is run with a computing time limit of 18,000 seconds; if any time remains available, the second submodel is run with a time limit equal to the remaining computing time; and so on.

H2_SHM consists of assigning half of the available computing time to the submodel to be solved. The first submodel is run with a computing time limit of 9,000 seconds; next, the second submodel is run for half of the available computing time, which is greater than 4,500 seconds if the first submodel has guaranteed the optimal solution in less than the assigned 9,000 seconds; and so on. The last submodel to be solved is run with all of the remaining time until the 18,000 seconds are spent.

Note that the submodels of the H2_SHM heuristic do not always guarantee optimal solutions. Therefore, when the submodel q is solved, it is possible to find a value T_j $\left(j<q\right)$ that is smaller than the value obtained by solving the submodel j (if this submodel j did not guarantee an optimal solution). This possibility is taken into account, and the values T_j are updated when these values improve.

4.3. Computational experiment

We evaluated the effectiveness of the proposed heuristic procedures under the same conditions used in Section 3.5.

Table 3 shows, for each heuristic (H_GHM, H1_SHM and H2_SHM), the number of instances $(N^{\circ}\,Sol)$ in which a feasible solution is obtained.

Insert Table 3

Whereas the heuristics derived from the SHM solve all 302 instances, the H_GHM heuristic solves only 195 instances. The high value of the parameters α_k of the H_GHM heuristic may explain the optimization software's difficulty in finding a feasible solution.

Next, we present three analyses of the results. First, the results obtained for the 195 instances that are solved by all three heuristic procedures are briefly analyzed (Section 4.3.1). Then, the results obtained for the 302 instances that are solved by the H1_SHM and H2_SHM heuristics are analyzed (Section 4.3.2). Finally, the results obtained with a procedure for the minimization of the smoothness index are analyzed (Section 4.3.3).

It should be remembered that the worse a solution is with regard to the reference solution, the larger the value of δ is.

4.3.1. Comparison of H_GHM, H1_SHM and H2_SHM for 195 instances

This section briefly analyzes the results obtained for the 195 instances solved by all three heuristic procedures. For each heuristic, Table 4 shows the number of instances for which the heuristic obtains the best solution $(N^{\circ}\,Best)$; the minimum $(\delta_{\min})$, average $(\overline{\delta})$ and maximum $(\delta_{\max})$ values of the parameter δ (which is calculated, for each instance, by comparing each solution with the best solution obtained using the three heuristic procedures); and the minimum $(t_{\min})$, average $(\overline{t})$ and maximum $(t_{\max})$ computing times in seconds.

Insert Table 4

The H_GHM heuristic has the worst behavior. H_GHM only obtains a feasible solution for 195 of the 302 instances. Of these 195 instances, the best solution is obtained only 57 times. Moreover, when H_GHM does not obtain the best solution, its solution is very far from the best (note the high average value of the parameter δ, $\overline{\delta} = 20.24$). Furthermore, H_GHM obtains the greatest average computing times.

The H2_SHM heuristic gives slightly better results than the H1_SHM heuristic; in any event, these are partial results and an analysis of all 302 instances must be carried out.

4.3.2. Comparison of H1_SHM and H2_SHM for the 302 instances

This section analyzes the results obtained for the 302 instances solved by the heuristic procedures H1_SHM and H2_SHM. For each heuristic, Table 5 shows the number of instances for which the heuristic obtains the best solution $(N^{\circ}\,Best)$; the minimum

$(\delta_{\min})$, average $(\overline{\delta})$ and maximum $(\delta_{\max})$ values of the parameter δ (which is calculated, for each instance, by comparing each solution with the best solution obtained using the two heuristic procedures); and the minimum $(t_{\min})$, average $(\overline{t})$ and maximum $(t_{\max})$ computing times in seconds.

Insert Table 5

The H1_SHM heuristic obtains the best solution in 7 more instances than the H2_SHM heuristic. Specifically, in 161 instances, both heuristics obtain the same solution; in 74 instances, H1_SHM obtains the best solution; and in 67 instances, H2_SHM obtains the best solution.

H1_SHM has a better behavior than H2_SHM when the best solution is not obtained. The 67 solutions obtained with H1_SHM are very close to the 67 better solutions obtained with H2_SHM, as the low average value of the parameter δ ($\overline{\delta} = 0.05$) shows. On the other hand, the 74 solutions obtained with H2_SHM are farther from the 74 best solutions obtained with H1_SHM, as the larger average value of the parameter δ ($\overline{\delta} = 0.48$) shows. Moreover, the results obtained with H1_SHM have less dispersion than the results obtained with H2_SHM: the maximum values of the parameter δ $(\delta_{\max})$ are 2.95 and 20.26, respectively.

The average computing time of the H1_SHM heuristic is slightly larger than that of the H2_SHM heuristic (14,736 and 14,243 seconds, respectively).

4.3.3. Comparison with a procedure for the maximization of the smoothness index

As it is exposed, the LB-ALBP is a new problem and the *lexicographic bottleneck* objective, LB, is different from the objective of minimizing the smoothness index, SI, –as it has been introduced, for assignment problems, by Pentico (2007)–. We can observe that both objectives are similar; thus we have also solved the 302 instances with a heuristic procedure taken from the literature designed for the maximization of the SI. We have used the "Hoffman precedence matrix" heuristic together with the "trade and transfer phase" of the Moodie and Young method, HPM_T&T, which is one of the heuristics that performs best for the SI criteria (Ponnambalam et al., 1999).

The results obtained for the 302 instances solved with the heuristic procedures H1_SHM and HPM_T&T are analyzed below. For each heuristic, Table 6 shows the number of instances for which the heuristic obtains the best solution $(N^{\circ}\,Best)$; the average $(\overline{\delta})$ values and the standard deviation (σ_{δ}) of the parameter δ (which is calculated, for each instance, by comparing each solution with the best solution obtained using the two heuristic procedures); and the average $(\overline{t})$ computing times, in seconds, and the standard deviation of the computing times (σ_t).

Insert Table 6

The 302 SALBP-2 instances have been classified according to their size (number of tasks, N): i) *Low/Middle-N* $(21 \le N \le 94)$ and ii) *High-N* $(148 \le N \le 297)$. Table 7 and 8 show the values of the previous parameters (N^o *Best*, $\overline{\delta}$, σ_δ, $\overline{t}$ and σ_t) for the two set of instances, respectively.

Insert Table 7

Insert Table 8

When all instances are considered, the H1_SHM heuristic obtains the best solution for 62 more instances than the HPM_T&T heuristic; but HPM_T&T has a better global behavior than H1_SHM according to the average value of the parameter δ and the average computing time of the H1_SHM is larger than that of the HPM_T&T heuristic. Similar results are obtained when the 66 *High-N* instances are analyzed. Nevertheless, H1_SHM heuristic has a better behavior than HPM_T&T heuristic (according to the number of instances for which the heuristic obtains the best solution and the average value of the parameter δ) when the 236 *Low/Middle-N* instances are analyzed.

Finally, we compare the results obtained with both procedures for the 62 instances that we know their optimal solution (which are obtained in Section 3.5). H1_SHM heuristic obtains the optimal solution for 61 instances, with an average value of the parameter δ equal to 0.00005, whereas HPM_T&T heuristic only obtains the optimal solution for 21 instances, with an average value of the parameter δ equal to 0.35.

5. Conclusions and future research

In the type 2 assembly line balancing problem (ALBP-2), the objective is to minimize the cycle time given a number of workstations; i.e., to minimize the workload of the most heavily loaded workstation (the bottleneck). However, considering the second-biggest workload, the third-biggest workload, etc., can allow us to obtain a more reliable, better-balanced solution.

The main objective of this paper is to present, formalize and solve a new assembly line balancing problem: the *Lexicographic Bottleneck Assembly Line Balancing Problem* (LB-ALBP). The LB-ALBP hierarchically minimizes the workload of the most heavily loaded workstation, followed by the workloads of the second and third most heavily loaded workstations, and so on. Two mixed-integer linear programming (MILP) models (called GHM and SHM, respectively) are designed to optimally solve the LB-ALBP, but the number of instances that are solved optimally is not operative. Finally, taking advantage of the designed mathematical programs, we design and evaluate three heuristic procedures based on these MILPs.

Future research work may involve designing and analyzing different ad hoc heuristic and metaheuristic methods for solving the LB-ALBP. An idea to be explored consists of solving the LB-ALBP with a heuristic for the SALBP-2. Then, the tasks assigned to the

most heavily loaded workstation are fixed. Next the two subproblems generated, one with the tasks assigned before the fixed workstation and the another with the tasks assigned after the fixed workstation, are independently solved with the same heuristic for SALBP-2; and so on.

The distribution of the most critical workstations of an assembly line is another area of future research.

6. Acknowledgments

The author is very grateful to Professor Albert Corominas and to Alberto García (Technical University of Catalonia) for their valuable comments which have helped to enhance this paper. The author wish to express his gratitude to the anonymous reviewers for their detailed and valuable comments, which have helped to improve the quality of this paper.

7. References

Ağpak, K., Gökçen, H., 2005. Assembly line balancing: Two resource constrained cases. *International Journal of Production Economics*, 96(1), 129-140.

Amen, M., 2001. Heuristic methods for cost-oriented assembly line balancing: A comparison on solution quality and computing time. *International Journal of Production Economics*, 69, 255-264.

Amen, M., 2006. Cost-oriented assembly line balancing: Model formulations, solution difficulty, upper and lower bounds. *European Journal of Operational Research*, 168, 747-770.

Andrés, C., Miralles, C., Pastor, R., 2008. Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, 187, 1212-1223.

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

Baykasoglu, A., Dereli, T., Erol, R. Sabancu, I., 2003. *An ant colony based optimization algorithm for solving assembly line balancing problems.* International XII Turkish Symposium on Artificial Intelligence and Neural Networks – TAINN.

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

Boysen, N., Fliedner, M., Scholl, A., 2006. A classification of assembly line balancing problems. *Working paper 12/2006*, Friedrich-Schiller-Universität Jena.

Boysen, N., Fliedner, M., Scholl, A., 2007. A classification of assembly line balancing problems. *European Journal of Operational Research*, 183, 674-693.

Boysen, N., Fliedner, M., Scholl, A., 2008. Assembly line balancing: Which model to use when? *International Journal of Production Economics*, 111 (2), 509-528.Burkard, R.E., Rendl, F., 1991. Lexicographic bottleneck problems. *Operations Research Letters*, 10, 303–308.

Capacho, L., Pastor, R., 2008. ASALBP: the Alternative Subgraphs Assembly Line Balancing Problem. *International Journal of Production Research*, 46, 3503-3516.

Capacho, L., Pastor, R., Dolgui, A., Gunshinskaya, O. 2009. An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem. *Journal of Heuristics*, 15, 109-132.

Corominas, A., Pastor, R., Plans, J., 2008. Balancing assembly line with skilled and unskilled workers. *OMEGA*, 36, 1126-1132.

Corominas, A., Pastor, R., 2009. A note on "A comparative evaluation of assembly line balancing heuristics". *International Journal of Advanced Manufacturing Technology*, 44, 817.

Cortés, P., Larrañeta, J., Onieva, L., García J.M., Caraballo, M.S., 2001. Genetic algorithm for planning cable telecommunication networks. *Applied Soft Computing*, 1, 21-33.

Cortés, P., Muñuzuri, J., Onieva, L., Larrañeta, J., Vozmediano J.M., Alarcón, J.C., 2006. Andalucía assesses the investment needed to deploy a fiber-optic network. *Interfaces*, 36, 105-117.

Cortés, P., Onieva, L., Guadix, J., 2009. Optimising and simulating the assembly line balancing problem in a motorcycle manufacturing company: a case study. *International Journal of Production Research*, doi: 10.1080/00207540902926522.

Ding, F-Y., Zhu, J., Sun, H., 2006. Comparing two weighted approaches for sequencing mixed-model assembly lines with multiple objectives. *International Journal of Production Economics*, 102 (1), 108-131.

Erel, E., Sarin, S.C., 1998. A survey of the assembly line balancing procedures. *Production Planning & Control*, 9, 414-434.

Gamberini, R., Grassi, A., Rimini, B., 2006. A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem. *International Journal of Production Economics*, 102, 226-243.

Ghosh, S., Gagnon, R.J., 1989. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of production research*, 27, 637-670.

Gökçen, H., Ağpak, K., Benzer, R., 2006. Balancing of parallel assembly lines. *International Journal of Production Economics*, 103(2), 600-609.

Helgeson, W.P., Birnie, D.P., 1961. Assembly line balancing using Ranked Positional Weight Techniques. *Journal of Industrial Engineering*, 12, 394-398.

Hoffman, T.R., 1992. EUREKA: a hybrid system for assembly line balancing. *Management Science*, 38, 39-47.

Johnson, R.V., 1988. Optimally balancing large assembly lines with Fable. *Management Science*, 34, 240-253.

Kao, E.P.C., Queyranne, M., 1982. On dynamic programming methods for assembly line balancing. *Operations Research*, 30, 375-390.

Kim, Y.K., Kim, Y.J., Kim, Y., 1996. Genetic algorithms for the assembly line balancing with various objectives. *Computers Industrial Engineering*, 30, 397-409.

Krajewski, L.J., Ritzman, L.P., 1979. Disaggregation in manufacturing and service organizations: Survey of problems and research, en Ritzman, L.P., Krajewski, L.J., Berry, W.L., Goodman, S.H., Hardy, S.T. and Vitt, L.D Eds. *Disaggregation. Problems in Manufacturing and Service Organizations*, Martinus Nijholff Publishing.

Malakooti, B., Kumar, A., 1996. A knowledge-based system for solving multi-objective assembly line balancing problems. *International Journal of Production Research*, 34, 2533-2552.

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

Miyazaki, S., Ohta, H., 1987. *An optimal solution of the assembly line balancing problem with the balancing index as a primary objective function.* Collection: System Modelling and Optimization, 681-690.

Moodie, C.L., Young, H.H., 1965. A heuristic method of assembly line balancing for assumptions of constant or variable work element times. *Journal of Industrial Engineering*, 16, 23-29.

Park, K., Park, S., Kim, W., 1997. A heuristic for an assembly line balancing problem with incompatibility, range, and partial precedence constraints. *Computers & Industrial Engineering*, 32, 321-332.

Pastor, R., Andrés, C., Duran, A., Pérez, M., 2002. Tabu search algorithms for an industrial multi-product and multi-objective assembly line balancing problem, with reduction of the tasks dispersion. *Journal of the Operational Research Society*, 53, 1317-1323.

Pastor, R., Corominas, A., 2000. Assembly line balancing with incompatibilities and bounded workstation loads. *Ricerca Operativa*, 30, 23-45.

Pastor, R., Ferrer, L., 2009. An improved mathematical program to solve the simple assembly line balancing problem. *International Journal of Production Research*, 47, 2943-2959.

Pastor, R., Andrés, C., Miralles, C., 2010. Corrigendum to "Balancing and scheduling tasks in assembly lines with sequence-dependent setup" [European Journal of Operational Research 187 (2008) 1212-1223]. *European Journal of Operational Research*, 201, 336.

Pentico, D.W., 2007. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176, 774-793.

Ponnambalam, S.G., Aravindan, P., Mogileeswar, G., 1999. A comparative evaluation of assembly line balancing heuristics. *International Journal of Advanced Manufacturing Technology*, 15, 577–586.

Rekiek, B., de Lit, P., Delchambre, A., 2002a. A hybrid assembly line design and user's preferences. *International Journal of Production Research*, 40, 1095-1111.

Rekiek, B., Dolgui, A., Delchambre, A., Bratcu, A., 2002b. State of art of optimization methods for assembly line design, *Annual Reviews in Control*, 26, 163-174.

Rubinovitz, J., Levitin, G., 1995. Genetic algorithm for assembly line balancing. *International Journal of Production Economics*, 41, 343-354.

Scholl, A., 1999. Balancing and sequencing of assembly lines. *Physica-Verlag Heidelberg*: Germany. 2nd edition.

Scholl, A., Becker, C., 2006. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168, 666-693.

Scholl, A., Klein, R., 1997. SALOME: a bidirectional branch and bound procedure for assembly line balancing. *Informs Journal on Computing*, 9, 319-334.

Sokkalingam, P.T., Aneja, Y.P., 1998. Lexicographic bottleneck combinatorial problems. *Operations Research Letters*, 23, 27–33.

Starr, M.K., 1979. Perspectives on disaggregation, en Ritzman, L.P., Krajewski, L.J., Berry, W.L., Goodman, S.H., Hardy, S.T. and Vitt, L.D Eds. *Disaggregation.*

Problems in Manufacturing and Service Organizations, Martinus Nijholff Publishing.

Suresh, G., Sahu, S., 1994. Stochastic assembly line balancing using simulated annealing. *International Journal of Production Research*, 32, 1801-1810.

Talbot, F.B., Patterson, J.H., 1984. An integer programming algorithm with network cuts for solving the assembly line balancing problem. *Management Science*, 30, 85-99.

Talbot, F., Patterson, J.H., Gehrlein, W.V., 1986. A comparative evaluation of heuristic line balancing techniques. *Management Science*, 32, 431-453.

Wee, T.S., Magazine, M.J., 1982. Assembly line balancing as generalized bin packing. *Operations Research Letters*, 1, 56-58.

White, W.W., 1961. Comments on a paper by Bowman. *Operations Research*, 9, 274-276.

Optimal solution	Sta_1	Sta_2	Sta_3	Sta_4	Sta_5	*LB* value	*SI* value
LB	3-4(29)	2-6(33)	1-5-7(33)	8-10(34)	9(19)	34-33-33-29-19	15.87
SI	3-4(29)	1-2(25)	5-10(29)	6-7(30)	8-9(35)	35-30-29-29-25	14.04

Table 1. Optimal solutions for each objective

Model	$N°\,Opt$	t_{min}	$\overline{t}$	t_{max}
GHM	32	0	561	7,725
SHM	57	0	715	14,513

Table 2. Results of the mathematical programming models

Heuristic	**H_GHM**	**H1_SHM**	**H2_SHM**
$N°\,Sol$	195	302	302

Table 3. Number of instances with obtained solution

Heuristic	$N°\,Best$	δ_{min}	$\overline{\delta}$	δ_{max}	t_{min}	$\overline{t}$	t_{max}
H_GHM	57	0	20.24	256.64	0	13,273	18,000
H1_SHM	144	0	0.06	2.54	0	13,038	18,000
H2_SHM	162	0	0.11	5.32	0	12,271	18,000

Table 4. Results of the heuristic procedures for 195 instances

Heuristic	*N° Best*	δ_{min}	$\overline{\delta}$	δ_{max}	t_{min}	$\overline{t}$	t_{max}
H1_SHM	235	0	0.05	2.95	0	14,736	18,000
H2_SHM	228	0	0.48	20.26	0	14,243	18,000

Table 5. Results of the heuristic procedures for 302 instances

Heuristic	*N° Best*	$\overline{\delta}$	σ_{δ}	$\overline{t}$	σ_t
H1_SHM	193	1.22	3.32	14,736	6,847
HPM_T&T	131	0.55	1.29	3,864	11,979

Table 6. Results of the heuristic procedures H1_SHM and HPM_T&T (302 instances)

Heuristic	*N° Best*	$\overline{\delta}$	σ_{δ}	$\overline{t}$	σ_t
H1_SHM	184	0.43	1.64	14,052	7,346
HPM_T&T	72	0.69	1.42	550	1,356

Table 7. Results of the heuristic procedures H1_SHM and HPM_T&T (236 *Low/Middle-N* instances)

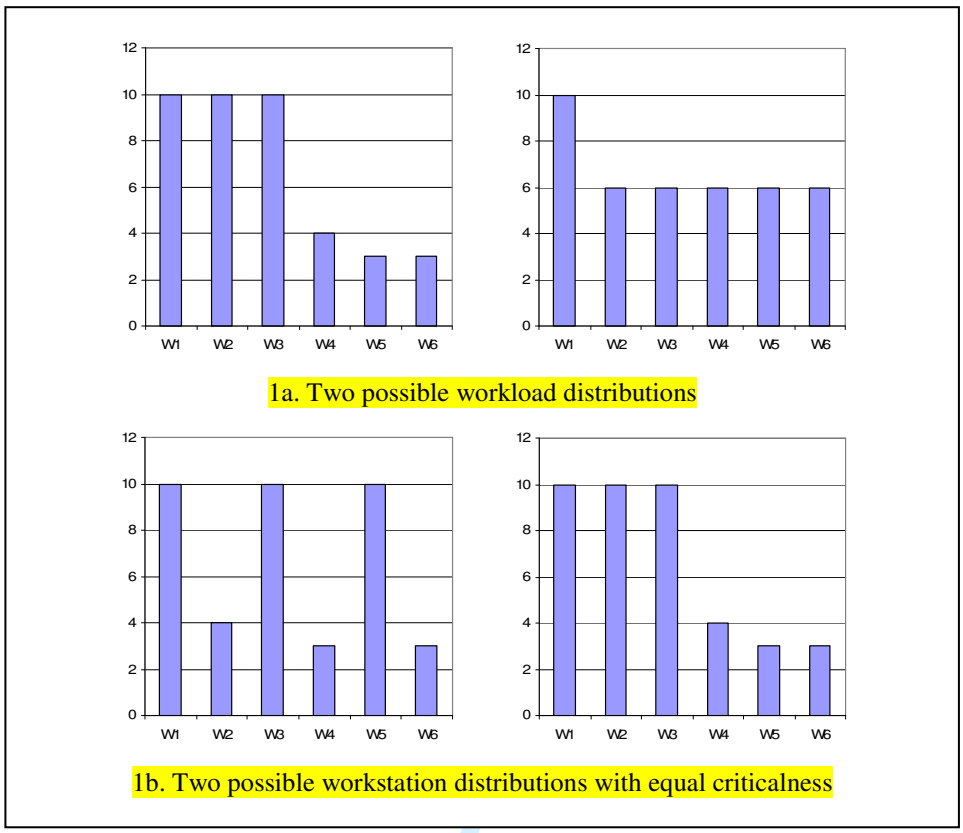
Heuristic	*N° Best*	$\overline{\delta}$	σ_{δ}	$\overline{t}$	σ_t
H1_SHM	9	4.02	5.58	17,185	3,373
HPM_T&T	59	0.04	0.17	15,716	21,803

Table 8. Results of the heuristic procedures H1_SHM and HPM_T&T (66 *High-N* instances)

FIGURE CAPTION

Figure 1. Possible workload distributions

Figure 2. Assembly line balancing instance



1a. Two possible workload distributions

1b. Two possible workstation distributions with equal criticalness

Figure 1. Possible workload distributions

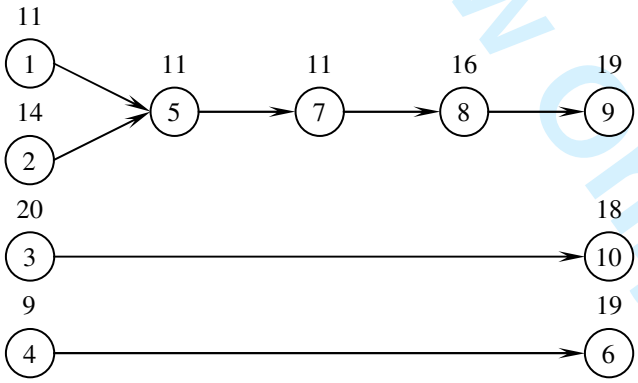


Figure 2. Assembly line balancing instance