Heuristic procedures for solving the General Assembly Line Balancing Problem with Setups (GALBPS)

Luigi Martino, Rafael Pastor

Heuristic procedures for solving the General Assembly Line Balancing Problem with Setups (GALBPS)

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2008-IJPR-0602.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	01-Oct-2008
Complete List of Authors:	Martino, Luigi; Technical University of Catalonia Pastor, Rafael; Universidad Politécnica de Cataluña, Instituto de Organización y Control de Sistemas Industriales
Keywords:	ASSEMBLY LINE BALANCING, ASSEMBLY LINES
Keywords (user):	sequence-dependent setup times, production

Heuristic procedures for solving the General Assembly Line Balancing Problem with Setups (GALBPS)[ϒ]

Luigi Martino and Rafael Pastor[*]
IOC Research Institute
Technical University of Catalonia
Av. Diagonal 647, 11th floor, 08028, Barcelona, Spain
luigi.martino@gmx.de / rafael.pastor@upc.edu

Abstract: The General Assembly Line Balancing Problem with Setups (GALBPS) was recently defined in the literature. It adds sequence-dependent setup time considerations to the classical Simple Assembly Line Balancing Problem (SALBP) as follows: whenever a task is assigned next to another at the same workstation, a setup time must be added to compute the global workstation time, thereby providing the task sequence inside each workstation. This paper proposes heuristic procedures, based on priority rules, for solving GALBPS, many of which are an improvement upon heuristic procedures published to date.

Keywords: assembly line balancing, sequence-dependent setup times, production

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 to the numerous tasks and constraints characteristic of real-life situations. The classic Assembly Line Balancing Problem (ALBP) basically consists of assigning a set of tasks (each characterized by its processing time) to an ordered sequence of workstations, such that the precedence constraints between tasks are maintained and a given efficiency measure is optimized.

The problem of designing and balancing assembly lines has been examined extensively in the literature. A number of surveys have been published, including Baybars (1986), Ghosh and Gagnon (1989), Erel and Sarin (1998), Scholl (1999), Rekiek *et al.* (2002), Becker and Scholl (2006), Scholl and Becker (2006) and Boysen *et al.* (2007). However, most of these papers focus on the simple ALBP (SALBP). This problem has been approached using heuristic procedures (*e.g.*, Talbot *et al.* (1986) and Ponnambalam *et al.* (1999)) as well exact procedures (*e.g.*, Scholl and Klein (1997) and Pastor and Ferrer (2008)). Myriad complex cases have been examined, including problems that consider lines with parallel workstations or parallel tasks (e.g., Bukchin and Rubinovitz (2003)); mixed or multi-models (e.g., Bard *et al.* (1992)); multiple products (e.g., Pastor *et al.* (2002)); U-shaped, two-sided, buffered or parallel lines (e.g., Miltenburg (2001)); incompatibility between tasks (e.g., Park *et al.* (1997)); processing times that depend on the sequence (e.g., Capacho and Pastor (2008)) or on the operator (e.g., Corominas *et al.* (2008)) or are stochastic (e.g., Gamberini *et al.* (2006)); and equipment selection (*e.g.*, Amen (2006)). Consequently, generalized problems have garnered much interest.

[ϒ] Supported by the Spanish MCyT 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.

Articles on assembly line balancing typically focus on the problem in a pure sense—as if, once the tasks were assigned to the workstations, there was nothing left to do. However, in some real production lines, the sequence in which tasks are executed inside the workstation is very important, since there are sequence-dependent setup times between tasks. Andrés *et al.* (2008) introduced the General Assembly Line Balancing Problem with Setups (GALBPS). GALBPS not only requires that the assembly line has to be balanced, but also that the sequence of tasks assigned to every workstation must be defined (due to the existence of sequence-dependent setup times). Therefore, both the inter-station balancing and intra-station task sequencing must be solved simultaneously. This reflects a more realistic scenario for many assembly lines, especially those from the electronics industry or similar sectors featuring low cycle times.

In this paper, we propose heuristic procedures, based on priority rules, for solving GALBPS, many of which are an improvement upon heuristic procedures published to date.

The remainder of the paper is organized as described below. GALBPS is outlined in Section 2, and the heuristic procedures designed to solve it are explained in Section 3. These heuristic procedures were tested and evaluated through a computational experiment, the main results of which are presented in Section 4. Finally, conclusions on this work and ideas for further research are presented in Section 5.

2. The General Assembly Line Balancing Problem with Setups

GALBPS adds sequence-dependent setup time considerations to the classical Simple Assembly Line Balancing Problem (SALBP) as follows: whenever a task j is assigned next to another task i at the same workstation, a setup time $\tau_{i,j}$ must be added to compute the global workstation time, thereby providing the task sequence inside each workstation. Furthermore, if a task p is the last one assigned to the workstation in which task i was the first task assigned, then a setup time $\tau_{p,i}$ must also be considered. This is because the tasks are repeated cyclically; the last task in one cycle of the workstation is performed just before the first task in the next cycle.

Hence, GALBPS consists of assigning a set of tasks to an ordered sequence of workstations, such that the precedence constraints between tasks are maintained, the setup times between tasks are considered and a given efficiency measure is optimized. As in the classification of Baybars (1986), when the objective is to minimize the number of workstations for a given upper bound on the cycle time, the problem is referred to as GALBPS-1; when the objective is to minimize the cycle time given a number of workstations, the problem is called GALBPS-2. Herein are presented improved heuristic procedures based on priority rules to solve GALBPS-1.

As an example, we can take a case in which there are three tasks (A, B and C) assigned to a workstation and having processing times (t_i) of $t_A = 10$, $t_B = 12$ and $t_C = 9$, respectively. Moreover, we consider that are no precedence constraints between the tasks, and that the setup times $\left(\tau_{i,j}\right)$ are the following: $\tau_{A,B} = 3$, $\tau_{A,C} = 4$, $\tau_{B,A} = 2$, $\tau_{B,C} = 1$, $\tau_{C,A} = 3$ and $\tau_{C,B} = 4$. Table 1 shows two possible sequences for the three

tasks, with the times to be considered as well as the global workstation time (which equals the sum of all processing times and setup times). As observed, the two solutions differ by three units of time.

Insert Table 1

In most industrial assembly lines these setup times exist but are usually not considered because they are very short compared to processing times. In certain cases, the setup times do not depend on the sequence of tasks, and are added to the processing times of the tasks. In other cases, the task sequence for every workstation is defined only after the tasks have been assigned and the line has subsequently been balanced; the problem is therefore solved in two separate stages. However, a better strategy to solve GALBPS is to simultaneously solve the line-balancing and the task-sequencing problems.

Andrés *et al.* (2008) introduced GALBPS and provided different real examples, including that of workers using different tools for different tasks and that of robotic lines. What is important in this situation is to define the best work sequence for the worker in order to minimize the global workstation time, including setup times. Robotic lines are another real case: often, the robot must remove one tool, select the corresponding new tool from a set and then make adjustments before starting the next assigned task. As mentioned in Graves and Lamar (1983), tool changes are especially important in robotic assembly because they may involve times that are comparable in magnitude to operation times. Another practical case is that in which components are located in separate containers: the time required to get to one container depends on the last component that was assembled for the product.

An overview of the relevant literature reveals a shortage of publications on this topic. On the one hand, we have focused on literature about scheduling research involving setup considerations (Allahverdi *et al*. (1999, 2008) and Zhu and Wilhelm (2006)), but we were unable to find any references to evaluation of the work sequence inside the assembly line.

On the other hand, we referred to the surveys on problems and methods in assembly line balancing commented in Section 1. In these, setup times are only included when mixed-model and multi-model lines are considered. However, in both cases the sequence refers to the products or models to be assembled on the line, not to the work sequence of tasks inside the workstations.

One survey which does include the sequence-dependent task time increments is Boysen et al. (2007), in which it is commented that if two tasks are executed at a station, one directly after the other, setup time may be required for tool changes and repositioning of workpieces (Arcus (1966) or Wilhelm (1999)). In that paper it is also commented that sequence-dependent time increments occur if the status achieved by completing particular tasks has an effect on the processing time of other tasks which are executed later in the same or another station. This problem is handled in Scholl *et al*. (2008), in which the sequence-dependent assembly line balancing problem (SDALBP) is defined, and in Capacho and Pastor (2008); however, the aforementioned problem is not the same as the problem at hand: in GALBPS a setup time $\tau_{i,j}$ must be considered whenever a task j is assigned next to another i at the same workstation.

Finally, in Sawik (2002) both the line balancing problem and the sequencing problem are handled simultaneously for the specific case of printed circuit board production lines; whereas Agnetis and Arbib (1997) face a related problem consists of assigning operations to machines, and then sequencing them in every workstation to maximize defined performance indicators.

For a cyclic case in which the tasks p and i are the last and first assigned to a given workstation, respectively, then a setup time $\tau_{p,i}$ must be also considered. However, the majority of works cited above do not apply it. Only Andrés *et al.* (2008) describe rapid and facile solution procedures that can be applied by any practitioner. Specifically, that paper, after introducing GALBPS and modelling GALBPS-1 through a binary programming model (which only provides optimal solutions for very small instances), designs eight heuristic priority rules and presents a GRASP algorithm.

3. The heuristic procedures

In ALBP, most heuristic algorithms are based on generating feasible solutions by successively assigning tasks, or subsets of tasks, to workstations. Therefore, these algorithms consider *partial solutions* containing a number of assigned tasks and (partial) workstation loads, whereas the remaining tasks and workstation idle times constitute a *residual problem* (Scholl and Becker, 2006). The aim is to assign tasks to workstations and sequence them such that no precedence relationships are violated, and the value global time (including setup times) is less than the cycle time. Almost every solution procedure is based on one of the two following construction schemes (introduced in Subsection 3.2 and 3.3), which define the main way of assigning tasks to workstations: workstation-oriented and task-oriented assignment.

This Section is organized as follows: the terminology used is presented (Subsection 3.1); the workstation-oriented procedures based on not-weighted priority rules are described (Subsection 3.2); use of the task-oriented procedure and designed heuristic rules for said procedure are explained (Subsection 3.3); the workstation-oriented procedures based on weighted priority rules (which are fine-tuned by means of the Nelder and Mead algorithm) are introduced (Subsection 3.4); and, finally, improved tasks assignation schemes within a workstation are described (Subsection 3.5).

3.1. Terminology

The principal data and parameters used are described below:

i, j	index for the tasks
k	index for the workstations
N	number of tasks $(i = 1,..., N)$
TC	upper bound on the cycle time
S_i	set of successor tasks, at any step, of the task i
P_i	set of preceding tasks, at any step, of the task i
NS_i	number of successor tasks, at any step, of the task i
NIS_i	number of immediate successors of task i

t_i processing time of task i

$\tau_{i,j}$ setup time when task j is performed directly after task i inside the same workstation, assuming that $\tau_{i,i} = 0$

$\tau_{last,i}$ setup time between the last task assigned to the workstation which is being completed and the task i

$\tau_{i,first}$ setup time between the task i and first task assigned to the workstation which is being completed

$\bar{\tau}_i$ average setup time of the task i (between i and either its successor or preceding tasks, at any step)

E_i, L_i earliest and latest workstation, respectively, to which task i can be assigned to (see, *e.g.*, Scholl, 1999). The calculation of the range of workstations $[E_i, L_i]$ also considers the minimum number of setup times between the task i and either its successor or preceding tasks.

3.2. *Workstation-oriented procedure based on not-weighted priority rules (WH)*

The *workstation-oriented procedure (WH)* is an iterative procedure which, at each iteration and according to a priority rule, assigns one of a group of *candidate* tasks to the workstation k which is being completed. A task i is considered a candidate once its preceding tasks have been assigned and it fits in the workstation k. If there are no candidate tasks available, but there are still tasks left to assign, then k is closed, and workstation $k+1$ is opened. The procedure ends once all of the tasks have been assigned.

A vital element in the definition of the *WH* procedure is the definition of the priority rule, which orders the candidate tasks at the time of choosing the next task to be assigned. Table 2 lists the not-weighted priority rules used in the *WH* procedure. In all cases, the task x^* is assigned with $v_{x^*} = \max_i v_i$. Rules called *A-01* to *A-04* are described for GALBPS in Andrés *et al.* (2008); and priority rules denoted *R-01* to *R-12* are new rules developed in this work.

Insert Table 2

Trying to comprehend the influence of setup times on selecting tasks, and consequently design proprity rules appropriate for GALBPS, we analyzed firstly the most common priority rules in the literature for the SALBP: t_i by Moodie and Young (1965); NS_i, $\frac{t_i + \sum_{j \in S_i} t_j}{NS_i + 1}$, $-E_i$, $-L_i$, $-(L_i - E_i)$, $\frac{t_i}{L_i}$, $-\left(\frac{L_i}{NS_i + 1}\right)$ and $\frac{NS_i}{L_i - E_i}$ by Talbot *et al.*, (1986); NIS_i by Tonge (1961); $t_i + \sum_{j \in S_i} t_j$ by Helgeson and Birnie (1961); and $t_i + NS_i$ by Bhattacharjee and Sahu (1988). The new priority rules (*R-01* to *R-12*) are based on taking into account the elements of the priority rules for SALBP reported in the

literature, which do not take into account setup times between tasks (*i.e.* t_i, NS_i, NIS_i, $\sum_{j \in S_i} t_j$, E_i and L_i), as well as the setup times between tasks ($\tau_{last,i}$, $\tau_{i,first}$ and $\bar{\tau}_i$).

3.3. *Task-oriented procedure (TH)*

The *task-oriented procedure (TH)* is an iterative procedure which, at each iteration and according to a priority rule, assigns one of a group of *candidate* tasks to a workstation. A task is considered a candidate once all of its preceding tasks have been assigned. The chosen task is assigned to the first workstation in which it can be assigned (provided that it fits in the workstation and that all of its preceding tasks have been assigned). All of the workstations remain open until all of the tasks have been assigned, at which point the procedure ends.

As mentioned in Andrés *et al.* (2008), most computational experiments reported in the literature indicate that, for SALBP, workstation-oriented procedures provide better results than task-oriented ones, although they are not theoretically dominant (Scholl and Voß, 1996). In addition, task-oriented procedures imply much higher computation times. All of the priority rules designed for the workstation-oriented procedure can be used here. However, in line with the aforementioned comments, only the priority rules shown in Table 3 were tested. In all cases the task x^* is assigned with $v_{x^*} = \max_i v_i$. Rules *A-01* to *A-04* were tested by Andrés *et al.* (2008); and priority rules *R-05*, *R-08* and *R-09* are tested in this work.

Insert Table 3

3.4. *Workstation-oriented procedure based on weighted priority rules (WH _ NM)*

In this Subsection, the workstation-oriented procedure based on weighted priority rules (which are fine-tuned by means of the Nelder and Mead algorithm) is introduced.

Analysis of the results of preliminary computational tests revealed that the best results are obtained when assignment of tasks with the following characteristics is prioritized: those with the longest processing time (t_i); those with the shortest setup time with the last task assigned to the workstation which is being completed ($\tau_{last,i}$); and those with the most successor tasks (NS_i) or those which have longest times of their successor tasks, considering the average setup time of these successor tasks ($\sum_{j \in S_i} \left(t_j + \bar{\tau}_j \right)$). Table 4 shows the three new weighted priority rules (*R-13* to *R-15*) that were designed for consideration (again, the task x^* is assigned with $v_{x^*} = \max_i v_i$), illustrating the advantages of the three characteristics. The parameter $\tau_{last,i}$ is negative, since the smallest values are preferred.

It would certainly be interesting to consider the setup time between the candidate task and the next task in the sequence $\left(\tau_{i,next} \right)$; however, the latter may be unknown. If the candidate task is definitely the last one that can be sequenced in the workstation which

is being completed (*i.e.* no additional task would fit), then $\tau_{i,next} = \tau_{i,first}$. However, in the contrary case, $\tau_{i,next} = \bar{\tau}_{i,new\_candidates}$, whereby $\bar{\tau}_{i,new\_candidates}$ is the average setup time between the task i and the candidate tasks present once it has been sequenced. Table 4 shows the new weighted priority rule (*R-16*) that was designed for consideration.

Insert Table 4

In the previous priority rules, the weight of each of their elements had to be fine-tuned. Fine-tuning the parameters of a new heuristic is almost always difficult. The parameters greatly influence the results of the heuristic; hence, their values are crucial. Nonetheless, fine-tuning is usually done by intuitively testing several values. For fine-tuning, we used EAGH (Empirically Adjusted Greedy Heuristics), introduced in Corominas (2005). EAGH is a procedure to design greedy algorithms for a given combinatorial optimization problem, whose starting point is to consider greedy heuristics as members of an infinite set, H, defined by a function that depends on several parameters (in our case, each of the rules shown in Table 4). Searching for the best element of H can then be approached as an optimization problem, for which the solution consists of finding the parameter values that optimize the value of the objective function for the problem being solved. Since the set of instances of a problem is infinite, we must resign ourselves to a representative training set for performing the optimization.

EAGH employs the Nelder and Mead (N&M) algorithm (Nelder and Mead, 1965; Lagarias *et al.*, 1998) for solving the fine-tuning problem because it is a direct one (*i.e.* it uses only the values of the function). Albeit other algorithms could be used to solve this fine-tuning optimization problem, the N&M algorithm has yielded good results since its publication and is referred to in recent papers (Anjos *et al.*, 2004; Chelouah and Siarry, 2005). A detailed description of the N&M algorithm can be found in the publications cited above.

A set of 64 training instances (generated as explained in Section 4) was used to fine-tune the priority rules shown in Table 4. The new, fine-tuned priority rules are shown in Table 5 (the values of the parameters have been rounded to the first decimal place).

Insert Table 5

As observed, the values of the parameters δ_2, δ_3 and δ_4 are lower than those of the other parameters. This does not imply that $\sum_{j \in S_i} \left(t_j + \lambda \cdot \bar{\tau}_j \right)$ is less important, as the values have not been normalized, and $\sum_{j \in S_i} \left(t_j + \lambda \cdot \bar{\tau}_j \right)$ tends to have a much higher value than the other elements considered.

3.5. Improved tasks assignation schemes within a workstation

In this Subsection, improved tasks assignation schemes within a workstation are introduced into the workstation-oriented procedure (*WH*). These are based on considering all of the positions at which a candidate task can be assigned (Subsection 3.5.1); performing a local optimization of the tasks assigned to a workstation, once the workstation can be considered closed (Subsection 3.5.2); and performing a local

optimization of the tasks assigned to a workstation, every time that a new task is assigned there (Subsection 3.5.3).

3.5.1. The position at which a candidate task can be assigned to (WH _ pos)

In the *WH* procedure, a task i is always assigned after the last task assigned to the workstation k which is being completed. Completion of said condition yields a set of candidate tasks and enables calculation of the priority rule associated with each of them.

In the *WH _ pos* procedure, a task i can also be assigned to intermediate positions in the partial task sequence that have already been assigned to the workstation k. Obviously, in this case precedence among tasks must be respected, and, considering the setup times for assigning a task i to position s of the sequence, the task i must fit in the workstation k. A task i can thereby have different values for the priority rule (as long as the rule accounts for setup times): one value for each possible position s of the sequence in which i can be assigned. The greatest value of the priority rule is assigned to the task i for all possible positions s at which i can be assigned. In the event of a tie, the position s which corresponds to the lowest value of the sum of the setup time with the previous task in the sequence, the processing time, and the setup time with the following task in the sequence is assigned.

As may be deduced, the number of candidate tasks can—and does—increase: once a non-candidate task is sequenced after the last assigned task, it can become a candidate when it is assigned to an intermediate position of the partial sequence of already assigned tasks.

A set of four priority rules (*R-05*, *R-08*, *R-09* and *R-14*) which gave good results and used complementary criteria were tested with the *WH _ pos* procedure.

3.5.2. Local optimization of the tasks assigned to a workstation (WH _ swap)

The *WH _ swap* procedure consists of performing local optimization of the sequence of tasks assigned to the workstation k which has just closed because no additional tasks can fit, before opening a new workstation $k+1$. As a result of said optimization, the tasks assigned to the workstation k can be tracked (*i.e.* candidate tasks reappear).

The procedure used for local optimization consists of iteratively calculating all of the neighbouring sequences of a given sequence of tasks in the workstation k $\left(Seq_{current}\right)$ and then substituting $Seq_{current}$ with the best neighbouring sequence $\left(Seq_{current}^{best\_nei}\right)$. The local optimization continues as long as $Seq_{current}^{best\_nei}$ is better than $Seq_{current}$, and stops once $Seq_{current}^{best\_nei}$ is not better than $Seq_{current}$. For a given sequence $Seq_{current}$, only feasible neighbouring sequences are considered. A sequence Seq_1 is considered to be better than another sequence Seq_2, if it has a shorter total required time (the sum of processing and setup times) than Seq_2. The neighbouring sequences of $Seq_{current}$ are generated by swapping the tasks assigned to every pair of its consecutive positions in the workstation k.

WH_swap was tested with the same priority rules used to test WH_pos.

3.5.3. Local optimization of the tasks assigned to a workstation after each assignment (WH_opt)

The WH_opt procedure consists of performing a local optimization of the sequence of tasks assigned to the workstation k which is being completed; such optimization takes place every time that a new task is assigned to the workstation. WH_opt differs from WH_swap in that the neighbouring sequences of $Seq_{current}$ are generated by inserting every task assigned to the workstation k at each possible position of the sequence.

In the WH_opt procedure, in order to increase the number of candidate tasks, a task i is initially assigned after the last task assigned to the workstation k, and then the local optimization described in the previous paragraph is immediately performed. This differs from the procedure WH (whereby the task i is assigned after the last task assigned to the workstation k which is being completed), and from the procedure WH_pos (in which the task i is assigned to the intermediate positions of the partial sequence of tasks already assigned to the workstation k). Here, only feasible sequences are considered.

The number of candidate tasks can and does increase: a non-candidate task, having not been sequenced in any position of the partial sequence of already assigned tasks, can become a candidate upon execution of the local optimization. WH_opt was tested with the same priority rules used to test WH_pos and WH_swap.

4. Computational experiment

The heuristic procedures proposed in Section 3 were tested with a set of self-made instances. The results demonstrate that some of the heuristic procedures based on the new priority rules improve upon those described to date (the best of which are described in Andrés *et al.* (2008)), including the metaheuristic GRASP proposed in the aforementioned paper.

This Section is broken down as follows: the method used to generate the set of benchmark instances is detailed (Subsection 4.1); a lower bound on GALBPS and a GRASP metaheuristic both defined by Andrés *et al.* (2008) are briefly introduced (Subsection 4.2 and 4.3); and lastly, the results of the computational experiment and a discussion of results are provided (Subsection 4.4 and 4.5).

4.1. Generation of benchmark instances

Since GALBPS is a novel problem, there is no set of benchmark instances with setup times available for testing. Therefore a set of self-made instances was generated from a well-known set of problems obtained from Scholl's and Klein's assembly line balancing research website (Scholl and Klein, 2008). The basic data used for the experiment are as follows:

- 16 instances from Scholl's and Klein's website were used. Table 6 lists each instance with its respective name; number of tasks (N); minimum, maximum and average processing times of the tasks ($t_{\min}$, $t_{\max}$ and $\overline{t}$, respectively); *order strength* of the precedence graph (OS); and upper bounds on the minimum and the maximum cycle times ($TC_{\min}$ and $TC_{\max}$, respectively). The instances contain a wide range of values of the cycle time (from 11 to 10,816 units of time), number of tasks (from 21 to 297 tasks), *order strength* of the precedence graph (from 22.49 to 83.82) and average task processing time (from 5 to 912.1 units of time). These values were considered to be sufficiently representative.
- Four levels of variability of the setup times were set. The setup times were randomly generated according to a uniform discrete distribution $U\left[0,\ \lceil 0.25 \cdot t_{\min} \rceil\right]$, $U\left[0,\ \lceil 0.75 \cdot t_{\min} \rceil\right]$, $U\left[0,\ \lceil 0.25 \cdot \overline{t} \rceil\right]$ and $U\left[0,\ \lceil 0.75 \cdot \overline{t} \rceil\right]$.
- Ten instances were created from each problem by randomly generating the upper bound on the cycle time according to a uniform discrete distribution $U\left[TC_{\min},\ TC_{\max}\right]$.

Insert Table 6

We were thus able to generate 640 cases that enabled us to extract conclusions on the overall behaviour of each procedure presented in Section 3. We solved these cases using each procedure, running nearly 26,500 experiments.

4.2. A lower bound on GALBPS

A lower bound on GALBPS, LB_{GALBPS}, was used to evaluate the efficiency of the proposed heuristic procedures. The lower bound used was that proposed by Andrés *et al.* (2008), $LB1_{GALBPS}$. $LB1_{GALBPS}$ is an adaptation of the most common lower bound on SALBP, which considers the total process time of the tasks to be executed, plus the sum of a certain number of setup times among them, divided by the workstation cycle time, TC. Further details on $LB1_{GALBPS}$ can be found in Andrés *et al.* (2008).

4.3. GRASP metaheuristic for GALBPS (from Andrés et al., 2008)

The GRASP (Greedy Randomized Adaptative Search Procedure) metaheuristic, first described by Feo and Resende (1995) and used in Andrés *et al.* (2008), is the most efficient heuristic procedure for solving GALBPS published to date. It involves two steps: constructing a solution and improving it. The two steps are repeated a prescribed number of times, NS_GRASP.

We programmed GRASP to compare its efficiency to that of our new heuristic procedures. The GRASP metaheuristic of Andrés *et al.* (2008) is briefly summarised below.

In the first phase, in which an initial solution is constructed, two greedy procedures were used: the procedure used in Andrés *et al.* (2008), which corresponds to the *WH* procedure with the priority rule *A-01*; and the *WH* procedure with the priority rule *R-*

14, which, as it can be seen in Subsection 4.4, is one of the procedures which yields the best results.

The second phase, in which the solution is improved, comprised a local optimization procedure similar to that described for the WH_swap procedure: all of the neighbouring solutions of a given solution $\left(Sol_{current}\right)$ are iteratively generated, and then $Sol_{current}$ is substituted with the best neighbouring solution $\left(Sol_{current}^{best\_nei}\right)$, as long as the latter is better than the former. The process stops once $Sol_{current}^{best\_nei}$ is no better than $Sol_{current}$. The neighbouring solutions of $Sol_{current}$ are generated by swapping the tasks assigned to each pair of consecutive positions of the complete sequence of tasks with which it can be described. It should be noted that in this case, as opposed to that of WH_swap, the tasks assigned to different workstations can be also interchanged.

NS_GRASP (number of iterations of these 2 phases) was set to 5, since it provides a computational time comparable to that of computationally-intensive heuristic procedures (TH procedures).

4.4. Performance parameters and results

We evaluated the performance of the heuristic procedures in order to identify the best one. The solutions obtained by using each procedure for each instance were compared by means of performance measures usual in the literature about ALBP (e.g., Capacho *et al*. (2007) or Miralles *et al*. (2008)) and about other scheduling problems like the flowshop problem (Framinan *et al*. (2005) or Ruiz and Stützle (2008)). The results are shown in Table 7, in which the following notation is used: $TofP$, type of procedure; $Rule$, priority rule used; ARD, average relative deviation from the value of the best solution BS (for each instance, BS is the value of the best of all solutions found by the heuristic procedures (the best known solution), and ARD is computed, for each heuristic solution HS, as follows: $ARD = 100 \cdot \frac{HS - BS}{BS}$); PBS, percentage of best solutions obtained; and $Time$, the computing time (in seconds) required to solve all the instances.

Insert Table 7

4.5. Discussion of results

The best not-weighted workstation-oriented procedure is that which used the priority rule *R-09* (WH_R09), with an average relative deviation from the value of the best solution of $ARD = 3.30\%$ and a percentage of best solutions obtained of $PBS = 55.31\%$. The best task-oriented procedure is that which used the priority rule *R-09* (TH_R09) as well, with values of $ARD = 3.51\%$ and $PBS = 53.13\%$. WH_R09 not only has better results than TH_R09, but it is also 790 times faster (26.9 seconds of computational time required vs. 21,238.8 seconds). These results justified the development of the additional workstation-oriented procedures, presented in Subsections 3.4 and 3.5.

The WH_NM procedure improves upon the results obtained using the WH or TH procedures, indicating that procedures based on weighted priority rules, whose parameters have to be accurately fine-tuned, should be considered. Specifically, the WH_NM procedure with priority rule *R-14* (WH_NM_R14) obtained values of $ARD = 2.17\%$ and $PBS = 68.59\%$.

Considering all positions at which a candidate task can be assigned provides good results when priority rule *R-14* is used (WH_pos_R14). But compared to the results obtained with WH_NM_R14, the average relative deviation from the value of the best solution is worse.

The procedures which perform a local optimization of the tasks assigned to a workstation, either once the workstation is considered to be closable (the WH_swap procedure) or each time that a new task is assigned there (the WH_opt procedure), afford better results tan those obtained with WH_NM_R14. The WH_opt procedure with priority rule *R-14* (WH_opt_R14) has values of $ARD = 1.09\%$ y $PBS = 82.97\%$. For this procedure, which is the best of all designed procedures, the computational time required to solve all instances is just 50.2 seconds. As observed in Table 7, the results obtained with the two GRASP procedures are worse than those obtained with WH_opt_R14, and also require much longer computational times.

An ANOVA analysis was made for evaluating both the ARD and the relative behaviour between seven procedures: the three best procedures by Andrés *et al.* (2008) –which are one for each type: WH, TH and $GRASP$– and the four best procedures proposed in this work. We also analyzed the influence of the characteristics of the problem instances –in particular, order strength OS (which gives information on the complexity of the instance), number of tasks N (which indicates the size of the instance) and variability of setups times Var– on the quality of the obtained solutions. The solved instances have been classified according to OS, N and Var, as follows: i) *Low-OS* $(22.49 \le OS \le 25.80)$, *Middle-OS* $(44.80 \le OS \le 59.42)$ and *High-OS* $(70.95 \le OS \le 83.82)$; ii) *Low-N* $(21 \le N \le 32)$, *Middle-N* $(53 \le N \le 94)$ and *High-N* $(148 \le N \le 297)$; iii) *Low-Var* $\left(U\left[0,\ \lceil 0.25 \cdot t_{\min} \rceil\right]\right.$ *and* $\left.U\left[0,\ \lceil 0.75 \cdot t_{\min} \rceil\right]\right)$, *Middle-Var* $\left(U\left[0,\ \lceil 0.25 \cdot \bar{t} \rceil\right]\right)$ and *High-Var* $\left(U\left[0,\ \lceil 0.75 \cdot \bar{t} \rceil\right]\right)$.

From our ANOVA analysis we may summarize the main conclusions obtained by means of the Fisher Test Graphics provided by ANOVA. Figure 1 confirms the results shown in Table 7: the procedure with a best overall behaviour was WH_opt_R14, and WH_opt_R09 was not far from it.

Insert Figure 1

As we can see in Figure 2, the developed procedures show a robust behaviour to the characteristics N and OS and, except for the procedure WH_opt_R09, to parameter Var too. The procedure with a best overall behaviour was WH_opt_R14, unless the characteristic Var is high: in this case WH_opt_R09 was the procedure with a best

overall behaviour. This suggests to do a more in-depth analysis of the existing interactions among the considered characteristics: (number of tasks, order strength and variability of setups times, $N/OS/Var$) for both best procedures WH_opt_R14 and WH_opt_R09. In detail, as we can see in Figure 3, the best procedure was WH_opt_R14, except with the characteristics combinations $Low/Middle/High$, $Low/Middle/Middle$, $Middle/High/High$ and $Middle/Middle/High$. Thus, it is recommended to use procedure WH_opt_R14 or WH_opt_R09 according to the instance characteristics.

Insert Figure 2

Insert Figure 3

To measure the quality of the solutions of these seven procedures, we calculated the workstations percentage increase $(NWPI)$. This indicator shows the percentage deviation between the number of workstations provided by a heuristic and LB_{GALBPS} (the lower bound on GALBPS presented in Subsection 4.2). Table 8 shows the following information: the procedure (type of procedure and priority rule used), the average relative deviation from the value of the best solution (ARD); and the value of $NWPI$. For the best heuristic procedure designed, WH_opt_R14, the maximum average error obtained from the optimal solution was 14.96%, which is acceptable given the complexity of the problem at hand, its newness and the quality of the availaible lower bound (which is usually less than the exact solution, Andrés *et al*. (2008)).

Insert Table 8

Lastly, we would like to point out that the results obtained in this work are better than the best results published to date for solving GALBPS (obtained by Andrés *et al.* (2008)).

5. Conclusions and further research

The General Assembly Line Balancing Problem with Setups (GALBPS) was recently defined in the literature. GALBPS adds sequence-dependent setup time considerations to the classical SALBP such that, whenever a task is assigned next to another at the same workstation, a setup time must be added to compute the global workstation time, thereby providing the task sequence inside each workstation. This reflects a more realistic scenario for many assembly lines. In Andrés *et al.* (2008) GALBPS is modelled through a binary programming model; however, the model only provides optimal solutions for very small instances. These authors presented and evaluated eight different heuristic rules and a GRASP algorithm (which are the best heuristic procedures published to date for solving GALBPS).

In this paper, we present several heuristic procedures, based on priority rules, for solving GALBPS-1 (*i.e.* for minimizing the number of workstations for a given upper bound on the cycle time): a workstation-oriented procedure based on not-weighted

priority; a task-oriented procedure with several priority rules; a workstation-oriented procedures based on weighed priority rules (which are fine-tuned with the Nelder and Mead algorithm); and, finally, improved tasks assignation schemes within a workstation. These schemes are based on considering all positions at which a candidate task can be assigned; performing a local optimization of the tasks assigned to a workstation, once the workstation can be considered closed; and performing a local optimization of the tasks assigned to a workstation, every time that a new task is assigned there.

We tested the proposed heuristic procedures with a set of self-made instances. The results demonstrate that some of the heuristic procedures based on the new priority rules improve upon those described to date, including the metaheuristic GRASP proposed by Andrés *et al.* (2008). In detail, the procedure with a best overall behaviour was WH_opt_R14; although for the following characteristics combinations of the problem instances (number of tasks / order strength / variability of setups times) $Low / Middle / High$, $Low / Middle / Middle$, $Middle / High / High$ and $Middle / Middle / High$, procedure WH_opt_R09 was the best one.

To measure the quality of the solutions, we calculated the workstations percentage increase that shows the percentage deviation between the number of workstations provided by a heuristic and a lower bound on GALBPS. For the best heuristic procedure designed, WH_opt_R14, the maximum average error obtained with the optimal solution was 14.96%, which is acceptable given the complexity of the problem at hand, its newness and the quality of the availaible lower bound (which is usually less than the exact solution, Andrés *et al*. (2008)).

Our future work will focus on the design of metaheuristic procedures for the problem.

6. Acknowledgments

The authors are very grateful to Professor Albert Corominas (Technical University of Catalonia) and to the anonymous reviewers for their valuable comments which have helped to enhance this paper.

7. References

Agnetis A, Arbib C., 1997. Concurrent operations assignment and sequencing for particular assembly problems in flow lines. *Annals of Operations Research*, 69, 1–31.

Allahverdi A, Gupta JND, Aldowaisan T., 1999. A review of scheduling research involving setup considerations. *Omega*, 27, 219–239.

Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY., 2008. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187, 985–1032.

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.

Anjos MF, Cheng RCH, Currie CSM., 2004. Maximizing revenue in the airline industry under one-way pricing. *Journal of the Operational Research Society*, 55, 535–541.

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

Arcus AL., 1966. COMSOAL: A computer method of sequencing operations for assembly lines. *International Journal of Production Research*, 4, 259–277.

Bhattacharjee TK, Sahu S., 1988. A heuristic approach to general assembly line balancing. *International Journal of Operations and Production Management*, 8, 67–77.

Bard JF, Dar-El E, Shtub A., 1992. An analytic framework for sequencing mixed model assembly lines. *International Journal of Production Research*, 30, 35–48.

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

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., 2007. A classification of assembly line balancing problems. *European Journal of Operational Research*, 183, 674–693.

Bukchin J, Rubinovitz J., 2003. A weighted approach for assembly line design with station paralleling and equipment selection. *IIE Transactions*, 35, 73–85.

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., 2007. An evaluation of constructive heuristic methods for solving the Alternative Subgraphs Assembly Line Balancing Problem. *Journal of Heuristics* (*Online first on 27 October 2007*) doi: 10.1007/s10732-007-9063-x.

Chelouah R, Siarry P., 2005. A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for the global minimization of multiminima functions. *European Journal of Operational Research*, 161, 636–654.

Corominas A., 2005. Empirically Adjusted Greedy Algorithms (EAGH): A new approach to solving combinatorial optimisation problems. *Working paper IOC-DT-P-2005-22*. Universidad Politécnica de Cataluña, Barcelona.

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

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

Feo TA, Resende MG., 1995. Greedy randomized adaptative search procedures. *Journal of Global Optimization*, 6, 109–133.

Framinan JM, Leisten R, Ruiz-Usano R., 2005. Comparison of heuristics for flowtime minimisation in permutation flowshops. *Computers & Operations Research*, 32, 1237–1254.

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 RJ., 1989. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, 27, 637–670.

Graves SC, Lamar BW., 1983. An Integer Programming Procedure for Assembly System Design Problems. *Operations Research*, 31, 522–545.

Helgeson WB, Birnie DP., 1961. Assembly line balancing using the ranked positional weight technique. *Journal of Industrial Engineering*, 12, 394–398.

Lagarias JC, Reeds JA, Wright MH, Wright PE., 1998. Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9, 112–147.

Miltenburg J., 2001. U-shaped production lines: a review of theory and practice. *International Journal of Production Economics*, 70, 201–214.

Miralles C, García-Sabater JP, Andrés C, Cardós M., 2008. Branch and Bound procedures for solving the Assembly Line Worker Assignment and Balancing Problem. Application to Sheltered Work Centres for Disabled. *Discrete Applied Mathematics*, 156, 352–367.

Moodie CL, Young HH., 1965. A heuristic method of assembly line balancing for assumptions of constant or variable work element times. *Journal of Industrial Engineering*, 16, 23–29.

Nelder JA, Mead R., 1965. A simplex method for function minimization. *The Computer Journal*, 7, 308–313.

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, Ferrer L., 2008. An improved mathematical program to solve the simple assembly line balancing problem. *International Journal of Production Research* (to be published, doi: 10.1080/00207540701713832).

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

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

Ruiz R, Stützle T., 2008. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187, 1143-1159.

Sawik T., 2002. Balancing and scheduling of surface mount technology lines. *International Journal of Production Research*, 40, 1973–1991.

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, Boysen N, Fliedner M., 2008. The sequence-dependent assembly line balancing problem. *OR Spectrum*, 30, 579-609.

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

Scholl A, Klein R. The assembly line balancing research website. *www.assembly-line-balancing.de*, 28/04/2008.

Scholl A, Voß S., 1996. Simple assembly line balancing-Heuristic approaches. *Journal of Heuristics*, 2, 217–244.

Talbot FB, Patterson JH, Gehrlein WV., 1986. A comparative evaluation of heuristic line balancing techniques. *Management Science*, 32, 431–453.

Tonge FM., 1961. *A heuristic program for assembly line balancing*. Prentice-Hall, Englewood Cliffs, NJ.

Wee TS, Magazine MJ., 1982. Assembly line balancing as generalized bin packing. *Operations Research Letters*, 1, 56–58.

Wilhelm WE., 1999. A column-generation approach for the assembly system design problem with tool changes. *International Journal of Flexible Manufacturing Systems*, 11, 177–205.

Zhu X, Wilhelm WE., 2006. Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions*, 38, 987–1007.

Sequence	Times to be considered	Global workstation time
A-B-C	10+3+12+1+9+3	38
B-A-C	12+2+10+4+9+4	41

Table 1. Two possible sequences for the tasks A, B and C

Rule	v_i
A-01	$\tau_{last,i} + t_i$
A-02	$-\left(\tau_{last,i} + t_i\right)$
A-03	$\tau_{last,i}$
A-04	$-\tau_{last,i}$

Rule	v_i
R-01	$\tau_{last,i} + t_i + \tau_{i,first}$
R-02	$-\left(\tau_{last,i} + \tau_{i,first}\right)$
R-03	$t_i + \sum_{j \in S_i} \left(t_j + \bar{\tau}_j\right)$
R-04	$t_i + \tau_{last,i} + NS_i$
R-05	$t_i - \tau_{last,i}$
R-06	$t_i - \tau_{last,i} + NS_i$

Rule	v_i
R-07	$t_i - \tau_{last,i} + \sum_{j \in S_i} \left(t_j + \bar{\tau}_j\right)$
R-08	$\frac{t_i}{\tau_{last,i}}$
R-09	$\frac{t_i + NS_i}{\tau_{last,i}}$
R-10	$\frac{t_i}{\tau_{last,i}} + NS_i$
R-11	$\frac{t_i}{L_i - E_i}$
R-12	$\frac{t_i - \tau_{last,i}}{L_i - E_i}$

Table 2. Not-weighted priority rules for the *WH* procedure

Rule	v_i
A-01	$\tau_{last,i} + t_i$
A-02	$-\left(\tau_{last,i} + t_i\right)$
A-03	$\tau_{last,i}$
A-04	$-\tau_{last,i}$

Rule	v_i
R-05	$t_i - \tau_{last,i}$
R-08	$\frac{t_i}{\tau_{last,i}}$
R-09	$\frac{t_i + NS_i}{\tau_{last,i}}$

Table 3. Priority rules for the *TH* procedure

Rule	v_i before fine-tuning
R-13	$\alpha_1 \cdot t_i - \beta_1 \cdot \tau_{last,i} + \gamma_1 \cdot NS_i$
R-14	$\alpha_2 \cdot t_i - \beta_2 \cdot \tau_{last,i} + \delta_2 \cdot \sum_{j \in S_i} \left(t_j + \lambda_2 \cdot \bar{\tau}_j\right)$
R-15	$\frac{\alpha_3 \cdot t_i - \beta_3 \cdot \tau_{last,i} + \delta_3 \cdot \sum_{j \in S_i} \left(t_j + \lambda_3 \cdot \bar{\tau}_j\right) + \gamma_3 \cdot NS_i}{\pi_3 \cdot L_i - \omega_3 \cdot E_i}$
R-16	$\alpha_4 \cdot t_i - \beta_4 \cdot \tau_{last,i} + \vartheta_4 \cdot \tau_{i,next} + \delta_4 \cdot \sum_{j \in S_i} \left(t_j + \lambda_4 \cdot \bar{\tau}_j\right)$

Table 4. Weighted priority rules for the *WH _ NM* before fine-tuning

Rule	v_i after fine-tuning
R-13	$1.0 \cdot t_i - 10.3 \cdot \tau_{last,i} + 2.6 \cdot NS_i$
R-14	$5.0 \cdot t_i - 45.3 \cdot \tau_{last,i} + 0.3 \cdot \sum_{j \in S_i} \left(t_j + 3.9 \cdot \bar{\tau}_j \right)$
R-15	$\dfrac{1.5 \cdot t_i - 8.5 \cdot \tau_{last,i} + 0.1 \cdot \sum_{j \in S_i} \left(t_j + 1.4 \cdot \bar{\tau}_j \right) + 1.7 \cdot NS_i}{2.0 \cdot L_i - 1.5 \cdot E_i}$
R-16	$1.7 \cdot t_i - 7.5 \cdot \tau_{last,i} + 4.8 \cdot \tau_{i,next} + 0.2 \cdot \sum_{j \in S_i} \left(t_j + 1.8 \cdot \bar{\tau}_j \right)$

Table 5. Weighted priority rules for the *WH _ NM* procedure after fine-tuning

Name	(N)	$t_{\min}$	$t_{\max}$	$\bar{t}$	OS	$TC_{\min}$	$TC_{\max}$
Arcus1	83	233	3,691	912.1	59.09	3,786	10,816
Barthold	148	3	383	38.1	25.80	403	805
Barthol2	148	1	83	28.6	25.80	84	170
Hahn	53	40	1,775	264.6	83.82	2,004	4,676
Heskiaoff	28	1	108	36.6	22.49	138	342
Lutz1	32	100	1,400	441.9	83.47	1,414	2,828
Lutz2	89	1	10	5.4	77.55	11	21
Lutz3	89	1	74	18.5	77.55	75	150
Mitchell	21	1	13	5.0	70.95	14	39
Mukherje	94	8	171	44.8	44.80	176	351
Roszieg	25	1	13	5.0	71.67	14	32
Sawyer	30	1	25	10.8	44.83	25	75
Scholl	297	5	1,386	234.5	58.16	1,394	2,787
Tonge	70	1	156	50.1	59.42	160	527
Warnecke	58	7	53	26.7	59.10	54	111
Wee-Mag	75	2	27	20.0	22.67	28	56

Table 6. Instances from Scholl's and Klein's website (Scholl and Klein, 2008)

TofP	Rule	ARD	PBS	Time
WH	A-01	7.95	36.41	31.0
	A-02	14.44	17.03	29.5
	A-03	13.33	20.47	28.5
	A-04	5.86	39.06	31.1
	R-01	8.10	36.41	31.4
	R-02	7.32	32.19	27.3
	R-03	6.56	41.41	26.7
	R-04	7.11	40.31	30.1
	R-05	5.29	42.97	29.9
	R-06	4.13	51.56	31.1
	R-07	4.64	45.94	27.3
	R-08	4.49	45.63	30.2
	R-09	3.30	55.31	26.9
	R-10	3.60	51.56	26.9
	R-11	6.26	42.66	30.1
	R-12	4.04	49.69	33.1
TH	A-01	8.54	30.00	21,465.8
	A-02	14.45	17.03	22,137.8
	A-03	12.17	22.03	22,438.6
	A-04	5.81	39.69	23,659.2
	R-05	6.05	37.66	21,345.6
	R-08	4.80	44.06	24,895.8
	R-09	3.51	53.13	21,238.8
WH _ NM	R-13	2.62	63.59	28.8
	R-14	2.17	68.59	26.7
	R-15	2.35	66.41	31.7
	R-16	2.48	65.78	28.1
WH _ pos	R-05	3.29	55.00	32.0
	R-08	4.08	49.38	34.1
	R-09	2.77	59.38	33.7
	R-14	2.51	69.53	32.9
WH _ swap	R-05	4.72	46.56	30.3
	R-08	4.34	48.13	31.2
	R-09	2.37	63.59	34.6
	R-14	2.07	69.38	39.0
WH _ opt	R-05	2.71	59.84	65.3
	R-08	3.12	57.50	60.9
	R-09	1.80	70.47	88.4
	R-14	1.09	82.97	50.2
GRASP	A-01	3.47	53.44	34,814.3
	R-14	7.74	30.16	38,529.1

Table 7. Results of the computational experiment

Procedure	ARD	NWPI
GRASP _ A01	3.47	17.60
TH _ A04	5.81	20.37
WH _ A04	5.86	20.44
WH _ opt _ R14	1.09	14.96
WH _ opt _ R09	1.80	15.68
WH _ swap _ R14	2.07	16.06
WH _ NM _ R14	2.17	16.17

Table 8. Results of the workstations percentage increase

FIGURE CAPTION

Figure 1. Means and 95.0% LSD intervals graphic for procedures

Figure 2. Interaction plots for order strength OS, number of tasks N and variability of setups times Var

Figure 3. Interaction plot for WH_opt_R09 and WH_opt_R14 procedures

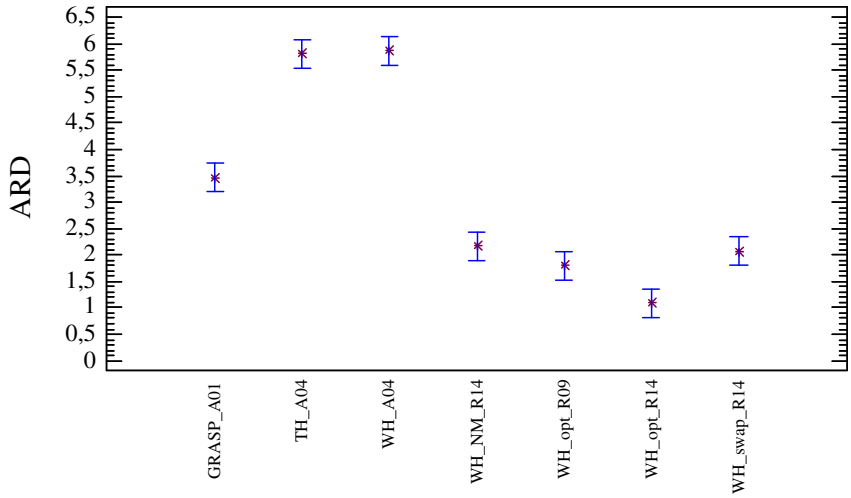

Figure 1. Means and 95.0% LSD intervals graphic for procedures

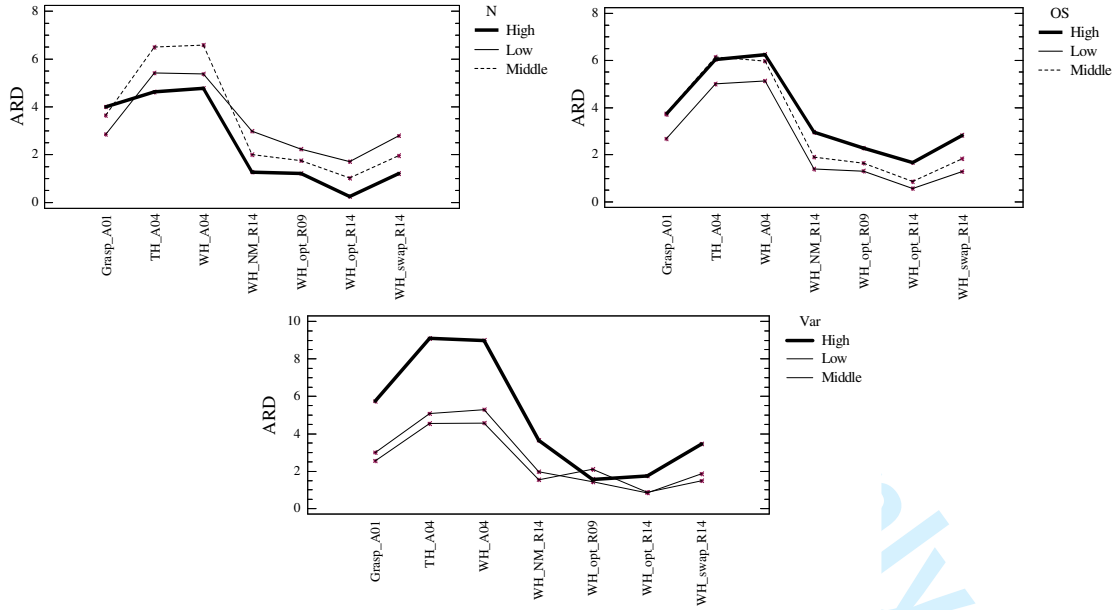

Figure 2. Interaction plots for order strength *OS* , number of tasks *N* and variability of setups times *Var*

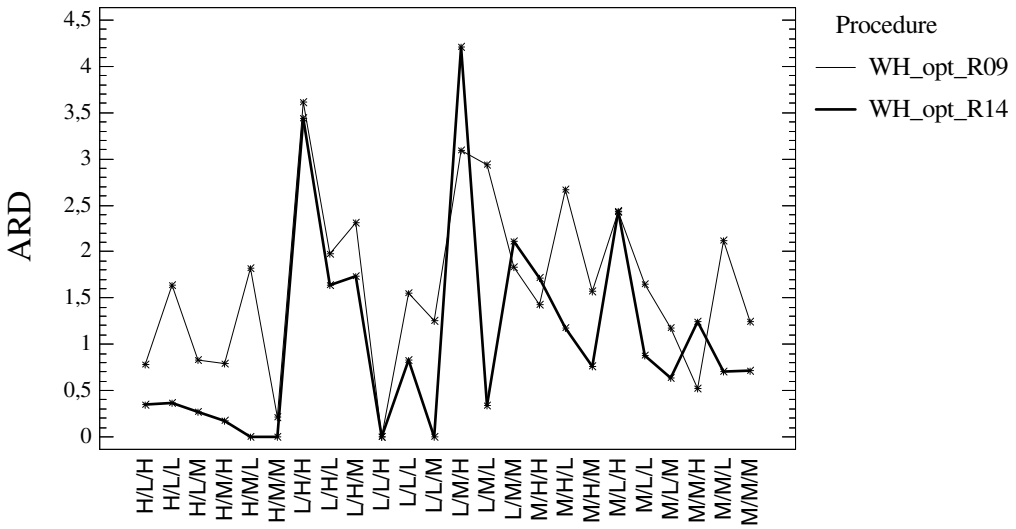

Figure 3. Interaction plot for *WH _ opt _ R*09 and *WH _ opt _ R*14 procedures