Electronic version of an article published as [European Journal of Operational Research, 2008, Vol. 187, No. 3, p. 1212-1223]
[DOI: http://dx.doi.org/10.1016/j.ejor.2006.07.044]

Balancing and scheduling tasks in assembly lines with sequence-dependent setup times*

Carlos Andrés[a], Cristóbal Miralles[a] and Rafael Pastor[b]

[a]ROGLE Group – Universidad Politécnica de Valencia, Depto. Organización de Empress, Camino de Vera s/n, 46071 Valencia, Spain

[b]IOC Research Institute – Universidad Politécnica de Cataluña, Av. Diagonal 647, p. 11, 08028 Barcelona, Spain

Abstract

The classical Simple Assembly Line Balancing Problem (SALBP) has been widely enriched over the past few years with many realistic approaches and much effort has been made to reduce the distance between the academic theory and the industrial reality. Despite this effort, the scheduling of the execution of tasks assigned to every workstation following the balancing of the assembly line has been scarcely reported in the scientific literature. This is supposed to be an operational concern that the worker should solve himself, but in several real environments, setups between tasks exist and optimal or near-optimal tasks schedules should be provided inside each workstation. The problem presented in this paper adds sequence-dependent setup time considerations to the classical SALBP in the following way: whenever a task is assigned next to another at the same workstation, a setup time must be added to compute the global workstation time. After formulating a mathematical model for this innovative problem and showing the high combinatorial nature of the problem, eight different heuristic rules and a GRASP algorithm are designed and tested for solving the problem in reasonable computational time.

Keywords: Assembly line balancing; Sequence-dependent setup times; Scheduling

*This work was developed under research project DELIMER (DPI2004-03472) supported by the Spanish National Science and Technology Commission CICYT, co-financed by FEDER.

1. Introduction

Traditional assembly line balancing research has focused on the simple assembly line balancing problem (SALBP), but in recent years a growing amount of literature has been produced dealing with more realistic and generalized problems. The literature contains additional characteristics such as cost functions, equipment selection, paralleling, U-shaped line layout and mixed-model production among many others (see, for example, Becker and Scholl, 2006). Thus many relevant problems have been identified and modelled.

However, many realistic situations that need modelling and design of efficient solving procedures remain on the agenda of researchers. In this sense, the assembly line balancing literature – even in those references closer to reality – used to focus on the problem in a pure sense as if, once assigned the tasks to the workstations, no further definition of parameters was necessary. In many real production lines, however, the sequence in which tasks are developed inside the workstation matters, since sequence-dependent setup times between tasks are present. This paper presents a previously scarcely explored problem in which the assembly line not only requires balancing, but also the scheduling of tasks assigned to every workstation must be defined due to the existence of sequence-dependent setup times. Therefore, both the inter-station balancing issue and the intra-station scheduling of tasks must be solved simultaneously; this supposes a new more realistic scenario for many real assembly lines, especially in the electronics industry and other similar industrial sectors with low cycle times.

In most industrial assembly lines these setup times exist but are usually not considered because they are very low compared to operation times; moreover, they are considered independently as they are executed just before or after the tasks (and then their times are added to task times). An example of the problem is the practice of different tools being handled by workers for developing different tasks. In this situation what is important is to define the best work schedule for the worker in order to minimize the workstation global time, including setup times. This is an issue that becomes even more relevant when cycle time is low, since setup times may represent a high percentage of it.

Another real situation in which it is possible to find setup times inside assembly line workstations are robotic lines. Often the robot must take off the actual tool; select the corresponding new tool from a set; and make adjustments before starting next task assigned. The time for this tool adjustment can vary and depends on the sequence of tasks. Obviously this tool adjustment can also depend on the next product to be assembled when talking about mixed-model assembly lines, but this would widen our scenario and make it even more complex. Hence, in the first step of the analysis and modelling of this original problem, we prefer to focus on the issue of balancing tasks & scheduling inside a paced mono-model assembly line.

Another practical question that can be answered through this proposal is the existence of components placed in distanced containers. The time to arrive at a container depends on the last component that was assembled for the product, resulting in the work schedule inside the workstation proving significant. We will consider a problem in which different tools must be handled by workers for developing different tasks and, therefore, whenever more than one task is assigned to the same workstation corresponding intra-station setup times between tasks appear and must be considered.

So far in the cases where setup times exist, the task schedule for every station is usually defined only after the line balancing once the tasks have been assigned. The scheduling problem inside every workstation can be solved approximately or as a TSP problem in which the setup times are the distances between the cities, although there are some forbidden paths due to the existence of precedence relations. But, with the usual approach of solving the problem in two separate stages, normally only suboptimal solutions can be reached; what provides evidence of the need of procedures for simultaneously solving both the line balancing and the scheduling problems. An example will be showed in the next section in order to completely understand the problem.

1.1. Example

This instance was generated from the example that appears in Scholl and Becker (2006), see Fig. 1, randomly generating the setup times that appear in Table 1.

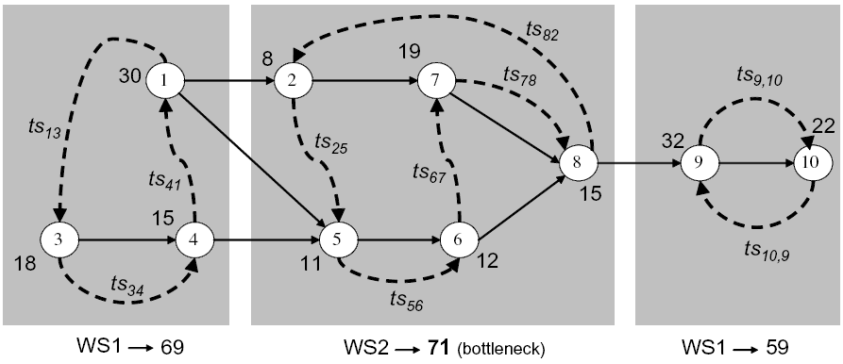

Fig. 1. Precedence network and solution given to the example.

Table 1.

Setup times matrix

0	3	1	3	1	1	0	1	1	1
2	0	2	1	0	1	1	1	3	3
3	2	0	2	2	1	2	2	1	1
3	1	3	0	2	3	1	1	2	1
1	1	4	1	0	0	3	2	1	2
1	3	1	0	2	0	1	4	1	3
1	1	4	1	1	1	0	4	1	0
1	1	1	2	1	1	1	0	2	1
1	1	2	1	2	2	3	1	0	3
3	1	1	2	3	1	3	1	2	0

In this table, for example, the setup time tsu_{32} equal to 2, in third row second column, corresponds to the time that is required for performing task 3 immediately before task 2. The operation times appear in Fig. 1 besides the circles (where we have represented the original precedence relations between tasks in straight fine lines).

For an input cycle time of 71, one solution was the following:

The task assignment for the three workstations provided by the solution are shaded in Fig. 1, and the work schedules inside each workstation are represented with bold, discontinuous lines. The global time in every station is the sum of operation times and setup times; therefore the second workstation becomes the bottleneck in this example of an assembly line with sequence-dependent setup times.

1.2. Structure of the paper

The remainder of the paper is organized as follows: in Section 2, we review the works developed so far both for the assembly line balancing scientific literature and the sequencing literature. In Section 3, we model the problem proposed, highlighting the basic assumptions. After formulating a mathematical model for this scheduling problem, eight different heuristic rules for solving it are presented in Section 4, relating to the search strategy implied (station or task oriented) and the task selection ordering. In the same section, a GRASP algorithm is designed and presented.

Both the heuristics and the GRASP procedure are tested and evaluated through an experimentation with the main results presented in Section 5. Since there is no standard benchmark for this innovative problem, the experimental study is carried out with a set of self-made instances generated from the well-known set of instances available in the assembly line balancing research homepage (www.assembly-line-balancing.de). Conclusions are reported and summarized in the last section, and also point out some interesting further research lines.

2. State of the art

The assembly line balancing problem has been studied extensively since the very first publication of Salveson (1955). Simple Assembly Line Balancing Problem (SALBP) (Baybars, 1986) is known to be NP-hard (Gutjahr and Nemhauser, 1964), and much research has been generated for solving the problem in reasonable computational time. When we talk about SALBP, the aim can be to minimize the number of stations given a desired cycle time (SALBP-1) or to minimize the cycle time given a fixed number of workstations (SALBP-2). Several techniques have been proposed for the solution of both problems with different considerations of tasks times and configurations (see the reviews of [Ghosh and Gagnon, 1989], [Erel and Sarin, 1998], [Rekiek et al., 2002], [Scholl and Becker, 2006] and [Dolgui, 2006]).

The industrial situations described in the previous section were the source of inspiration for this research, and very soon we had clear evidence of the shortage of scientific literature related to this topic. In fact, our research mixes in the same scenario of the classical assembly line balancing problem with the problem of sequencing tasks due to the existence of sequence-dependent setup times. Therefore, in a first overview we referred to the most important reviews done so far in both fields: assembly line balancing and sequencing.

On the one hand, we referred to the "survey on problems and methods in generalized assembly line balancing" done by Becker and Scholl (2006), in which all the realistic variants of assembly lines are categorized, and the best papers for every category are summarized. In that paper no reference at all is made about assembly lines with sequence-dependent setups; only a little reference to setups is done when talking mixed-model and multi-model lines. Depending on the type of intermixing of the units, two variants of assembly lines arise: a mixed-model line produces units of different models in an arbitrarily intermixed sequence (see Bukchin and Tzur, 2002) differing from each other with respect to size, colour, used material, or equipment; whereas a multi-model line produces a sequence of batches (each containing units of only one model or a group of similar models) with intermediate setup operations. Therefore, in both cases balancing and sequencing are connected, and many examples are given, but the sequence always refers to the products or models to perform in the line, not to the work sequence of tasks inside the workstations. In Merengo et al. (1999) a review is done for both line the balancing problem and sequencing problem in mixed-

model assembly lines, whereas in Sawik (2002) both problems are faced simultaneously in a specific environment of printed circuit board production lines.

On the other hand, if we focus on literature about scheduling research involving setup considerations we have many different points of view, like all the references we can find in Allahverdi et al. (1999) and other reviews done later like Allahverdi et al. (in press), the authors have not found references about the evaluation of the work schedule inside the assembly line.

Only Agnetis and Arbib (1997) face a related problem but where the tasks precedence network has a "comb" shape. This means that the previous operations to the different parts of the final product are performed separately and there is a later stage of assembling those different parts. The problem consists of assigning operations to machines and sequencing them in every workstation in order to maximize defined performance indicators. Having the special "comb" precedence network makes the combinatory nature of the problem much lower, thereby it can be solved in polynomial time. Although the problem is not mathematically modelled, the ideas for the dynamic programming procedure proposed are inspiring.

In Jayashakandar et al. (1999), a specific kind of mixed-model assembly line with somewhat similar features to ours is presented, despite this, the aim is completely different. This problem focuses on those product lines that delay product differentiation and where only at the last workstations the tasks need a sequence in order to perform the major quantity of different products.

Some other references also face a double assignment of tasks and resources to stations. For example some cost-oriented models assume that the equipment of the stations is given and that the production process is fixed, then the total cost must be minimized by optimally integrating design (selecting the machine type to locate at each activated station) and operating issues (assigning tasks to observe precedence relationships and cycle time restrictions). When these decisions are connected, the term Assembly Line Design Problem (see Baybars, 1986) or Assembly System Design Problem (see Pinnoi and Wilhelm, 1997) are frequently used in the literature. But in these kinds of approaches also no reference is given to the problem addressed in this paper.

3. Mathematical modelling of the problem

3.1. Introduction

As presented in the above State of the Art there is a small but growing volume of literature on assembly line balancing problems when considering realistic scenarios. The classical approaches were usually limited to the traditional (and difficult to find in reality) SALBP, which comprises the assignment of tasks to stations. Precedence relations between some of the tasks impose a partial ordering and reflect which task has to be completed before others. The tasks are related to the assembly of a product to be performed at consecutive stations and the assigned tasks must be developed within the cycle time.

The problem addressed in this paper adds sequence-dependent setup time considerations to the classical SALBP in the following way: whenever a task k is assigned next to the task i at the same workstation, a setup tsu_{ik} must be added to compute the global operation time. 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 tsu_{pi} must be also considered. This is because the operations are repeated cyclically;

therefore the last task of the workstation is performed in one cycle just before the first task in the next cycle.

3.2. Mathematical model

The aim is to assign tasks to workstations and sequence them so that no precedence relationships are violated and the global time, including setup times, is under the cycle time. As has been mentioned, this problem has barely been reported in previous literature; despite the fact that it represents a common situation in most robotic production lines and in almost every manually operated assembly line since the worker must change the tool or adjust the machine to pass from one task to the next.

Therefore, similar to the classification of Baybars (1986), when the objective is to minimize the number of stations for a given cycle time we will call this problem GALBPS-1 (General Assembly Line Balancing Problem with Setups-1). When the aim is to minimize the cycle time assigned to a predefined set of workstations we will call the problem GALBPS-2. For the sake of brevity, we prefer to focus this paper just on GALBPS-1, although the model presented is easily adaptable to a GALBPS-2 scenario and the procedures presented in the next section have a modular design that enables the resolution of both kinds of problems.

Before presenting the notation and the binary linear program (BLP) model generated, certain basic assumptions must be stated in order to completely define the problem:

(1) Task processing times, setup times matrix and precedence relationships are known deterministically.

(2) Processing and setup times are independent on the workstation in which tasks are processed.

(3) A single model of one product is assembled on the line.

(4) We define a paced serial line where buffers are not considered.

(5) Every task is assigned to only one workstation.

(6) Tasks must be processed only once.

(7) All workstations are equally equipped and any task can be assigned to any workstation.

(8) None of the task processing times may be greater than the cycle time.

(9) Workstations can process only one task at a time.

With these basic assumptions, the BLP model will be defined using the notation in Table 2.

Table 2.

Notation used for GALBPS

i, k	Task
j	Workstation
s	Position inside the schedule of a workstation
N	Set of tasks ($i = 1, \ldots, N$)
$m_{\min}$	Lower bound for number of workstations
$m_{\max}$	Upper bound for number of workstations
t_i	Duration of task i
TC	Upper bound for cycle time
E_i, L_i	Earliest and Latest workstation where task i can be assigned
T_j	Set of tasks that can be assigned to workstation j
P	Set of couples of tasks (i, k) where i is immediate predecessor of k
PT_i	Set of tasks all predecessors of task i, including non immediate predecessors
Nm_j	Maximum number of tasks that can be assigned to workstation j
NTm	Maximum number of tasks that can be assigned to any workstation $NTm = \max_j\{Nm_j\}$
tsu_{ik}	Setup time when task k is performed just after task i in side the same workstation

$x_{ijs} \in \{0, 1\}$	1 if task i is assigned to workstation j in position s of its schedule ($i = 1, \dots, N; j = E_i, \dots, L_i; s = 1, \dots, Nm_j$)
$y_j \in \{0, 1\}$	1 if any task is assigned to the workstation j ($j = m_{\min} + 1, \dots, m_{\max}$)
$z_{ikj} \in \{0, 1\}$	1 if task i is performed immediately before task k in the workstation j in the same or in the next cycle ($\forall j;\ \forall\ (i, k)|(i \neq k)\ \wedge\ (i, k \in T_j)$)
$w_{ij} \in \{0, 1\}$	1 if task i is the last one in the sequence of tasks assigned to workstation j ($\forall i; j = E_i, \dots, L_i$)

According to this notation we have the following BLP formulation:

$$[\text{MIN}]\, Z = \sum_{j=m_{\min}+1}^{m_{\max}} j \cdot y_j \quad (1)$$

subject to:

$$\sum_{j=E_i}^{L_i} \sum_{s=1}^{Nm_j} x_{ijs} = 1 \quad (\forall i), \quad (2)$$

$$\sum_{\forall i \in T_j} x_{ijs} \leq 1 \quad (\forall j; s = 1, \dots, Nm_j) \quad (3)$$

$$\sum_{\forall i \in T_j} x_{ij,s+1} \leq \sum_{\forall i \in T_j} x_{ijs} \quad (\forall j; s = 1, \dots, Nm_j) \quad (4)$$

$$\sum_{j=E_i}^{L_i} \sum_{s=1}^{Nm_j} (NTm \cdot (j-1) + s) \cdot x_{ijs} \leq \sum_{j=E_k}^{L_k} \sum_{s=1}^{Nm_j} (NTm \cdot (j-1) + s) \cdot x_{kjs} \quad (\forall (i,k) \in P) \quad (5)$$

$$\sum_{\forall i \in T_j} \sum_{s=1}^{Nm_j} t_i \cdot x_{ijs} + \sum_{\forall (i,k) | (i \neq k) \wedge (i,k \in T_j)} tsu_{ik} \cdot z_{ikj} \leq TC \quad (j = 1, \dots, m_{\min}), \quad (6)$$

$$\sum_{\forall i \in T_j} \sum_{s=1}^{Nm_j} t_i \cdot x_{ijs} + \sum_{\forall (i,k) | (i \neq k) \wedge (i,k \in T_j)} tsu_{ik} \cdot z_{ikj} \leq TC \cdot y_j \quad (j = m_{\min} + 1, \dots, m_{\max}), \quad (7)$$

$$x_{ijs} + x_{kj,s+1} \leq 1 + z_{ikj}$$

$$(\forall j; s = 1, \dots, Nm_j - 1;$$

$$\forall (i,k) | (i \neq k) \wedge (i,k \in T_j) \wedge (k \notin PT_i)) \quad (8)$$

$$x_{ijs} - \sum_{\forall k \in T_j | (i \neq k) \wedge (k \notin PT_i)} x_{kj,s+1} \leq w_{ij}$$

$$\left(\forall j; s = 1,..., Nm_j - 1; \forall i \in T_j\right), \quad (9)$$

$$w_{ij} + x_{kj1} \leq 1 + z_{ikj}$$

$$\left(\forall j; \forall (i,k) | \left(i \neq k\right) \wedge \left(i,k \in T_j\right) \wedge \left(i \notin PT_k\right)\right) \quad (10)$$

The objective function (1) minimizes the number of workstations (since we focus on GALBPS-1); (2) implies that every task must be assigned to only one position inside only one workstation; with (3) in every position inside every workstation there will be no more than one task assigned; (4) ensures that the tasks have to be assigned in increasing positions in the schedule of every workstation; constraints set (5) imply that the precedence constraints are not violated, regarding not only the assignment to different workstations, but also the positions inside the same workstation; (6) and (7) ensure that the global time in every workstation, including both tasks duration and setup times, is under the cycle time; with (8) the variable z_{ikj} is 1 whenever task i is assigned to position s and task k is assigned to position $s + 1$ in the schedule of workstation j; with (9) the variable w_{ij} is 1 whenever task i is assigned to the last position in the schedule of workstation j; with (10) the variable z_{ikj} is 1 whenever task i is assigned to the last position and task k to the first one in the schedule of workstation j.

3.3. Results

The BLP model was implemented with CPLEX 9® and tested with a set of self-made samples generated from a well-known set of problems available in the assembly line balancing research homepage (www.assembly-line-balancing.de). The results, detailed in Section 5.2, provide evidence of the combinatorial nature of the problem and the difficulty of obtaining exact resolution in reasonable computational time. Therefore, the design of alternative heuristic methods like those presented in the next section is fully justified.

The BLP model was very useful since the value obtained after certain computation time can be used as a lower bound to evaluate the efficiency of heuristic procedures, as will be exposed in the experimental study carried out in Section 5. Furthermore, the BLP model may be the starting point for one of the further research lines commented in Section 6.

4. Heuristic procedures developed

As it has been introduced, SALBP is known to be NP-hard, and is only a special case of GALBPS where no setup times exist. Therefore GALBPS is also NP-hard, and it is fully justified to develop heuristic methods in order to achieve good results in a computational time short enough to be applied in industrial real environments.

In this sense we have developed eight simple heuristic rules that have been compared when solving different kinds of problems, extracting conclusions about their convenience when facing them. Moreover, a specific GRASP heuristic method has been designed and tested against the same set of

instances in order to report the comparative behaviour of this heuristic. In this section both the heuristic rules and the GRASP method will be described, being experimentally tested in Section 5.

4.1. Heuristic rules defined

Most heuristic algorithms are based on generating feasible solutions by successively assigning tasks or subsets of tasks to stations. Therefore, these algorithms consider partial solutions containing a number of already assigned tasks and (partial) station loads, while the remaining tasks and station idle times constitute a residual problem (Scholl and Becker, 2006). Almost every solution procedure for SALBP-1 is based on one of the two following construction schemes, which define the main way of assigning tasks to stations:

• Station-oriented assignment: In any step of the procedure a complete load of assignable tasks is built for a station j, before the next one $j + 1$ is considered.

• Task-oriented assignment: Procedures which are task-oriented iteratively select a single available task and assign it to a station, to which it is assignable.

Most computational experiments reported in the literature indicate that, for SALBP, station-oriented procedures get better results than task-oriented ones though no theoretical dominance exists (Scholl and Voß, 1996). The eight simple heuristic rules that have been tested are based on the combination between following strategies and ordering rules:

• Strategies:

– Station oriented strategy heuristic (SH).

– Task oriented strategy heuristic (TH).

• Ordering rules:

– Maximum setup time plus processing times (max_ts).

– Minimum setup time plus processing times (min_ts).

– Maximum setup time (max_s).

– Minimum setup time (min_s).

When we refer to heuristic rule *SH-max_ts* we mean a Station oriented Heuristic that selects next task ordering candidate tasks (those whose predecessors have been already assigned and can fit in the actual open station) by MAXimum processing time plus Setup time. Hence, the list of heuristic rules that have been defined and tested in Section 5 are: *SH-max_ts, SH-max_s, SH-min_ts, SH-min_s, TH-max_ts, TH-max_s, TH-min_ts* and, *TH-min_s.*

In any of the station oriented strategy heuristics in every iteration the setup time considered when ordering the candidate tasks is the setup time between the last task assigned to the actual open station and every candidate task. In the case of the task oriented strategy heuristics, this setup time will be the one between the last task assigned to the first station to which the task could be assigned and the candidate task. In both cases, if the station is empty the mean setup time between the

candidate task i and all the other tasks j ($j \neq i$) is considered; and also the mean setup time between all the j and the candidate task i.

4.2. GRASP method

We propose to use a metaheuristic, namely GRASP (Feo and Resende, 1995), which is rather fast, quite simple to program, particularly apt at handling these types of constraints and is widely used. It has proven successful in many similar combinatorial optimization problems, including academic and industrial problems in scheduling, routing, partitioning, location and layout, graph theory, assignment, manufacturing, transportation, telecommunications and electrical power systems (see Laguna and Martı, 1999, as well as the extensive bibliography presented by (Festa and Resende, 2001) and (Resende and Ribeiro, 2003)).

Several studies have shown that GRASP produces good quality solutions for hard combinatorial optimization problems, particularly the set covering problems (Feo and Resende, 1995), the node packing problems (Feo et al., 1994) and the set packing problems (Delorme et al., 2004). Due to the fact that ALBPs can be seen as a special packing problem, we selected GRASP algorithm expecting a good behaviour of this search strategy when facing this kind of problem.

GRASP involves two steps: constructing a solution and improving it. Both steps are repeated a prescribed number of times.

The construction phase starts, following a station-oriented strategy, with the empty sequence and iteratively inserts a task randomly chosen among a restringed list of candidate tasks (RCL), which are a subset of tasks inside the *Candidate List* (CL). CL is composed of all tasks whose predecessors have been already assigned and fit into the actual open station; whereas RCL only includes those tasks from CL with better greedy index (see formula (11)), that indicates how interesting it is to insert each task in the current position of the sequence given all previously sequenced tasks.

In every iteration of the construction phase both CL and RCL evolve as tasks are assigned, and this process ends when all tasks have been sequenced and a feasible solution is available.

Let π_n the partial sequence of tasks until iteration n. The solution construction mechanism requires a number of iterations equal to the number of tasks to be sequenced less one (the last task is obviously directly determined). The CL in iteration n (CL_n) is formed by all candidate tasks en that iteration. For each task in CL_n the following greedy index is computed:

$$Index_j^n = \frac{1}{tsu_j^n + t_j}, \quad (11)$$

where tsu_j^n is the setup time between the last task sequenced in the sequence π and the task j, always that actual open station is not empty, or zero if it is; and t_j is the processing time of task j.

This index is a measure of the impact of sequencing task j in the iteration n, given the tasks previously sequenced. The lower the index is, the more interesting the task to be sequenced in iteration n.

Let Max_Index^n and Min_Index^n the maximum and minimum index values at iteration n, i.e.

$$Max\_Index^n = \max_{j \in CL_n} Index_j^n, \quad (12)$$

$$Min\_Index^n = \min_{j \in CL_n} Index_j^n, \quad (13)$$

In order to form the RCL in iteration n (RCL_n) a threshold is defined so that all tasks $j \in CL_n$ whose $Indexj^n$ is above that threshold will be selected.

$$Threshold^n = Min\_Index^n + \alpha \cdot \left(Max\_Index^n - Min\_Index^n \right), \quad (14)$$

$$j \in RCL_n \Leftrightarrow Index_j^n \leq Threshold^n. \quad (15)$$

The threshold depends on a parameter $\alpha \in [0, 1]$, which controls the trade-off between randomness and greediness in the constructive process. Thus, a value $\alpha = 1$ means that all candidate tasks can be selected in iteration n. On the contrary, a value $\alpha = 0$ means that only the task(s) with the most positive impact will be selected and the constructive process will behave, barring ties in the minimum index values, deterministically. After several testings the value of α has been experimentally set to 0.3.

The improvement step consists in a Local Search using an exchange movement that tries to exchange the positions of every two tasks in the sequence π, provided the exchange is feasible. For every feasible exchange, the variation in the objective function value is computed. If the variation is positive – i.e. the feasible exchange improves the objective function – then it is accepted and the modified solution becomes the current solution and the exploration of feasible exchanges restarted. The Local Search ends when a local minimum is attained, i.e. no feasible exchange can improve the current solution.

An important comment about the Local Search phase must be emphasised here: a solution with less number of stations than the current solution is always considered better. In case of having exactly the same number of stations, the following two functions were implemented to select one of them diversifying the evaluation of solution fitness:

• The first objective function's aim is to create solutions with their workload as balanced as possible. In order to get these kind of solutions, the objective function to minimize is

$$f^1(\pi) = \sum_{j=1}^{UM} \left(\frac{t(S_j)}{TC} \right)^2, \quad (16)$$

where UM is the number of workstations; $t(S_j)$ is the sum of processing and setup times of tasks assigned to workstation j; and TC is the cycle time.

• Maximizing the second objective function, imbalanced solutions are created. Obviously, the aim of this objective function is the diversification of solutions in the Local Search

$$f^2(\pi) = \sum_{j=1}^{UM} \left(\frac{1}{TC - t(S_j) + \varepsilon} \right), \quad (17)$$

where ε is a very low number (in our case 0.001).

In each GRASP iteration there is a probability p of using $f^1(\pi)$ and a probability $(1 - p)$ of using $f^2(\pi)$. Through this parameter, these two objectives are combined to avoid local optimums and efficiently diversify the solution space.

After several tests, the value of p has been experimentally set to 0.75; and the number of iterations of the GRASP algorithm has been experimentally set to 5 and 10, since no significant improvements appear after 10 iterations.

5. Experimental study

5.1. Benchmark generated

Since GALBPS is an innovative problem, there is no standard set of benchmark instances with setup times available for testing. Therefore a two-level three factors full factorial experimental study was constructed, containing problems with high and low *Size* (number of tasks); high and low *Order Strength* (which measures the structural properties of the precedence network); and high and low *Variability* of setup times.

According to this scheme, we selected 8 problems from the well-known classical SALBP instances collection, available in the assembly line balancing research homepage (www.assembly-line-balancing.de), which are representative enough for our aim:

• *Mitchell*: low *Size* (N = 21 tasks), high *Order strength* (OS = 70.95);

• *Roszieg*: low *Size* (N = 25 tasks), high *Order strength* (OS = 71.67);

• *Heskiaoff:* low *Size* (N = 28 tasks), low *Order strength* (OS = 22.49);

• *Sawyer:* low *Size* (N = 30 tasks), low *Order strength* (OS = 44.83);

• *Arcus1*: high *Size* (N = 83 tasks), high *Order strength* (OS = 59.09);

• *Lutz2:* high *Size* (N = 89 tasks), high *Order strength* (OS = 77.55);

• *Wee-Mag:* high *Size* (N = 75 tasks), low *Order strength* (OS = 22.67);

• *Bartholdi*: high *Size* (N = 148 tasks), low *Order strength* (OS = 25.80).

From each problem, the original precedence network and operation times were preserved and 10 instances were generated, randomly generating the input cycle time for every instance according to a uniform discrete distribution $U[c_{min}, c_{max}]$, where c_{min} and c_{max} are the minimum and maximum cycle times considered in the assembly line balancing research homepage (e.g. [14, 39] for *Mitchel* problem).

Finally, for every problem, the following two levels of setup times variability was set:

• For low *Variability* we randomly generated the matrix of setup times according to a uniform discrete distribution $U[0, \lceil 0.25 \cdot \min \forall_{i \in N} t_i \rceil]$.

• For high *Variability* we randomly generated the matrix of setup times according to a uniform discrete distribution $U[0, \lceil 0.75 \cdot \min \forall_{i \in N} t_i \rceil]$.

Therefore, the benchmark is composed of 160 instances with different combinations of *Size, Order Strength* and *Variability* of setups, that will enable us to extract conclusions about the overall behaviour of every procedure presented in Section 4; as well as the convenience (if any exist) of some procedures against a specific kind of problem.

5.2. BLP model results

The BLP model was implemented with CPLEX 9® and tested with all the 160 instances presented in the last section. The instances were launched on a Pentium IV, CPU 3.4 GHz with 512 de RAM with a maximum computation time set to 2000 seconds. Results showed, as it was expected due to the NP-hard nature of the problem, a great difficulty to calculate exact solutions. Table 3 reports the number of samples where the optimum was reached (*Opt*); those samples where only feasible sub-optimal solutions were obtained (*Fea*); and those where no feasible solution was provided by CPLEX $(\overline{Fea})$.

Table 3.

Results for BLP model

Opt	***Fea***	$\overline{Fea}$
12	5	143

When modelling the problem some known ways for efficiently modelling ALB problems (Pastor et al., 2004) were applied; and also some subset variables were carefully defined and applied in order to save as many constraints as possible to the model. Despite this, results were not satisfactory, but in any case the BLP model was very useful. As was introduced, the solution values here obtained can be used as a lower bound to evaluate the efficiency of heuristic procedures, as will be exposed in the experimental study carried out in the next section.

5.3. Lower bound and solution quality indicator

Both the eight heuristic rules and the GRASP procedure defined in Section 4 were applied to solve the 160 instances generated. As it was impossible to solve exactly all these instances, it was necessary to define a lower bound to be calculated for every instance, so that this value can be

compared to the solution provided by every heuristic. The first lower bound designed, named as $LB1_{\text{GALBPS}}$, is an adaptation of the most usual SALBP lower bound:

Step 1. Calculate $LB1^{0}_{GALPS} = \left\lceil \frac{\sum_{i=1,\ldots,N} t_i}{ct} \right\rceil$, where *ct* is the cycle time

$k = 1$ and GO to *Step 2*.

Step 2. IF $LB1^{k-1}_{GALPS} = N$ then END and $LB1_{GALPS} := N$;

ELSE: calculate $LB1^{k}_{GALPS} = \left\lceil \frac{\sum_{i=1,\ldots,N} t_i + SUT}{ct} \right\rceil$,

where *SUT* is the sum of the $N - LB1^{k-1}_{GALPS} + 1$ lowest setup times between the *N* tasks.

Step 3. IF $LB1^{k}_{GALPS} \leq LB1^{k-1}_{GALPS}$ END and $LB1_{GALPS} = LB1^{k-1}_{GALPS}$;

ELSE $k = k + 1$ and GO to *Step 2*.

For the example presented in Section 1.1:

$$LB1^{0}_{GALPS} = \left\lceil \frac{182}{71} \right\rceil = \lceil 2.56 \rceil = 3,$$

$$LB1^{1}_{GALPS} = \left\lceil \frac{182 + (0+0+0+0+0+1+1+1)}{71} \right\rceil = \left\lceil \frac{185}{71} \right\rceil = \lceil 2.61 \rceil = 3,$$

$$LB1_{GALPS} = 3.$$

And, as it was introduced, the second lower bound $LB2_{\text{GALBPS}}$ is the lowest integer number upper to the number of stations provided by the BLP model after 2000 seconds of runtime. Therefore the global lower bound used in the following experimentation will be

$LB_{\text{GALBPS}} = \max(LB1_{\text{GALBPS}}, LB2_{\text{GALBPS}})$. (18)

Solving all the 160 instances with all the 10 procedures (eight heuristic rules and GRASP algorithm with 5 and 10 iterations), a total of 1600 experiments were launched. To measure the quality of the solutions, the adimensional indicator *Number of Workstations Percentage Increase* (NWPI) was defined, and calculated for each one of the 1600 experiments. This indicator shows the percentage deviation between the number of stations provided by a heuristic and LB_{GALBPS}. Using NPWI, an ANOVA analysis was made for evaluating both the absolute deviation from the lower bound LB_{GALBPS}, and the relative behaviour between the different heuristics.

The main conclusions of this ANOVA analysis are reported in the next section (note that percentage increases NWPI's are higher since they were obtained against the number of stations provided by the lower bound, which are usually less than the exact solution).

5.4. Solution quality results

From our ANOVA analysis we may summarize the main conclusions obtained by means of the Fisher Test Graphics provided by ANOVA.

In Fig. 2, we observe that the heuristic with a better overall behaviour was *GRASP-10*, which means GRASP with 10 iterations. But we may highlight how the simple heuristic rule *SH_max_ts* (station oriented heuristic ordering the tasks from maximum to minimum setup time plus task time when selecting the task) was not far from *GRASP*, despite its simplicity. In both cases percentage deviations close to 10% from the lower bound (not from the exact solution) are little enough to recommend their application, also taking account how fast these heuristics become compared to exact methods.

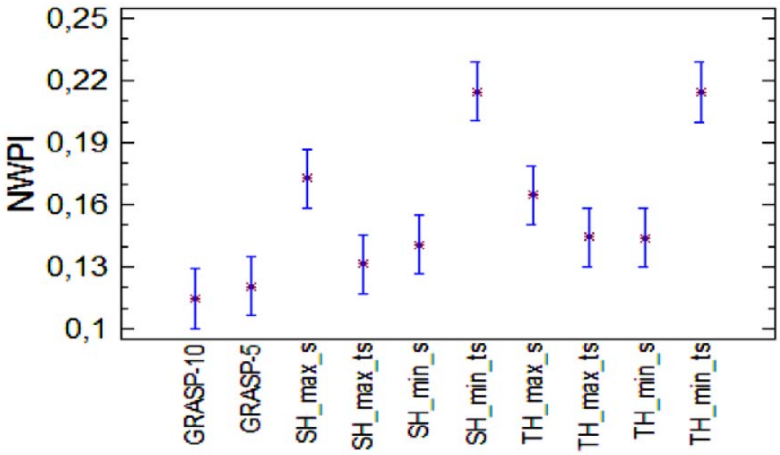

Fig. 2. Means and 95.0% LSD intervals graphic for methods.

About the size of the problems (*N*) it was noticed, as it was expected, a quite robust behaviour where every procedure had always better results with the small problems (*low N*). About the other two variables, in Fig. 3 it is showed how better results were obtained by every procedure when facing problems with low Order Strength (*OS*) and low Variability of setup times (*Var*).

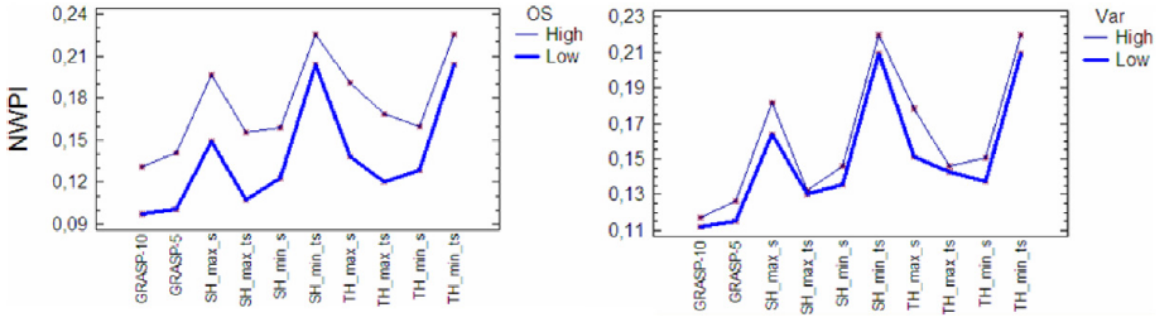

Fig. 3. Interaction plots for Order Strength (OS) and setup times variability (*Var)*.

With such a robust behaviour it's not worth displaying any more double interaction plots. The conclusions are quite clear: the problems with low *N*, low *OS* and low *Var* are easier to solve, and GRASP procedures are always the winners.

5.5. Computational times

The problems were launched on a Pentium IV CPU 3.4 GHz with 512 de RAM, the same machine that was used to solve the examples shown in 5.2, under Windows XP and using Java JRE 1.5.06.

In the case of the heuristic rules, the computational time for the eight procedures defined was always less than one second, therefore no further computational analysis is necessary for them. In the case of the GRASP algorithm, computational results are summarized through the Fisher Test Graphics (computational time in seconds) shown in Fig. 4.

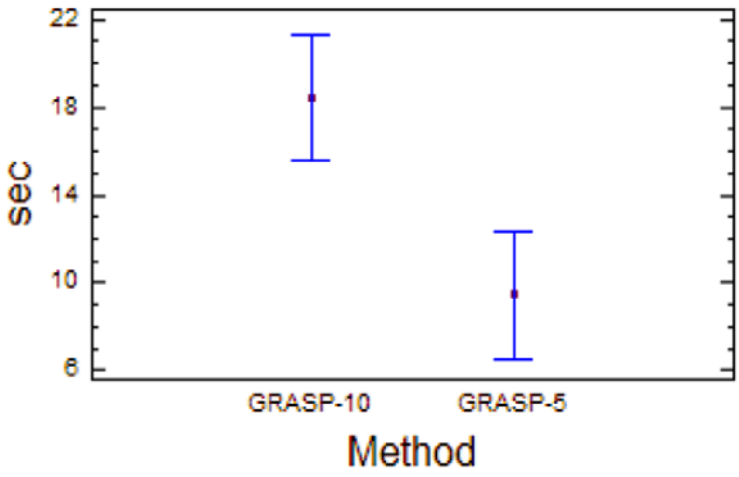

Fig. 4. Means and 95.0% LSD intervals graphic for GRASP computational time.

After several tests, the number of iterations of the GRASP algorithm have been experimentally set to 5 and 10, since no significant improvements appear after 10 iterations (even the percentage improvement from 5 to 10 iterations is not so high). Fig. 5 evidences this fact. It shows how the computational time increases very quickly with the size of the problem, and this increase is bigger when more iterations are set:

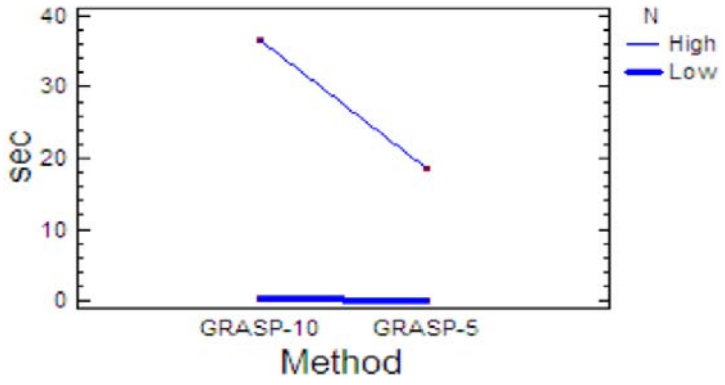

Fig. 5. Interaction plot for size of the problem (*N*).

In fact it was tested how is no worth to run the GRASP up to more than 10 iterations, since the relative solution quality increase is very weak. In fact, the test shows that it is not worth running the GRASP with more than 10 iterations, since the relative solution quality increase is very weak.

The heuristic rule that showed the best behaviour was *SH_max_ts*. This heuristic is recommended due to the fact that it gets almost the same quality of solutions as GRASP, but always consumes less than one second of computational time.

6. Conclusions and further research

In this paper an innovative assembly line balancing problem which adds sequence-dependent setup time considerations has been addressed. After an extensive bibliographical search – both of sequencing and line balancing literature – very few references addressing similar problems were found. Therefore, as the problem is not codified in any of the last surveys published, the problem has been named as GALBPS (General Assembly Line Balancing Problem with Setups) and it has been completely defined through nine basic assumptions and a binary programming model; also having been tested against several problems. These initial tests proved the requirement of heuristic procedures for solving problems of realistic sizes in reasonable computational time for industrial environments. Thus, eight different heuristic rules were designed regarding task-oriented and station-oriented strategies with different task selection ordering criteria.

Furthermore, a GRASP algorithm has been presented and evaluated against the heuristic rules through an experimental study, whose main results reported the best behaviour of GRASP algorithm and the relative good behaviour of the simple heuristic rule *SH_max_ts*; which is a station oriented heuristic ordering the tasks from maximum to minimum setup time plus task time.

As a first approach to this innovative problem, the results obtained are encouraging and our work will continue basically with two interesting research lines: on one hand the design of heuristics derived from the mathematical model through *fix & relax* or other strategies; whilst on the other hand, the application of some other metaheuristic to the problem. Initially, genetic algorithms and tabu search are the best candidates.

References

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

Allahverdi et al., 1999 A. Allahverdi, J.N.D. Gupta and T. Aldowaisan, A review of scheduling research involving setup considerations, *Omega* 27 (1999), pp. 219–239.

Allahverdi et al., in press Allahverdi, A., Ng, C.T., Cheng, T.C.E., Kovalyov, M.Y., in press. A survey of scheduling problems with setup times or costs. European Journal of Operational Research, doi:10.1016/j.ejor.2006.06.060.

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

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

Bukchin and Tzur, 2002 J. Bukchin and M. Tzur, Design of flexible assembly line to minimize equipment cost, *IIE Transactions* 32 (2002), pp. 585–598.

Delorme et al., 2004 X. Delorme, X. Gandibleux and J. Rodrı´guez, GRASP for set packing problems, *European Journal of Operational Research* 153 (2004), pp. 564–580.

Dolgui, 2006 A. Dolgui, Balancing assembly and transfer lines, *European Journal of Operational Research* 168 (2006), pp. 663–665.

Erel and Sarin, 1998 E. Erel and S. Sarin, A survey of the assembly line balancing procedures, *Production Planning and Control* 9 (5) (1998), pp. 414–434.

Feo and Resende, 1995 T.A. Feo and M.G. Resende, Greedy randomized adaptative search procedures, *Journal of Global Optimization* 6 (1995), pp. 109–133.

Feo et al., 1994 T.A. Feo, M.G. Resende and S.H. Smith, A greedy randomized adaptative search procedure for maximum independent set, *Operations Research* 42 (1994), pp. 860–878.

Festa and Resende, 2001 P. Festa and M.G. Resende, GRASP: An annotated bibliography. In: C.C. Ribeiro and P. Hansen, Editors, *Essays and Surveys on Metaheuristics*, Kluwer Academic Publishers (2001), pp. 325–367.

Ghosh and Gagnon, 1989 S. Ghosh and R.J. Gagnon, A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems, *International Journal of Production Research* 27 (1989), pp. 637–670.

Gutjahr and Nemhauser, 1964 A. Gutjahr and G.L. Nemhauser, An algorithm for the line balancing problem, *Management Science* 11 (1964), p. 2.

Jayashakandar et al., 1999 M. Jayashakandar, S. Swaminathan and R. Tayur, Managing design of assembly sequences for product lines that delay product differentiation, *IIE Transactions* 31 (1999), pp. 1015–1026.

Laguna and Martı, 1999 M. Laguna and R. Martı, GRASP and path relinking for 2-layer straight line crossing minimization, *INFORMS Journal on Computing* 11 (1999), pp. 44–52.

Merengo et al., 1999 F. Merengo and Nava A. Pozzetti, Balancing and sequencing manual mixed model assembly lines, *International Journal of Production Research* 37 (12) (1999), pp. 2835–2860.

Pastor et al., 2004 R. Pastor, A. Corominas and A. Lusa, Different ways of modelling and solving precedence and incompatibility constraints in the Assembly Line Balancing Problem, *Frontiers in Artificial Intelligence and Applications* 113 (2004), pp. 359–366.

Pinnoi and Wilhelm, 1997 A. Pinnoi and W.E. Wilhelm, A family of hierarchical models for assembly system design, *International Journal of Production Research* 35 (1997), pp. 253–280.

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

Resende and Ribeiro, 2003 M.G. Resende and C.C. Ribeiro, Greedy randomized adaptive search procedures. In: F. Glover and G. Kochenberger, Editors, *Handbook of Metaheuristics*, Kluwer Academic Publishers (2003), pp. 219–249.

Salveson, 1955 M.E. Salveson, The assembly line balancing problem, *Journal of Industrial Engineering* 6 (3) (1955), pp. 18–25.

Sawik, 2002 T. Sawik, Balancing and scheduling of surface mount technology lines, *International Journal of Production Research* 40 (9) (2002), pp. 1973–1991.

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

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