An improved mathematical program to solve the simple assembly line balancing problem

Rafael Pastor, Laia Ferrer

An improved mathematical program to solve the simple assembly line balancing problem

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2007-IJPR-0383
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	24-May-2007
Complete List of Authors:	Pastor, Rafael; Technical University of Catalonia, IOC Research Institute Ferrer, Laia; Technical University of Catalonia, IOC Research Institute
Keywords:	ASSEMBLY LINE BALANCING, ASSEMBLY LINES
Keywords (user):	ASSEMBLY LINE BALANCING, MIXED INTEGER LINEAR PROGRAMMING

An improved mathematical program to solve the simple assembly line balancing problem[†]

Rafael Pastor[*] and Laia Ferrer

IOC Research Institute

Technical University of Catalonia

Av. Diagonal 647, p.11, 08028, Barcelona, Spain

{rafael.pastor/laia.ferrer}@upc.edu

Abstract

The Simple Assembly Line Balancing Problem (SALBP) has been extensively examined in the literature. Various mathematical programs have been developed to solve SALBP type-1 (minimizing the number of workstations, m, for a given cycle time, ct) and SALBP type-2 (minimizing ct given m). Usually, an initial pre-process is carried out to calculate the range of workstations to which a task i may be assigned, in order to reduce the number of variables of task-workstation assignment. This paper presents a more effective mathematical program than those released to date to solve SALBP-1 and SALBP-2. The key idea is to introduce additional constraints in the mathematical program, based on the fact that the range of workstations to which a task i may be assigned depends either on the upper bound on the number of workstations or on the upper bound on the cycle time (for SALBP-1 and SALBP-2, respectively). A computational experiment was carried out and the results reveal the superiority of the mathematical program proposed.

Keywords: assembly line balancing, mixed integer linear mathematical programming

[†] Supported by the Spanish MCyT projects DPI2004-03472 and DPI2007-61905, co-financed by FEDER.

1. Introduction

The Simple Assembly Line Balancing Problem (SALBP), as was defined by Baybars (1986), for instance, basically consists of assigning a set of indivisible tasks (which are characterized by their processing times and by a set of precedence relations) to workstations in such a way that precedence constraints are fulfilled, the work content of each workstation does not exceed the cycle time and a given efficiency measure is optimized. When the objective is to minimize the number of workstations m for a given cycle time ct, the problem is usually referred to as SALBP-1; if the objective is to minimize ct given m, the problem is called SALBP-2 –see Baybars (1986).

The design of assembly lines has been extensively examined in the literature, particularly the SALB problem. Several reviews have been published –the latest by Erel and Sarin (1998), Rekiek et al. (2002) and Scholl and Becker (2006)– and a huge amount of specific research exists. For SALBP, both heuristic –e.g., Scholl and Voss (1996)– and exact procedures –e.g., Johnson (1988), Hoffmann (1992) and Scholl and Klein (1997)– have been developed.

Some of the exact procedures are based on binary linear programming and mixed integer linear programming. SALBP was first formulated as a 0-1 mathematical program by Bowman (1960) and was later modified by White (1961). An improved version of Bowman's model was presented by Patterson and Albracht (1975), in which four 0-1 formulations of the assembly line balancing problem are compared. This paper also defines for each task i, the earliest and the latest stations to which task i can be

* Corresponding author: Rafael Pastor, IOC Research Institute, Av. Diagonal 647 (edif. ETSEIB), p.11, 08028 Barcelona, Spain; Tlf. + 34 93 401 17 01; Fax. + 34 93 401 66 05; e-mail: rafael.pastor@upc.edu

assigned, E_i and L_i respectively, based on the precedence relations; this is used together with lower and upper bounds on the number of workstations needed to significantly reduce the size of the formulation of the problem. Thangavelu and Shetty (1971) report similar simplifications. An alternative formulation as a general integer programming problem was proposed by Talbot and Patterson (1984); it was solved by using an adaptation of the Balas' algorithm (see Amen (2006) for an improved formulation of the precedence relations in the models designed for solving by use of Balas ideas). The effectiveness of several ways of modelling the technological precedence constraints of SALBP and its solution using mathematical programming and constraint logic programming are shown in Pastor et al. (2004); therefore, here we consider one of the two most successful models identified in Pastor et al. (2004) as the referring model to test the effectiveness of our ideas.

Amen (2006) indicates that, according to in the literature, the outline of the exact formulations developed for SALBP is different depending on the solution technique to be used: branch-and-bounds with LP-relaxation or implicit enumeration techniques (for further references see Amen, 2006). In this work we present a new approach belongs to the branch-and-bound technique with LP-relaxation.

In most of the mathematical models, an initial pre-process is carried out to reduce the number of variables of task-workstation assignment. As explained above, the aim of this pre-process is to calculate, for each task i, the earliest and the latest workstations to which task i can be assigned, E_i and L_i (see for example Saltzman and Baybars, 1987).

Take t_i as the processing time of task i; PT_i the set of tasks which precede task i; ST_i the set of tasks which succeed task i; and $\lceil x \rceil$ the smallest integer value not smaller than x. For SALBP-1, given a value of ct and an upper bound on the number of workstations, $m_{\max}$, E_i and $L_i(m_{\max})$ are calculated as (1) and (2):

$$E_i = \left\lceil \frac{t_i + \sum_{k \in PT_i} t_k}{ct} \right\rceil \tag{1}$$

$$L_i(m_{\max}) = m_{\max} + 1 - \left\lceil \frac{t_i + \sum_{k \in ST_i} t_k}{ct} \right\rceil \tag{2}$$

For SALBP-2, given a value of m and an upper bound on the cycle time, $ct_{\max}$, $E_i(ct_{\max})$ and $L_i(ct_{\max})$ are calculated as (3) and (4):

$$E_i(ct_{\max}) = \left\lceil \frac{t_i + \sum_{k \in PT_i} t_k}{ct_{\max}} \right\rceil \tag{3}$$

$$L_i(ct_{\max}) = m + 1 - \left\lceil \frac{t_i + \sum_{k \in ST_i} t_k}{ct_{\max}} \right\rceil \tag{4}$$

These calculations provide the range of workstations $[E_i, L_i]$ to which any task i can be assigned.

Let us consider the precedence graph shown in Figure 1, in which the value over the nodes indicates the processing time of each task.

Insert Figure 1

Now let us consider a SALBP-1 with a given cycle time $ct = 9$, an upper bound on the number of workstations $m_{\max} = 6$ and a lower bound on the number of workstations of $m_{\min} = \left\lceil \sum t_i / ct \right\rceil = \left\lceil 25/9 \right\rceil = 3$ (so that the range of values of the number of workstations is: $m \in [3,6]$). Table 1 shows the first and the last workstation to which each task can be assigned according to the value of m, calculated as (1) and (2).

Insert Table 1

In the initial stage of the pre-process, with $m_{\max} = 6$, the range of workstations to which task i can be assigned is calculated, for instance $[1,5]$ for task B. However, additional knowledge has not been exploited: if only four workstations are used (that is, the fifth and the sixth stations are not needed) the range of workstations to which the tasks can be assigned is reduced and becomes $[1,3]$ for task B ($E_B = 1$ and $L_B(4) = 3$).

The aim of this study is to provide a more effective mathematical program than those that have previously been published to solve SALBP-1 and SALBP-2. The new idea is to introduce additional constraints into the mathematical program, based on the fact that the workstation interval $[E_i, L_i]$ depends either on the upper bound on the number of

workstations or on the upper bound on the cycle time, for SALBP-1 and SALBP-2, respectively. The idea of a feasible workstation interval $[E_i, L_i]$ is not new idea (it has already been defined in the literature, see for example Saltzman and Baybars, 1987). What has not yet been done, hence our contribution, is modelling the additional knowledge introduced by this range and introducing this range in the mathematical program as additional constraints. In mathematical programs, a binary variable associated to the existence of each workstation j, $y_j \in \{0,1\}$, is normally used –so it is immediately known if a workstation j is not used $(y_j = 0)$– and binary variables of task-workstation assignment are also included: $x_{ij} \in \{0,1\}$ $(\forall i; j = E_i,...,L_i)$, which take a value of 1 if (and only if) task i is assigned to workstation j. The key idea is to force the variable x_{ij} to take a value of 0 if the last workstation q that makes $L_i(q) = j$ is not used (i.e. $y_q = 0$); in the particular case of task B this can be achieved by adding the constraints $x_{B5} \le y_6$, $x_{B4} \le y_5$ and $x_{B3} \le y_4$. If workstations 6 and 5 are not used (i.e. $y_6 = y_5 = 0$) task B cannot be assigned to workstations 5 and 4, respectively ($x_{B5} \le y_6 = 0$ and $x_{B4} \le y_5 = 0$).

Now consider a SALBP-2 with a given number of workstations $m = 5$ and the range of values for the cycle time: $ct \in [5,10]$. Tables 2 and 3, respectively, show the first and the last workstation to which each task can be assigned according to the value of ct, calculated as indicated in (3) and (4). A shaded cell shows the value of the cycle time ct in which either the first workstation to which a task can be assigned decreases or the last workstation to which a task can be assigned increases, with regard to a value

$ct' = ct - 1$. Similar reasoning to that used for SALBP-1 can be applied to SALBP-2, provided that ct is an integer value, which is assumed without losing generality.

Insert Table 2

Insert Table 3

At present, state-of-the-art optimization tools are already able to solve middle-sized assembly line balancing problems within reasonably limited calculation times by means of mathematical programming. Moreover, the working environment is becoming more and more powerful, thanks to both the hardware and the software available –according to Bixby (2002), in the last decade, the problem-solving speed of mathematical programs has increased by a factor of approximately 1,000,000 due to improvements in these two elements–. On the other hand, Atamtürk and Savelsbergh (2005) mention that "*integer programming is rapidly gaining acceptance as a powerful computational tool that can provide optimal or near-optimal solutions to real-life strategic and operational planning problems*". Therefore, any improvements in the modelling and solving of this problem are critical in order to solve real-life cases in an optimal way, as has already been achieved, for example, in Corominas et al. (2006).

The remaining paper is organized as follows. In Section 2 the initial models for SALBP-1 and SALBP-2 are presented. The resulting models obtained by incorporating the proposed additional constraints are explained in Section 3. In Section 4 the models are tested on a set of well-known instances and the computational evaluation reveals that

the new models perform better than the previous ones. Finally, Section 5 presents the conclusions of the study.

2. Mathematical programming models for SALBP

In order to solve SALBP, several common and equivalent mathematical programming models can be considered. In particular, here we consider one of the two most successful models of Pastor et al. (2004). First the model for SALBP-1 is presented, and then the changes made in order to obtain the model for SALBP-2 are explained. The previous models for SALBP-1 and SALBP-2 are refered to as *SALBP-1-i* and *SALBP-2-i*, respectively.

Data:

n	Number of tasks $(i = 1,...,n)$.
$m_{\max}$	Upper bound on the number of workstations $(j = 1,...,m_{\max})$.
$m_{\min}$	Lower bound on the number of workstations.
t_i	Processing time of task i.
ct	Cycle time.
E_i	Earliest possible workstation for task i.
$L_i(m_{\max})$	Latest possible workstation for task i, given a value of $m_{\max}$; to simplify the terminology, only L_i is used.

P Set of pairs of tasks (i,k) such that there is immediate precedence between them.

Variables:

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

$y_j \in \{0,1\}$ 1, if and only if any task is assigned to workstation j $(j = m_{\min}+1,...,m_{\max})$

Model SALBP-1-i:

$$[MIN]Z = \sum_{j=m_{\min}+1}^{m_{\max}} j \cdot y_j \quad (5)$$

$$\sum_{j=E_i}^{L_i} x_{ij} = 1 \qquad \forall i \quad (6)$$

$$\sum_{\forall i | j \in [E_i, L_i]} t_i \cdot x_{ij} \le ct \qquad (j = 1,...,m_{\min}) \quad (7)$$

$$\sum_{\forall i | j \in [E_i, L_i]} t_i \cdot x_{ij} \le ct \cdot y_j \qquad (j = m_{\min}+1,...,m_{\max}) \quad (8)$$

$$\sum_{j=E_i}^{L_i} j \cdot x_{ij} \le \sum_{j=E_k}^{L_k} j \cdot x_{kj} \qquad \forall (i,k) \in P \quad (9)$$

The objective function (5) minimizes the number of workstations (moreover, this function forces $y_{j+1} = 0$ when $y_j = 0$); constraint set (6) implies that each task i is assigned to one and only one workstation; constraints (7) and (8) ensure that the total

task processing time assigned to workstation j does not exceed the cycle time; constraint set (9) imposes the technological precedence conditions.

In SALBP-2 the objective is to minimize the cycle time ct given the number of workstations m. Therefore, new data and variables are needed (in addition to n, t_i, P and x_{ij}):

Data:

m	Number of workstations $(j = 1,...,m)$.
$ct_{\max}$	Upper bound on the cycle time.
$E_i(ct_{\max})$	Earliest possible workstation for task i, given a value of $ct_{\max}$; to simplify the terminology, only E_i is used.
$L_i(ct_{\max})$	Latest possible workstation for task i, given a value of $ct_{\max}$; to simplify the terminology, only L_i is used.

Variable:

$ct \geq 0$	Cycle time

Model SALBP-2-i:

$$[MIN] Z = ct \quad (10)$$

$$\sum_{j=E_i}^{L_i} x_{ij} = 1 \qquad \forall i \qquad (6)$$

$$\sum_{\forall i \mid j \in [E_i, L_i]} t_i \cdot x_{ij} \leq ct \qquad \forall j \qquad (11)$$

$$\sum_{j=E_i}^{L_i} j \cdot x_{ij} \leq \sum_{j=E_k}^{L_k} j \cdot x_{kj} \qquad \forall (i,k) \in P \qquad (9)$$

The objective function (10) minimizes the cycle time and constraint set (11) ensures that the total task processing time assigned to workstation j does not exceed the cycle time.

3. New mathematical programming models for SALBP

Next, the new mathematical programming models proposed for SALBP-1 and SALBP-2 are presented. We refer to these models as *SALBP-1-n* and *SALBP-2-n*, respectively. As has been introduced in Section 2, the key idea is to introduce additional constraints to the mathematical program, based on the fact that the range of the interval of workstations $[E_i, L_i]$ depends on the upper bound on the number of workstations or on the upper bound on cycle time (for SALBP-1 and SALBP-2, respectively).

For SALBP-1 the data and the variables are the same as those used in the previous model.

Model SALBP-1-n:

$$[MIN]Z = \sum_{j=m_{\min}+1}^{m_{\max}} j \cdot y_j \tag{5}$$

$$\sum_{j=E_i}^{L_i} x_{ij} = 1 \qquad \forall i \tag{6}$$

$$\sum_{\forall i \mid j \in [E_i, L_i]} t_i \cdot x_{ij} \le ct \qquad (j = 1, ..., m_{\min}) \tag{7}$$

$$\sum_{\forall i \mid j \in [E_i, L_i]} t_i \cdot x_{ij} \le ct \cdot y_j \qquad (j = m_{\min}+1, ..., m_{\max}) \tag{8}$$

$$\sum_{j=E_i}^{L_i} j \cdot x_{ij} \le \sum_{j=E_k}^{L_k} j \cdot x_{kj} \qquad \forall (i,k) \in P \tag{9}$$

$$x_{i,L_i-q} \le y_{m_{\max}-q} \qquad \forall i; q = 0, ..., m_{\max} - m_{\min} - 1 \tag{12}$$

The new constraints (12) annuls the assignment variable of task i to a particular workstation (namely p), when the number of workstations used (determined by the variable $y_j = 1$ with the biggest value of j) is such that workstation p is outside the current range of possible workstations: $[E_i, ..., L_i(j)]$.

For SALBP-2 the following data must be redefined:

$ct_{\min}$	Lower bound on the cycle time.
$E_i(ct)$	Earliest possible workstation for task i, given a value of ct.
$L_i(ct)$	Latest possible workstation for task i, given a value of ct.

as well as the following variables:

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

$r_t \in \{0,1\}$ 1, if and only if the cycle time is equal to t $\left(t = ct_{\min},...,ct_{\max}\right)$; if the processing times are integer values, ct is an integer value (which is assumed without losing generality).

Model SALBP-2-n:

$$\left[MIN\right] Z = ct \tag{10}$$

$$\sum_{j=E_i(ct_{\max})}^{L_i(ct_{\max})} x_{ij} = 1 \qquad \forall i \tag{13}$$

$$\sum_{\forall i \mid j \in \left[E_i(ct_{\max}),L_i(ct_{\max})\right]} t_i \cdot x_{ij} \le ct \qquad \forall j \tag{14}$$

$$\sum_{j=E_i(ct_{\max})}^{L_i(ct_{\max})} j \cdot x_{ij} \le \sum_{j=E_k(ct_{\max})}^{L_k(ct_{\max})} j \cdot x_{kj} \qquad \forall (i,k) \in P \tag{15}$$

$$\sum_{t=ct_{\min}}^{ct_{\max}} r_t = 1 \tag{16}$$

$$ct = \sum_{t=ct_{\min}}^{ct_{\max}} t \cdot r_t \tag{17}$$

$$x_{i,E_i(ct_{\max})+a-1} \le 1 - \sum_{\forall t \in \left[ct_{\min},ct_{\max}-1\right] \mid E_i(t) > E_i(ct_{\max})+a-1} r_t \qquad \forall i; a = 1,...,E_i\left(ct_{\min}\right) - E_i\left(ct_{\max}\right) \tag{18}$$

$$x_{i,L_i(ct_{\max})-b+1} \le 1 - \sum_{\forall t \in \left[ct_{\min},ct_{\max}-1\right] \mid L_i(t) < L_i(ct_{\max})-b+1} r_t \qquad \forall i; b = 1,...,L_i\left(ct_{\max}\right) - L_i\left(ct_{\min}\right) \tag{19}$$

Constraints (13), (14) and (15) are equivalent to (6), (11) and (9), respectively. The new constraints (16) forces the cycle time to be between its lower bound $\left(ct_{\min}\right)$ and its

upper bound $\left(ct_{\max}\right)$; (17) links the cycle time variable $\left(ct\right)$ with the variables that specify the cycle time reached $\left(r_t\right)$; finally, (18) and (19) impose that when the cycle time is equal to t $\left(r_t = 1\right)$ the assignment variables of each task i to the workstations that do not belong to its current range of possible workstations $\left[E_i\left(t\right),...,L_i\left(t\right)\right]$ are annulled.

To make the understanding of the new constraints (16) to (19) easier, these constraints are shown below for tasks A and G of the second example given in Section 1 (figure 1, and tables 2 and 3). Constraint (20) specifies constraint (16); (21) specifies (17); (22) and (23) specify (19) for task A (A has no constraints of the type (18)); and (24) and (25) are (18) for task G (G has not constraints of the type (19)).

$$r_5 + r_6 + r_7 + r_8 + r_9 + r_{10} = 1 \tag{20}$$

$$ct = 5 \cdot r_5 + 6 \cdot r_6 + 7 \cdot r_7 + 8 \cdot r_8 + 9 \cdot r_9 + 10 \cdot r_{10} \tag{21}$$

$$x_{A3} \le 1 - \left(r_5 + r_6 + r_7 + r_8\right) \tag{22}$$

$$x_{A2} \le 1 - \left(r_5 + r_6\right) \tag{23}$$

$$x_{G3} \le 1 - \left(r_5 + r_6 + r_7\right) \tag{24}$$

$$x_{G4} \le 1 - \left(r_5\right) \tag{25}$$

The logic of the idea proposed in this paper is as follows. Let assume that the optimization process has already obtained a feasible solution with a cycle time equal to 6 $\left(ct = 6\right)$. Taking into account $ct = 6$ and constraints (20) and (21), we obtain $r_6 = 1$ and constraints (22) to (25) become (22') to (25'):

$$x_{A3} \leq 0 \qquad (22')$$

$$x_{A2} \leq 0 \qquad (23')$$

$$x_{G3} \leq 0 \qquad (24')$$

$$x_{G4} \leq 1-(r_5) \qquad (25')$$

That is to say, variables x_{A2}, x_{A3} and x_{G3} become zero directly, what prevents task A to be assigned to workstations 2 and 3, and also avoids the assignment of task G to workstation 3.

The two complete models for the first example given in Section 1, a SALBP type 1 (figure 1 and table 1), are detailed in the Annex, in order to clarify the new additional constraints and allow for the comparison of the two models.

4. Computational experiment

A computational experiment was carried out in order to evaluate the effectiveness of the new models compared to previous models for SALBP.

The basic data used for the experiment are as follows:

- All the well-known instances available on the Scholl's and Klein's homepage for assembly line balancing research (www.assembly-line-balancing.de) were used: 269 instances for SALBP-1 and 302 for SALBP-2. The instances contain a wide range of

values of the cycle time for SALBP-1 (from 6 to 17,067 units of time), the number of workstations for SALBP-2 (from 3 to 52 workstations), the number of tasks (from 7 to 297 tasks), the number of precedences (from 7 to 423 precedences) and the average task processing time (from 4.11 to 1,354.95 units of time).

- Mathematical programs were solved using the ILOG CPLEX 9.0 Optimizer, with a PC Pentium IV at 3.7 GHz and with 512 Mb of RAM.
- A maximum computing time of 2,000 seconds was set (this is not an excessive computing time if we take into account that we are solving a design problem of a production system).

- $m_{\min}$ was calculated as $\left\lceil \sum_{i=1}^{n} t_i \Big/ ct \right\rceil$
- $m_{\max}$ was calculated as $\min\left(2 \cdot m_{\min}; n\right)$
- $ct_{\min}$ was calculated as $\max\left(\max_{\forall i}\left(t_i\right); \left\lceil \sum_{i=1}^{n} t_i \Big/ m \right\rceil \right)$
- $ct_{\max}$ was calculated as $2 \cdot ct_{\min}$

Before carrying out the complete computational experiment, several initial experiments were done in order to set the previous values.

For SALBP-1, the value of $m_{\max}$ was defined as $\min\left(2 \cdot m_{\min}; n\right)$ and n, and 150 instances were solved. For SALBP-2, the value of $ct_{\max}$ was defined as $2 \cdot ct_{\min}$ and $\sum_{i=1}^{n} t_i$, and 80 instances were solved. It was verified that the models are not very sensitive to the value of any of these parameters and that the best results are obtained with $m_{\max} = \min\left(2 \cdot m_{\min}; n\right)$ and $ct_{\max} = 2 \cdot ct_{\min}$.

Sometimes the solution process can be faster when a pre-calculation of a feasible heuristic solution is done, since it gives tighter bounds. However, in this work this possibility has not been considered, because when the values of $m_{\max}$ and $ct_{\max}$ are adjusted, the number of additional constraints decreases (and, it could even be zero) making it difficult to evaluate its influence on the mathematical model.

The most important elements of the new model are the additional constraints (constraints (12) for SALBP-1, constraints (18) and (19) for SALBP-2). The number of additional constraints to be generated could be high and it could influence the effectiveness of the new models. For constraints type (12), (18) and (19) may be $n \cdot (m_{\max} - m_{\min})$, $n \cdot (E_i(ct_{\min}) - E_i(ct_{\max}) - 1)$ and $n \cdot (L_i(ct_{\max}) - L_i(ct_{\min}) - 1)$, respectively. A brief computational experiment is carried out to decide the quantity of additional constraints to be included. Three quantities of additional constraints are tested: including all the possible constraints (All), including half of the possible constraints $(Half)$ and including only the first one (One). Table 4 shows the results obtained for SALBP-1 and SALBP-2: the number of instances with a proved optimal solution $(Opt - prov)$; the number of instances with a feasible solution (Fea); and the number of instances without a feasible solution $(\overline{Fea})$.

Insert Table 4

Better solutions are obtained when all the additional constraints are included in the model, that is, generating all the constraints as stated in the model. Therefore, all the additional constraints will be included in the computational experiment.

The mathematical models can be compared on the basis of the number of binary variables, real variables and constraints required. Table 5 shows (separated by "/") the minimum, the average and the maximum number of (binary and real) variables and constraints for the initial models (*SALBP-1-i* and *SALBP-2-i*) and for the improved models (*SALBP-1-n* and *SALBP-2-n*).

Insert Table 5

To evaluate the performance of the models and identify the best one, we compare the solutions obtained by the two models for each instance. For SALBP-1, for example, for the instance Tonge with a cycle time of 195, *SALBP-1–i* obtained a solution with 20 workstations whereas the new model *SALBP-1–n* obtained a better solution with 19 stations.

Table 6 and table 7 show the results of the computational experiment for SALBP-1 and SALBP-2. Table 6 shows, both for the previous and for the new models, the type of solutions obtained: whether the solutions are optimal, feasible but not optimal or not feasible. This information allows the comparison of the models and determines the one that reaches the best results. Table 7 gives information on the calculation time of the instances in which both models guarantee the optimal solution. Table 7 also shows

which model has the best behaviour when both models find a feasible but not optimal solution.

In table 6, for each model, the following information is summarized: the number of instances with a proved optimal solution $(Opt-prov)$; the number of instances with an unproved optimal solution $(Opt-\overline{prov})$ even though its optimality is known; the number of instances with a feasible but not an optimal solution $(Fea-\overline{opt})$; and the number of instances without a feasible solution $(\overline{Fea})$.

Insert Table 6

Table 7 shows the number of instances with the minimum calculation time required to obtain a proved optimal solution $(Best-time)$ when both models guarantee the optimal solution (132 instances for SALBP-1 and 74 for SALBP-2) and the total time (in seconds) taken to solve these 132 and 74 instances. This table also shows the number of instances with the best solution $(Best-sol)$ when none of the models guarantees the optimal solution in the computing time allowed (28 instances for SALBP-1, although in 22 instances both obtain the same solution and are not included; and 146 instances for SALBP-2, although in 23 instances both obtain the same solution are not included).

Insert Table 7

The results are very satisfactory for SALBP-1 and show the superiority of the proposed new model, *SALBP-1-n*. With *SALBP-1-n* a greater number of instances are solved optimally (149 vs. 136, table 6) and there are less instances for which no solution is found (70 vs. 97, table 6). However, when both models guarantee an optimal solution (132 instances), *SALBP-1-n* needs a little more time than *SALBP-1-i* to solve the instances (7,410 s vs. 5,908 s, table 7). That makes an average of 56.1 seconds and 44.8 seconds to solve each instance with *SALBP-1-n* and *SALBP-1-i*, respectively. Finally, in the instances in which the models find different solutions and none of them guarantee its optimality; the solution reached by *SALBP-1-n* is better than the solution reached for *SALBP-1-i* in more instances (4 vs. 2, table 7).

The results obtained for SALBP-2 with the new model (*SALBP-2-n*) are also better than the results obtained with *SALBP-2-i*. When *SALBP-2-n* is used, a greater number of instances are solved optimally (102 vs. 83, table 6) and no solution is found for two less instances than with the previous model (36 vs. 38, table 6). In the 74 instances for which both models guarantee the optimal solution, although *SALBP-2-i* finds the optimal solution in less time in more instances (46 vs. 28, table 7), the total processing time of *SALBP-2-n* is only 58.37% of the time of *SALBP-2-i* (7,157 s vs. 12,260 s, table 7). That makes an average of 96.7 seconds and 165.7 seconds to solve each instance with *SALBP-2-n* and *SALBP-2-i*, respectively. Finally, in the instances in which the models find different solutions and none of them guarantee its optimality; the solution reached by *SALBP-2-n* is better than the one reached by *SALBP-2-i* in more instances (64 vs. 59, table 7). Therefore, the model *SALBP-2-n* also obtains better results when both models find a solution but neither can guarantee its optimality.

Finally we calculate the validity of the conclusion "the obtained results with models found in the literature are inferior to those obtained with the new model" to the parameter number of instances with a proved optimal solution. In order to do so we carried out the two proportions test using a statistical software package: the conclusion is true with a maximum confidence level of 86.95% for *SALBP-1* and for *SALBP-2* with a maximum confidence level of 95.35%.

Additionally we analyze the influence of the characteristics of the problem instances –in in particular, order strength (which gives information on the complexity of the instance: "*The higher OS, the less permutations of all tasks are possible, and the less difficult will be the problem instance*", Amen (2006), p. 761) and number of tasks (which indicates the size of the instance)– on the quality of the obtained solutions and on the calculation time on those instances in which both models guarantee the optimal solution. The solved instances have been classified according to the order strength, OS, and the number of tasks, NT: i) *Low-OS* $(22.49 \le OS \le 25.80)$, *Middle-OS* $(40.38 \le OS \le 60.0)$ and *High-OS* $(70.95 \le OS \le 83.82)$; ii) *Low-NT* $(7 \le NT \le 45)$, *Middle-NT* $(53 \le NT \le 111)$ and *High-NT* $(148 \le NT \le 297)$. Table 8 shows the following information: the percentage of instances with a proved optimal solution $(\%\, Opt - prov)$ and the percentage of instances with the minimum calculation time required to obtain a proved optimal solution $(\%\, Best - time)$ when both models guarantee the optimal solution. The results show that, in terms of proved optimal solutions, the behaviour of the new models with respect to the initial ones is similar in *SALBP-1* and in *SALBP-2*: its effectiveness increases when the order strength increases and is remarkably high for middle number of tasks. When both models guarantee the

optimal solution, the results do not show any conclusive relation in terms of number of instances with the minimum calculation time.

Insert Table 8

In the new model for SALBP-2, *SALBP-2-n*, the number of additional binary variables depends on the range of the possible values of ct, $R = ct_{\max} - ct_{\min} + 1$. In order to study the possible influence of this parameter, the results were analyzed according to the value of R. It can be seen that from a value $R = 2{,}000$ upwards, the effectiveness of *SALBP-2-n* is no longer greater than the effectiveness of *SALBP-2-i*. However, $R = 2{,}000$ is a very large range for the possible values of ct, which is not realistic in industrial systems.

5. Conclusions

The SALB problem has been extensively examined in the literature and several equivalent mathematical models have been developed in order to solve it.

Usually, an initial pre-process is carried out to reduce the number of assignment variables task-workstation; this pre-process calculates the interval of workstations $[E_i, L_i]$ to which any task i may be assigned. Our contribution is the design of a more effective mathematical program than those that have previously been released to solve SALBP-1 and SALBP-2. In other words, when the mathematical program proposed is

used, a greater number of instances are solved optimally and no solution is found for fewer instances than with the previous model. Moreover, the new model also obtains better results when both models find a solution but none of them can guarantee its optimality. The new idea is to introduce additional constraints into the mathematical program, based on the fact that the workstation interval $[E_i, L_i]$ depends either on the upper bound on the number of workstations or on the upper bound on the cycle time (for SALBP-1 and SALBP-2, respectively). The idea of feasible workstation interval $[E_i, L_i]$ is not new; but modelling the additional knowledge introduced by this range and introducing it in the mathematical program as additional constraints is new.

The computational evaluation, with a set of well-known instances, reveals the superiority of the proposed mathematical program; therefore, its utilization is recommended to solve assembly line balancing problems by mathematical programming models. From an industrial point of view, and considering the production increase that may be obtained when reducing either the number of workstations (objective of SALBP type-1) or the cycle time (objective of SALBP type-2), we recommend to solve the real problem optimally by means of mathematical programming (e.g. Corominas et al., 2006); when the optimal solution is not guaranteed, then we recommend to use a heuristic or metaheuristic procedure adhoc.

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.

References

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

Atamtürk, A., Savelsbergh, M.W.P. 2005. Integer-programming software systems. *Annals of Operations Research*, 140, 67-124.

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

Bixby, R.E. 2002. Solving real-world linear programs: a decade and more of progress. *Operations Research*, 50, 3-15.

Bowman, E.H. 1960. Assembly-line balancing by linear programming. *Operations Research*, 8, 385-389.

Corominas, A., Pastor, R., Plans, J. 2006. Balancing assembly line with skilled and unskilled workers. *OMEGA* (in press, available online 9 May 2006) doi: 10.1016/j.omega.2006.03.003.

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

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

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

Pastor, R., Corominas, A., Lusa, A. 2004. Different ways of modelling and solving precedence and incompatibility constraints in the assembly line balancing problem. *Frontiers in Artificial Intelligence and Applications*, 113, 359-366.

Patterson, J.H., Albracht, J.J. 1975. Assembly-line balancing: Zero-one programming with Fibonacci search. *Operations Research*, 23, 166-172.

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.

Saltzman, M.J., Baybars, I. 1987. A two-process implicit enumeration algorithm for the simple assembly line balancing problem. *European Journal of Operational Research*, 32, 118-129.

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

Scholl, A., Klein, R. 1997. SALOME: A bidirectional branch and bound procedure for assembly line balancing. *INFORMS J Comp*, 9, 319-334.

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

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

Thangavelu, S.R., Shetty, C.M. 1971. Assembly line balancing by zero-one integer programming. *AIEE Transactions*, 3, 61-68.

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

Annex

This section includes the two complete mathematical models (*SALBP-1-i* and *SALBP-1-n*) for the first example given in Section 1 (figure 1 and table 1), in order to clarify the new additional constraints and allow for the comparison of the two models.

Model SALBP-1-i:

$$[MIN] Z = 4 \cdot y_4 + 5 \cdot y_5 + 6 \cdot y_6 \tag{5}$$

$$\begin{aligned}
&x_{A1} + x_{A2} + x_{A3} + x_{A4} = 1 \\
&x_{B1} + x_{B2} + x_{B3} + x_{B4} + x_{B5} = 1 \\
&x_{C1} + x_{C2} + x_{C3} + x_{C4} + x_{C5} = 1 \\
&x_{D1} + x_{D2} + x_{D3} + x_{D4} + x_{D5} = 1 \\
&x_{E2} + x_{E3} + x_{E4} + x_{E5} + x_{E6} = 1 \\
&x_{F2} + x_{F3} + x_{F4} + x_{F5} = 1 \\
&x_{G3} + x_{G4} + x_{G5} + x_{G6} = 1 \\
&x_{H3} + x_{H4} + x_{H5} + x_{H6} = 1
\end{aligned} \tag{6}$$

$$\begin{aligned}
&2 \cdot x_{A1} + 3 \cdot x_{B1} + 5 \cdot x_{C1} + 3 \cdot x_{D1} \le 9 \\
&2 \cdot x_{A2} + 3 \cdot x_{B2} + 5 \cdot x_{C2} + 3 \cdot x_{D2} + 2 \cdot x_{E2} + 5 \cdot x_{F2} \le 9 \\
&2 \cdot x_{A3} + 3 \cdot x_{B3} + 5 \cdot x_{C3} + 3 \cdot x_{D3} + 2 \cdot x_{E3} + 5 \cdot x_{F3} + 3 \cdot x_{G3} + 2 \cdot x_{H3} \le 9
\end{aligned} \tag{7}$$

$$\begin{aligned}
&2 \cdot x_{A4} + 3 \cdot x_{B4} + 5 \cdot x_{C4} + 3 \cdot x_{D4} + 2 \cdot x_{E4} + 5 \cdot x_{F4} + 3 \cdot x_{G4} + 2 \cdot x_{H4} \le 9 \cdot y_4 \\
&3 \cdot x_{B5} + 5 \cdot x_{C5} + 3 \cdot x_{D5} + 2 \cdot x_{E5} + 5 \cdot x_{F5} + 3 \cdot x_{G5} + 2 \cdot x_{H5} \le 9 \cdot y_5 \\
&2 \cdot x_{E6} + 3 \cdot x_{G6} + 2 \cdot x_{H6} \le 9 \cdot y_6
\end{aligned} \tag{8}$$

$$
\begin{aligned}
&1\cdot x_{A1}+2\cdot x_{A2}+3\cdot x_{A3}+4\cdot x_{A4}\le 1\cdot x_{B1}+2\cdot x_{B2}+3\cdot x_{B3}+4\cdot x_{B4}+5\cdot x_{B5}\\
&1\cdot x_{A1}+2\cdot x_{A2}+3\cdot x_{A3}+4\cdot x_{A4}\le 1\cdot x_{C1}+2\cdot x_{C2}+3\cdot x_{C3}+4\cdot x_{C4}+5\cdot x_{C5}\\
&1\cdot x_{A1}+2\cdot x_{A2}+3\cdot x_{A3}+4\cdot x_{A4}\le 1\cdot x_{D1}+2\cdot x_{D2}+3\cdot x_{D3}+4\cdot x_{D4}+5\cdot x_{D5}\\
&1\cdot x_{B1}+2\cdot x_{B2}+3\cdot x_{B3}+4\cdot x_{B4}+5\cdot x_{B5}\le 2\cdot x_{E2}+3\cdot x_{E3}+4\cdot x_{E4}+5\cdot x_{E5}+6\cdot x_{E6}\\
&1\cdot x_{C1}+2\cdot x_{C2}+3\cdot x_{C3}+4\cdot x_{C4}+5\cdot x_{C5}\le 2\cdot x_{E2}+3\cdot x_{E3}+4\cdot x_{E4}+5\cdot x_{E5}+6\cdot x_{E6}\\
&1\cdot x_{D1}+2\cdot x_{D2}+3\cdot x_{D3}+4\cdot x_{D4}+5\cdot x_{D5}\le 2\cdot x_{F2}+3\cdot x_{F3}+4\cdot x_{F4}+5\cdot x_{F5}\\
&2\cdot x_{E2}+3\cdot x_{E3}+4\cdot x_{E4}+5\cdot x_{E5}+6\cdot x_{E6}\le 3\cdot x_{G3}+4\cdot x_{G4}+5\cdot x_{G5}+6\cdot x_{G6}\\
&2\cdot x_{F2}+3\cdot x_{F3}+4\cdot x_{F4}+5\cdot x_{F5}\le 3\cdot x_{G3}+4\cdot x_{G4}+5\cdot x_{G5}+6\cdot x_{G6}\\
&3\cdot x_{G3}+4\cdot x_{G4}+5\cdot x_{G5}+6\cdot x_{G6}\le 3\cdot x_{H3}+4\cdot x_{H4}+5\cdot x_{H5}+6\cdot x_{H6}
\end{aligned}
\tag{9}
$$

Model SALBP-1-n:

$$[MIN]Z=4\cdot y_4+5\cdot y_5+6\cdot y_6 \tag{5}$$

$$
\begin{aligned}
&x_{A1}+x_{A2}+x_{A3}+x_{A4}=1\\
&x_{B1}+x_{B2}+x_{B3}+x_{B4}+x_{B5}=1\\
&x_{C1}+x_{C2}+x_{C3}+x_{C4}+x_{C5}=1\\
&x_{D1}+x_{D2}+x_{D3}+x_{D4}+x_{D5}=1\\
&x_{E2}+x_{E3}+x_{E4}+x_{E5}+x_{E6}=1\\
&x_{F2}+x_{F3}+x_{F4}+x_{F5}=1\\
&x_{G3}+x_{G4}+x_{G5}+x_{G6}=1\\
&x_{H3}+x_{H4}+x_{H5}+x_{H6}=1
\end{aligned}
\tag{6}
$$

$$
\begin{aligned}
&2\cdot x_{A1}+3\cdot x_{B1}+5\cdot x_{C1}+3\cdot x_{D1}\le 9\\
&2\cdot x_{A2}+3\cdot x_{B2}+5\cdot x_{C2}+3\cdot x_{D2}+2\cdot x_{E2}+5\cdot x_{F2}\le 9\\
&2\cdot x_{A3}+3\cdot x_{B3}+5\cdot x_{C3}+3\cdot x_{D3}+2\cdot x_{E3}+5\cdot x_{F3}+3\cdot x_{G3}+2\cdot x_{H3}\le 9
\end{aligned}
\tag{7}
$$

$$
\begin{aligned}
&2\cdot x_{A4}+3\cdot x_{B4}+5\cdot x_{C4}+3\cdot x_{D4}+2\cdot x_{E4}+5\cdot x_{F4}+3\cdot x_{G4}+2\cdot x_{H4}\le 9\cdot y_4\\
&3\cdot x_{B5}+5\cdot x_{C5}+3\cdot x_{D5}+2\cdot x_{E5}+5\cdot x_{F5}+3\cdot x_{G5}+2\cdot x_{H5}\le 9\cdot y_5\\
&2\cdot x_{E6}+3\cdot x_{G6}+2\cdot x_{H6}\le 9\cdot y_6
\end{aligned}
\tag{8}
$$

$$
\begin{aligned}
&1\cdot x_{A1}+2\cdot x_{A2}+3\cdot x_{A3}+4\cdot x_{A4}\le 1\cdot x_{B1}+2\cdot x_{B2}+3\cdot x_{B3}+4\cdot x_{B4}+5\cdot x_{B5}\\
&1\cdot x_{A1}+2\cdot x_{A2}+3\cdot x_{A3}+4\cdot x_{A4}\le 1\cdot x_{C1}+2\cdot x_{C2}+3\cdot x_{C3}+4\cdot x_{C4}+5\cdot x_{C5}\\
&1\cdot x_{A1}+2\cdot x_{A2}+3\cdot x_{A3}+4\cdot x_{A4}\le 1\cdot x_{D1}+2\cdot x_{D2}+3\cdot x_{D3}+4\cdot x_{D4}+5\cdot x_{D5}\\
&1\cdot x_{B1}+2\cdot x_{B2}+3\cdot x_{B3}+4\cdot x_{B4}+5\cdot x_{B5}\le 2\cdot x_{E2}+3\cdot x_{E3}+4\cdot x_{E4}+5\cdot x_{E5}+6\cdot x_{E6}\\
&1\cdot x_{C1}+2\cdot x_{C2}+3\cdot x_{C3}+4\cdot x_{C4}+5\cdot x_{C5}\le 2\cdot x_{E2}+3\cdot x_{E3}+4\cdot x_{E4}+5\cdot x_{E5}+6\cdot x_{E6}\\
&1\cdot x_{D1}+2\cdot x_{D2}+3\cdot x_{D3}+4\cdot x_{D4}+5\cdot x_{D5}\le 2\cdot x_{F2}+3\cdot x_{F3}+4\cdot x_{F4}+5\cdot x_{F5}\\
&2\cdot x_{E2}+3\cdot x_{E3}+4\cdot x_{E4}+5\cdot x_{E5}+6\cdot x_{E6}\le 3\cdot x_{G3}+4\cdot x_{G4}+5\cdot x_{G5}+6\cdot x_{G6}\\
&2\cdot x_{F2}+3\cdot x_{F3}+4\cdot x_{F4}+5\cdot x_{F5}\le 3\cdot x_{G3}+4\cdot x_{G4}+5\cdot x_{G5}+6\cdot x_{G6}\\
&3\cdot x_{G3}+4\cdot x_{G4}+5\cdot x_{G5}+6\cdot x_{G6}\le 3\cdot x_{H3}+4\cdot x_{H4}+5\cdot x_{H5}+6\cdot x_{H6}
\end{aligned}
\tag{9}
$$

$$
\begin{aligned}
&x_{A4} \le y_6 \\
&x_{A3} \le y_5 \\
&x_{A2} \le y_4 \\
&x_{B5} \le y_6 \\
&x_{B4} \le y_5 \\
&x_{B3} \le y_4 \\
&x_{C5} \le y_6 \\
&x_{C4} \le y_5 \\
&x_{C3} \le y_4 \\
&x_{D5} \le y_6 \\
&x_{D4} \le y_5 \\
&x_{D3} \le y_4 \\
&x_{E6} \le y_6 \\
&x_{E5} \le y_5 \\
&x_{E4} \le y_4 \\
&x_{F5} \le y_6 \\
&x_{F4} \le y_5 \\
&x_{F3} \le y_4 \\
&x_{G6} \le y_6 \\
&x_{G5} \le y_5 \\
&x_{G4} \le y_4 \\
&x_{H6} \le y_6 \\
&x_{H5} \le y_5 \\
&x_{H4} \le y_4
\end{aligned}
\tag{12}
$$

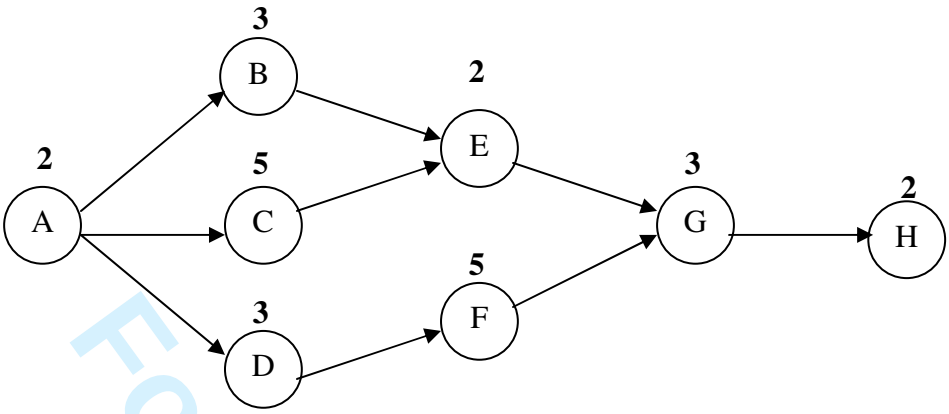

Figure 1. Precedence graph

Task	E_i	$L_i(6)$	$L_i(5)$	$L_i(4)$	$L_i(3)$
A	1	4	3	2	1
B	1	5	4	3	2
C	1	5	4	3	2
D	1	5	4	3	2
E	2	6	5	4	3
F	2	5	4	3	2
G	3	6	5	4	3
H	3	6	5	4	3

Table 1. First and last workstation to which each task can be assigned according to the value of m

Task	$E_i(5)$	$E_i(6)$	$E_i(7)$	$E_i(8)$	$E_i(9)$	$E_i(10)$
A	1	1	1	1	1	1
B	1	1	1	1	1	1
C	2	2	1	1	1	1
D	1	1	1	1	1	1
E	3	2	2	2	2	2
F	2	2	2	2	2	1
G	5	4	4	3	3	3
H	5	5	4	4	3	3

Table 2. First workstation to which each task can be assigned depending on the value of ct

Task	$L_i(5)$	$L_i(6)$	$L_i(7)$	$L_i(8)$	$L_i(9)$	$L_i(10)$
A	1	1	2	2	3	3
B	4	4	4	4	4	5
C	3	4	4	4	4	4
D	3	3	4	4	4	4
E	4	4	5	5	5	5
F	4	4	4	4	4	5
G	5	5	5	5	5	5
H	5	5	5	5	5	5

Table 3. Last workstation to which each task can be assigned depending on the value of ct

		$Opt - prov$	Fea	$\overline{Fea}$
SALBP-1 (150 instances)	*All*	87	44	19
	Half	79	34	37
	One	74	35	41
SALBP-2 (80 instances)	*All*	48	32	0
	Half	36	42	2
	One	37	42	1

Table 4. Results for different depth levels of the additional constraints

Model	Binary variables	Real variables	Constraints
SALBP-1-i	27/3,986/21,077	0	17/262/820
SALBP-1-n	27/3,986/21,077	0	24/2,712/15,373
SALBP-2-i	126/1,891/11,057	1	69/264/772
SALBP-2-n	215/4,023/50,437	1	126/879/5,244

Table 5. Number of binary variables, real variables and constraints of the models

Model	$Opt - prov$	$Opt - \overline{prov}$	$Fea - \overline{opt}$	$\overline{Fea}$
SALBP-1-i	136	17	19	97
SALBP-1-n	149	22	28	70
SALBP-2-i	83	18	163	38
SALBP-2-n	102	11	153	36

Table 6. Results of the computational experiment

Model	*Best – time*	*Total – time*	*Best – sol*
SALBP-1-i	96	5,908	2
SALBP-1-n	36	7,410	4
SALBP-2-i	46	12,260	59
SALBP-2-n	28	7,157	64

Table 7. Results when both models guarantee the optimal solution or when neither model guarantees the optimal solution.

	% Opt – prov		*% Best – time*	
	SALBP-1-i / SALBP-1-n	***SALBP-2-i / SALBP-2-n***	***SALBP-1-i / SALBP-1-n***	***SALBP-2-i / SALBP-2-n***
Low-OS	29.23 / 30.77	22.73 / 21.21	93.75 / 6.25	90.00 / 10.00
Medium-OS	55.92 / 60.53	28.02 / 33.52	66.67 / 33.33	44.68 / 55.32
High-OS	61.54 / 71.15	31.48 / 50.00	78.13 / 21.88	94.12 / 5.88
Low-NT	98.72 / 100	97.50 / 97.50	72.73 / 27.27	42.11 / 57.89
Medium-NT	43.85 / 52.31	19.90 / 29.08	74.07 / 25.93	81.82 / 18.18
High-NT	3.28 / 4.92	7.58 / 9.09	0 / 100	100 / 0

Table 8. Results depending on characteristics of the problem instances