Evaluating optimization models to solve SALBP[*]

Rafael Pastor, Laia Ferrer, Alberto García

Technical University of Catalonia, IOC Research Institute, Av. Diagonal 647, Edif. ETSEIB, p.11, 08028 Barcelona, Spain
{rafael.pastor, laia.ferrer, alberto.garcia}@upc.edu

Abstract. This work evaluates the performance of constraint programming (CP) and integer programming (IP) formulations to solve the Simple Assembly Line Balancing Problem (SALBP) exactly. Traditionally, its exact solution by CP or IP and standard software has been considered to be inefficient to real-world instances. However, nowadays this is becoming more realistic thanks to recent improvements both in hardware and software power. In this context, analyzing the best way to model and to solve SALBP is acquiring relevance. The aim of this paper is to identify the best way to model SALBP-1 (minimizing the number of stations, for a given cycle time) and SALBP-2 (minimizing the cycle time, for a given number of stations). In order to do so, a wide computational experiment is carried out to analyze the performance of one CP and three IP formulations to solve each problem. The results reveal which of the alternative models and solution techniques is the most efficient to solve SALBP-1 and SALBP-2, respectively.

Keywords: assembly line balancing

1. Introduction

An assembly line consists in set of workstations, through which the product to be processed flows. In each workstation, a number of tasks are done, which are characterized by their processing times and by a set of technological precedence relations between them. The Simple Assembly Line Balancing Problem (SALBP) consists of assigning a set of tasks to workstations in such a way that precedence constraints are fulfilled, the total processing time assigned to a station do not exceed a cycle time *tc* and a given efficiency measure is optimized. When the objective is to minimize the number of workstations m for a given cycle time *tc*, the problem is usually referred to as SALBP-1; if the objective is to minimize *tc* given *m*, the problem is called SALBP-2; and SALBP-F consists of finding a feasible solution, given *tc* and *m* (see e.g. [1]).

The design of assembly lines has been extensively examined in the literature, especially the SALB Problem. Several reviews have been published –the last is [2]–, and a huge amount of specific research exists, both for heuristic and exact procedures.

[*] This work is supported by the Spanish MCyT project DPI2004-03472, co-financed by FEDER.

Some of the exact procedures are based on mathematical programming and different integer linear programming and mixed-integer linear programming models (IP models) have been developed. Scholl highlights three basic formulations to solve SALBP-F [1] (finding a feasible solution, given a cycle time and a number of stations) based on different sets of assignment variables. Other exact procedures have also used constraint programming (CP) [3].

Recent improvements both in software and hardware power have reduced remarkably the computing time needed to solve combinatorial problems by constraint programming or mathematical programming. Nowadays, these techniques are gaining acceptance as a powerful computational tools [4]. In this context, analyzing the best way to model and to solve combinatorial problems is acquiring relevance. Constraint programming and mathematical programming can solve similar combinatorial problems, but their effectiveness depends on the class of problems studied [5]. A number of papers have compared the performance of CP and IP approaches for solving different problems – for example, [6].

To our knowledge, the efficiency of a CP model and the IP enhanced by Scholl [1] models has not been compared. In this work, a wide computational experiment is carried out to analyze the performance of one CP model and three IP models to solve SALBP-1 and SALBP-2. The results reveal which of the alternative models is the most efficient to solve these SALBP problems.

The remaining paper is organized as follows. In Section 2 the different formulations for SALBP-1 and SALBP-2 are presented. In Section 3 the results of the computational experiment are analyzed and the performances of the models are compared. Finally, in Section 4 the main conclusions of the study are summarized.

2. Models for SALBP

In this section four alternative formulations for SALBP-1 and SALBP-2 are developed.

First, we present a CP model – *constraint programming* model-.

Then, the three SALBP-F models presented in Scholl (1999) are adapted to SALBP-1 and SALBP-2. In sum, the main difference between these three linear models is the definition of the assignment variables used:

- *impulse* variables based model: binary variables x_{ij} take value 1 if and only if task i is assigned to workstation j (see also [7] and [8])
- *step* variables based model: binary variables x_{ij} take value 1 if and only if task i is assigned to workstation j or earlier (see also [7] and [8]).
- *mixed-integer* variables model: integer variables z_i denotes the number of the station to which task i is assigned.

In the following sections the formulations for the four models are detailed. In each section, first the model for SALBP-1 is presented. Then the model for SALBP-2 is explained highlighting the new data, the new variables and the changes to be done in the formulation with respect to the model for SALBP-1.

2.1. The *constraint programming* models

Next, the *constraint programming* model for SALBP-1 (*SALBP-1-c*) is presented and the changes for SALBP-2 (*SALBP-2-c*) are explained.

SALBP-1-c

Data:

Note subindexes *i* and *k* are related with tasks and subindex *j* with workstations.

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.
TC	Cycle time.
P	Set of pairs of tasks (i, k) such that there is immediate precedence between them.
S	Set of tasks without any successive task.
E_i	Earliest possible workstation for task i.
L_i	Latest possible workstation for task i, given a value of $m_{\max}$. Before a task is assigned the total processing time of the tasks that precede it must be assigned, and afterwards the total time of the tasks that follow it; as a result, the range of workstations [E_i, L_i] to which each task can be assigned is obtained and the number of binary variables is reduced (see, for example, [1]).

Variables:

ws	Number of workstations used.
z_i	Number of the workstation to which task *i* is assigned $(\forall i; z_i \in [E_i, L_i])$

Model SALBP-1-c:

$$[MIN]\, Z = ws \tag{1}$$

$$ws = \max\left(z_i \,\middle|\, i \in S\right) \tag{2}$$

$$\sum_{\forall i | j \in [E_i, L_i] \wedge (z_i = j)} t_i \le TC \qquad \forall j \tag{3}$$

$$z_i \le z_k \qquad \forall (i,k) \in P \tag{4}$$

The objective function (1) consists in minimizing the number of workstations, which is calculated in (2); constraints (3) ensures that the total task processing time assigned to workstation j does not exceed the cycle time; constraint set (4) imposes the technological precedence conditions.

SALBP-2-c

Data:
The model uses the same data of *SALBP-1-c*; furthermore we redefine:

m	Number of workstations $\left(j = 1,\ldots,m\right)$.
C	Upper bound on the cycle time.
E_i	Earliest possible workstation for task i, given a value of C.
L_i	Latest possible workstation for task i, given a value of C.

Variables:

tc	Cycle time.

Model SALBP-2-c:

$$\left[MIN\right] Z = tc \tag{5}$$

$$\sum_{\forall i \mid j \in [E_i, L_i] \wedge (z_i = j)} t_i \le tc \qquad \forall j \tag{6}$$

Constraint (4) has to be added.

The objective function (5) minimizes the cycle time and constraint set (6) ensures that the total task processing time assigned to workstation j does not exceed the cycle time.

2.2. The *impulse* variables based models

Next, the *impulse* variables based model for SALBP-1 (*SALBP-1-i*) is presented and the changes for SALBP-2 (*SALBP-2-i*) are explained.

SALBP-1-i

Data:
The data used in this model is the same as the previous one.

Variables:

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

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

Model SALBP-1-i:

$$[MIN]\, Z = \sum_{j=m_{\min}+1}^{m_{\max}} j \cdot y_j \tag{7}$$

$$\sum_{j=E_i}^{L_i} x_{ij} = 1 \quad \forall i \tag{8}$$

$$\sum_{\forall i | j \in [E_i, L_i]} t_i \cdot x_{ij} \le TC \quad j = 1, ..., m_{\min} \tag{9}$$

$$\sum_{\forall i | j \in [E_i, L_i]} t_i \cdot x_{ij} \le TC \cdot y_j \quad j = m_{\min} + 1, ..., m_{\max} \tag{10}$$

$$\sum_{j=E_i}^{L_i} j \cdot x_{ij} \le \sum_{j=E_k}^{L_k} j \cdot x_{kj} \quad \forall (i,k) \in P \tag{11}$$

The objective function (7) consists in minimizing the number of workstations; constraint set (8) implies that each task i is assigned to one and only one workstation; constraints (9) and (10) are equivalent to (3) and they ensure the cycle time is not exceeded; constraint set (11) replaces (4) and imposes the precedence conditions.

SALBP-2-i

Data:
The data used in this model is the same as the previous ones.

Variables:
The variables have been defined in the previous models.

Model SALBP-2-i:

$$[MIN]\, Z = tc \tag{5}$$

$$\sum_{\forall i | j \in [E_i, L_i]} t_i \cdot x_{ij} \le tc \quad \forall j \tag{12}$$

Constraints (8) and (11) have to be added.

The objective function (5) minimizes the cycle time; constraint set (12) is equivalent to (6) and ensures that the total task processing time assigned to workstation j does not exceed the cycle time.

2.3. The *step* variables based models

Next, the *step* variables based model for SALBP-1 (*SALBP-1-s*) is presented and the changes for SALBP-2 (*SALBP-2-s*) are explained.

SALBP-1-s

Data:
The data used in this model is the same as the previous ones.

Variables:

The variables used in the *step* variables based models are the same of the *impulse* variables based models but x_{ij} are redefined:

$x_{ij} \in \{0,1\}$	1, if and only if task *i* is assigned to workstation *j* or earlier, 0 otherwise $(\forall i;\ j = E_i, \ldots, L_i - 1)$. Note that $x_{i,L_i} = 1$ and it is not defined.
$y_j \in \{0,1\}$	1, if and only if any task is assigned to workstation *j*, 0 otherwise $(j = m_{\min} + 1, \ldots, m_{\max})$.

Model SALBP-1-s:

$$[MIN]\,Z = \sum_{j=m_{\min}+1}^{m_{\max}} j \cdot y_j \tag{7}$$

$$x_{ij} \le x_{i,j+1} \qquad \forall i;\ j = E_i, \ldots, L_i - 2 \tag{13}$$

$$\sum_{\forall i \mid j=E_i} t_i \cdot x_{ij} + \sum_{\forall i \mid j \in [E_i+1, L_i-1]} t_i \cdot \left(x_{ij} - x_{i,j-1}\right) + \sum_{\forall i \mid j=L_i} t_i \cdot \left(1 - x_{i,j-1}\right) \le TC; \quad j = 1, \ldots, m_{\min} \tag{14}$$

$$\sum_{\forall i \mid j=E_i} t_i \cdot x_{ij} + \sum_{\forall i \mid j \in [E_i+1, L_i-1]} t_i \cdot \left(x_{ij} - x_{i,j-1}\right) + \sum_{\forall i \mid j=L_i} t_i \cdot \left(1 - x_{i,j-1}\right) \le TC \cdot y_j \quad j = m_{\min} + 1, \ldots, m_{\max} \tag{15}$$

$$x_{kj} \le x_{ij} \qquad \forall (i,k) \in P; \forall j \in [E_i, L_i - 1] \cap [E_k, L_k - 1] \tag{16}$$

Constraint sets (13), (14) and (15) are equivalent to constraint sets (8), (9) and (10), respectively. Now, the technological precedence conditions –constraint set (4) or (11)– is modeled by (16).

SALBP-2-s

Data:
The data used in this model is the same as the previous ones.

Variables:
The variables have been defined in the previous models.

Model SALBP-2-s:

$$[MIN]\, Z = tc \qquad (5)$$

$$\sum_{\forall i | j = E_i} t_i \cdot x_{ij} + \sum_{\forall i | j \in [E_i + 1, L_i - 1]} t_i \cdot \left(x_{ij} - x_{i,j-1} \right) + \sum_{\forall i | j = L_i} t_i \cdot \left(1 - x_{i,j-1} \right) \le tc \quad \forall j \qquad (17)$$

Constraints (13) and (16) have to be added. Constraint set (17) is equivalent to (6) and (12).

2.4. The *mixed-integer* variables based models

Next, the *mixed-integer* variables based model for SALBP-1 (*SALBP-1-m*) is presented and the changes for SALBP-2 (*SALBP-2-m*) are explained.

SALBP-1-m

Data:
The model uses the same data of the previous ones; furthermore we define:

$P*$	Set of pairs of tasks (i,k) such that there is an immediate or transitive precedence between them.
T	Upper-bound of the total time of the workstations.

Variables:
This formulation introduces continuous non-negative variables b_i for the clock time at which task i is started and binary variables w_{ik}:

$b_i \in \{0,1\}$	Clock time at which task i is started (measured in the time elapsed since entering the first workstation).
$w_{ik} \in \{0,1\}$	1, if and only if task i is performed before task k, value 0 otherwise $\left(i < k;\, (i,k) \notin P*; [E_i, L_i] \cap [E_k, L_k] \neq \varnothing \right)$.
ws	Number of workstations used.
z_i	Number of the workstation to which task i is assigned $\left(\forall i;\, z_i \in [E_i, L_i] \right)$.

Model SALBP-1-m:

$$\left[MIN\right] Z = ws \qquad (1)$$

$$ws \geq z_i \qquad \forall i \qquad (18)$$

$$b_i \geq TC\,(z_i - 1) \qquad \forall i \qquad (19)$$

$$b_i + t_i \leq TC\, z_i \qquad \forall i \qquad (20)$$

$$(1 - w_{ik}) \cdot T + b_k \geq b_i + t_i \quad i < k, (i,k) \notin P^*, \left[E_i, L_i\right] \cap \left[E_k, L_k\right] \neq \varnothing \qquad (21)$$

$$w_{ik} \cdot T + b_i \geq b_k + t_k \quad i < k, (i,k) \notin P^*, \left[E_i, L_i\right] \cap \left[E_k, L_k\right] \neq \varnothing \qquad (22)$$

$$b_i + t_i \leq b_k \qquad (i,k) \in P, L_i \geq E_k \qquad (23)$$

$$E_i \leq z_i \qquad \forall i \qquad (24)$$

$$z_i \leq L_i \qquad \forall i \qquad (25)$$

The objective function (1) consists in minimizing the number of workstations calculated by constraint set (18); constraint sets (19) and (20) ensure that each task *i* is fully performed within one workstation; the disjuntive constraints (21) and (22) guarantee that for each pair of tasks, which are not related by precedence and may interfere which each other, either task *i* is completely processed before task *k* , or vice versa; constraint set (23) ensure the fullfilment of the precedence constraints; the assignment task is restricted to the possible workstation interval by (24) and (25).

SALBP-2-m

The adaptation of *mixed-integer* variables based model for SALBP-F to SALBP-2 produces a non-linear model since the variable cycle time *tc* replaces data *TC* in constraints (19) and (20). This non-linear formulation of *SALBP-2-m* is linearised as follows.

Variables:

p_i	not negative real variable that indicates the total time of the workstations until the workstation in which task *i* is assigned (this one also included)
$r_{ij} \in \{0,1\}$	1, if and only if task i is assigned to workstation j, value 0 otherwise $(\forall i; j = E_i, ..., L_i)$.

Model SALBP-2-m:

$$\left[MIN\right] Z = tc \qquad (5)$$

$$b_i + tc \geq p_i \qquad \forall i \qquad (26)$$

$$b_i + t_i \le p_i \qquad \forall i \tag{27}$$

$$z_i = \sum_{j=E_i}^{L_i} j \cdot r_{ij} \qquad \forall i \tag{28}$$

$$\sum_{j=E_i}^{L_i} r_{ij} = 1 \qquad \forall i \tag{29}$$

$$p_i - j \cdot tc \le (1 - r_{ij}) \cdot T \qquad \forall i,\ j = E_i, \ldots, L_i \tag{30}$$

$$j \cdot tc - p_i \le (1 - r_{ij}) \cdot T \qquad \forall i,\ j = E_i, \ldots, L_i \tag{31}$$

The real variables p_i replaces the product $tc \cdot z_i$ in (19) and (20) obtaining (26) and (27); the variables z_i are expressed as shown in (28); constraint sets (29), (30) and (31) are added.

Constraint sets (21)-(23) need to be added too.

3. Computational experiment

A computational experiment is carried out to compare the efficiency of the models.

The basic data used for the experiment are all the well-known instances available in the assembly line balancing research homepage (www.assembly-line-balancing.de). A total of 269 instances for SALBP-1 and 302 for SALBP-2 were used.

The CP models were solved using ILOG Solver 6.0 and the MILP models were solved by CPLEX 9.0, with a PC Pentium IV at 3.4 GHz and with 512 Mb of RAM. A maximum computing time of 2,000 seconds was set.

The analysis of the results of the computational experiment starts with a initial comparison of the performance of the models in terms of the type of the solutions obtained: whether the model finds a solution or not and whether this solution is optimal or feasible. This initial analysis identifies the best models to be analyzed in detail. Next, the computing time used by these models is studied, focusing on the instances in which the optimal solution is found. Next, the solutions obtained in the instances in which the optimality is not guaranteed are presented. Finally, considering all these aspects, a detailed analysis of the performance of the different models is carried out.

3.1. Results of the type the solutions

Table 1 and table 2 show the results of the computational experiment for SALBP-1 and SALBP-2, respectively, focusing on the type of the solutions obtained. For each model, the following information is summarized:

- the number of instances with a proved optimal solution $(Opt - prov)$: an optimal solution is found and the solving software guarantees it.
- the number of instances in which an unproved optimal solution $(Opt - \overline{prov})$: an optimal solution is found but the solving software does not guarantee its optimality. The optimal solution of the instances is available in the assembly line balancing research homepage.
- the number of instances with a feasible but not optimal solution $(Fea - \overline{opt})$.
- the number of instances in which the solving software does not find any solution $(\overline{Sol})$.

Table 1. Results of the computational experiment for SALBP-1

	SALBP-1				*SALBP-2*			
	c	*i*	*s*	*m*	*c*	*i*	*s*	*m*
$Opt - prov$	98	136	123	51	55	84	122	0
$Opt - \overline{prov}$	12	17	5	24	0	16	12	4
$Fea - \overline{opt}$	2	19	14	14	199	174	168	64
$\overline{Sol}$	157	97	127	180	48	28	0	234

The results show that the performance of the *mixed-integer* based model is worse than the performance of the *constraint programming* model, the *impulse* variables and the *step* variables based models. For SALBP-1, *SALBP-1-m* obtains 51 proved optimal solutions; nearly half of the optimal solutions reached by *SALBP-1-c*, *SALBP-1-i* or *SALBP-1*-s (98, 136 and 123, respectively). For SALBP-2, *SALBP-2-m* does not obtain any proved optimal solution. Moreover, the *mixed-integer* variables based models do not reach a feasible solution in more instances than the other models, both for SALBP-1 and for SALBP-2.

Due to clear inferiority of the *mixed-integer* variables based model, we focus the detailed comparison of the results only in the *constraint programming*, the *impulse* variables and the *step* variables based models. We analyze the percentage of proved optimal solutions depending on the the number of tasks (*NT*) and the order strength (*OS* = number of all precedence relations / (*NT* * (*NT* - 1))) of the instances. We classify: i) *Low-OS* $(22.49 \leq OS \leq 25.80)$, *Middle-OS* $(40.38 \leq OS \leq 60.0)$ and *High-OS* $(70.95 \leq OS \leq 83.82)$; ii) *Low-NT* $(7 \leq NT \leq 45)$, *Middle-NT* $(53 \leq NT \leq 111)$ and *High-NT* $(148 \leq NT \leq 297)$. Table 2 shows the percentage of proved optimal solutions obtained with the *constraint programming (c)*, *impulse* variables *(i)* and *step* variables *(s)* based models.

Table 2. Percentage of proved optimal solutions depending on *OS* and *NT*

	SALBP-1			*SALBP-2*		
	c	*i*	*s*	*c*	*i*	*s*
Low-OS	10.77	29.23	12.31	6.061	21.21	25.76
Middle-OS	40.79	55.92	53.95	18.68	28.02	36.81
High-OS	55.77	61.54	63.46	31.48	35.19	70.37
Low-NT	100.00	98.72	98.72	77.50	97.50	100.00
Middle-NT	14.62	43.85	33.85	10.71	19.90	36.73
High-NT	1.64	3.28	3.28	4.55	9.09	15.15

3.2. Results of the computing time

We compare the computing time used by the *constraint programming*, the *impulse* variables and the *step* variables based models when all of them obtain a proved optimal solution (95 instances in SALBP-1 and 48 instances in SALBP-2). Table 3 shows, for each model: the number of instances with the minimum calculation time (in seconds) to obtain a proved optimal solution (*Best time*); the total of time used by these instances (*Total time*); and the number of instances in which the time used by the model is less than 75% of the time used by each of the other two models (*time(a/b)<0.75*).

Table 3. Results when the 3 models find an optimal solution.

	SALBP-1			*SALBP-2*		
	c	*i*	*s*	*c*	*i*	*s*
Best-time	26	47	22	27	15	7
Total time	2084.4	3302.4	2824.7	2491.3	6608.2	508.9
time(a/b)<0.75	9	1	2	8	0	5

3.3. Results of the solutions with no optimality guaranteed

Next, we summarize the results obtained when the constraint programming, the impulse variables and the step variables based models find a feasible solution but none of them guarantees optimality. This situation occurs in 1 instance for SALBP-1, and in 119 instances for SALBP-2: in 10 of them *SALBP-2-c* obtains the best solution, *SALBP-2-i* in 9 instances and *SALBP-2-s* in 82. When *SALBP-2-s* obtains a better solution, the average solution is 95.5% and 71.1% of the average obtained by *SALBP-2-i* and *SALBP-2-c*, respectively.

3.4. Analysis of the performance of models

In this section, a detailed analysis of the performance of the models is carried out. First, we study the results for *SALBP-1* and then a similar study is presented for *SALBP-2*. Each study starts with a brief final conclusion to facilitate the comprehension of the analysis of the results. These conclusions are justified through a detailed analysis that compares the type of solutions obtained, the computing time used and the results of the solutions in which their optimality is not guaranteed. Due to clear inferiority of the performance of the *mixed-integer* variables based model (Section 3.1), these analyses focus on the *constraint programming*, the *impulse* variables and the *step* variables based models.

For SALBP-1:
In sum, in terms of number of optimal and feasible solutions the results of *SALBP-1-i* are better than results of *SALBP-1-c* and *SALBP-1-s*. However, concerning the time used, *SALBP-1-c* is the quickest model.

In terms of the type of solutions obtained (Table 1), the number of proved and unproved optimal solutions obtained by *SALBP-1-i* is higher than those obtained by *SALBP-1-c* and those obtained by *SALBP-1-s* (136 and 17, 98 and 12, 123 and 5, respectively). Moreover, *SALBP-1-i* does not obtain a feasible solution in less instances than *SALBP-1-c* and *SALBP-1-s* (97, 157 and 127, respectively) The results of *SALBP-1-c* are worse than the results of the other models, in particular, for instances with middle and high levels of *NT*; the performance of *SALBP-1-i* is especially the best for instances with low *OS* (Table 2).

In terms of the computing time (Table 3), when the three models guarantee the optimal solution, *SALBP-1-i* need less time in more instances than *SALBP-1-c* and *SALBP-1-s* (47, 26 and 22, respectively). Nevertheless, for solving all 95 instances the time needed by *SALBP-1-c* is considerably less than the time required by *SALBP-1-i* and by *SALBP-1-s* (2084.4 s, 3302.64 s and 2824.7 s, respectively). Moreover, among the 26 instances where *SALBP-2-c* is quicker, there the 9 instances in which the time used is less than 75% of the time needed by each of the other two models, whereas this difference only occurs in 1 instance for *SALBP-1-i* and 2 for *SALBP-1-c*.

For SALBP-2:
In sum, the results of *SALBP-2-s* are much better than the results of *SALBP-2-c* and *SALBP-2-i*, in terms of optimal and feasible solutions obtained and total computing time. *SALBP-2-c* is only superior to *SALBP-2-s* in the number of instances that use the minimum time.

In terms of the type of solutions obtained (Table 1), the *SALBP-2-s* model obtains more proved solutions than *SALBP-2-c* and *SALBP-2-i*: (122, 55 and 84, respectively). The total number of optimal solutions (proved and unproved) is also superior for *SALBP-2-s* than for *SALBP-2-c* and *SALBP-2-i* (134, 55 and 100, respectively). In addition, *SALBP-2-s* always obtains a feasible solution whereas *SALBP-2-c* does not obtain a feasible solution in 48 instances and *SALBP-2-i* in 28. The influence of NT is similar in the 3 models, *SALBP-1-c* is remarkably the best model for instances of low *OS* (Table 2).

In terms of the computing time (Table 3), when the three models guarantee an optimal solution, *SALBP-2-c* uses less time in more instances than *SALBP-2-i* and *SALBP-2-s* (27, 15 and 7, respectively). Moreover, the time used by *SALBP-2-c* is in 8 instances less than 75% of the time used by the other models (*SALBP-2-i* does not have this difference in any instance and *SALBP-2-s* only in 5). However, for solving all the instances in which the three models guarantee an optimal solution, the total time used for *SALBP-2-s* is much less than the time used by *SALBP-2-c* and *SALBP-2-i* (508.9 s, 2492.4 s and 6608.2 s, respectively).

Finally, when none of the models guarantees the optimal solution, *SALBP-2-s* obtains a better solution in considerably more instances than the others.

4. Conclusions

The SALB Problem has been extensively examined in the literature and different and equivalent CP models and IP models have been developed in order to solve it. However, their efficiency has not been compared and the best one is not known. The best way to model and to solve the hard combinatorial problems has a high relevance. The use of constraint programming or mathematical programming techniques to solve these problems is becoming more realistic thanks to recent improvements both in software and hardware power.

This paper focus on comparing one CP formulation *–constraint programming* model- and three IP formulations that were highlighted by Scholl [1] -the *impulse* variables, the *step* variables and the *mixed-integer* variables based model-. A wide computational experiment is carried out to compare the efficiency of these models, both for SALBP-1 and SALBP-2.

The analysis of the results shows the bad performance of the *mixed-integer* variables models. For SALBP-1, the *impulse* variables based model obtains the best solutions although *constraint programming model* is the quickest. The *step* variables based model obtains the best results for SALBP-2.

References

1. Scholl, A.: Balancing and sequencing of assembly lines. Physica, Heidelberg, 2nd edition (1999).
2. Scholl, A., Becker, C.: State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research, Vol. 168, 666-693 (2006).
3. Bockmayr, A. Pisaruk, N.: Solving an assembly line balancing problem combining IP and CP. Proceedings of the 6th Annual Workshop of ERCIM Working Droup on Constraints, Prague, Czech Republic (2001).
4. Atamtürk, A., Savelsbergh, M.W.P.: Integer-programming software systems. Annals of Operations Research, Vol. 140, 67-124 (2005)
5. Jain, V., Grossman, I.E.: Algorithms for hybrid MILP/CP models for a class of optimization problems. Journal on computing, Vol. 13 (4), 258-276 (2001).
6. Darby-Dowman, K., Little, J.: Properties of some combinatorial optimization problems and their effect on the performance of integer programming and constraint logic programming. Journal on computing, Vol. 10, 276-286 (1998).
7. Andreatta, G., Brunetta, L.: Multiairport ground holding problem: a computational evaluation of exact algorithms. Operations Research, 46, 57-64 (1998).
8. Alonso-Ayuso, A., Escudero, L.F., Garín, A., Ortuño, M.T., Pérez, G.: An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. Journal of Global Optimization, 26, 97-124 (2003).