SORT 40 (2) July-December 2016, 227-242

Improving the resolution of the simple assembly line balancing problem type E

Albert Corominas, Alberto García-Villoria* and Rafael Pastor**

Abstract

The simple assembly line balancing problem type E (abbreviated as SALBP-E) occurs when the number of workstations and the cycle time are variables and the objective is to maximise the line efficiency. In contrast with other types of SALBPs, SALBP-E has received little attention in the literature. In order to solve optimally SALBP-E, we propose a mixed integer liner programming model and an iterative procedure. Since SALBP-E is NP-hard, we also propose heuristics derived from the aforementioned procedures for solving larger instances. An extensive experimentation is carried out and its results show the improvement of the SALBP-E resolution.

MSC: 90C27 (Combinatorial Optimisation).

Keywords: Assembly line balancing, SALBP, manufacturing optimisation.

1. Introduction

Assembly line balancing problems (ALBPs) consist in assigning optimally (according to a given objective function) the tasks of an assembly or production process to the ordered workstations of an assembly line (or several assembly lines) such that some specific conditions are satisfied. These NP-hard problems (Gutjahr and Nemhauser, 1964) have an important relevance in many production systems such as in automotive and electronic industries (Battaïa and Dolgui, 2013). Thus, ALBPs have been extensively studied in the literature and several surveys have been published. Some recent surveys are Erel and Sarin (1998), Rekiek et al. (2002), Becker and Scholl (2006), Scholl and Becker (2006a,b), Boysen et al. (2008), Battaïa and Dolgui (2013).

* *Corresponding author:* Alberto García-Villoria, Institute of Industrial and Control Engineering (IOC), Av. Diagonal 647 (Edif. ETSEIB), 11th floor, 08028 Barcelona, Spain; e-mail: alberto.garcia-villoria@upc.edu

** Institute of Industrial and Control Engineering (IOC), Universitat Politècnica de Catalunya (UPC). {albert.corominas/ alberto.garcia-villoria/rafael.pastor}@upc.ed

Received: January 2015
Accepted: July 2016

The most basic family of ALBPs is the simple assembly line balancing problem (SALBP). SALBP is defined with the following assumptions (Baybars, 1986): 1) a task cannot be split among workstations; 2) there are precedence relations between tasks; 3) all tasks must be processed; 4) the task process times are independent of the workstation, they are known with certainty, they are not sequence dependent and they are additive at any workstation; 5) all workstations have the same associated costs; 6) any task can be processed at any workstation; 7) the line is serial and without feeder or parallel subassembly lines; 8) the line is designed for a unique model of a single product.

According to Baybars' nomenclature, when the objective is to minimise the number m of workstations for a given upper bound on the cycle time ct, the SALBP is called SALBP-1; if the objective is to minimise ct for a given m, the problem is called SALBP-2. On the other hand, SALBP-E is a more general version of SALBP which consists in finding the combination of m and ct such that the line efficiency is maximised. The efficiency is measured as the sum of task process times, t_{sum}, divided by the product $m \cdot ct$. In practice SALBP-E has a lower bound on m (due to a desired degree of the division of the labour) and/or an upper bound on ct (due to a minimum desired production rate); otherwise, SALBP-E would be trivial since a line with $m = 1$ and $ct = t_{sum}$ has an efficiency equal to 1. When the aim is to find a feasible line balance for a given combination of m and ct, the problem is called SALBP-F.

SALBP-1 is the most studied problem in the ALBP literature and a lot of exact and heuristic procedures have been designed for its resolution (see Scholl and Becker, 2006b). SALBP-2 has been also studied although there exist fewer procedures and most of them are based on repeatedly solving SALBP-1 with different values (see Scholl and Becker, 2006b, Uğurdağ et al., 1997). SALBP-F can be solved with modified SALBP-1 or SALBP-2 procedures (Scholl and Becker, 2006b).

In the last years researchers have intensified their efforts studying ALBPs with additional characteristics. For instance, among others: general assignment constraints (e.g. Scholl et al., 2010), task times depending on the sequence (e.g. Capacho et al., 2009), setup times between tasks (e.g. Martino and Pastor, 2010), uncertainty on task times (e.g. Saif et al., 2014), task times dependent on the workers (e.g. Moreira et al., 2015), space constraints (e.g. Chica et al., 2016), constrained resources (e.g. Corominas et al., 2011), lengths of the workpieces larger than the accessibility windows of the workstations (e.g. Calleja et al., 2014), ergonomics considerations (e.g. Bautista et al., 2016), mixed-model lines (e.g. Battaïa et al., 2015), robotic lines (Levitin et al., 2006, Gao et al., 2009, Yoosefelahi et al., 2012), U-shaped lines (e.g. Ogan and Azizoglu, 2015), machining transfer lines (e.g. Battaïa and Dolgui, 2012) and parallel two-sided assembly lines (e.g. Tapkan et al., 2016).

In contrast, to the best of our knowledge, very few procedures have been discussed for SALBP-E. Plans and Corominas (1999) formulated a MILP model which solves optimally SALBP-E. The model is tested only on seven instances and, thus, the computational experiment is insufficient. Scholl and Becker (2006b) outlined the following exact approach. All combinations of m and ct values are (implicitly) considered and

SALBP-F is examined (that is, whether there is a feasible solution with m workstations and a cycle time equal to ct). The feasible combination with best efficiency would be the optimal SALBP-E solution. The drawback is that a lot of combinations may be considered and SALBP-F is itself NP-hard. Wei and Chao (2011) designed an exact procedure based on solving optimally as many SALBP-2 as admissible values for m; each SALBP-2 is solved by means of mixed integer linear programming (MILP). That work contains some errors which are corrected in García-Villoria and Pastor (2013). Additionally, the computational experiment in Wei and Chao (2011) is limited to small instances that can be solved optimally in a short computing time. An ALBP type-E considering stochastic task times has been dealt in Gurevsky et al. (2012) and Zacharia and Nearchou (2013), in which two heuristic procedures and a genetic algorithm are developed, respectively. A genetic algorithm is also used in Al-Hawari et al. (2015) to solve a multi-objective ALBP, which includes the maximization of the line efficiency. Esmaeilbeigi et al. (2015) proposed mathematical programming for SALBP-E and different variants of a model and redundant constraints are compared.

The aim of this study is to improve the resolution of SALBP-E. We propose a MILP model together with an enhanced procedure based on the iterative one designed by Wei and Chao (2011). A computational experiment shows that our proposal outperforms the previously published methods.

The remaining paper is organised as follows. First, Section 2 presents the terminology, bounds on the objective function and the cycle time, and the MILP model. The enhanced iterative procedure and its derived heuristics are presented in Section 3. In Section 4 the proposed procedures are tested on a well-known benchmark set of instances; the procedures are compared and the results show that the SALBP-E resolution is improved. We conclude with the final conclusions in Section 5.

2. Terminology, lower and upper bounds and MILP

The data that define a SALBP-E instance are the following:

n	Number of tasks
t_i	Process time for task i $(i = 1,\dots,n)$
IP	Set of immediate precedence relations, such that $(h,i) \in IP$ means that task h must be performed before task i
m_{min}, m_{max}	The minimum and maximum number of workstations allowed, respectively

The SALBP-E objective is to maximise the efficiency E of the line (recall that $E = t_{sum}/(m \cdot ct)$, where $t_{sum} = \sum_{i=1\dots n} t_i$). Note that this objective is equivalent to minimise the line capacity $Z = m \cdot ct$. We propose the following bounds on the cycle time and the line capacity, where $\lceil x \rceil$ $(\lfloor x \rfloor)$ is the operator that returns the smallest (greatest) integer that is equal to or greater (smaller) than x:

LB^{ct}	Lower bound of cycle time: $LB^{ct} = LB^{ct}_{m_{max}}$, where $LB^{ct}_m = \max(\max_{i=1\ldots n} t_i, \lceil t_{sum}/m \rceil)$
UB^{ct}	Upper bound of cycle time: $UB^{ct} = UB^{ct}_{m_{min}}$, where $UB^{ct}_m = \max(\max_{i=1\ldots n} t_i, 2 \cdot \lfloor t_{sum}/m \rfloor)$
LB^Z	Lower bound on the line capacity: $LB^Z = \min_{m=m_{min}\cdots m_{max}}(m \cdot LB^{ct}_m)$
UB^Z	Upper bound on the line capacity: $UB^Z = \min_{m=m_{min}\cdots m_{max}}(m \cdot UB^{ct}_m)$

LB^{ct}_m and UB^{ct}_m are a lower and upper bound, respectively, on the cycle time for a given number of workstations m; that is, they are bounds of SALBP-2 (Scholl, 1999). Thus, $LB^{ct}_{m_{max}}$ and $UB^{ct}_{m_{min}}$ are lower and upper bounds on the cycle time of SALBP-E, respectively. With respect to the bounds on the line capacity, they are straightforwardly deduced from the bounds on the efficiency formulated in Scholl (1999).

Moreover, we define the following additional data that are derived from the above data and bounds:

P_i	Set of all tasks which must precede task i $(i = 1 \ldots n)$: $P_i = \bigcup_{h=1\ldots n \mid (h,i) \in IP}(\{h\} \cup P_h)$
S_i	Set of all tasks which must succeed task i $(i = 1 \ldots n)$: $S_i = \bigcup_{h=1\ldots n \mid (i,h) \in IP}(\{h\} \cup S_h)$
E_i	Earliest workstation to which task i can be assigned $(i = 1 \ldots n)$: $E_i = \lceil (t_i + \sum_{h \in P_i} t_h)/UB^{ct} \rceil$
L_i	Latest workstation to which task i can be assigned $(i = 1 \ldots n)$: $L_i = \max_{m=m_{min}\cdots m_{max}}(m + 1 - \lceil (t_i + \sum_{h \in S_i} t_h)/UB^{ct}_m \rceil)$
W_j	Set of tasks that can be assigned to workstation j $(j = 1 \ldots m_{max})$: $W_j = \{i = 1 \ldots n : E_i \leq j \leq L_i\}$

Formulations of SALBP-1 (also of SALBP-2) usually define the earliest and latest workstation in which each task can be assigned based on the precedence relations (e.g. Saltzman and Baybars, 1987). Analogously for SALBP-E, we define E_i and L_i as the earliest and latest workstation in which task i can be assigned, respectively. They are used to reduce significantly the size of the MILP model of the problem.

The mathematical model is formulated as follows:

Variables

z	Line capacity: $LB^Z \leq z \leq UB^Z$
ct	Cycle time: $LB^{ct} \leq ct \leq UB^{ct}$
$x_{ij} \in \{0,1\}$	1 if task i is assigned to station j; 0 otherwise $(i = 1 \ldots n; j = E_i \ldots L_i)$
$y_j \in \{0,1\}$	1 if station j exists; 0 otherwise $(j = m_{min} + 1 \ldots m_{max})$

Model

$$[\text{MIN}]\ z \tag{1}$$

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

$$\textstyle\sum_{i \in W_j} t_i \cdot x_{ij} \leq ct \qquad j = 1 \ldots m_{max} \tag{3}$$

$$\textstyle\sum_{j=E_h}^{L_h} j \cdot x_{hj} \leq \sum_{j=E_i}^{L_i} j \cdot x_{ij} \qquad \forall (h,i) \in IP \tag{4}$$

$$\textstyle\sum_{i \in W_j} x_{ij} \leq \|W_j\| \cdot y_j \qquad j = m_{min} + 1 \ldots m_{max} \tag{5}$$

$$z \geq m_{min} \cdot ct \tag{6}$$

$$z \geq j \cdot ct - M_j \cdot (1 - y_j) \qquad j = m_{min} + 1 \ldots m_{max}$$

$$\text{where } M_j = j \cdot UB^{ct} - \textstyle\min_{m=m_{min}}^{j-1} (m \cdot LB_m^{ct}) \tag{7}$$

$$y_j \geq y_k \qquad j = m_{min} + 1 \ldots m_{max} - 1; k = j + 1 \ldots m_{max} \tag{8}$$

Objective function (1) minimises the line capacity (recall that it is equivalent to maximise the line efficiency). Constraints (2) ensure that each task is assigned to one and only one workstation. Constraints (3) imply that the cycle time is not lower than the total task process time assigned to any workstation. Constraints (4) impose the precedence relations. Constraints (5) force a workstation to be open when some task is assigned to it. Constraints (6) and (7) link the line capacity with the number of open workstations and the cycle time. Finally, constraints (8) impose that the open workstations must be contiguous (as it is done in Pastor et al., 2011) and break symmetries.

The main differences between our enhanced MILP model and the model proposed in Plans and Corominas (1999) (let they be named Enh-MILP and CP-MILP, respectively) are: (i) the addition of a lower bound on the cycle time and a lower and upper bound on the line capacity; and (ii) the constraints that impose the contiguousness of the open workstations.

3. Iterative procedures

First we explain in Section 3.1 the procedure proposed in Wei and Chao (2011) and corrected in García-Villoria and Pastor (2013); let it be named Iterative Procedure (IP). Then we propose our enhanced iterative procedure in Section 3.2; let it be named Enhanced Iterative Procedure (EIP). When non-small instances are solved, the above procedures may need a huge computational time to solve optimally them, so a maximum global time has to be set. Section 3.3 discuses several heuristics based on IP and EIP in which the maximum time is shared among their iterations in different ways.

3.1. Iterative Procedure (IP)

The Wei and Chao's iterative procedure (IP) consists in solving the corresponding SALBP-2 for each value of m between m_{min} and m_{max}. Figure 1 shows its pseudocode.

```
Let $CT(Sol)$ be the cycle time of solution $Sol$
$Z^* = \infty$
For $m = m_{min} \dots m_{max}$ do:
    $Sol$ = Solve SALBP-2 with $m$ workstations
    If $m \cdot CT(Sol) \leq Z^*$ then $Sol^* = Sol, Z^* = m \cdot CT(Sol)$ End if
End for
Return $Sol^*$
```

Figure 1: *Pseudocode of* Iterative Procedure *for SALBP-E.*

The following MILP model is used to solve SALBP-2:

Additional data

m	Number of workstations
E'_i	Earliest workstation to which task i $(i = 1 \dots n)$ can be assigned: $E'_i = \lceil (t_i + \sum_{h \in P_i} t_h)/UB^{ct}_m \rceil$
L'_i	Latest workstation to which task i $(i = 1 \dots n)$ can be assigned: $L'_i = m + 1 - \lceil (t_i + \sum_{h \in S_i} t_h)/UB^{ct}_m \rceil$
W'_j	Set of tasks that can be assigned to workstation j $(j = 1 \dots m)$: $W'_j = \{i = 1 \dots n : E'_i \leq j \leq L'_i\}$

Variables

ct	Cycle time: $LB^{ct}_m \leq ct \leq UB^{ct}_m$
$x_{ij} \in \{0,1\}$	1 if task i is assigned to station j; 0 otherwise $(i = 1 \dots n; j = E'_i \dots L'_i)$

Model

$$[\text{MIN}]\ ct \tag{9}$$

$$\sum_{j=E'_i}^{L'_i} x_{ij} = 1 \qquad i = 1 \dots n \tag{10}$$

$$\textstyle\sum_{i \in W'_j} t_i \cdot x_{ij} \leq ct \qquad j = 1 \ldots m \tag{11}$$

$$\textstyle\sum_{j=E'_h}^{L'_h} j \cdot x_{hj} \leq \sum_{j=E'_i}^{L'_i} j \cdot x_{ij} \qquad \forall (h,i) \in IP \tag{12}$$

3.2. Enhanced Iterative Procedure (EIP)

One drawback of IP is that each resolution of SALBP-2 does not use any information of the previous SALBP-2 solutions. To improve the performance of IP we propose an enhanced iterative procedure (EIP), which takes advantage of the best solution known up to the current iteration.

Let $ni = m_{max} - m_{min} + 1$ be the number of iterations of the IP and let Z^*_p be the best line capacity value found at the beginning of iteration p ($p = 1 \ldots ni$); we consider $Z^*_1 = \infty$. The SALBP-2 to solve at iteration p has $m = p + m_{min} - 1$ workstations; let Sol_m be its optimum solution. The cycle time of Sol_m, $CT(Sol_m)$, must fulfil the following condition in order to have a better line capacity than the best SALBP-E solution known at the moment:

$$m \cdot CT(Sol_m) < Z^*_p \equiv CT(Sol_m) \leq \lfloor (Z^*_p - 1)/m \rfloor \tag{13}$$

Eq. 13 assumes, without loss of generality, that the process task times are integers, restricting cycle times and line capacities to integer values.

EIP is an adaptation of IP in order to reduce the search space of each SALBP-2 thanks to the condition expressed in Eq. 13. Thus, each iteration of EIP may be more efficient. The EIP pseudocode is very similar to the one shown in Figure 1. The differences are at each iteration p the Z^*_p value is available and the domain of variable ct of the MILP model to solve SALBP-2 may be tighter. The ct domain is expressed in Eq. 14:

$$LB^{ct}_m \leq ct \leq \min(UB^{ct}_m, \lfloor (Z^*_p - 1)/m \rfloor) \tag{14}$$

Note that the SALBP-2 model used by EIP may be infeasible. Its infeasibility at iteration p means that the optimum solution of SALBP-2 with $m = p + m_{min} - 1$ workstations is not better than the best SALBP-E solution found up to iteration $p - 1$. Thus, EIP continues the search at the next iteration.

3.3. Heuristics derived from IP and EIP

Wei and Chao (2011) assumed that, at each iteration of IP, its corresponding SALBP-2 would be solved optimally. However, for non-small instances, the required time may be huge in practice. Heuristics can be derived from IP and EIP limiting the maximum total computing time to D and returning the best solution found. In that case, one question that arises is how to distribute the available time among the ni iterations. We propose three ways similarly as it is done in the ALBP literature (see, for example,Pastor 2011):

T1 The maximum computing time for solving SALBP-2 at the first iteration is D, at the second iteration is the remaining time (if any), and so on. That is, the time limit at iteration p, TL_p, is the following:
$TL_p = D - \sum_{q=1\ldots p-1} \tau_q$ for $p = 1\ldots ni$, where τ_q is the time spent for solving SALBP-2 at iteration q.

T2 The maximum computing time for solving SALBP-2 is half of the remaining time (except for the last SALBP-2, which is all remaining time). That is:
$TL_p = (1/2) \cdot (D - \sum_{q=1\ldots p-1} \tau_q)$ for $p = 1\ldots ni-1$, and $TL_{ni} = D - \sum_{q=1\ldots ni-1} \tau_q$.

T3 The maximum computing time for solving SALBP-2 is the remaining time divided by the number of the remaining iterations. That is:
$TL_p = (D - \sum_{q=1\ldots p-1} \tau_q)/(ni - p + 1)$ for $p = 1\ldots ni$.

Combining the two iterative procedures (IP and EIP) and the three ways of splitting the available computing time (T1, T2 and T3) results in a total of six heuristics: IP-T1, IP-T2, IP-T3, EIP-T1, EIP-T2 and EIP-T3. Note that two heuristics can also be derived from the introduced mathematical models (Section 2) limiting their maximum computing time to D.

Illustrative example of the heuristics mechanism

In order to clarify how the proposed heuristics work, we will show as example the iterations of EIP-T1 when it is applied to solve the instance named Lutz3 with $m_{min} = 12$ and $m_{max} = 15$ with a total computing time $D = 3600$ s. Thus, EIP-T1 will iterate 4 times ($ni = m_{max} - m_{min} + 1 = 4$).

EIP-T1

Iteration $p = 1$. $Z_1^* = \infty$. $TL_1 = D = 3600$. SALBP-2 is solved with $m = 12$ workstations and a solution is found; let it be called Sol^{12}, with $ct(Sol^{12}) = 138$. Let $Sol^* = Sol^{12}$ (whose line capacity is equal to $12 \cdot 138 = 1656$). The time spent in this iteration is $\tau_1 = 14$.

Iteration $p = 2$. $Z_2^* = 1656$. $TL_2 = D - \tau_1 = 3586$. SALBP-2 is solved with $m = 13$ workstations and the model is infeasible (therefore, there is no solution with 13 workstations and a line capacity smaller than Z_2^*, recall Eq. 14). The time spent in this iteration is $\tau_2 = 1$.

Iteration $p = 3$. $Z_3^* = 1656$. $TL_3 = D - \tau_1 - \tau_2 = 3585$. SALBP-2 is solved with $m = 14$ workstations and a solution is found; let it be called Sol^{14}, with $ct(Sol^{14}) = 118$.

Let $Sol^* = Sol^{14}$ (whose line capacity is equal to $14 \cdot 118 = 1652$). The time spent in this iteration is $\tau_3 = 5$.

Iteration $p = 4$. $Z_4^* = 1652$. $TL_4 = D - \tau_1 - \tau_2 - \tau_3 = 3580$. SALBP-2 is solved with $m = 15$ workstations and a solution is found; let it be called Sol^{15}, with $ct(Sol^{15}) = 110$. Let $Sol^* = Sol^{15}$ (whose line capacity is equal to $54 \cdot 110 = 1650$).

Return solution Sol^*

EIP-T2 and EIP-T3 would iterate in a similar way but the time limits at each iteration would be the following. For EIP-T2: $TL_1 = D/2 = 1800$, $TL_2 = (D - \tau_1)/2 = 1793$, $TL_3 = (D - \tau_1 - \tau_2)/2 = 1792$ and $TL_4 = D - \tau_1 - \tau_2 - \tau_3 = 3580$. And for EIP-T3: $TL_1 = D/4 = 900$, $TL_2 = (D - \tau_1)/3 = 1195$, $TL_3 = (D - \tau_1 - \tau_2)/2 = 1792$ and $TL_4 = D - \tau_1 - \tau_2 - \tau_3 = 3580$. Note that the results at each iteration would not be different since $\tau_p < TL_p$ for $p = 1 \ldots 4$.

4. Computational experiments

The MILP models are solved using the IBM ILOG CPLEX 12.2 Optimiser; the absolute MIP gap tolerance is set to 0.9999 (since process task times are integer values). The iterative procedures are implemented in Java SE 1.6.21. The experiments are run on a PC 3.16 GHz Pentium Intel Core 2 Duo E8500 with 3.46 GB of RAM. The maximum computing time D per instance and procedure is limited to 3,600 seconds. Note that in a real application, when the design of the assembly line is a strategic problem, the computational time could be much greater; however, one hour seems a reasonable compromise in order to use a variety of instances and, at the same time, make the computational experiment affordable.

Section 4.1 presents the test instances used in the experiments. Section 4.2 shows the obtained results and Section 4.3 compares the heuristic procedures between them. Section 4.4 studies the quality of the proposed heuristics according to the characteristics of the instances. Finally, Section 4.5 analyses how the distribution of the computing time among the iterations affects the performance of the heuristics.

4.1. Description of the test instances

Our experiments are performed on the 256 benchmark SALBP-E instances that are available in Scholl and Klein's assembly line balancing research website (www.assembly-line-balancing.de). Scholl (1993) generated these instances from twenty-four problems varying the m_{min} and m_{max} values. Table 1 shows the problem name, its number of tasks,

Table 1: *Test problems.*

Name	Number	Process time			Order	m_{min}	m_{max}
	of tasks	Minimum	Maximum	Average	strength	range	range
Arcus1	83	233	3691	912.1	59.09	3 to 19	9 to 21
Arcus2	111	10	5689	1354.9	40.38	3 to 22	7 to 27
Barthold	148	3	383	38.1	25.8	3 to 14	5 to 15
Barthol2	148	1	83	28.6	25.8	3 to 51	10 to 52
Bowman	8	3	17	9.4	75	3 to 4	5 to 5
Buxey	29	1	25	11.2	50.74	3 to 10	7 to 13
Gunther	35	1	40	13.8	59.5	3 to 12	8 to 13
Hahn	53	40	1775	264.6	83.82	3 to 7	4 to 8
Heskiaoff	28	1	108	36.6	22.49	3 to 9	10 to 10
Jackson	11	1	7	4.2	58.18	3 to 5	7 to 7
Jaeschke	9	1	6	4.1	83.33	3 to 6	7 to 7
Killbridge	45	3	55	12.3	44.55	3 to 10	11 to 11
Lutz1	32	100	1400	441.9	83.47	3 to 10	11 to 11
Lutz2	89	1	10	5.4	77.55	3 to 46	13 to 49
Lutz3	89	1	74	18.5	77.55	3 to 19	9 to 23
Mansoor	11	2	45	16.8	60	3 to 4	5 to 5
Mertens	7	1	6	4.1	52.38	3 to 4	5 to 5
Mitchell	21	1	13	5	70.95	3 to 7	9 to 9
Roszieg	25	1	13	5	71.67	3 to 9	10 to 10
Sawyer	30	1	25	10.8	44.83	3 to 10	7 to 13
Scholl	297	5	1386	234.5	58.16	3 to 50	4 to 51
Tonge	70	1	156	50.1	59.42	3 to 21	12 to 23
Warnecke	58	7	53	26.7	59.1	3 to 28	13 to 30
Wee-Mag	75	2	27	20	22.67	3 to 36	9 to 38

their minimum, maximum and average process times, the order strength of the precedence graph and the ranges on the m_{min} and m_{max} values used to generate the instances.

4.2. Results of the procedures

When the objective function value of the obtained solution is equal to the lower bound, LB, value (or, strictly speaking, when the difference between them is less than one), then the solution optimality is demonstrated. In the case of the MILP procedures for SALBP-E, LB is equal to the lower bound returned by CPLEX. In the case of the iterative procedures, LB is calculated as follows. Let LBcplex_m^{ct} be the lower bound on the cycle time returned by CPLEX when it solves SALBP-2 with workstations; thus, LB is equal to $\min_{m=m_{min}\ldots m_{max}}(m \cdot \text{LBcplex}_m^{ct})$. Moreover, we consider that $\text{LBcplex}_m^{ct} = \text{LB}_m^{ct}$ when no lower bound is returned by CPLEX within the maximum time assigned and $\text{LBcplex}_m^{ct} = \infty$ when the SALBP-2 model is infeasible.

Table 2 summarises the type of solutions obtained with each procedure. The following information is given: the number of instances with a proved optimal solution (#Opt); the number of instances with a feasible solution whose optimality is not proven (#Fea); and the number of instances without finding a feasible solution (#Uns).

Table 2: *Type of the obtained solutions.*

	CP-MILP	Enh-MILP	IP-T1	IP-T2	IP-T3	EIP-T1	EIP-T2	EIP-T3
#Opt	72	90	121	124	120	144	143	141
#Fea	175	156	135	132	135	112	113	114
#Uns	9	10	0	0	1	0	0	1

In terms of proved optimal solutions, our proposed MILP model, Enh-MILP, outperforms the model proposed in Plans and Corominas (1999), CP-MILP. Nevertheless, both approaches are clearly worse than any iterative procedure. Moreover, Enh-MILP and CP-MILP cannot find a solution in 10 and 9 instances, respectively, whereas the iterative procedures always find a solution (IP-T1, IP-T2, EIP-T1 and EIP-T2) or only the same 1 instance remains unsolved (IP-T3 and EIP-T3).

4.3. Comparison between the heuristic procedures

We focus on the comparison of the heuristic pairs (IP-T1, EIP-T1), (IP-T2, EIP-T2) and (IP-T3, EIP-T3). Table 3 shows, for each pair, the number of instances in which both procedures guarantee the optimal solution (#Opt) and the average computational time, in seconds, for solving these instances ($\overline{\text{Time}}$). Table 3 also shows, for each pair, the number of instances when none of the procedures guarantees the optimal solution in the computing time allowed (#Fea) and the number of times that a procedure finds a better solution than the another procedure (#Best).

Table 3: *Results when both procedures guarantee the optimal solution or when neither procedure guarantees the optimal solution.*

Time distribution	Optimal solutions			Feasible solutions		
	#Opt	$\overline{\text{Time}}$		#Fea	#Best	
		IP	EIP		IP	EIP
T1	121	1057.1	201.39	112	0	24
T2	124	1189.39	180.38	113	0	42
T3	120	1045.42	191.94	114	0	32

Results in Tables 2 and 3 confirm the effectiveness of the proposed improvements regardless of how the global time is shared among the iterations of the procedures. EIP heuristics are able to find around 15% more proven optimal solutions than the IP heuristics and all instances solved optimally by one IP heuristic are also solved optimally by its analogous EIP heuristic. Regarding the computational time, the average times spent by EIP heuristics when an instance is solved optimally are at least five times less than the IP average times. Furthermore, all feasible solutions reached by an EIP heuristic has the same or better quality than the solutions reached by its analogous IP heuristic.

4.4. Quality of the proposed heuristics according to the instance characteristics

We now focus on the comparison of the three EIP heuristics, which are the best heuristics, taking into account the influence of the characteristics of the instances (as outlined in Table 1) on the quality of the results. Specifically, we create subset of instances according to the number of tasks and order strength. The number of tasks is respectively classified as low, medium and high according to the ranges (7, 35), (45, 111) and (148, 297). Likewise, the order strength is respectively considered low, medium and high according to the ranges (22.49, 25.80), (40.38, 60.00) and (70.95, 83.82).

Table 4: *Number of proven optimal solutions for each combination of number of tasks and order strength.*

Procedure	LL (6)	LM (30)	LH (20)	ML (22)	MM (67)	MH (40)	HL (39)	HM (32)
EIP-T1	6	30	20	14	25	31	15	3
EIP-T2	6	30	20	14	24	32	15	2
EIP-T3	6	30	20	14	22	31	15	3

Table 5: *Quality of the solutions for each combination of number of tasks and order strength.*

Procedure	LL (6)	LM (30)	LH (20)	ML (22)	MM (67)	MH (40)	HL (39)	HM (32)
=	6	30	20	20	32	30	19	2
EIP-T1	0 (0)	0 (0)	0 (0)	0 (2)	5 (9)	0 (1)	3 (4)	15 (2)
EIP-T2	0 (0)	0 (0)	0 (0)	0 (0)	12 (3)	1 (0)	4 (4)	7 (2)
EIP-T3	0 (0)	0 (0)	0 (0)	0 (0)	7 (15)	2 (7)	1 (8)	5 (23)

Tables 4 and 5 report the results obtained by each procedure for combinations of order strength and number of tasks. The column headers use L, M and H for low, medium and high, respectively. The first letter is for the number of tasks and the second letter for the order strength. This is followed by the number of instances in the subset, which is shown between parentheses. Table 4 shows the number of instances with a proved optimal solution. Table 5 contains the number of times that all procedure obtains the same quality solution (”=”) and, for each procedure, the number of times that it obtains a better solution (and a worst solution between parentheses) than the other two solutions obtained with the other procedures; we consider the non-existent solution of the only instance unsolved with EIP-T3 (see Table 2) worse than any feasible solution.

Results show that all EIP procedures are very similar with respect to the number of optimal solutions regardless of the characteristics of the instances. However, with respect to the quality of the solutions, we can see EIP-T3 that is, on average, worse than EIP-T1 and EIP-T2. EIP-T2 tends to be better than EIP-T1 in terms of quantity of best and worst solutions; the exception is in instances with high number of tasks and medium order strength, in which EIP-T1 performs, on average, better.

4.5. Analysis of the proposed heuristics

Lastly, we investigate the different performances of the EIP heuristics. To do so, we record for each instance the number of workstations of the best known solution, m^* (i.e. the best solution obtained with any of the eight procedures). If multiple best solutions have been obtained, then the one with the lowest number of workstations is considered. Table 6 reports the number of times (#Ins) that the difference between m^* and m_{min} is 0, 1, 2, etc.

Table 6: *Differences between the best number of workstations and m_{min}.*

$m^* - m_{min}$	$= 0$	$= 1$	$= 2$	$= 3$	$= 4$	≥ 5
#Ins	168	44	26	11	3	4

We can observe that most of the number of workstations of the best found solutions are equal to or close to the minimum value allowed. Thus, it seems reasonable that EIP-T3, which tends to share equally the maximum global time among all SALBP-2 resolutions, performs worse than EIP-T1 and EIP-T2, which tend to give priority to SALBP-2 with fewer number of workstations. On the other hand, EIP-T1 might perform slightly worse than EIP-T2 because in some cases the time spent by EIP-T1 in the first SALBP-2 may be too much (potentially all time could be spent in it and no SALBP-2 is solved with other numbers of workstations).

5. Conclusions and future research

SALBP is the type of assembly line balancing problems most studied in the literature. However, most research efforts are reduced to SALBP-1 and SALBP-2. The resolution of SALBP-E, in which the number of workstations and cycle time are variables, has not received enough attention in the literature with the notable exceptions of Plans and Corominas (1999) and Wei and Chao (2011).

The special interest in our work has been the exact resolution of SALBP-E. We propose an enhanced MILP model, together with an iterative procedure (based on solving SALBP-2 at each iteration) which improves the one proposed by Wei and Chao (2011). Since we expected that large instances cannot be solved optimally in a practical time, we proposed several heuristics based on limiting the maximum computing time and returning the best solution found. Specifically, we propose three ways of distributing the available time among the different SALBP-2 resolutions of the iterative procedures.

Through extensive experimentation, we have been able to determine the benefits of adding the proposed improvements to the existing iterative procedure. On the other hand, we have detected that a direct approach as the proposed MILP model performs worse than any iterative procedure. Nevertheless, other direct approaches should not be dismissed and a procedure based on, for instance, branch & bound will be studied. Regarding the heuristic resolution of this problem, another line of search that we will follow is the use of metaheuristics, which may obtain better results for large instances.

Acknowledgements

The authors wish to express their gratitude to the anonymous reviewers for their valuable comments, which have helped to improve the quality of this paper.

References

Al-Hawari, T., Ali, M., Al-Araidah, O. and Mumani, A. (2015). Development of a genetic algorithm for multi-objective assembly line balancing using multiple assignment approach. *The International Journal of Advanced Manufacturing Technology*, 77, 1419–32.

Battaïa, O. and Dolgui, A. (2012). Reduction approaches for a generalized assembly line balancing problem. *Computers and Operations Research*, 39, 2337–45.

Battaïa, O. and Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, 142, 259–77.

Battaïa, O., Delorme, X., Dolgui., A., Hagemann, J., Horlemann, A., Kovalev, S. and Malyutin, S. (2015). Workforce minimization for a mixed-model assembly line in the automotive industry. *International Journal of Production Economics*, 170, 489–500.

Bautista, J., Batalla-García, C. and Alfaro-Pozo, R. (2016). Models for assembly line balancing by temporal, spatial and ergonomic risk attributes. *European Journal of Operational Research*, 251, 814–29.

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

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

Boysen, N. and Fliedner, M., Scholl, A. (2008). Assembly line balancing: Which model to use when?. *International Journal of Production Economics*, 111, 509–28.

Calleja, G., Corominas, A., García-Villoria, A. and Pastor, R. (2014). Combining matheuristics and MILP to solve the Accessibility Windows Assembly Line Balancing Problem Level 2 (AWALBP-L2). *Computers & Operations Research*, 48, 113–23.

Capacho, L., Pastor, R., Dolgui, A. and Guschinskaya, O. (2009). An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem. *Journal of Heuristics*, 15, 109–32.

Chica, M., Bautista, J., Cordón, O. and Damas, S. (2016). A multiobjective model and evolutionary algorithms for robust time and space assembly line balancing under certain constraints. *Omega*, 58, 55–68.

Corominas, A., Ferrer, L. and Pastor, R. (2011). Assembly line balancing: general resource-constrained case. *International Journal of Production Research*, 49, 3527–42.

Erel, E. and Sarin, C.S. (1998). A survey of the assembly line balancing procedures. *Production Planning & Control*, 9, 414–34.

Esmaeilbeigi, R., Naderi, B. and Charkhgard, P. (2015). The type E simple assembly line balancing problem: A mixed integer linear programming formulation. *Computers & Operations Research*, 64, 168–177.

Gao, J., Sun, L., Wang, L. and Gen, M. (2009). An efficient approach for type II robotic assembly line balancing problems. *Computers & Industrial Engineering*, 56, 1065–80.

García-Villoria, A. and Pastor, R. (2013). Erratum to "A solution procedure for type E simple assembly line balancing problem". *Computers & Industrial Engineering*, 66, 201–22.

Gurevsky, E., Battaïa, O. and Dolgui, A. (2012). Balancing of simple assembly lines under variations of task processing times. *Annals of Operational Research*, 201, 265–86.

Gutjahr, A. L. and Nemhauser, G.L. (1964). An algorithm for the line balancing problem. *Management Science*, 11, 308–15.

Levitin, G., Rubinovitz, J. and Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Research*, 168, 811–25.

Martino, R. and Pastor, R. (2010). Heuristic procedures for solving the general assembly line balancing problem with setups. *International Journal of Production Research*, 48, 1787–804.

Moreira, M.C.O., Cordeau, J-F., Costa, A.M. and Laporte, G. (2015). Robust assembly line balancing with heterogeneous workers. *Computers & Industrial Engineering*, 88, 254–63.

Ogan, D. and Azizoglu, M. (2015). A branch and bound method for the line balancing problem in U-shaped assembly lines with equipment requirements. *Journal of Manufacturing Systems*, 36, 46–54.

Pastor, R. (2011). LB-ALBP: The lexicographical bottleneck assembly line balancing problem. *International Journal of Production Research*, 49, 2425–42.

Pastor, R., García-Villoria, A. and Corominas, A. (2011). Comparing ways of breaking symmetries in mathematical models for SALBP-1. *Assembly Automation*, 31, 385–7.

Plans, J. and Corominas, A. (1999). Modelling and solving the SALBP-E problem. *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning*, Porto, Portugal, 356–60.

Rekiek, B., Dolgui, A., Deichambre, A. and Bratcu, A. (2002). State of art of optimization methods for assembly line design. *Annual Reviews in Control*, 26, 163–74.

Saif, U., Guan, Z., Liu, W., Zhang, C. and Wang, B. (2014). Pareto based artificial bee colony algorithm for multi objective single model assembly line balancing with uncertain task times. *Computers & Industrial Engineering*, 76, 1–15.

Saltzman, M.J. and Baybars, I. (1987). A two-process implicit enumeration algorithm for the simple assembly line balancing problem. *European Journal of Operational Research*, 32, 118–29.

Scholl, A. (1993). Data of assembly line balancing problem. *Schriften zur Quantitativen Betriebswirtschaftslehre*, Darmstadt University of Technology, Number 16/93.

Scholl, A. (1999). *Balancing and Sequencing of Assembly Lines*. Heidelberg, Physica-Verlag, 2nd rev.

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

Scholl, A. and Becker, C. (2006b). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168, 666–93.

Scholl, A., Fliedner, M. and Boysen, N. (2010). Absalom: Balancing assembly lines with assignment restrictions. *European Journal of Operational Research*, 200, 688–701.

Tapkan, P., Özbakir, L. and Baykasoğlu, A. (2016). Bee algorithms for parallel two-sided assembly line balancing problem with walking times. *Applied Soft Computing*, 39, 275–91.

Uğurdağ, H.F., Rachamadugu, R. and Papachristou, C.A. (1997). Designing paced assembly lines with fixed number of stations. *European Journal of Operational Research*, 102, 488–501.

Wei, N.C. and Chao, I.M. (2011). A solution procedure for type E simple assembly line balancing problem. *Computers & Industrial Engineering*, 61, 824–30.

Yoosefelahi, A., Aminnayeri, M., Mosadegh, H. and Ardakani, D. (2012). Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model. *Journal of Manufacturing Systems*, 31, 139–51.

Zacharia, P.Th. and Nearchou, A.C. (2013). A meta-heuristic algorithm for the fuzzy assembly line balancing type-E problem. *Computers & Operations Research*, 12, 3033–44.