ASALBP: The Alternative Subgraphs Assembly Line Balancing Problem

Liliana Capacho, Rafael Pastor

ASALBP: The Alternative Subgraphs Assembly Line Balancing Problem

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2006-IJPR-0126.R2
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	21-Dec-2006
Complete List of Authors:	Capacho, Liliana; Universidad Politécnica de Cataluña, Instituto de Organización y Control de Sistemas Industriales; Universidad de Los Andes, Departamento de Investigación de Operaciones y Centro de Simulación y Modelos Pastor, Rafael; Universidad Politécnica de Cataluña, Instituto de Organización y Control de Sistemas Industriales
Keywords:	ASSEMBLY LINE BALANCING, PRODUCTION MANAGEMENT
Keywords (user):	

ASALBP:

The Alternative Subgraphs Assembly Line Balancing Problem

Liliana Capacho[1,2] and Rafael Pastor[2]

[1] *Departamento de Investigación de Operaciones y Centro de Simulación y Modelos*

Universidad de Los Andes, Mérida, Venezuela

[2] *Instituto de Organización y Control de Sistemas Industriales,*

Universidad Politécnica de Cataluña, Barcelona, España

{liliana.capacho/rafael.pastor@upc.edu}

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

ASALBP:
The Alternative Subgraphs Assembly Line Balancing Problem

Abstract

Assembly line balancing problems basically consist in assigning a set of tasks to a group of workstations while maintaining the tasks' precedence relations, which are represented by a predetermined precedence graph. However, one or more parts of a product's assembly process may admit alternative precedence subgraphs, which represent possible assembly variants. In general, because of the great difficulty of the problem and the impossibility of representing alternative subgraphs in a precedence graph, the system designer will decide to select, a priori, one of such alternative subgraphs. This paper presents, characterizes and formulates a new general assembly line balancing problem with practical relevance: the Alternative Subgraphs Assembly Line Balancing Problem (ASALBP). Its novel characteristic is that it considers the possibility of having alternative assembly subgraphs, with the processing times and/or the precedence relations of certain tasks dependent on the assembly subgraph selected. Therefore, solving this problem implies simultaneously selecting an assembly subgraph for each part of the assembly that allows alternatives and balancing the line. The potentially positive effects of this on the solution of the problem are shown in a numerical example. Finally, a simple mathematical programming model is described and the results of a brief computational experiment are presented.

***Keywords*:** assembly line balancing, production.

1. Introduction

Basically, the Assembly Line Balancing Problem (ALBP) consists in assigning a set of indivisible tasks (any one characterized by its processing time and a set of precedence relations) to an ordered sequence of workstations in such a way that precedence constraints are maintained, the work content of each workstation does not exceed the cycle time and a given efficiency measure is optimised (e.g., the number of workstations).

A well-known classification of ALBPs is the one proposed by Baybars (1986), which differentiates between two classic problems: the Simple Assembly Line Balancing Problem

(SALBP) and the General Assembly Line Balancing Problem (GALBP). The SALBP includes problems characterized as follows (Baybars 1986): serial (straight) assembly lines processing a unique model of a single product are considered; all input parameters are known with certainty; the task processing times are independent of the workstation at which they are performed and of the preceding or following tasks; all workstations are equipped and manned to process any one of the tasks and any task can be processed at any workstation; a task cannot split among two or more workstations; tasks cannot be processed in an arbitrary sequence due to technological precedence requirements; all tasks must be processed; and no assignments restrictions apart from precedence constraints are considered. GALBPs are those problems in which one or more assumptions of the simple case are relaxed. If one reviews the literature concerning assembly line balancing problems, such as that by Baybars (1986), Ghosh and Gagnon (1989), Erel and Sarin (1998), Rekiek *et al*. (2002), Becker and Scholl (2006) or Scholl and Becker (2006), one can see that a huge amount of research exists, although most authors focus on the simple case. Nevertheless, it seems that generalized problems are becoming a widespread subject, since a significant variety of complex cases have already been examined, such as, for example, problems that consider lines with parallel workstations or parallel tasks; mixed or multi-models; multiple products; U-shaped, two-sided or buffered lines; incompatibility between tasks; stochastic processing times; equipment selection; or different types of objective functions (for example, Pinnoi and Wilhelm 1997, Pastor *et al*. 2002, Aase *et al*. 2003, Erel *et al*. 2005, Amen 2006, Andrés *et al*. 2006 and Vilarinho and Simaria 2006).

Numerous algorithms have been developed to solve ALBP, most of which focus on solving SALBP. Two major groups can be outlined: exact methods, which are mainly based on linear programming, dynamic programming and branch-and-bound procedures, and heuristic and metaheuristic methods. Information concerning both types of solving procedures can be found, for example, in Baybars (1986), Talbot *et al*. (1986), Erel and Sarin (1998), Becker and Scholl (2006) and Scholl and Becker (2006).

The objective of this paper is to present a new GALBP with practical relevance, referred to by the authors as the Alternative Subgraphs Assembly Line Balancing Problem (ASALBP). Generally, it is considered that there exists a predetermined precedence graph, although in

reality there may be several alternative precedence subgraphs for various parts in the assembly process of a product. When the processing time of one or more tasks is dependent on the processing sequence, various alternative subgraphs may appear, one of which must be selected; normally, processing times are considered to be inherent to the tasks or, in some cases, dependent on the equipment that performs them (see, for example, Bukchin and Tzur 2000). Alternative precedence subgraphs may also be needed when there are assembly alternatives. Nevertheless, the system designer normally selects a priori one alternative from all the possible alternatives in order to determine the precedence graph. The ASALBP considers the possibility of having alternative assembly subgraphs, in which the processing times of some tasks and/or their precedence relations are dependent on the assembly subgraph selected. Therefore, a decision problem, regarding the selection of an assembly subgraph for each part of the assembly that allows alternatives, must be solved together with the balancing problem.

The literature presents a variety of problems in which alternative precedence subgraphs are considered in the assembly/disassembly process of certain products. Examples include the toy manufacturing problem mentioned in Das and Nagendra (1997), the production of commercial hand-held drills (Senin et al., 2000) and the process of disassembling complex products (Gungor and Gupta, 1997). The authors are also familiar with a real-life case related to the process of assembling car dashboards in the automotive industry.

In the comprehensive literature review carried out by the authors, this type of problem has not been addressed before. In Pinto *et al*. (1983), and according to Bukchin and Tzur (2000), the problem of selecting limited equipment, which involves processing alternatives, is considered: each alternative represents a limited equipment selection that may be added to the existing equipment in the workstation; in any case, the precedence relations between tasks are always maintained. Pinto et al. discuss a new possibility: '*In practice it is possible that a particular processing alternative can change the nature of the precedence requirements such that the requirements for the replacing task are not the same as the union for the requirement of the replaced tasks… Such special situations are not dealt with here*' (p. 823). However, as stated, this possibility is neither formalized nor developed.

The remaining paper is organized as follows: Section 2 describes and characterizes the ASALBP, providing numerical examples to illustrate its potential benefits; aiming at formalizing the ASALBP, Section 3 presents a simple mathematical programming model and the results of a brief computational experiment; and finally, Section 4 provides several conclusions and ideas for further research.

2. The Alternative Subgraphs Assembly Line Balancing Problem (ASALBP)

2.1. *Alternative Subgraphs*

Normally, to assemble a part of a product a unique precedence subgraph is taken into account; this notwithstanding, it may sometimes be possible to consider alternative assembly subgraphs for the same part. Consider, for example, an intermediate phase in the process of assembling a motorbike, which consists of three tasks (B, C and D): two parts of a piece, including the axle, have to be attached to the motorbike's main body. First, one of the two parts is attached to the axle (task B or C), then the axle is placed onto the motorbike's body (task D), and finally the second part of the piece is attached to the axle (task C or B).

The assembly process described above can be carried out in two different ways, by determining two alternative precedence subgraphs (also referred to in this paper as assembly subgraphs): S1, which consist in performing task B first, then task D and lastly task C; and S2, which consists in performing task C first, then task D, and task B at the end. Finally, consider that task durations (tasks B, C and D last 3, 6 and 15 time units respectively) are fixed and independent of the order in which the tasks are processed (see Figure 1).

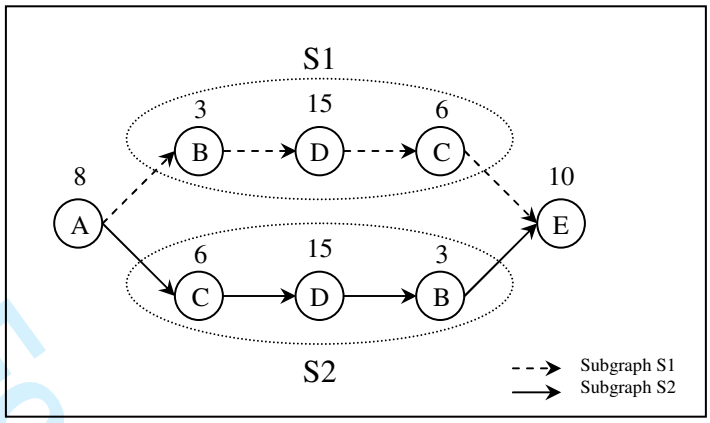

Figure 1. Alternative precedence subgraphs for the intermediate phase in the assembly of the motorbike

As previously mentioned, in assembly line balancing problems task processing times are usually considered to be independent of the way in which tasks are performed. However, in some cases the processing times may depend on the sequence in which tasks are processed. Consider, for example, the final phase in the process of assembling a motorbike (see Figure 2), which consists of three main sets of tasks: Z, which is the decoration of the motorbike's fairing (it involves several subtasks, such as sticking different colour stickers and text labels onto the fairing); K, which entails attaching the fairing to the motorbike; and L, which involves making the final adjustments. Possibly, there is not any technological precedence relation between Z and K; hence, these two tasks are represented in parallel in a standard precedence graph, whereas task L is preceded by tasks Z and K. Consider also that the processing time of task Z and/or K depends on the order in which they are processed (which hinder their representation in a precedence graph). In this example, task Z is considered to last 22 time units if performed before task K and 25 time units if it is performed afterwards; task K, on the other hand, lasts 13 time units regardless of the assembly sequence and task L lasts 7 time units.

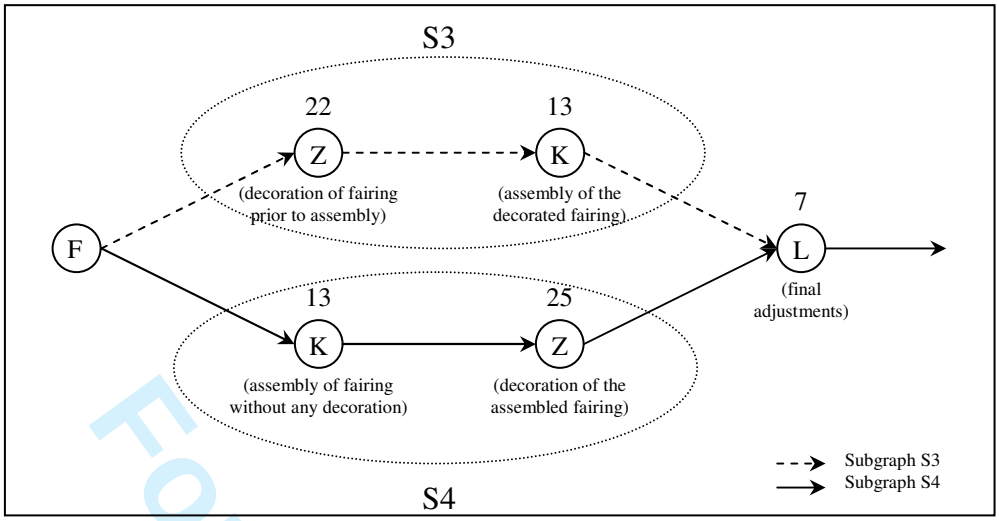

Figure 2. Alternative precedence subgraphs for the final assembly process of the motorbike

Therefore, in this case it is also possible to consider alternative precedence subgraphs (to represent each processing alternative): the first, S3, which has a total processing time of 35 time units and entails the decoration of the unattached fairing first and then its assembly; and the second, S4, which has a total processing time of 38 time units, and entails decorating the fairing provided it has already been attached to the motorbike (see Figure 2).

Using the standard diagramming representation, it is not possible to depict alternative precedence subgraphs. A potential way of representing precedence subgraphs S1 and S2, and S3 and S4, which is referred to by the authors as the precedence S-graph, is illustrated in Figure 3.

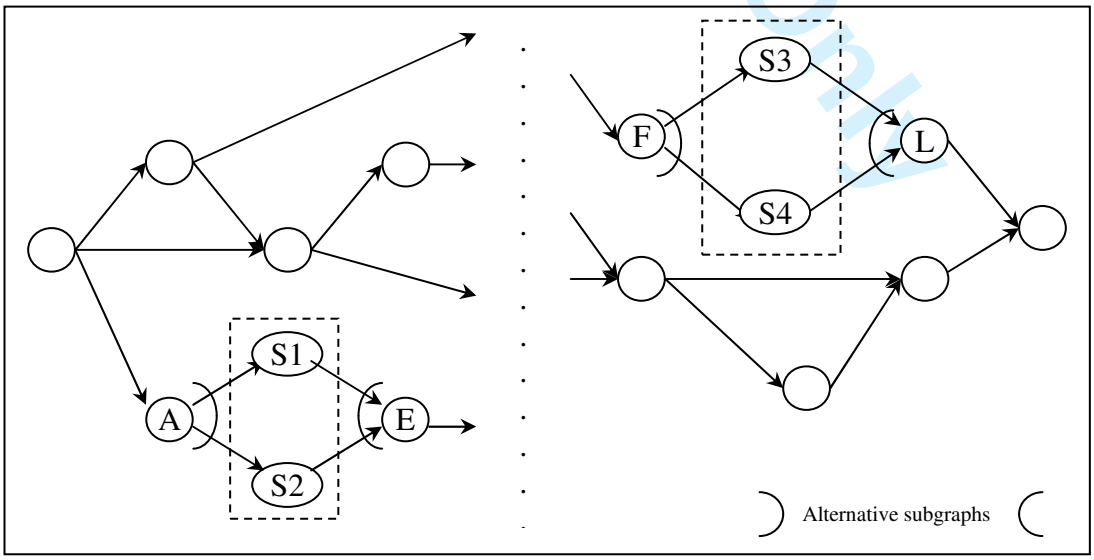

Figure 3. Precedence S-graph for the assembly process of the motorbike

Furthermore, it is also possible to consider situations involving the two cases previously described: alternative precedence subgraphs with task processing times that are dependent on their assembly sequence.

In order to make a more comprehensive definition of the S-graph as an alternative precedence diagramming tool, two aspects need to be discussed. On one hand, it is assumed that assembly alternatives do not overlap between each other; therefore, each alternative for each available subassembly is represented by a unique and independent precedence subgraph. On the other hand, fictitious tasks, with nil processing time, are used to facilitate the representation of two subassemblies with processing alternatives that are consecutive (this case is represented in Figure 4 by the fictitious task α).

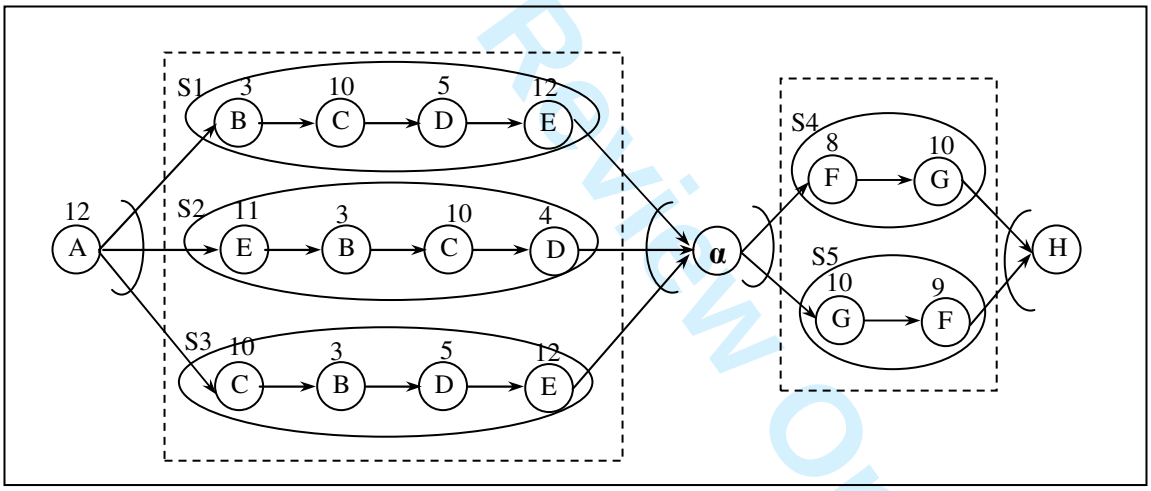

Figure 4. S-graph including fictitious tasks

The Alternative Subgraphs Assembly Line Balancing Problem (ASALBP) is a general assembly line balancing problem that considers alternative assembly subgraphs for task processing. Then, apart from considering cycle time, subgraph constraints have to be taken into account to assure that tasks belonging to a particular subassembly are processed in a unique assembly subgraph. Furthermore, if one considers task processing times not to be fixed, yet all known, but dependent on the subgraph through which tasks are processed, then the total processing time may vary from one processing alternative to another. Taking into

account these assumptions, two problems have to be solved simultaneously: 1) the precedence subgraphs or assembly subgraphs must be selected, which make it possible to reduce the precedence S-graph into a standard precedence graph and, in some cases, determines task processing times; and 2) the line must be balanced, which gives an assignment of tasks that optimises a given objective.

In practice, a procedure in which there are two independent stages is used to solve a problem like the one described above. In the initial stage, the system designer either decides, a priori, all the task durations (by fixing a precedence subgraph from all the possible alternatives, which is equivalent to imposing additional precedence relations other than the existing technological ones), or selects one assembly subgraph from the possible alternatives, if there are any. Different criteria, such as the shortest total processing time, for example, may be used to select the precedence subgraph. Lambert (2006) considers selecting an optimal assembly sequence on the basis of maximum task parallelism. Senin *et al*. (2000) consider that an assembly plan should be ranked according to multiple objectives, including line balancing; however, in their work on assembly planning they adopt a simplified objective measure based on planning the overall execution time. Once the assembly subgraphs are selected from amongst the alternatives and a precedence graph is available, the line is balanced in a second stage. By following this two-stage procedure, it cannot be guaranteed that an optimal solution of the global problem will be obtained, because the decisions taken by the system designer restrict the problem and cause information loss, which affects the assembly line balancing.

By considering alternative precedence subgraphs (precedence S-graphs), a higher level of difficulty is imposed on an assembly line balancing problem and the NP-hard condition of the ASALB problem is verified given that the simple case (SALBP) is also NP-hard (see e.g. Wee and Magazine 1982). However, as real industrial processes may involve assembly alternatives, the possibility of considering alternative subgraphs not only enables more practical and realistic instances of ALBP to be addressed, but may also favour an assignation of tasks to workstations in order to optimise a given objective. Regarding the conventional terminology (see e.g. Baybars 1986 or Scholl 1999), when the objective is to minimize the number of workstations for a given upper bound on the cycle time, the problem is referred to as

ASALBP-1. If the objective is to minimize the cycle time given the number of workstations, the problem is called ASALBP-2.

Two examples that, on the one hand, clarify the ideas previously introduced and, on the other, illustrate the benefits of selecting the precedence subgraphs and balancing the line simultaneously, rather than independently, are presented below.

2.2. *Example 1: the final process of assembling a motorbike*

Let us again consider the final process of assembling a motorbike, as described above: decorating the motorbike's fairing and assembling the fairing on the motorbike (see Figure 2). Additionally, the task of decorating the fairing (Z) has been further divided into four subtasks (G, H, I and J). Table 1 shows the disaggregated tasks, and, for each subgraph, the task processing times, the tasks' predecessors and the total processing time (including task L).

Task		Subgraph S3		Subgraph S4	
		Processing time	Predecessors	Processing time	Predecessors
Z	G: Decoration of fairing with yellow stickers	5	F	6	K
	H: Decoration of fairing with blue stickers	5	F	7	K
	I: Decoration of fairing with text labels	8	F	8	K
	J: Decoration of fairing with black stickers	4	F	4	K
K	Assembly of fairing	13	G, H, I and J	13	F
L	Final adjustment	7	K	7	G, H, I and J
	Total processing time	42		45	

Table 1. Data for Example 1

As can be seen in Table 1, some of the decorating tasks require longer processing times if they are performed on the attached fairing instead of on the unattached fairing. Alternative 2 therefore has a longer total processing time than Alternative 1. If this fact is taken into account, subgraph S3 would, in general, be chosen a priori over subgraph S4.

Table 2 presents the solutions obtained by optimally balancing each of the two resulting problems, one for each alternative subgraph, and aiming to minimize the number of workstations given a cycle-time upper bound equal to 17 time units. These results include task assignments (in addition to the workstation's load), total processing times and the number of workstations required.

Alternative subgraph	Station load (station time)				Total processing time	Number of stations
	I	II	III	IV		
S3	G, I, J (17)	H (5)	K (13)	L (7)	42	4
S4	K, J (17)	H, I (15)	G, L (13)	-	45	3

Table 2. Results for ASALBP-1

Considering both selecting the assembly subgraph and balancing the line simultaneously, subgraph S4 is the one that provides the best solution of the problem, in which three workstations are required instead of the four workstations required by subgraph S3. If S3 had been selected a priori, then a better solution would have been discarded.

The following results are obtained by optimally balancing the problem for each alternative subgraph and aiming to minimize the cycle time given a number of workstations equal to 3:

Alternative subgraph	Station load (station time)			Total processing time	Cycle time
	I	II	III		
S3	G, H, I (18)	J, K (17)	L (7)	42	18
S4	K, J (17)	G, H (13)	I, L (15)	45	17

Table 3. Results for ASALBP-2

As can be seen in Table 3, subgraph S4 again provides the best solution, even though it has a longer total processing time, which requires a cycle time of 17 instead of the 18 required by S3.

2.3. *Example 2: the intermediate process of assembling a motorbike*

Consider again the intermediate process of assembling a motorbike, as previously described: the attaching of two parts of a piece, including the axle, to the motorbike's main body (see Figure 1). By optimally balancing the problem for each alternative subgraph (including tasks A and E) and aiming to minimize the number of workstations, given a cycle time upper bound that is equal to 15 time units, the following results are obtained:

Alternative subgraph	Station load (station time)				Total processing time	Number of stations
	I	II	III	IV		
S1	A, B (11)	D (15)	C (6)	E (10)	42	4
S2	A, C (14)	D (15)	B, E (13)	-	42	3

Table 4. Results for Example 2

As shown in Table 4, the possibility of having alternative assembly subgraphs may favour an assignation of tasks to workstations, even when task processing times are not dependent on the tasks' processing sequence.

2.4. *Conclusions*

The examples outlined show how to consider alternative precedence subgraphs (assembly subgraphs) while simultaneously balancing the line may favour the assignation that minimizes the number of workstations (ASALBP-1) or the cycle time (ASALBP-2).

3. Mathematical programming model of the ASALBP

Consider the example of the process of assembling a motorbike introduced in Section 2 (see Figure 3). A way of solving the problem would be to keep the best solution when solving a SALBP considering each precedence graph obtained by combining the alternative subgraphs of each available subassembly contained by the S-graph. In the example, four precedence graphs are obtained when subgraphs S1-S3, S1-S4, S2-S3 and S2-S4 are considered. However, this process becomes infeasible for an S-graph with a large number of subassemblies with alternative subgraphs.

In this way, it becomes highly relevant to consider a unique model which simultaneously decides on both the assembly subgraph and the line balancing. In order to formalize the problem previously introduced, a simple binary linear program (01ILP) has been developed. This model is not proposed to solve the ASALBP of practical size to optimum within acceptable computing time, since even the simple case (SALBP) is expected to be intractable by mathematical programming and standard software in real-world instances. Therefore, the purpose of the 01ILP is merely to formalize in a simple way the new problem ASALBP.

3.1. *Mathematical model for ASALBP*

ASALBP-1 consists in minimizing the number of workstations for the upper bound on a given cycle time. To facilitate the use of the terminology, in the following formulation, any precedence graph is regarded as an alternative assembly route (hereafter, a route). It may be useful to remember that a precedence graph is obtained by the combination of all the subassembly subgraphs available.

- *Indices:*

  i for tasks

  j for workstations

  r for routes

- *Parameters:*

  n number of tasks ($i = 1,\ldots,n$)

  m_{max} upper bound on the number of workstations ($j = 1,\ldots,m_{max}$)

  m_{min} lower bound on the number of workstations

  nr number of alternative routes ($r = 1,\ldots,nr$)

  t_{ir} duration of task i when processed through route r ($i = 1,\ldots,n;\ r = 1,\ldots,nr$); in some cases this value is independent of route r (t_i)

  C_{max} upper bound on the cycle time

  PD_{ir} set of the immediate predecessors of task i, if task i is processed through route r ($i = 1,\ldots,n;\ r = 1,\ldots,nr$)

E_{ir}, L_{ir} earliest and latest station respectively that task i can be assigned to, if task i is processed through route r ($i = 1,\ldots,n$; $r = 1,\ldots,nr$). E_{ir} and L_{ir} can be obtained by considering the precedence relations and the task processing times. Furthermore, a task cannot be assigned to a workstation until all its predecessors have been assigned. As a result, the range of workstations to which each task can be assigned is obtained and the number of binary variables is reduced (see, for example, Patterson and Albracht 1975. For instance, E_{ir} is computed by rounding up to the nearest integer the result of dividing the task time plus the times of its predecessors by the cycle time. For example, considering that tasks A, B and C, all with processing time equal to 8, must be processed in the following way: task A precedes task B and task B precedes task C; furthermore, considering a cycle time equal to 20; then, workstation 2 is the first station to which task C can be assigned. Because tasks A and B should be assigned before task C, they load workstation 1 enough to avoid task C be assigned to workstation 1.

T_{jr} set of tasks potentially assignable to workstation j, $\{i \mid j \in [E_{ir}, L_{ir}]\}$, if the tasks are processed through route r ($j = 1,\ldots,m_{max}$; $r = 1,\ldots,nr$)

- *Decision variables:*

x_{ijr} = 1 if task i is assigned to workstation j and processed through route r ($\forall i$, $\forall r$, $\forall j \in [E_{ir}, L_{ir}]$) ; 0 otherwise.

y_j = 1 if there is any task assigned to workstation j ($j = m_{\min}+1,\ldots,m_{\max}$) ; 0 otherwise.

- *Model:*

$$\text{Minimize } z = \sum_{j=m_{\min}+1}^{m_{max}} j \cdot y_j \qquad (1)$$

$$\sum_{r=1}^{nr} \sum_{j=E_{ir}}^{L_{ir}} x_{ijr} = 1 \qquad \forall i \qquad (2)$$

$$\sum_{r=1}^{nr} \sum_{\forall i \in T_{jr}} t_{ir} \cdot x_{ijr} \le C_{\max} \qquad j = 1,\ldots,m_{\min} \qquad (3)$$

$$\sum_{r=1}^{nr} \sum_{\forall i \in T_{jr}} t_{ir} \cdot x_{ijr} \le C_{\max} \cdot y_j \qquad j = m_{\min}+1,\ldots,m_{\max} \qquad (3')$$

$$\sum_{j=E_{kr}}^{L_{kr}} j \cdot x_{kjr} \leq \sum_{j=E_{ir}}^{L_{ir}} j \cdot x_{ijr} \qquad \forall r, \forall i, \forall k \in PD_{ir} \qquad (4)$$

$$\sum_{j=E_{1r}}^{L_{1r}} x_{1jr} \leq \sum_{j=E_{ir}}^{L_{ir}} x_{ijr} \qquad \forall r;\ i = 2,...,n \qquad (5)$$

$$x_{ijr} \in \{0,1\} \qquad \forall i, \forall r, \forall j \in [E_{ir}, L_{ir}] \qquad (6)$$

$$y_j \in \{0,1\} \qquad j = m_{\min}+1,...,m_{\max} \qquad (7)$$

The objective function (1) minimizes the sum of the ordered numbers related to the used workstations that are greater than the lower bound m_{min} (thus, the number of workstations is also minimized). Constraints (2) guarantee that every task i is assigned to one and only one workstation and to one and only one route. Constraints (3) and (3') ensure that the total task processing time assigned to workstation j does not exceed the upper bound on the cycle time. Constraints (4) impose the precedence conditions. The route uniqueness constraints (5), together with constraints (2), ensure that all tasks are assigned to the same route. Finally, (6) and (7) express the binary conditions of the variables.

If one analyzes the previous model, it can be observed that, if the precedence graph is connected, then constraints (5) can be removed, due to the fact that constraints (4) are sufficient to guarantee route uniqueness. Constraints (4) oblige all tasks to be assigned to the same route as their immediate predecessors. In a connected graph, all the tasks are related to one another, direct or indirectly, through their predecessors and successors; therefore, all the tasks are assigned to the same route. In any case, a connected graph can be obtained by defining an initial (or final) fictitious task for which the processing time is nil.

The mathematical formulation of ASALBP-1 can be easily modified for ASALBP-2 by using cycle time C_{max} as the variable that is to be minimized.

3.2. *Computational experiment*

As mentioned above, the 01ILP model is proposed to easily formalize the new ASALBP problem. Taking into account the ASALBP NP-hard condition, the model is not expected to be effective enough for real-world instances.

A brief computational experiment was carried out to prove the above prediction. The mathematical model for ASALBP-1 was implemented and several test instances were solved using the ® ILOG CPLEX 8.1 optimisation software on a PC Pentium 4, CPU 2.80 GHz with 512 Mb of RAM. The data sets used in the computational experiment were designed by incorporating various alternative assembly subgraphs into problem instances obtained from Scholl and Klein's homepage for assembly line balancing research (www.assembly-line-balancing.de). A total number of 60 problem instances were considered, using from 8 to 70 tasks and from 2 to 24 processing alternatives (called also routes); additionally, three different cycle time values, also based on the available benchmark datasets, were used for each problem instance.

The computational experiment showed (as it was expected) that optimal solutions can only be obtained and guaranteed in a reasonable amount of time for small sized problem instances, such as ASALBPs involving about 20 tasks and from 6 to 12 assembly routes. Notwithstanding some problems were optimally solved in a significantly low computing time, the time required by CPLEX to solve ASALB problems increases exponentially with the number of tasks and the number of processing alternatives that are available.

Let us make two comments concerning the modeling process. First, alternative model variations could be considered: a) the objective function (1) can be replaced by Minimize $z = \sum_{j=m_{\min}+1}^{m_{max}} y_j$ and constraints must be added to arrange the workstations consecutively, and b) the precedence relations (4) can be disaggregated. See, for example, Amen (2006), who mentions that a) and b) may perform better together when CPLEX is used. Second, the complexity of the model could be reduced by defining task-workstation assignment variables, regardless of the assembly route, for tasks not affected by subassemblies involving alternative subgraphs. However, due to the NP-hard nature of the ASALBP, both possibilities are considered to be impractical for optimally solving industrial problems (only

small or medium-sized problems could be solved optimally). Therefore, heuristic and metaheuristic procedures need to be developed to solve this new problem efficiently.

To better understand the potential benefits of using the *simultaneous* model, we compare the results of the proposed model with the results of a model designed to carry out subgraph selection and line balancing sequentially rather than simultaneously. The *sequential* line balancing model can be obtained by eliminating all route references and route constraints (5) from the *simultaneous* model presented in Section 3.1. Therefore, in the *sequential* scheme, the subgraphs are selected first—as usual (Senin *et al*., 2000), those with the smallest total processing time are chosen—and then the resulting assembly line is balanced. Both schemes were tested using an ASALBP instance based on Hann's benchmark problem with 53 tasks, which was adapted to consider 12 assembly routes (generated from the combination of the 7 available subgraphs): the *simultaneous* model only requires 8 workstations while the *sequential* model requires 9. Another possibility for the *sequential* solution is to solve a line balancing problem for each available assembly route. Considering the same instance of 53 tasks, 12 different assembly line balancing problems were solved: the *simultaneous* model takes 12.1 seconds to be solved, and the 12 resulting balancing problems take a total of 21.1 seconds. This shows the benefits of applying the proposed *simultaneous* model rather than a *sequential* scheme.

4. Conclusions and future research

In this paper, a new general assembly line balancing problem with practical relevance is presented, characterized and formulated: the Alternative Subgraphs Assembly Line Balancing Problem (ASALBP). A graphical representation scheme in the form of S-graphs is proposed that enables the alternative assembly subgraphs to be represented. Furthermore, numerical examples were used to illustrate the potential benefits of solving the two problems simultaneously, selecting an assembly subgraph for each part of the precedence S-graph that admits processing alternatives, and balancing the line. Finally, in order to formalize the ASALB problem, a binary linear programming mathematical model was developed, and its

performance was explored, being only useful in optimally solving small problem instances (as was to be expected, due to the NP-hard nature of the problem).

The core research work will involve designing and analyzing different heuristic and metaheuristic solution methods, to enable more realistic cases to be addressed. Another line of research consists in examining exact resolution procedures, such as dedicated branch-and-bound or column-generation algorithms. Finally, further research considers adding other features to the ASALB problem. Specifically, some authors have proposed considering the cost of the resources required to perform the tasks assigned to each workstation (see, e.g., Amen 2006).

Acknowledgments

Supported by the Spanish MCyT project DPI2004-03472, co-financed by FEDER. The authors are very grateful to Professor Albert Corominas (Technical University of Catalonia) for his valuable comments, which have helped to enhance this paper. Moreover, the authors wish to thank the anonymous reviewers for their invaluable insights, as we believe that these have considerably improved the paper.

References

Aase, G.R., Schniederjans, M.J. and Olson, J.R., U-OPT: an analysis of exact U-shaped line balancing procedures. *International Journal of Production Research*, 2003, **41**, 4185-4210.

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

Andrés, C., Miralles, C. and Pastor, R., Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, 2006 (In Press, Corrected Proof, Available online 15 November 2006)

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

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

Bukchin, J. and Tzur, M., Design of flexible assembly line minimize equipment cost. *IIE Transactions*, 2000, **32**, 585-598.

Das, S.K. and Nagendra, P., Selection of routes in a flexible manufacturing facility. *International Journal of Production Economics*, 1997, **48**, 237-247.

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

Erel, E., Sabuncuoglu, I. and Sekerci, H., Stochastic assembly line balancing using beam search. *International Journal of Production Research*, 2005, **43**, 1411-1426.

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

Gungor, A. and Gupta, S.M., An evaluation methodology for disassembly processes. *Computers & Industrial Engineering*, 1997, **33**, 1-4.

Lambert, A.J.D., Generation of assembly graphs by systematic analysis of assembly structures. *European Journal of Operational Research*, 2006, **168**, 932-951.

Pastor, R., Andrés, C., Durán, A. and Pérez, M., Tabu search algorithms for an industrial multi-product and multi-objective assembly line balancing problem, with reduction of the task dispersion. *Journal of the Operational Research Society*, 2002, **53**, 1317-1323.

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

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

Pinto, P.A., Dannenbring, D.G. and Khumawala, B.M., Assembly line balancing with processing alternatives: an application. *Management Science*, 1983, **29**, 817-830.

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

Scholl, A., *Balancing and sequencing of assembly lines*, 1999 (Physica-Verlag Heidelberg: Germany, 2nd edition)

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

Senin, N., Groppetti, R. and Wallace, D., Concurrent assembly planning with genetic algorithms. *Robotics and Computer Integrated Manufacturing*, 2000, **16**, 65-72.

Talbot, F., Patterson, J.H. and Gehrlein, W.V., A comparative evaluation of heuristic line balancing techniques. *Management Science*, 1986, **32**, 431-453.

Vilarinho, P.M. and Simaria, A.S., ANTBAL: an ant colony optimization algorithm for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research*, 2006, **44**, 291-303.

Wee, T.S. and Magazine, M.J., Assembly line balancing as generalized bin packing. *Operations Research Letters*, 1982, **1**, 56-58