UNIVERSITAT POLITÈCNICA
DE CATALUNYA

ASALBP: the Alternative Subgraphs Assembly Line Balancing Problem

Liliana Capacho Betancourt, Rafael Pastor Moreno

IOC-DT-P-2005-5
Gener 2005

Institut d'Organització i Control
de Sistemes Industrials

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*

Abstract

The classic assembly line balancing problem basically consists in assigning a set of tasks to a group of workstations while maintaining the tasks' precedence relations. Habitually, such relations are represented by a predetermined precedence graph. However, a product's assembly process may admit, for one or more of its parts, alternative precedence subgraphs. This may be due to the fact that the processing times for some tasks are dependent on their processing sequence, or because there are alternatives for such a subassembly. 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, which sometimes imposes additional constraints other than technological ones. In this paper we present, characterize and formulate a new general assembly line balancing problem: the Alternative Subgraphs Assembly Line Balancing Problem (ASALBP). Its novel characteristic is that it considers the possibility of having alternative assembly subgraphs, in which the processing times and/or the precedence relations of certain tasks are 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 (i.e., assigning the tasks to the workstations). The potentially positive effects of this on the solution of the problem are shown here in a numerical example. Finally, the mathematical programming model developed is described and the results of a brief computational experiment are presented.

***Keywords*:** assembly line balancing.

1. Introduction

Basically, the Assembly Line Balancing Problem (ALBP) consists in assigning a set of 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 and a given efficiency measure is optimized (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 very simple (and therefore restricted) problems (see Baybars for the assumptions of the simple case). 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) or Becker and Scholl (2004a, 2004b), 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,

[†] Supported by the Spanish MCyT project DPI2004-03472, co-financed by FEDER.
[*] 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

buffered or parallel lines; incompatibility between tasks; stochastic processing times; or equipment selection.

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) and Becker and Scholl (2004a, 2004b).

In this paper a new GALBP is presented, 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. Subgraphs may be required when the processing times of one or more tasks are dependent on their processing sequence; 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.

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; Section 3 presents a 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)

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).

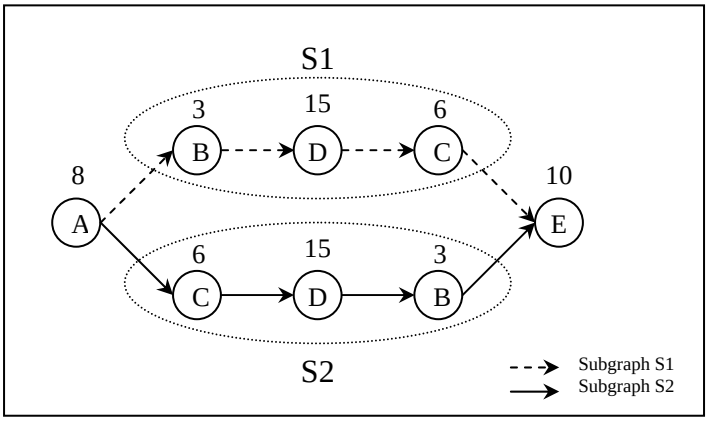

Figure 1. Alternative precedence subgraphs for the intermediate phase in the assembly of the motorbike

Using the standard diagramming representation, it is not possible to depict alternative precedence subgraphs. Figure 2 illustrates a potential way of representing precedence subgraphs S1 and S2, which is referred to by the authors as the precedence S-graph. Tasks F, K, L and Z (described later) are also considered in Figure 2.

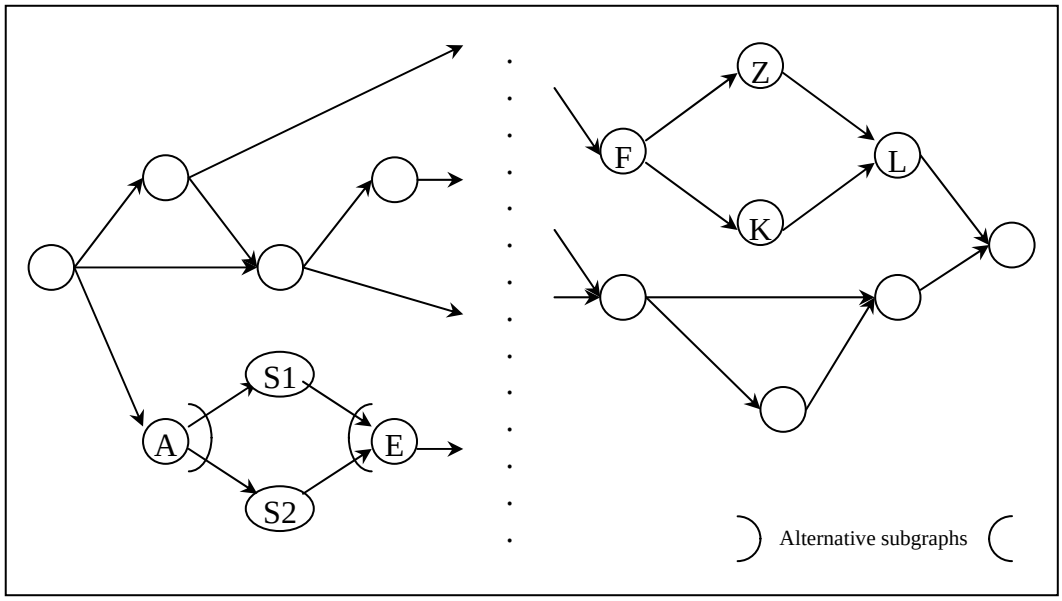

Figure 2. Precedence S-graph for the assembly process of the motorbike

As previously mentioned, 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, which consists of three main sets of tasks: Z, which is the decoration of the motorbike's fairing (it involves several tasks, 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 (see the upper part of Figure 2), 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 lasts 13 time units regardless of the assembly sequence and task L lasts 7 time units (as can be seen in Figure 3).

Therefore, in this case it is also possible to consider alternative precedence subgraphs: 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 4).

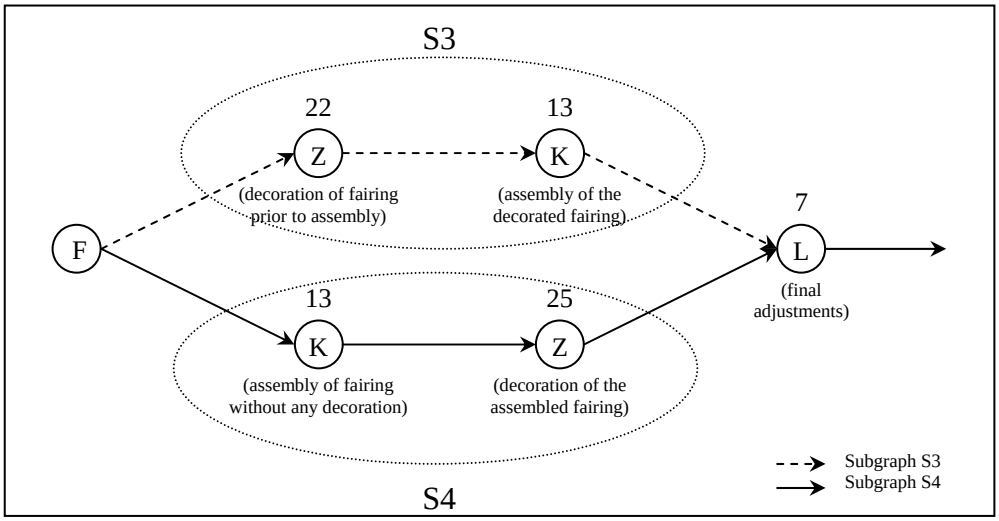

Figure 3. Alternative precedence subgraphs for the final assembly process of the motorbike

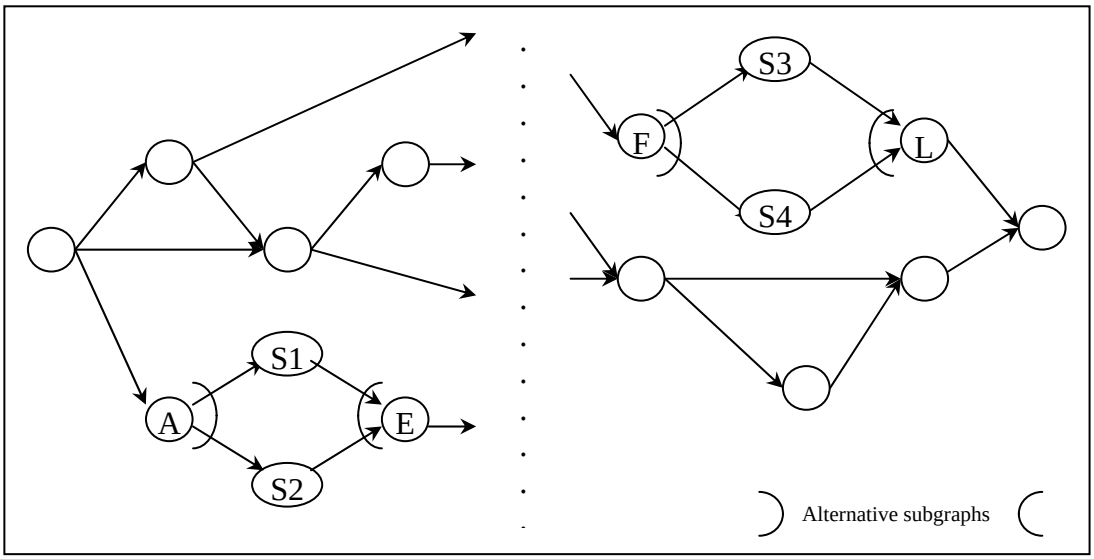

Figure 4. Precedence S-graph for the assembly process of the motorbike

Finally, 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.

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 optimizes 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 (2004) 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.

Considering alternative precedence subgraphs (precedence S-graph) imposes a higher level of difficulty on an assembly line balancing problem, regarding the simple case (SALBP), which is NP-hard. However, as real industrial processes may involve assembly alternatives, the possibility of considering alternative subgraphs not only enables more realistic instances of ALBP to be addressed, but may also favour an assignation of tasks to workstations in order to optimize a given objective. Regarding the conventional terminology (see for example 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.

2.1. Example 1: The Final Process of Assembling a Motorbike

- *Data for Example 1*

The final process of assembling a motorbike, which involves decorating the motorbike's fairing and assembling the fairing on the motorbike (see Figure 3) is again considered. Additionally, the task of decorating the fairing (Z) has been further divided into four tasks (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

Figure 5 shows, for each disaggregated subgraph (also including task L), a precedence graph. Each of these precedence graphs contains nodes for all the tasks and arcs indicating their immediate precedence relations; task processing times are represented as node weights.

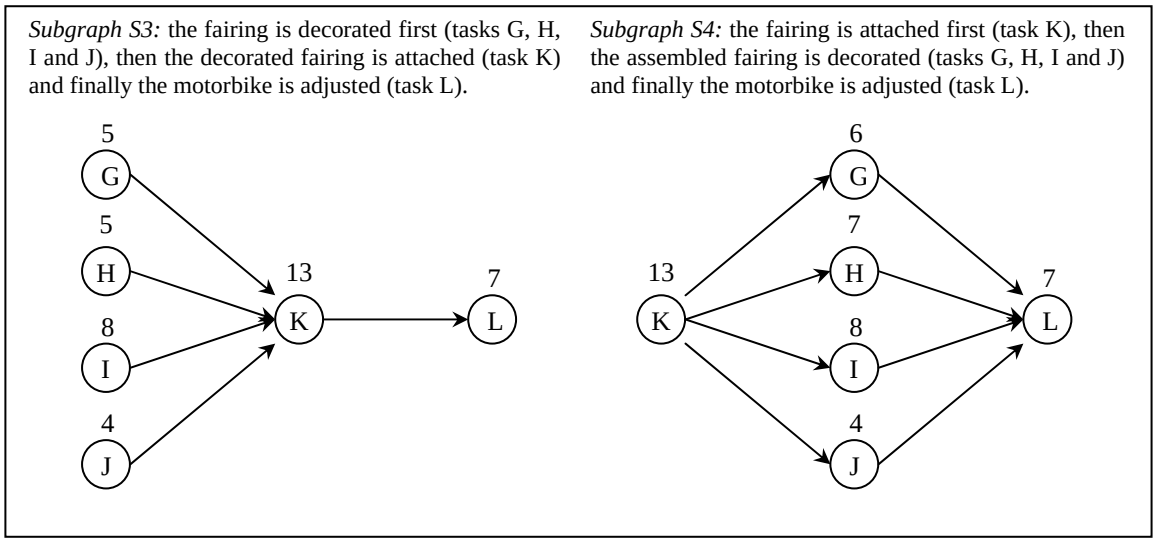

Figure 5. Precedence graph of the alternative subgraphs of Example 1

As can be seen in Table 1 and Figure 5, 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.

- *Considering an ASALBP-1 with an upper bound on a cycle time of 17*

By balancing each of the two resulting problems optimally, one for each alternative subgraph, the solutions presented in Table 2 are obtained. 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.

- *Considering an ASALBP-2 with 3 workstations*

By optimally balancing the problem for each alternative subgraph and aiming to minimize the cycle time given the number of workstations, the following results are obtained:

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.

2.2. 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.

2.3. 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 4). 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, there are four precedence graphs when subgraphs S1-S3, S1-S4, S2-S3 and S2-S4 are respectively 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, and aiming at solving it optimally, a binary linear program (BLP) has been developed (Appendix A describes the BLP model). In theory, the BLP is capable of solving the problem satisfactorily; in practice, mixed integer linear programming is somewhat impractical for real-sized problems (as was to be expected).

A brief computational experiment was carried out to prove the above statement. The mathematical model for ASALBP-1 was implemented and several test instances were solved using the ILOG CPLEX 8.1 optimization software on a PC Pentium 4, CPU 2.80 GHz with 512 Mb of RAM. The data sets used were designed by incorporating various alternative assembly subgraphs to problem instances obtained from the homepage for assembly line balancing research (www.assembly-line-balancing.de). The computational experiment showed that optimal solutions can only be obtained and guaranteed in a reasonable amount of time for small problem instances, such as ASALBPs involving about 20 tasks and from 6 to 10 assembly routes.

4. Conclusions and Future Research

In this paper, a new general assembly line balancing problem 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 several test problems were used to explore the model's performance, which are only useful in optimally solving small problem instances (as was to be expected, due to the NP-hard nature of the problem).

If the model is analyzed, it can be observed that it is possible to define task-workstation assignation variables, independently of the assembly route, for those tasks that are not affected by subassemblies with alternative subgraphs (this reduces the size of the mathematical program). Future research work may be focused on analysing the efficiency of considering such a possibility within the model; however, it could be expected that, in this case also, only small or medium-sized problems could be solved optimally. Therefore, the core research work will involve designing and analyzing different heuristic and metaheuristic solution methods, to enable more realistic cases to be addressed.

Acknowledgments

The authors would like to thank Dr. Albert Corominas (a Professor at the Technical University of Catalonia) for his valuable comments, which, without a doubt, have helped to enhance this paper.

References

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

Becker, C. and Scholl, A. (2004a). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research* (in press, corrected proof).

Becker, C. and Scholl, A. (2004b). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research* (in press, corrected proof).

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

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

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

Lambert, A. (2004). Generation of assembly graphs by systematic analysis of assembly structures. *European Journal of Operational Research* (in press).

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

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

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

Senin, N., Groppetti, R. and Wallace, D. (2000). Concurrent assembly planning with genetic algorithms. *Robotics and computer integrated Manufacturing*, 16, 65-72.

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

Appendix A. A Mathematical Programming Model for the ASALBP

The problem can be modelled as a binary linear program.

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 ($i = 1,\ldots,n;\ r = 1,\ldots,nr$) (see, for example, Scholl 1999 for details of how to calculate E_i and L_i)
  - 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 ($\forall i,\ \forall r,\ \forall j \in [E_{ir}, L_{ir}]$)

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

- *Model:*

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

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

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

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

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

$$\sum_{j=E_{1r}}^{L_{1r}} x_{1jr} \le \sum_{j=E_{ir}}^{L_{ir}} x_{ijr} \qquad \forall r;\ i = 2,\ldots,n \qquad [5]$$

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

$$y_j \in \{0,1\} \qquad j = m_{\min}+1,\ldots,m_{\max} \qquad [7]$$

The objective function [1] consists in minimizing the number of workstations. Constraints [2] guarantee that every task *i* is assigned to one and only one workstation. 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. Route uniqueness constraints [5] ensure that all the 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.