15th Triennial World Congress, Barcelona, Spain

GENERATION OF ASSEMBLY GRAPHS BY SYSTEMATIC ANALYSIS OF ASSEMBLY STRUCTURES

A.J.D. Lambert

Eindhoven University of Technology
P.O.Box 513 5600 MB Eindhoven, The Netherlands
a.j.d.lambert@tm.tue.nl

Abstract: Aimed at detection of parallel assembly modes, a method for assembly graph construction is described that is based on the systematic generation and analysis of precedence relations and their application to subassembly detection. This method dramatically reduces the search space for detailed analysis and avoids the use of precedence graphs. *Copyright © 2002 IFAC*

Keywords: Assembly, Boolean algebra, Combinatorial algebra, Disassembly, Graphs, Optimization, Production control, Sequences, Scheduling algorithms.

1. INTRODUCTION

Despite of the essential differences between assembly and disassembly processes, there have always existed strong ties between the analyses of both processes. If planning and sequencing are involved, different levels of aggregation should be considered: (1) Motion planning, (2) Assembly sequence planning and (3) Assembly task planning and sequencing. Motion planning is component oriented, assembly sequence planning is product oriented, and assembly task planning is process oriented, i.e. it is considered in an industrial engineering environment, including logistics. The approach (1) is beyond the framework of the present paper. The approaches (2) and (3) are less detailed compared with the approach (1). Both are closely intertwined, and should be considered as the basis for more detailed studies.

The assembly line balancing problem is closely related to approach (3). It should be stressed that one can not speak of "the" balancing problem, as there exists a class of various approaches, depending on the context in which the study has to be carried out (product design, process design, etc.), and external circumstances (product variety, degree of uncertainty etc.). A lot of papers on assembly line balancing problems have appeared. In textbooks such as (Scholl, 1999) an extensive review of balancing problems is presented. The assembly line balancing problem is defined here as "*the decision problem of optimally partitioning (balancing) the assembly work among the work stations*" (p. 5), which presupposes an existing assembly line consisting of a linear configuration of workstations. Many variations and extensions to this basic theme are relevant to practice and have been studied in so far.

Usually, balancing problems start with establishing a list of tasks accompanied by a time estimate of accomplishing each task and, subsequently, combining the tasks such that the assembly is completed within the shortest time, with a maximum use of the available capacity. As this type of problems originally referred to manual operations, time and costs were considered as directly proportional to each other. The introduction of automation and robotics has dramatically changed this concept.

2. PRECEDENCE GRAPHS

One of the basic tools in formulating and solving balancing problems is a diagram that graphically represents the different possible sequences of tasks that have to be accomplished. Because definite tasks have to be performed prior to the accomplishment of other tasks, these diagrams are subjected to precedence relations, and are called *precedence diagrams* or *precedence graphs*. These diagrams are the graphical representation of a set of precedence constraints or *precedence relations*. Precedence graphs were originally used for analyzing existing assembly lines that were applied for assembling existing products. This restricted the decision space to the resequencing of some tasks. One of the early descriptions of such diagrams is described by Prenting and Battaglin (1964). There, the use of *subassembly lines* has also been discussed. These enable the simultaneous performance of multiple assembly tasks, also called parallel assembly. This simplifies the problem because it is decomposed, and it might speed up the cycle time of the assembly line. The precedence graphs that are at the basis of this

method can be analyzed with basic tools in network analysis, such as Critical Path Method (CPM), and Project Evaluation and Review Technique (PERT).

Gradually, a more technical approach became necessary, because of the introduction of information technology in many domains, in the replacement of manual labor by robots, and in the increasing complexity of assembly processes. Apart from this, new models of management and operation were applied, with a need for mixed-model assembly (simultaneous manufacture of various items of a product family) and even multi-model assembly.

This technical approach was also required for design purposes. In order to design a product simultaneously with its assembly line, the consequence of product modifications on the assembly process had to be incorporated. Several authors introduced this design for assembly (DFA) concept. The evolution towards technically oriented precedence graphs is seen in the early literature on assembly sequencing, see Boothroyd *et al*. (1982); Miller and Stockman (1990). Here, the starting point was an assembly drawing of a definite product such as a power plug. Tasks corresponded there essentially with the attachment of a definite component or subassembly. An initial task (load the initial component) and a final task (remove the completed assembly) were added.

The precedence relations in those examples were restricted to simple AND relations. An AND relation means that a definite component or subassembly can be mounted only if, e.g., two other components (A AND B) are mounted beforehand. It should be stressed that reality is much more complicated than that, which puts severe restrictions on the use of precedence graphs. Usually, one has to deal with combinations of AND and OR relations, which are called complex AND/OR relations. These will be discussed in the sequel. Various authors have investigated and refined the construction of precedence graph. Heuristics-based precedence graphs were used by, e.g., Murayama *et al*. (1994, 2001) and Danloy et al. (1999). Application of precedence graphs to the assembly of product families can be found in (Fouda et al., 2001ab). A formal treatment of precedence graphs is in (Mînzu *et al*., 1999). Further analytical work on precedence graphs, including their application to disassembly line balancing, is described by Moore *et al*. (1998) and Güngör and Gupta (2001).

On the basis of the ideas of precedence graphs, the complete literature on assembly sequencing has been founded, starting with Bourjault's work (Bourjault, 1984) and the papers by De Fazio and Whitney, (1987), Baldwin, *et al*., (1991), and Homem de Mello and Sanderson (1990, 1991). In these papers, the problem of deriving all possible assembly sequences was dealt with. De Fazio, Whitney, and Baldwin *et al*. were focused on automatically deriving the precedence relations from a set of queries put to the designer. They used a *connection state* representation for depicting the sequences. Homem de Mello and Sanderson advocated a novel graphical presentation method, the *AND/OR (hyper-) graph* which resulted from considering the disassembly process as reverse assembly. Here the unit tasks or actions corresponded to the transition of a parent subassembly into a pair of child subassemblies. In the reverse, assembly actions are considered as the addition of two parent subassemblies to one child subassembly. In the AND/OR graphs, each action is represented by a set of two joined arcs (hyperarcs), pointing from the parent to both the children in case of disassembly graphs. Nodes correspond to subassemblies here. The construction of these graphs needs the set of precedence relations and the set of possible subassemblies as a prerequisite. The AND/OR graph can concisely represent all possible assembly sequences. Each of these sequences is represented by a subgraph of the AND/OR graph. Once the graph has been established and the costs of each action are estimated, the optimum sequence can be determined by applying mixed integer programming (Lambert, 1999, 2001; Lambert and Gupta, 2002). With respect to the state representation, the AND/OR graph notation is more compact as the number of nodes is reduced.

In the conventional approach, disassembly operations are described by cut-sets, i.e. a set of connections that has to be disestablished in order to separate a parent subassembly in two child subassemblies. The method that is described in this paper is on the full basis of subassemblies with the detachment of a part rather than the disestablishment of a connection as a basic operation. The principal advantage is that the additional complexity, which arises from the requirement of simultaneous disestablishment of multiple connections, is avoided. In the connection-oriented method, the precedence relations are therefore complex and the derivation requires dedicated software (Baldwin *et al*., 1991). It is demonstrated here, that a subassembly oriented method is applicable to a variety of problems, with a minimum of queries for visual inspection of the assembly drawing, and generates the precedence relations in a quite natural way.

Precedence graph are not required, because the precedence relations are implicitly present in the assembly graph. This will be explained in the following section.

Although the study of task precedence graphs is at the origin of assembly sequence determination, the methods that have been developed from an assembly sequencing point of view are in turn useful tools for assembly task planning. The subassembly oriented method that is described in this paper, can be used for quickly detecting opportunities for parallel assembly.

Parallel assembly is widely applied in industry, as it enables the simultaneous assembly of modules that are, subsequently, combined to form the final product. In the example of figure 3, e.g., the simultaneous assembly of the modules ABCDEF and GHJKL appears feasible. In a final step, these are joined.

3. ASSEMBLY GRAPHS

Assembly graphs can be derived from the equivalent disassembly graphs. The method for their establishment is illustrated by a basic case.

Consider an assembly that consists of three components, which is represented by the set {A, B, C}, see figure 1a. The different actions are indicated here by the figures 1 through 12. The number of components is denoted by *N*, thus $N = 3$. The maximum number of subassemblies is $2^N - 1$. In this case there are maximally 7 subassemblies possible, viz. ABC, AB, AC, BC, A, B, C. However, because these subassemblies should meet definite criteria, the actual number of subassemblies is restricted in realistic cases. From a disassembly point of view, the principal criteria are: (1) *coherence*, which means that the subassembly should be connected, and (2) *detachability*, which means that the subassembly should be obtained by disassembly operations only, without reassembly. By visual inspection, it is observed that subassembly AB cannot be obtained from ABC by disassembly operations. Coherence can be determined by using a connection diagram, see figure 1b.

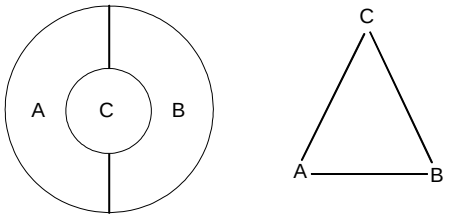

Figure 1. Basic example of an assembly. (a) Assembly structure, (b) Its connection diagram.

In this example, the assembly is *strongly connected*, which means that every part is connected with all the remaining parts and no incoherent subassemblies exist. In most of the more complex assemblies, the situation of strong connection is not obtained. The assembly graph can be derived from the connection diagram, which is presented in figure 2 in its most general form, without regarding detachability.

It reveals 6 possible assembly sequences. If an undetachable subassembly appears in an assembly sequence, this sequence is invalid. In figure 2, it is evident that AB is undetachable. This implies that, once AB has been assembled, the process cannot be continued towards final state, because the action 10 can not performed. This means that the assembly sequences 1,4,10 and 2,6,10 are discarded. The actions 4 and 6, although possible ones, should also be discarded for they do not make part of a valid assembly sequence.

Although the rejection of AB can be justified at a first glance in this simple example, this is no longer applicable in more complex cases.

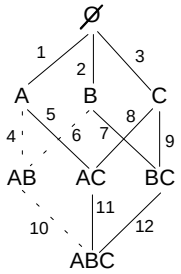

Figure 2. Assembly graph and actions belonging to the basic example. The dotted lines are the infeasible actions.

Therefore, a systematic method for subassembly rejection, which is based on precedence relations, will be presented. This approach is an extension of the work that has been performed by (Huang and Lee, 1989) and (Rajan and Nof, 1996). According to this method, one analyses, subsequently, the different components. For each component, one investigates whether a combination of two surrounding parts exists, that impedes the detachment of the respective component. In this example, there is only one precedence relation, namely:

$$R_A \text{ or } R_B \rightarrow R_C \qquad (1)$$

This expression means that component A or component B should be detached prior to the detachment of component C.

Inversion of (1) results in an expression that discards definite subassemblies on the criterion of undetachability, a so-called sieve. Here it reads:

$$(A \text{ and } B) \text{ and not } C \qquad (2)$$

This means that all subassemblies that contain A and B and that do not contain C are discarded.

It should be stressed that in more complex cases this method is a powerful instrument in reducing the amount of valid subassemblies.

This method does not fully guarantee the completeness of the set of conditions, so it results in a preliminary set of necessary conditions, which implies that no potentially valid subassemblies are discarded. More rigorous methods that include the search for appropriate combinations of three and more surrounding components, guarantee completeness but at the cost of

a more complex search and the generation of many redundant selection rules.

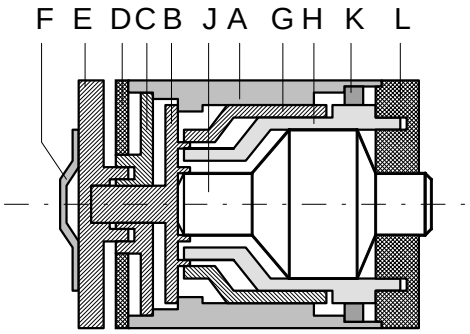

Figure 3. The Assembly for Industry example (De Fazio and Whitney, 1987).

It is reported that, in realistic subassemblies, the number of valid subassemblies strongly reduces with respect to the number of combinatorially possible ones. For the Assembly for Industry, depicted in figure 3, (De Fazio and Whitney, 1987), consisting of 11 components, only 83 subassemblies are valid, which is a reduction with a factor of 0.04. A somewhat more complex 2D-example that has been taken from (Chen, *et al.*, 1997) has only 135 valid subassemblies. This results in a reduction with the factor 0.008, see figure 4.

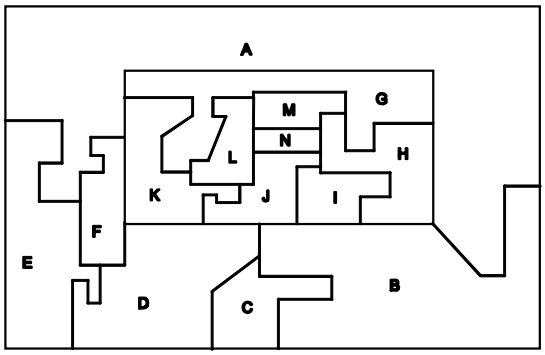

Figure 4. The 2D-example taken from (Chen *et al.*, 1997).

Completeness is not obtained in the general case without the simultaneous introduction of redundant constraints. In realistic cases such as that of figure 3 and 4, however, the set of precedence equations appears complete, which can be justified by visual inspection of the reduced set of selected subassemblies. If it happens to be that a subassembly is erroneously considered feasible, this requires an additional constraint to the sieve. In the examples of figure 3 and 4, no infeasible subassemblies were selected.

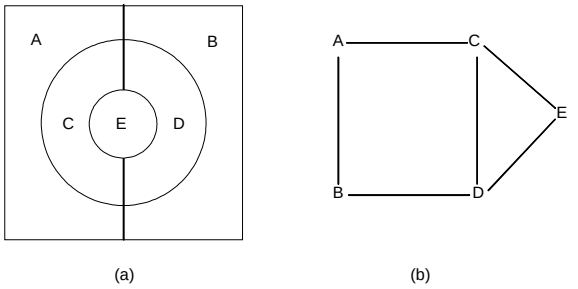

Figure 5. Demonstrating subassembly detection and parallel assembly. (a) Assembly, (b) Connection diagram

By applying this considerable reduction beforehand, the testing of the subassemblies on other criteria that are required for assembly, such as stability, can be performed more efficiently because a restricted amount of subassemblies should be considered. Such a test is usually elaborate, for it presupposes detailed information on geometry, the nature of connections (mating or the presence of different types of fasteners), fixtures, tools, tolerances etc.

5. PARALLEL ASSEMBLY

Subassembly detection will be illustrated via the assembly {A, B, C, D, E} of figure 5a. Its connection diagram is presented in figure 5b. The assembly has 31 combinatorially possible subassemblies, which results, after exclusion of the components and the complete assembly, in 25 nontrivial ones. From the connection diagram, the sieve can be derived that discards incoherent subassemblies. This proceeds by considering the neighboring components of every component. It is clear, for instance, that a subassembly that contains A, and that contains neither B nor C, cannot be coherent. Consequently applying this method results in the sieve that discards all subassemblies that contain:

A and not (B or C)
B and not (A or D)
C and not (A or D or E) (3)
D and not (B or C or E)
E and not (C or D)

Additionally, the precedence relations should be formulated. There are three of them, because the components A and B can each be removed without blocking by other parts, if the appropriate direction of motion is chosen. Because component C is surrounded by the components A, D, and E, the precedence relation that is related to C reads:

$$(R_A \text{ or } R_D) \rightarrow R_C$$

Similarly, one can formulate precedence relations for D and E, namely:

$(R_B \text{ or } R_C) \rightarrow R_D$
$(R_C \text{ or } R_D) \rightarrow R_E$

From this, the sieve is derived by inversion, which implies that all subassemblies are discarded that contain:

(A and D) and not C
(B and C) and not D (4a)
(C and D) and not E

Applying the sieves (3) and (4a) to the combinatorially possible subassemblies, discards 15 of the 25 combinations. Those that are left are:
AB, AC, BD, CE, DE, ACE, BDE, CDE, ACDE, BCDE.
Visual inspection reveals that AB is erroneously selected, because the set of surrounding components was too restricted. In the case of component C, B should be added to the surrounding components and the set of precedence relations thus is extended to:

$(R_A \text{ or } R_D) \text{ and } (R_A \text{ or } R_B) \rightarrow R_C$
$(R_B \text{ or } R_C) \text{ and } (R_B \text{ or } R_A) \rightarrow R_D$ (5)
$(R_C \text{ or } R_D) \rightarrow R_E$

Such as is visible in, e.g., (5), the "natural" shape of a precedence relation is a complex AND/OR relationship. This strongly complicates the graphical presentation of these relations in a precedence graph. The use of such diagrams is partly based on historical arguments for it is essentially a modification of a precedence graph for work elements.
From (5), the following sieve is obtained:

(A and D) or (A and B) and not C
(B and C) or (A and B) and not D (4b)
(C and D) and not E

Consequently, AB is discarded too and the sieve that consists of (3) and (4b) discards all incoherent and undetachable subassemblies. Only 9 out of 25 subassemblies remain, viz.:
AC, BD, CE, DE, ACE, BDE, CDE, ACDE, BCDE.
From this, the assembly graph can be derived. It is presented here in two parts.
The transitions that are obtained by subsequent addition of single components are depicted in figure 6a. Here only one significant arc of each hyperarc is drawn. In figure 6b the possibilities of parallel assembly are presented. Here the full hyperarcs are visualized. The parent subassemblies should be both feasible and complementary with respect to the child subassembly. Because of the strong reduction in the amount of subassemblies, this still more reduces the modes of parallel assembly. The combination of the figures 6a and 6b reveals the full assembly graph.
Further criteria for assembly, e.g. stability, can now be introduced. If, e.g. ACE is instable, it should not appear in the graph with the consequence that the related actions and subassemblies also are removed. In the example, this results in removal of the actions 14, 16, and 20. The subassembly AC is not removed because it also appears in parallel assembly as a parent of ACDE in action 26. One can start, e.g., with the assembly of AC and DE simultaneously, combine them to ACDE, and add B. An alternative parallel sequence is as follows: attach B to DE and next combine BDE and AC to the final assembly according to action 29. In more complex cases, discarding a subassembly on a detailed criterion might result in discarding a chain of subassemblies.

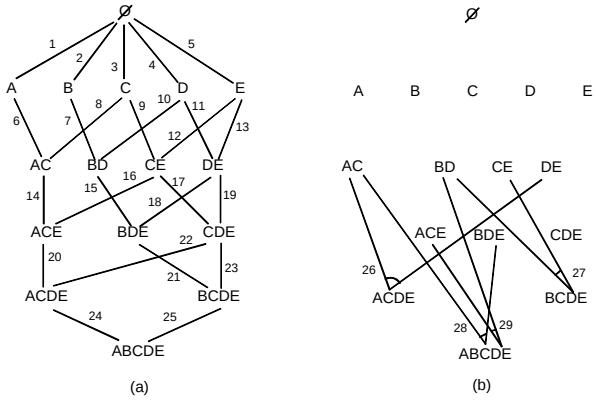

Figure 6. The assembly graph. (a) Simple assembly, (b) Parallel assembly.

In balancing of assembly lines, the possibility of carrying out multiple tasks in parallel is an interesting option. Such possibilities follow straight from the assembly graphs.
A further application of this method is in cases in which *modules* (functional units consisting of a definite subassembly) should appear which might be required, e.g., for testing purposes. If CDE were such a module, the subassemblies that contain elements of CDE but that do not contain CDE are discarded. This refers to ACE, BDE, AC, and BD. Only the actions 3, 4, 5, 9, 11, 12, 13, 17,19,22, 23, 24, 25 are permitted and no parallel assembly appears possible.

6. CONCLUSIONS

This paper demonstrates that the use of assembly graphs is a valuable tool in the analysis of assembly sequences and acts as an alternative to the use of precedence graphs. It shows that assembly graphs can be derived from component-oriented rather than connection-based precedence relations via subassembly detection. This strongly reduces the search space of subassemblies and actions that should be analyzed in more detail. Particularly, the complete set of parallel assembly operations is obtained in a logical way from this analysis. This is important in the design of assembly processes that permit parallel operations.

Acknowledgement

The research presented in this paper makes up part of the research on re-use in the context of the TMR project REVersed LOGistics sponsored by the European Union (ERB 4061 PL 97-650) in which take part the Aristoteles University of Thessaloniki (GR), Erasmus University Rotterdam (NL), INSEAD (F), the Otto-von-Guericke-Universität Magdeburg (D), Technische Universiteit Eindhoven (NL) and the University of Piraeus (GR).

REFERENCES

Baldwin D.F., T.E. Abell, M.C.M. Lui, T.L. De Fazio and D.E. Whitney D.E. (1991) An integrated computer aid for generating and evaluating assembly sequences for mechanical products, *IEEE Transactions on Robotics and Automation*, **7**(1), 78-94.

Boothroyd, G., C. Pol and L.E. Murch (1982). *Automatic assembly,* Manufacturing Engineering and Materials Processing, Vol. 6, Marcel Dekker Inc., New York. Chapter 9.

Bourjault A. (1984). *Contribution à une approche méthodologique de l'assemblage automatisé: elaboration automatique des séquences opératoires*, Ph.-D. Thesis, Université de Franche-Comté, Besançon, France (in French)

Chen S.F., J.H. Oliver, S.Y. Chou, and L.L. Chen (1997). Parallel disassembly by onion peeling, *Journal of Mechanical Design*, **119**(2), 267-274.

Danloy, J., F. Petit, A. Leroy, P. De Lit, and B. Rekiek (1999). A pragmatic approach for precedence graph generation, *Proceedings of 1999 IEEE International Symposium on Assembly and Task Planning*, 387-392.

De Fazio T.L. and D.E. Whitney (1987). Simplified generation of all mechanical assembly sequences, *IEEE Journal of Robotics and Automation*, **RA-3**(6), 640-658.

Fouda, P., P. De Lit, J. Danloy, B. Rekiek, T. L'Eglise, and A. Delchambre (2001a). A heuristic to generate a precedence graph between components for a product family, *Proceedings of 2001 IEEE International Symposium on Assembly and Task Planning*, 43-48.

Fouda, P., P. De Lit, B. Rekiek, and A. Delchambre (2001b). Generation of precedence graphs for a product family using a disassembly approach, *Proceedings of 2001 IEEE International Symposium on Assembly and Task Planning*, 226-231.

Güngör A. and S.M. Gupta (2001), A solution approach to the disassembly line balancing problem in the presence of task failures, *International Journal of Production Research*, **39**(7), 1427-1467.

Homem de Mello L.S. and A.C. Sanderson (1990). AND/OR graph representation of assembly plans, *IEEE Transactions on Robotics and Automation*, **6**(2), 188-199.

Homem de Mello L.S. and A.C. Sanderson (1991). A correct and complete algorithm for the generation of mechanical assembly sequences, *IEEE Transactions on Robotics and Automation*, **7**(2), 228-240.

Huang, Y.F. and C.S.G. Lee (1989). Precedence knowledge in feature mating operation assembly planning, *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, 216-221.

Lambert, A.J.D. (1999). Linear programming in disassembly/clustering sequence generation. *Computers and Industrial Engineering*, **36**(4), 723-738.

Lambert, A.J.D. (2001). Automatic determination of transition matrices in optimal disassembly sequence generation, *Proceedings of 2001 IEEE International Symposium on Assembly and Task Planning*, 220-225.

Lambert, A.J.D., and Gupta, S.M., (2002). Demand-driven disassembly optimisation for electronic consumer goods, *Journal of Electronics Manufacturing*, **11**(2), forthcoming.

Miller, J.M. and G.C. Stockman(1990). Precedence constraints and tasks: how many task orderings ?, *Proceedings of the 1990 IEEE International Conference on Systems Engineering*, 408-411.

Mînzu, V., A. Bratcu, and J.M. Henrioud (1999). Construction of the precedence graphs equivalent to a given set of assembly sequences, *Proceedings of 1999 IEEE International Symposium on Assembly and Task Planning*, 14-19.

Moore, K.E., A. Güngör, and S.M. Gupta (1998). Disassembly Petri net generation in the presence of XOR precedence relationships, *Proceedings of 1998 IEEE International Conference on Systems, Man, and Cybernetics*, 13-18.

Murayama, T., F. Oba, and S. Abe (1994). Assembly partitioning by genetic algorithm for generating assembly sequences efficiently, *Advancement of Intelligent Production, Publication no. 1 of The Japan Society for Precision Engineering*, 659-700.

Murayama T., F. Oba, S. Abe, and Y. Yamamichi (2001). Disassembly sequence generation using information entropy and heuristics for component replacement, *Proceedings of 2001 IEEE International Symposium on Assembly and Task Planning*, 208-213.

Prenting, T.O. and R.M. Battaglin (1964). The precedence diagram: a tool for analysis in assembly line balancing. *The Journal of Industrial Engineering*, **15**(4), 208-213.

Rajan, V.N. and S.Y. Nof (1996). Minimal precedence constraints for integrated assembly and execution planning *IEEE Transactions on Robotics and Automation*, **12**(2), 175-186.

Scholl, A. (1999). *Balancing and sequencing of assembly lines*, Physica-Verlag, Heidelberg (D).