On a Learning Precedence Graph Concept for the Automotive Industry

Hanne Klindworth, Christian Otto, Armin Scholl

09/2010

ISSN 1864-3108

Publisher:

Wirtschaftswissenschaftliche Fakultät
Friedrich-Schiller-Universität Jena
Carl-Zeiß-Str. 3, D-07743 Jena

www.jbe.uni-jena.de

Editor:

Prof. Dr. Hans-Walter Lorenz
h.w.lorenz@wiwi.uni-jena.de

Prof. Dr. Armin Scholl
armin.scholl@wiwi.uni-jena.de

www.jbe.uni-jena.de

On a learning precedence graph concept for the automotive industry

Hanne Klindworth[a], Christian Otto[a], Armin Scholl[a,*]

[a] Friedrich-Schiller-University of Jena, Chair of Management Science,

Carl-Zeiß-Straße 3, D-07743 Jena, Germany

* Corresponding author: phone +49-3641-943171, email: armin.scholl@uni-jena.de

Abstract

Assembly line balancing problems (ALBP) consist in assigning the total workload for manufacturing a product to stations of an assembly line as typically applied in automotive industry. The distribution of the tasks to the stations is due to restrictions which can be expressed in a precedence graph. However, automotive manufacturers usually do not know complete precedence graphs describing the production processes of their models. Unfortunately, the known approaches for graph generation are not suitable for the conditions in the automotive industry.

We describe a new graph generation approach that is based on learning from past production sequences and forms a sufficient precedence graph. This graph, indeed, restricts the ALBP instance but guarantees feasible line balances. Computational experiments indicate that the proposed procedure is able to approximate the real precedence graph sufficiently well to detect optimal or nearly optimal solutions for all instances of a benchmark data set. So, the new approach is applicable and effective and might be a major step to close the gap between theoretical line balancing research and practice of assembly line planning.

Keywords: assembly; line balancing; auto industry; manufacturing process; precedence graph; learning approach; decision support

1 Introduction

In today's automotive industry (and related industries manufacturing complex products, like appliance industry and electronic industry), flexible assembly lines are used in order to meet the varying customer demands. Each assembly line consists of (work) stations $k = 1, \dots, m$ established along a mechanical material handling equipment typically moving the workpieces down the line at a constant speed. The production process is subdivided into n tasks which are collected in (node) set $V = \{1, \dots, n\}$. Execution of task $j \in V$ takes task time t_j , which can be defined to be integral through an appropriate scaling of data. At each station k, a specific set (station load) $S_k \subset V$ of tasks is repeatedly executed with respect to a given cycle time c available per workcycle. The evolving decision problem of assigning tasks to stations in order to optimize a given objective function is known as the assembly line balancing problem (ALBP). Possible objectives include cost minimization, profit maximization and maximization of line efficiency (e.g. minimization of the number of stations given the cycle time). For a detailed problem statement as well as surveys and classification schemes for ALBP, we refer to Baybars (1986), Becker and Scholl (2006), and Boysen et al. (2007, 2008). An ALBP that considers specific conditions of the automotive industry is described in Becker and Scholl (2009).

The assignment of tasks to stations is restricted through technological and organizational conditions. In particular, precedence constraints have to be considered, which can be summarized and visualized in a precedence graph. The precedence graph is an acyclic digraph $G = (V, E)$ that contains a node for each task $j \in V$ and an arc $(i, j) \in E$ for each non-redundant precedence relation which requires that task $i \in V$ is finished before another task $j \in V$ can be started. The task times t_j are allocated as node weights. In a mixed-model setting, which is the standard in automotive industry, a joint precedence graph based on an average product model is utilized (cf. Boysen et al., 2009a). While the task times may change, the precedence relations are usually stable and independent of the current model-mix.

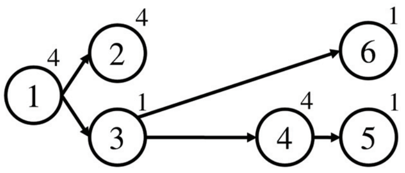

Figure 1. Example Precedence Graph

Figure 1 shows an example of a precedence graph with six tasks and task times between 1 and 4 time units. Task 1 has tasks 2 and 3 as direct successors as well as tasks 4, 5 and 6 as indirect successors, i.e., task 1 has to be completed before any other task can be started. For task 6 to be executed, task 1 (indirect predecessor) and task 3 (direct predecessor) have to be finished.

Besides precedence constraints, the precedence graph also shows (precedence) independencies of tasks. Task 4 and 6 exemplify this issue as they can be executed in either order. Two tasks i and j which are not related by precedence are called *independent* of each other. More generally, a subset of tasks, no pair of which is related by precedence, is called an *independent task set*.

Each solution of an ALBP instance is given by a precedence-feasible sequence π of tasks separated into subsets S_k forming the loads of stations $k = 1, \ldots, m$. These loads have to fulfill the cycle time constraints, i.e., the station time $t(S_k) = \sum_{j \in S_k} t_j$ of each station k must not exceed the cycle time c. Furthermore, these loads are assumed to be maximal, i.e., the next task in sequence π cannot be added without exceeding the cycle time. If existing, further constraints like assignment restrictions have to be met (cf., e.g., Scholl et al. 2010).

Since the approach presented in this paper is applicable to any type of ALBP, we use the simple ALBP with the objective of minimizing the number of stations given the cycle time (SALBP-1) as an example throughout the paper (cf. Baybars, 1986; Scholl and Becker, 2006).

As far as we know, no automotive manufacturer possesses complete precedence graphs for its production processes. Although line balancing experts would most likely be able to identify the relevant constraints, the huge manual input and the multitude of tasks (up to several thousands) prevent manufacturers from collecting and maintaining precedence graphs (cf. Ammer 1985, p. 17). Hence, most theoretical assembly line balancing concepts, which all require a known precedence graph, cannot be applied in practice up to now. Instead, planners, who are experts for parts of the production processes, balance lines by manually shifting tasks from one station to another. This is a very time-consuming and fault-prone job, which is solely driven by the experience and knowledge of planners. Case studies we performed at several real-world lines show that there is a great potential for improving balances, accelerating planning and making the planning process more independent of particular planners. Also, different major German car manufacturers, with which we co-operate, desire to have precedence graphs on hand as they are aware of the disadvantages of balancing lines without this important piece of information. In order to close this gap, we propose an approach to generate sufficient precedence graphs (predominantly) automatically by learning precedence constraints from feasible line balances (task sequences).

The remainder of the paper is organized as follows: In Section 2, we review related previous research work. The new concept of learning the precedence graph is presented in Section 3. Section 4 reports on computational experiments. Summary and conclusions are given in Section 5.

2 Analyzing previous work on generating precedence information

In the literature, we find some research work related to deriving precedence information. However, most researchers do not aim at constructing precedence graphs but to find feasible sequences.

Concepts, which deal with finding assembly sequences or precedence restrictions can be categorized in two main classes: On the one hand, there are manual and automated approaches that are intended to detect all feasible sequences. On the other hand, genetic algorithms and case-based reasoning procedures are applied to search for a few good sequences (see Table 1).

Detect all feasible sequences	Manual approaches	Precedence constraints and all feasible sequences are manually extracted by consulting experts and with the help of geometric product models.
	Automated approaches	Product models are evaluated widely automatically.
Detect a few feasible sequences	Genetic algorithms	Genetic algorithms are based on a few feasible sequences that are recombined by genetic selection, recombination and mutation.
	Case-based reasoning	Sequences for the same or similar subassemblies are recomposed to form the overall plan.

Table 1. Concepts to detect feasible sequences

The earliest approach to find feasible assembly sequences was introduced by Bourjault (1984). During his question-based procedure, an expert has to decide on the feasibility of assembly actions. However, as the number of liaisons between parts (connection points that correspond to mounting operations) rises, the number of questions grows exponentially. De Fazio and Whitney (1987) modified Bourjault's approach and reduced the number of questions to two times the number of liaisons. While both methods are based on the assembly of the product, Homen de Mello and Sanderson (1990, 1991) pioneered the disassembly approach. The basic idea is to enumerate all possibilities to disassemble a product by examining cut-sets of the liaison graph. The liaison graph is an undirected graph that contains parts as nodes, where two nodes are adjacent if they are connected in the assembly (by some mounting operations). Due to high error-proneness and the huge amount of input needed, manual approaches fail to work for highly complex products like automobiles. Therefore, researcher's interest turned to automated methods.

Ammer (1985) uses the subassembly structure of the multi-level bill of material for the outline of the precedence graph. Precedence constraints for each subassembly are automatically extracted with the help of CAD data and merged thereafter. Most automated methods to find assembly sequences are based on geometric reasoning and, thus, geometric product models (e.g. CAD models). For early automated methods based on collision analyses in direction of the orthogonal axes we refer to Romney et al. (1995) and Kaufman et al. (1996). Su (2009) expanded the collision analysis to 360° around the assembly.

There are certain issues that restrain the use of automated methods especially in the automotive industry: First of all, manufacturers do not necessarily possess geometric product models and often do not even have the CAD data. Furthermore, even if CAD models are available, soft components like clips and gasket rings cannot be handled (cf. Altemeier 2009, Chapter 3.2).

Since both manual and automated methods only work for relatively small assemblies, Bonneville et al. (1995) suggested the use of genetic algorithms. Feasibility of new generated individuals (i.e., sequences) is checked with the help of the liaison graph and expert knowledge. Accepted individuals are evaluated by means of a fitness function. This or other genetic methods, e.g. as proposed by Bonneville et al. (1995, Chapter 2.5) or Smith and Smith (2002), basically combine a (potentially successful) heuristic meta-strategy for solving the assembly line balancing problem with one of the (manual or automated) approaches for detecting or examining precedence relations. So, there is no methodological gain in finding the relations of the precedence graph.

Swaminathan and Barber (1996) included a case-based reasoning (CBR) approach into assembly planning. The basic CBR approach can be described as follows (cf. Riesbeck and Schank 1989, Chapter 2.1): A case-based reasoner solves new problems by matching and adjusting solutions used for similar problems in the past. In order to find feasible assembly sequences, Swaminathan and Barber (1996) divide the liaison graph into cyclic substructures and the case-based reasoner searches for a feasible sequence for each cycle. Dong et al. (2007) use subassemblies instead of cycles of the liaison graph. Criteria and the approach to find the best suitable subsequences are described in Su (2007, Chapter 3.2). Chen et al. (2006) developed the software "Body Build Advisor" that was used for the assembly of an automotive body.

Contrary to procedures, which manually search for assembly sequences, both genetic algorithms and case-based reasoning approaches also take older plans into account. However, a structured use of those methods requires extensive additional input.

Despite these recent developments in the literature, requirements on the precedence graph generation procedure put by automotive producers are still not satisfied by the available methods (see Table 2). These methods either need an extensive manual input, which manufacturers are not willing or able to provide, or refer to geometric product models, which are often not available. Furthermore, a precedence graph concept suited for the automotive industry must work for several hundreds or even thousands of tasks and have to incorporate all the options for assembly of specific car models; both cannot be guaranteed by the existing approaches.

Automobile manufacturers regularly re-balance their assembly lines both in the short term and in the long term (cf. Boysen et al. 2008, 2009b). Typically, actual customer orders are known in advance for about 4 to 6 weeks, which makes it possible to re-balance the line according to the current model-mix (share of different models and/or options in the sales plan). This will take place

	Literature	Automotive industry
1	Small number of parts and tasks	Large number of parts and tasks (several hundreds to some thousands of tasks)
2	Precedence graph from scratch (esp. manual methods)	Credited assembly plans (containing feasible sequences) for past assemblies available
3	Product is assembled in only one or a few variants (models)	Product is assembled in almost arbitrary number of models (up to several millions or even billions)
4	Geometric product models are available	Geometric product models are not or only partly available, many (simple) parts responsible for great portion of tasks are not contained in those models

Table 2. Differences between literature assumptions and conditions in the automotive industry

daily, weekly or monthly. Of course, adjustments introduced during re-balancing will not change the entire balance, but will consist of local re-arrangements (shifting some tasks to other stations). Long-run balancing takes place in case of new models or modifications in the production process. Many manufacturers hold yearly workshops, where experts re-balance the manufacturing lines. In order to internally charge production times and costs, all feasible plans (also called credited plans) are stored in the manufacturer's databases. This is a very valuable source of information as it implicitly contains the knowledge of planners about precedence relations (and other aspects of assembly conditions).

Recently, it has been a highly discussed topic by well known car-producers, on how to do assembly line balancing using these credited plans. A first promising step in this direction was done by Minzu et al. (1999), who noticed that the knowledge of all the feasible assembly sequences is equivalent to possessing the precedence graph. Further reflections of these ideas are summarized by Altemeier (2009, ch. 5.2), who also pointed out that the combination of several assembly sequences very often results in getting information on new feasible sequences.

We generalize these ideas and develop a learning concept in Section 3. The experimental results of our approach in Section 4 show that it is a very effective method which meets the outlined requirements of the automotive industry better than existing approaches.

3 Learning precedence graph concept

In this section, we describe the learning precedence graph concept for generating an adequate precedence graph from several known feasible (i.e., practicable) sequences (credited plans). Such a precedence graph allows for (heuristic) automated optimization of the assembly line.

3.1 Terms, definitions and basic idea

The learning precedence graph concept is based on three precedence graph types that are distinguished by their "fill level": target graph, maximum graph and minimum graph. As storage for these graphs an adjacency matrix is used. For a precedence graph with n tasks the *adjacency matrix* is an $(n \times n)$-matrix A with:

$$a_{ij} = \begin{cases} 1, & \text{if } i \text{ is a direct predecessor of } j \\ 0, & \text{otherwise} \end{cases} \quad \forall i,j \in V \tag{1}$$

For a precedence graph $G = (V, E)$ with adjacency matrix A, its *transitive closure* is defined as the precedence graph $G^T = (V, E^T)$ that contains in E^T an arc (i,j) from $i \in V$ to $j \in V - \{i\}$ if there exits at least one path from i to j in G. The corresponding adjacency matrix is called A^T.

The precedence graph that correctly describes the real production process is called **target graph** G. This is the graph, we try to approximate by our methods as closely as possible and necessary. The following two types of precedence graphs are utilized as approximations of the target graph.

The **maximum graph** $\bar{G} = (V, \bar{E})$ with its adjacency matrix $\bar{A}$ is a precedence graph that contains the same nodes (tasks) as the target graph and at least all the precedence constraints of the target graph, i.e., $E^T \subseteq \bar{E}^T$.

We propose a simple method to construct the maximum graph $\bar{G} = (V, \bar{E})$ based on a set of $p \geq 1$ precedence-feasible assembly sequences (permutations) $\pi_1, \pi_2, \dots, \pi_p$ of the *n* tasks available. All possible combinations of two tasks $i, j \in V$ are examined as follows: If there is one sequence π_x, in which task i is conducted before task j and another sequence π_y ($x, y \in \{1, \dots, p\}$ and $x \neq y$), in which i is executed after j, then i and j are *definitely independent* as they were already executed in both orders in the past, and $\overline{a}_{ij}^T$ (and, thus, also $\overline{a}_{ij}$) is set to 0. However, if i is executed before j in each of the sequences $\pi_1, \pi_2, \dots, \pi_p$, it is *assumed* that i is a predecessor of j and $\overline{a}_{ij}^T = 1$. Note that an assumed precedence relation (i,j) is not proven – it is just a potential constraint which could not be contradicted up to now. Possibly, there are (still unknown) feasible sequences in which j is executed prior to *i.*

Usually, the procedure will be applied in an iterative manner, starting with a single sequence ($p = 1$) which defines the maximum graph as a single chain arranging the nodes according to the given sequence. If, in a later iteration, a maximum graph representing p sequences is already present, the graph can be easily updated when a new sequence π_{p+1} becomes known. It is only necessary to examine if the chain representing the new sequence and the present maximum graph show contradictory precedence relations and to delete those relations from the maximum graph.

For example, first only one feasible sequence is known: π_1 = *1-2-3-4-5-6.* We build the maximum graph by setting all potential constraints $\overline{a}_{ij}^T = 1$ for $i < j$ and $\overline{a}_{ij}^T = 0$ otherwise (see Figure 2-I). Let the second sequence, applied in a second production period, be π_2 = *1-3-2-4-5-6.* As tasks 2 and 3 have swapped their places, no precedence constraint exists between both tasks and we update the maximum graph by setting $\overline{a}_{23}^T = 0$. This operation has no bearing on the other arcs, specifically the transitive relations from 1 to 3 and 2 to 4 still exist and transform into direct relations (see Figure 2-II). We assume a third practicable sequence π_3 = *1-2-3-6-4-5* is available and task 6 is now performed before tasks 4 and 5. Therefore, neither the direct precedence relation between 5 and 6 nor the indirect arc from 4 to 6 exists and we have to set $\overline{a}_{46}^T = \overline{a}_{56}^T = 0$. This operation has no effect on other arcs, e.g. $\overline{a}_{36}^T = 1$ or $\overline{a}_{45}^T = 1$ are still potential constraints.

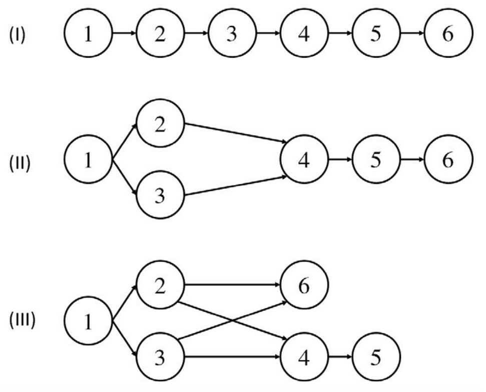

Figure 2. Direct arcs in the maximum graph after first, second and third sequence

In such a way we built the maximum graph (Figure 2-III) for the (unknown) target graph of Figure 1. For given cycle time $c = 5$, each of the three available practicable sequences needs $m = 4$ stations. Using the derived maximum graph as input for a line balancing procedure, we find the optimal sequence *1-3-2-6-4-5* that requires $m^* = 3$ stations though the graph is still more restrictive than the target graph.

Note that by enumerating on basis of the maximum graph, a set with eight feasible sequences is detected from the three already known without any further input by the planner. If problem size gets larger and sequences become more different, the ratio of detected sequences increases as well. It is this combination of simplicity and considerable effect that has impressed practitioners, to whom we presented the approach.

The **minimum graph** $\underline{G} = (V, \underline{E})$ with its adjacency matrix $\underline{A}$ is a precedence graph that contains the same nodes and task times as the target graph, but only a subset of the precedence restrictions of the target graph, i.e., $\underline{E}^T \subseteq E^T$. The precedence relations described by the minimum graph are valid, whereas the independencies are temporary.

The above relations can be summarized as:

$$\underline{E}^T \subseteq E^T \subseteq \bar{E}^T \tag{2}$$

An important measure characterizing graphs is the order strength, which is defined as $OS(G) = \frac{2\cdot|E^T|}{n\cdot(n-1)}$; cf. Mastor (1970). *OS* represents the ratio of the number of all (direct and indirect) precedence relations in G and the maximal number of precedence relations in a directed graph with *n* tasks, i.e., $n \cdot (n-1)/2$. Hence, for the above precedence graph types it follows:

$$OS(\underline{G}) \leq OS(G) \leq OS(\overline{G}) \tag{3}$$

Taking the differences between $\bar{A}^T$ and $\underline{A}^T$, we receive a matrix A^{pot} that contains all potential independencies.

Definition: $G^{pot} = (V, E^{pot})$ is the graph of potential independencies that contains the same nodes as the target graph and $E^{pot} := \bar{E}^T - \underline{E}^T$ and is formed as follows:

$$a_{ij}^{pot} = \bar{a}_{ij}^T - \underline{a}_{ij}^T \qquad \forall i,j = 1 \ldots n \tag{4}$$

We see that $a_{ij}^{pot} = 1$ implies a potential constraint in the maximum graph that has not yet been confirmed in the minimum graph. Two cases are possible: Either there really exists a precedence constraint (i,j) in the target graph ($a_{ij}^T = 1$) or the tasks are indeed independent ($a_{ij}^T = 0$) although i has been executed before j in each of the yet known feasible sequences.

A^{pot} also immediately shows confirmed relations. If $a_{ij}^{pot} = 0$, then tasks i and j are affirmed as dependent or independent in both the maximum and the minimum graph. Hence, their relation is verified yet.

3.2 Overall concept

Our approach consists in using the maximum and the minimum graph to approximate the target graph or, at least, the optimal solution of the target graph as closely as possible (see Figure 3).

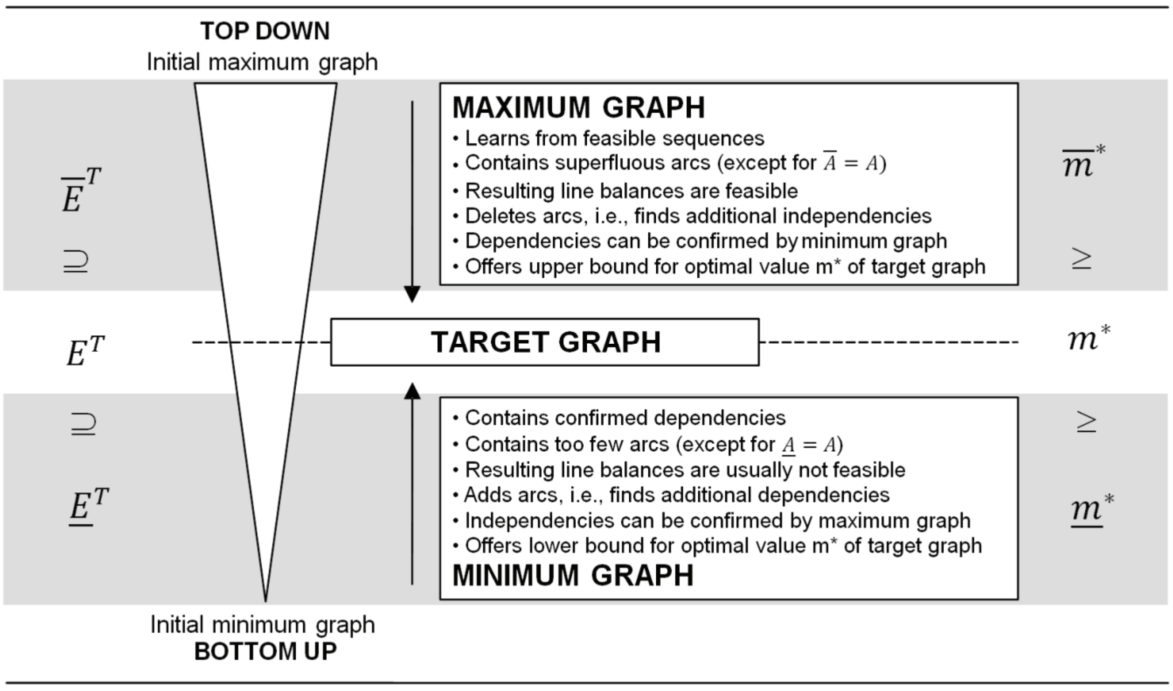
TOP DOWN
Initial maximum graph
MAXIMUM GRAPH
• Learns from feasible sequences
• Contains superfluous arcs (except for $\overline{A} = A$)
• Resulting line balances are feasible
• Deletes arcs, i.e., finds additional independencies
• Dependencies can be confirmed by minimum graph
• Offers upper bound for optimal value m* of target graph
$\overline{E}^T$
$\supseteq$
E^T
TARGET GRAPH
$\supseteq$
$\underline{E}^T$
• Contains confirmed dependencies
• Contains too few arcs (except for $\underline{A} = A$)
• Resulting line balances are usually not feasible
• Adds arcs, i.e., finds additional dependencies
• Independencies can be confirmed by maximum graph
• Offers lower bound for optimal value m* of target graph
MINIMUM GRAPH
$\overline{m}^*$
$\geq$
m^*
$\geq$
$\underline{m}^*$
Initial minimum graph
BOTTOM UP

Figure 3. Relationships and contributions of different graph types

As described earlier, the transitive closure of the maximum graph contains all the arcs of the transitive closure of the target graph plus additional arcs resulting from still possible further dependencies. In other words, $\bar{G}$ incorporates a subset of the sequences/solutions encoded in G. Since the maximum graph contains no constraints inconsistent with the target graph, it can serve as a basis for automated line balancing procedures to receive feasible solutions (provided the input data is correct). The guaranteed generation of feasible sequences is the main requirement to an automated concept coming from practice. For example, if we assume that the target of the line balancing procedure is to minimize the number m of stations, the best solution value $\overline{m}^*$ found for the maximum graph via a line balancing procedure represents an upper bound on the optimal number of stations in the target graph m^*. Note that for this upper bound property, it is not important if the ALBP instance defined by the maximum graph was solved to optimality or not. Anyway, the solution serves as a heuristic solution for the real ALBP instance defined by the target graph.

The minimum graph initially contains no arcs (zero matrix $\underline{A}^T$). In order to get closer to the target graph, the minimum graph is filled with confirmed constraints, e.g. from R&D department or CAD database (see Section 2), in a bottom up manner. All the arcs that are part of $\underline{E}^T$are also part of E^T. However, the minimum graph generally contains too few constraints and therefore resulting sequences might be infeasible or, in other words, the sequences represented by G build a subset of the sequences represented by the minimum graph $\underline{G}$. That is, the minimum graph ALBP is a relaxation of the original ALBP.

As a result, in case of minimization of the number of stations, the optimal solution with objective value $\underline{m}^*$ found via a line balancing procedure for the ALBP instance constructed from the minimum graph serves as a lower bound for the optimal solution value in the target graph. In contrast to the maximum graph, it is necessary to find an optimal solution in the minimum graph to assure that $\underline{m}^*$ is a valid lower bound. To summarize, we get the following relations:

$$\underline{m}^* \leq m^* \leq \overline{m}^* \tag{5}$$

Hence, we can conclude that if the best solution found for the maximum graph and the optimal solution in the minimum graph requires the same number of stations, this value is optimal for the target graph. In this case, the found maximum graph solution is optimal and also constitutes an optimal solution for the target graph.

$$\underline{m}^* = \overline{m}^* \Rightarrow \mathrm{m}^* = \underline{m}^* = \overline{m}^* \tag{6}$$

That is, it is not always necessary to continue approximating graphs until $\overline{A} = \underline{A}$ holds in order to find an optimal solution. This ability of the concept has great potential in practice as it will usually not be possible to close the gap between both graph types completely. Furthermore, the difference $\overline{m}^* - \underline{m}^*$ is an upper bound on the absolute deviation of $\overline{m}^*$ from the unknown optimal solution value m^*. So, it is possible to evaluate a current line balance with respect to capacity usage or other criteria in a worst-case manner.

Let us return to the example discussed in Section 3.1. Figure 4 repeats the target graph and the maximum graph. Dotted lines represent indirect precedence relations.

We now assume that the R&D department provided the following confirmed technological constraints: Task 1 has to be executed first and the precedence relations $(3,4)$ and $(3,6)$ must be met.

The combination of these precedence constraints results into the minimum graph in Figure 4. One can easily see that an optimal solution of the minimum graph (for $c = 5$) is sequence *1-3-2-5-4-6* with $\underline{m}^* = 3$ stations. The maximum graph leads to a feasible solution *1-3-2-6-4-5* that also needs $\overline{m}^* = 3$ (see Section 3.1). According to (6), a feasible and optimal solution for the target graph is sequence *1-3-2-6-4-5* with $m^* = 3$ stations and, hence, there is no need to approximate the target graph more closely (at least for the current planning period).

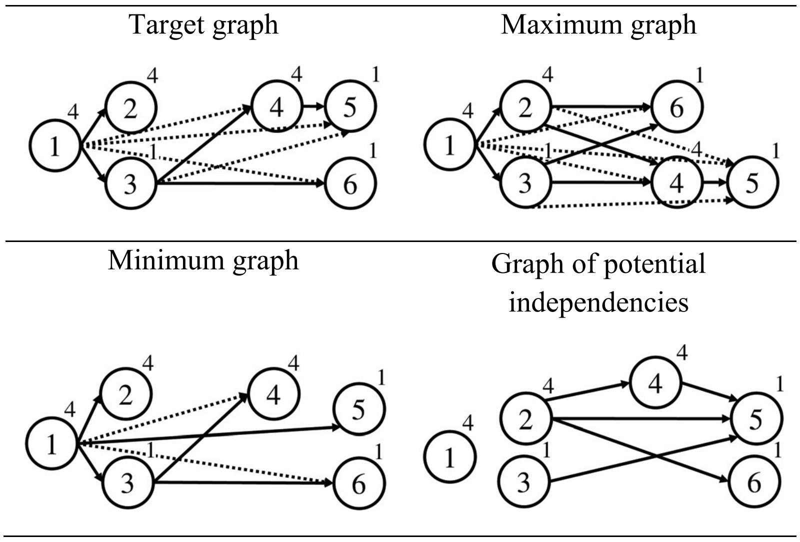

Figure 4. Example for precedence graph types and transitive closures

The graph of potential independencies in Figure 4 shows all the constraints from the maximum graph which are not confirmed by the minimum graph. In case the optimal solution would not have been found yet and in order to approach closer to the target graph, an expert would only need to decide about the five precedence relations (2,4), (2,5), (2,6), (3,5) and (4,5). Alternatively, further feasible sequences could be utilized to reduce the set of potential relations, e.g., *1-3-4-5-6-2*, which removes the first three potential arcs.

4 Computational experiments

In order to examine the potential performance of the learning precedence graph concept described in Section 3 in real-world applications, we conduct computational experiments based on well-known test instances. This is necessary as real-world data is not available due to the nescience about the real target graphs in practice.

4.1 Data set and experimental design

As a basis for testing, we use the well-known data set for SALBP with 269 instances based on 25 precedence graphs (interpreted as target graphs in our experiment), each having 8 to 297 nodes and connected to several cycle times. The average order strength is 0.510. For a detailed description of the data set and the characteristics of the underlying precedence graphs see Scholl (1999, Chapter 7.1) and the "homepage for assembly line optimization research" (www.assembly-line-balancing.de). We used the exact balancing procedure SALOME (Scholl and Klein 1997) with restricted computation time to optimize the maximum, minimum and target graph instances. As computing environment, a personal computer with Intel Core i7 processor (2.67 GHz) and 3 GByte of memory is utilized.

In the first part of our experiment (Section 4.2), we measure the quality of our learning concept by comparing the results of the optimization using the learned maximum graphs with the best practicable (input) sequences and the optimal solutions of the target graphs.

To better understand the learning precedence graph concept, we calculate the number of sequences represented by a maximum graph compared to the target graph in the second part of our computational experiment (Section 4.3).

In the final step (Section 4.4), minimum graphs are generated for a part of the data set in order to check the differences between $\overline{m}^*$and $\underline{m}^*$ (upper bound on the absolute deviations from optimality). As argued in Section 3.2, these values define the interval where the unknown optimal solution value m^* is bounded to.

4.2 Solution quality of line balancing using the maximum graph

To test our learning concept in a realistic manner, we assume the precedence graphs of the SALBP data set as target graphs (which are unknown in the real-world and, thus, are not available for tests). To simulate the (re-) balancing process in practice, where practicable (feasible) sequences are modified by a lot of single changes made by several planners between two planning periods, we generate a number of random sequences, which are feasible for the target graph of each test instance.

These sequences are utilized to form the maximum graph by adding one sequence after the other as described in Section 3.1.

In the first part of our experiment, we optimize on the maximum graphs with a computation time limit of 100 seconds for each instance. If the optimal solution could not be found or proven in this restricted time span, $\bar{m}^*$ denotes the number of stations in the incumbent solution (upper bound).

In order to assess the quality of the learned maximum graph $\bar{G}$, its optimal (incumbent) line balance, i.e., task sequence, and its order strength are compared to the corresponding values of the target graph G and the best of the input sequences (simulating the status quo in practice). We use the following measures:

$\frac{m_{best}}{m^*} - 1$	(relative) optimality gap of status quo; with m_{best} denoting the number of stations required by the best input sequence and m^*denoting the optimal value
$\frac{\bar{m}^*}{m^*} - 1$	(relative) optimality gap of learning concept; with $\bar{m}^*$ denoting the minimal (best) number of stations for the target graph
$1 - \frac{\bar{m}^*}{m_{best}}$	relative improvement of learning concept versus status quo
$OS(\bar{G}) - OS(G)$	closeness of maximum graph to target graph

Table 3 shows the average results for the complete data set in case of building the maximum graph of several (up to 20) input sequences.

Number of input sequences	2	3	4	5	10	15	20
$m_{best}/m^* - 1$	12.39 %	11.78 %	11.16 %	10.80 %	9.84 %	9.35 %	9.20 %
$\bar{m}^*/m^* - 1$	5.84 %	4.50 %	3.53 %	3.14 %	2.31 %	2.02 %	1.84 %
$1 - \bar{m}^*/m_{best}$	5.56 %	6.25 %	6.58 %	6.62 %	6.56 %	6.40 %	6.41 %
$OS(\bar{G}) - OS(G)$	0.373	0.325	0.298	0.278	0.224	0.201	0.180

Table 3. Average results for the line balancing using the maximum graph

Already after having learned from 5 input sequences, the best solution of the maximum graph is only 3.14 % worse than the optimal solution on average, whereas the best initial sequence (status quo) required about 10.80 % more stations, i.e., over-capacity, than the optimal solution on average. This good performance of our concept is remarkable as the maximum graphs still deviate from the target ones (documented by the difference of order strengths) considerably due to the rather small number of input sequences utilized (cf. Section 4.3).

With increasing number of input sequences, we approach closer the optimal solution of the target graph. Thus, after having learned from 20 practicable sequences, the solutions of the maximum graphs $\overline{m}^*$ were only 1.84 % worse than m^*, while the number of stations in the best input sequence m_{best} still exceeds m^* by 9.20 % on average. Furthermore, we notice that the maximum graph reduces capacity of the line by more than 6 % on average compared to the best input sequence (status quo). This result is rather stable irrespective of the number of input sequences.

Let us assume that the firm runs an assembly line with $m_{best} = 109$ stations. Then, the learning graph concept (with 20 input sequences) will economize about seven stations ($\overline{m}^* = 102$) on average, while the optimal solution would save about two additional stations ($m^* = 100$) on average. This shows that a large portion of the (theoretical) optimizing potential is exploited by our very simple and practicable approach.

Number of sequences	$m^* = m_{best}$	$m^* = \overline{m}^*$
5	20.82 %	54.28 %
10	23.05 %	62.83 %
15	25.65 %	66.17 %
20	25.65 %	67.29 %
Not found	74.35 %	32.71 %

Table 4. Optimal solutions found

Table 4 shows the portion of optimal solutions contained in the input sequences and the maximum graphs, respectively, for different numbers of input sequences. Again, the great potential for improvement becomes obvious. While the best input sequence among 20 is optimal only for a quarter of the instances (25.65%), an optimal sequence is contained in the maximum graph (and found within 100 seconds computing time) for more than two thirds of the instances (67.29%). Even if $\overline{m}^*$ still exceeds m^*, in only 3.35 % of the instances the difference is more than one station. The worst difference was four stations. In contrast, the best of 20 input sequences required two or more unnecessary stations for 31.97 % of the instances with the worst deviation being 12 stations (Bartholdi2, $c = 84$).

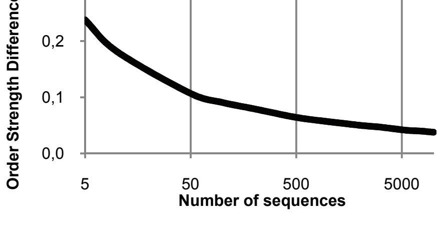

Figure 5. Average order strength gap

Figure 5 depicts the development of the average difference of the order strengths between the maximum graph and the target graph on a logarithmic scale. If $OS(\bar{G}) - OS(G) = 0$, the maximum graph equals the target graph. The order strength differences sharply decrease with the first input sequences. However, for large target graphs the maximum graph contains various unnecessary arcs even after learning from several thousands of input sequences. Given these results, it seems to be impossible for real-world problems with hundreds or thousands of tasks to generate the exact target graph.

However, our results above show that it is usually not necessary to approximate the target graph very closely. The analysis clearly indicates that a combination of just a few sequences even with humble solution quality generally leads to a maximum graph that contains significantly better, nearly optimal sequences. Especially for only five input sequences the decline in number of stations was surprisingly significant, while the order strength kept being rather high. The following section will give an idea why the learning precedence graph concept achieves such good results in face of large differences between maximum and target graph.

4.3 Number of feasible plans in maximum and target graph

In this section, we want to measure degrees of freedom, which arise from combining sequences to the maximum graph, by counting the number of feasible sequences represented by the maximum graph.

Counting the number of task orderings in a precedence graph is computationally hard; Brightwell and Winkler (1991) show its #P-completeness. So, it is unlikely to develop a polynomial time algorithm for this problem when defined on a general graph. In case of series parallel graphs, the problem is solvable in polynomial time as the graph can be evaluated by counting and collapsing serial and parallel subgraphs easily (cf. Miller and Stockman 1990a). Miller and Stockman (1990b) extend the polynomial approach to so-called NX-fold graphs which are composed of serial and parallel subgraphs as well as N- and X-shaped subgraphs with four nodes, respectively.

We extend this approach to general directed graphs by an enumerative approach. At first, the graph is reduced by collapsing (and counting) all the above mentioned subgraphs. If the remaining graph is a single node, the procedure stops. Otherwise, different subproblems are built by a branching step which enumerates on the possible orientation of arcs such that in leaf nodes of the resulting enumeration tree NX-fold graphs arise which can be evaluated by the procedure of Miller and Stockman (1990b). As this approach is only utilized to count the number of sequences for a subset of rather small graphs and this counting is not a part of our approach, we do not describe the procedure in detail and did not take care of an efficient implementation.

Already medium-sized precedence graphs may represent a multitude of feasible sequences. For example, due to its many parallel structures the graph Heskiaoff (cf. Scholl 1999, Chapter 7.1) with 28 tasks and order strength $OS(G) = 0.238$ represents more than $1.38 \cdot 10^{23}$ sequences.

Table 5 shows that by combination of just a few input sequences to form the maximum graph, we receive information on many new feasible sequences. At the same time, the stronger the order strength of a maximum graph declines the more sequences can be generated from that graph. Though the gap to the number of sequences represented by the target graph still is high, the solution set of the maximum graph often contains very good or even optimal solutions of the original (target) problem as documented in Section 4.2.

Precedence graph	Number of sequences in target graph	$OS(G)$	Number of sequences in maximum graph				$OS(\bar{G})$			
			2 seq.	3 seq.	4 seq.	5 seq.	2 seq.	3 seq.	4 seq.	5 seq.
Jackson (n = 11)	756	0.582	32	95	189	272	0.85	0.79	0.75	0.73
Mitchell (n = 21)	1,385,472	0.710	469	11,256	23,903	30,980	0.95	0.92	0.90	0.88
Lutz1 (n = 32)	$5.208 \cdot 10^{8}$	0.835	1,406	87,040	412,704	1,554,720	0.97	0.95	0.94	0.93

Table 5. Number of sequences and OS in target graphs and related maximum graphs

Basically, the smaller size of the solution set might even be an advantage as less enumeration is required to find the optimal solution (within the respective set). So, if an optimal solution is contained in the restricted solution set of the maximum graph, it could reduce computation time to utilize the maximum graph approach instead of optimizing on the target graph. If an optimal solution m^* was found and was proven in both graphs, the computation time for optimizing the maximum graph is only 87.38 % compared with the time optimizing the target graph on average in our experiment.

4.4 Lower bounds by the minimum graph

The number of stations in the optimal solution of the minimum graph $\underline{m}^*$ serves as a lower bound for (unknown) m^* and the difference $\overline{m}^* - \underline{m}^*$ as an upper bound on the deviation from optimality. Tables 6 and 7 show results for 189 of the 269 instances, for which minimum graphs were optimized. The minimum graphs were generated by randomly picking and fictitiously confirming about a quarter of the precedence relations in the target graph's transitive reduction. Note that only those minimum graphs were taken into consideration, whose optimal solutions could be found within the computation time limit, because only the *optimal* solution value of the minimum graph represents a lower bound for the initial problem. Nevertheless, each of the 25 precedence graphs from the data set is represented with at least one instance in this subset.

	$\emptyset$	min	max
$\overline{m}^* - \underline{m}^*$	0.720	0	6
$m^* - \underline{m}^*$	0.423	0	4
$OS(G)$	0.535	0.227	0.838
$OS(\underline{G})$	0.024	0.002	0.167

Table 6. Results for minimum graphs compared to target graphs and maximum graphs after 5 sequences

The results in Table 6 show that minimum graphs having significantly lower order strengths than the target graph still can lead to bounds close to the optimal solution value. While the mean order strength of the minimum graph is much lower than that of the target graphs, the average difference $m^* - \underline{m}^*$ is only 0.423 stations. The maximal deviation amounts to 4 stations.

From Table 7, we see that in 62% of 189 instances the optimal number of stations in the minimum graph equaled m^* and in further 34% the lower bound $\underline{m}^*$ was only one station too low.

$\overline{m}^* - \underline{m}^*$		$m^* - \underline{m}^*$	
0	41.80 %	0	62.4%
1	48.68 %	1	34.4%
2	6.35 %	2	2.1%
3	2.11 %	3	0.5%
4	1.06 %	4	0.5%

Table 7. Shares of instances with certain gaps

Further, with the help of the minimum graph, we can measure the worst-case deviation of the best solution of the maximum graph from the (actually unknown) optimal value m^*. For 41.80 % of instances $\underline{m}^*$ complies with $\overline{m}^*$and therefore the optimality of $\overline{m}^*$ was proven without knowing the actual value of m^*. It is impressive that in more than 90 % of the instances the simple overall concept could confirm that the solution deviates from optimality by at most one station though the minimum graphs $\underline{G}$ contain only a small part of the real precedence relations.

5 Conclusions

In this paper, we contribute to overcoming one of the main gaps between line balancing in theory and practice. Whereas the application of existing line balancing approaches requires a precedence graph, we are aware of no automotive manufacturer possessing the precedence graph describing the entire production process of any of its models. We examined the available approaches for generating the precedence graph and found out that none of the existing methods is suitable for the conditions in the automotive and related industries. The presented learning precedence graph concept takes requirements of manufacturers into account and relies on the analysis of available practicable sequences.

We tested our approach on a well-established benchmark data set. We showed that a graph constructed by learning from just a few practicable sequences, that we call maximum graph, frequently allows to approaching closely the optimal solutions of the (unknown) target graph. In our experiments, we found very small deviations from optimality (less than 2%) which can be considered sufficient in relevant real-world settings. A pilot study conducted at a major German automotive manufacturer showed that the maximum graph constructed from only six input sequences already contains information of about twenty thousand new feasible sequences. This result was highly appreciated by planners such that parts of the concept are already implemented in the manufacturer's planning software.

We also utilize the available information on sure precedence relations (e.g., geometrical information from CAD, hierarchical dependencies in the manufacturing bill of materials) to build a minimum graph that serves as indication of the remaining gap to the optimal solution of the target graph.

Further research steps might be to determine factors that influence characteristics of the solution sets of the maximum, the minimum and the target graph. Also it has to be clarified, how the way of generation and the quality of input solutions influences the best solution of the received maximum graph.

To complement the overall concept, all options to fill the minimum graph with approved precedence relations and to find independencies to approach the maximum graph to the target one need to be collected and combined to develop the full power of the approach. In particular, it seems to be promising to use interviews with experts in a selective manner by analyzing maximum and minimum graphs to identify still assumed relations that are crucial for the graph's structure.

Acknowledgement

We are indebted to our former students Johannes Sternatz and Julia Hillebrand for discussing the concept and conducting a supporting case study.

References

Altemeier, S. (2009) *Kostenoptimale Kapazitätsabstimmung in einer getakteten Variantenfließlinie*, PhD dissertation, University of Paderborn, Germany.

Ammer, E.-D. (1985) *Rechnergestützte Planung von Montageablaufstrukturen für Erzeugnisse der Serienfertigung*, IPA-IAO Forschung und Praxis 81, Springer, Berlin.

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

Becker, C. and Scholl, A. (2009) Balancing assembly lines with variable parallel workplaces: Problem definition and effective solution procedure. *European Journal of Operational Research,* **199/2**, 359-374.

Bonneville, F., Perrard, C., and Henrioud, J.M. (1995) A genetic algorithm to generate and evaluate assembly plans. *IEEE Symposium on Emerging Technologies and Factory Automation*, **2**, 231–239.

Bourjault, A. (1984) Contribution a une approche méthodologique de l'assemblage automatisé: Elaboration automatique des séquences opértiores, PhD dissertation, Université de Franche-Comte.

Boysen, N., Fliedner, M., and Scholl, A. (2007) A classification of assembly line balancing problems. *European Journal of Operational Research,* **183/2**, 674-693.

Boysen, N., Fliedner, M., and Scholl, A. (2008) Assembly line balancing: Which model to use when? *International Journal of Production Economics,* **111/2**, 509-528.

Boysen, N., Fliedner, M., and Scholl, A. (2009a) Assembly line balancing: Joint precedence graphs under high product variety. *IIE Transactions*, **41/3**, 183-193.

Boysen, N., Fliedner, M., and Scholl, A. (2009b) Production planning of mixed-model assembly lines: Overview and extensions. *Production Planning and Control,* **20/5**, 455-471.

Brightwell, G. and Winkler, P. (1991) Counting Linear Extensions. *Order*, **8**, 225–242.

Chen, G., Zhou, J., Cai, W., Lai, X., Lin, Z., and Menassa, R. (2006) A framework for an automotive body assembly process design system. *Computer Aided-Design*, **38**, 531–539.

Dong, T., Tong, R., Thang, L., and Dong, J. (2007) A knowledge-based approach to assembly sequence planning. *International Journal of Advanced Manufacturing Technology*, **32**, 1232–1244.

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

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

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

Kaufman, S. G., Wilson, R. H., Jones, R. E., Calton, T. L., and Ames, A. L. (1996) The Archimedes 2 mechanical assembly planning system. *IEEE International Conference on Robotics and Automation*, **4**, 3361–3368.

Mastor, A. A. (1970) An experimental investigation and comparative evaluation of production line balancing techniques. *Management Science*, **16**, 728–746.

Miller, J. M. and Stockmann, G. C. (1990a) On the number of linear extensions in a precedence graph. *IEEE International Conference on Robotics and Automation*, Cincinnati Ohio, pp. 2136–2141.

Miller, J. M. and Stockmann, G. C. (1990b) Precedence constraints and tasks: How many task orderings? *IEEE International Conference on Systems Engineering*, pp. 408–411.

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

Riesbeck, C. K. and Schank, R. C. (1989) *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Inc. Publishers, Hillside, New Jersey.

Romney, B., Godard, C., Goldwasser, M., and Ramkumar, G. (1995) An efficient system for geometric assembly sequence generation and evaluation. *Proceedings of the ASME International Computers in Engineering Conference*, Boston, Massachusetts, pp. 699–712.

Scholl, A. (1999) *Balancing and sequencing of assembly lines*, 2nd ed., Physica, Heidelberg.

Scholl, A. and Becker, C. (2006) State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, **168/3**, 666-693.

Scholl, A., Fliedner, M., and Boysen, N. (2010) Absalom: Balancing assembly lines with assignment restrictions. *European Journal of Operational Research*, **200/3**, 688-701.

Scholl, A. and Klein, R. (1997) SALOME: A bidirectional branch and bound procedure for assembly line balancing. *INFORMS Journal on Computing*, **9**, 319–334.

Smith, G. C. and Smith, S. S.-F. (2002) An enhanced genetic algorithm for automated assembly planning. *Robotics and Computer Integrated Manufacturing*, **18**, 355–364.

Su, Q. (2007) Applying case-based reasoning in assembly sequence planning. *International Journal of Production Research*, **45**, 29–47.

Su, Q. (2009): A hierarchical approach on assembly sequence planning and optimal sequences analyzing. *Robotics and Computer Integrated Manufacturing,* **25**, 224-234.

Swaminathan, A. and Barber, S. (1996) An experience-based assembly sequence planner for mechanical assemblies. *IEEE Transactions on Robotics and Automation*, **12**, 252–267.