Jena Research Papers in Business and Economics

Balancing assembly lines with variable parallel workplaces: Problem definition, model and exact solution procedure

Christian Becker, Armin Scholl

06/2008

Jenaer Schriften zur Wirtschaftswissenschaft

Working and Discussion Paper Series
School of Economics and Business Administration
Friedrich-Schiller-University Jena

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

Balancing assembly lines with variable parallel workplaces: Problem definition, model and exact solution procedure

Christian Becker

Railion Deutschland AG
Rheinstraße 2, D-55116 Mainz
e-Mail: christian.ch.becker@bahn.de

Armin Scholl

Friedrich-Schiller-Universität Jena
Fakultät für Wirtschaftswissenschaften
Lehrstuhl für Betriebswirtschaftliche Entscheidungsanalyse
Carl-Zeiß-Straße 3, D-07743 Jena
e-Mail: a.scholl@wiwi.uni-jena.de

Abstract: Assembly line balancing problems (ALBP) arise whenever an assembly line is configured, redesigned or adjusted. An ALBP consists of distributing the total workload for manufacturing any unit of the products to be assembled among the work stations along the line subject to a strict or average cycle time. Traditionally, stations are considered to be manned by one operator, respectively, or duplicated in form of identical parallel stations, each also manned by a single operator. In practice, this assumption is usually too restrictive. This is particularly true for large products like cars, trucks, busses and machines, which can be handled by several operators performing different tasks at the same time. Only restricted research has been done on such parallel workplaces within the same station though they have significant relevance in real-world assembly line settings.

In this paper, we consider an extension of the basic ALBP to the case of flexible parallel workplaces (VWALBP) as they typically occur in the automobile and other industries assembling large products. The problem is defined and modelled as an integer linear program. As a solution approach a branch-and-bound procedure is proposed which also can be applied as a heuristic. Finally, computational experiments documenting the solution capabilities of the procedure are reported.

Keywords: Assembly line balancing – Mass-production – Combinatorial optimization – Sequencing

1 Introduction and literature review

Assembly lines are flow-oriented production systems which are typical in the industrial production of high quantity standardized commodities and even gain importance in low volume production of customized products. Among the decision problems which arise in managing such systems, assembly line balancing problems are important tasks in medium-term production planning (cf., e.g., Baybars,1986; Becker and Scholl 2006; Boysen et al. 2007).

An assembly line consists of *(work) stations* k=1,...,m arranged along a conveyor belt or a similar mechanical material handling equipment. The workpieces are consecutively launched down the line and are moved from station to station. At each station, certain operations are repeatedly performed regarding the *cycle time* (maximum or average time available for each workcycle). The decision problem of optimally partitioning (balancing) the assembly work among the stations with respect to some objective is known as the *Assembly Line Balancing Problem (ALBP)*.

Manufacturing a product on an assembly line requires partitioning the total amount of work into a set $V = \{1,...,n\}$ of *tasks* which constitute the nodes of a precedence graph. Performing a task i takes a *task time* (node weight) t_i. A *precedence relation* (i,j) means that task i must be finished before task j can be started and is contained in the arc set E of the *precedence graph*. Within the acyclical and topologically numbered graph redundant arcs are omitted.

The following sets are useful to describe the precedence relations:

- $P_i = \{h \in V \mid (h,i) \in E\}$ set of direct predecessors of task $i \in V$
- $F_i = \{j \in V \mid (i,j) \in E\}$ set of direct successors (followers) of task $i \in V$

Assuming E^* to be the transitive closure of E, we further define:

- $P_i^* = \{h \in V \mid (h,i) \in E^*\}$ set of all predecessors of task $i \in V$
- $F_i^* = \{j \in V \mid (i,j) \in E^*\}$ set of all successors of task $i \in V$

Any type of ALBP consists in finding a feasible *line balance*, i.e., an assignment of tasks to stations such that precedence constraints and possible further restrictions are fulfilled. The set S_k of tasks assigned to a station k (=1,...,m) constitutes its *station load*, the cumulated task time $t(S_k) = \sum_{j \in S_k} t_j$ is called *station time*. When a fixed common cycle time c is given, a line balance is feasible only if the station time of neither station exceeds c. In case of $t(S_k) < c$, the station k has an *idle time* of $c - t(S_k)$ time units in each cycle, i.e., it is repeatedly unproductive for this time span.

The most popular ALBP is called *Simple Assembly Line Balancing Problem (SALBP)*. It has the following characteristics (cf. Baybars 1986; Scholl and Becker 2006; Boysen et al. 2007):

(S-1) Mass-production of one homogeneous product.
(S-2) All tasks are processed in a predetermined mode (no processing alternatives exist).
(S-3) Paced line with a fixed common cycle time according to a desired output quantity.
(S-4) The line is considered to be serial with no feeder lines or parallel elements.
(S-5) The processing sequence of tasks is subject to precedence restrictions.
(S-6) Deterministic (and integral) task times t_j with $t_j \leq c$.
(S-7) No assignment restrictions of tasks besides precedence constraints.
(S-8) A task cannot be split among two or more stations.
(S-9) All stations are equally equipped with respect to machines and workers.

The general objective consists of maximizing the *line efficiency* $Eff = t_{sum}/(m \cdot c)$ with total task time $t_{sum} = \sum_{j=1}^{n} t_j$. Several problem versions arise from varying the objective as shown in Table 1. The tuple-notations specify the characterizations of the problem versions within the recent classification scheme of Boysen et al. (2007). All versions of SALBP and, thus, all generalizations like the new problem VWALBP, are NP-hard (cf. Wee and Magazine 1982; Scholl 1999, ch. 2.2.1.5).

		cycle time c	
		given	minimize
no. m of stations	given	SALBP-F [| |]	SALBP-2 [| | c]
	mini-mize	SALBP-1 [| | m]	SALBP-E [| | E]

Table 1. Versions of SALBP

The basic SALBP has been extended and generalized in many research papers constituting a large number of generalized assembly line balancing problems (GALBP). For recent surveys and classifications of those problems see Becker and Scholl (2006) and Boysen et al. (2007).

Among these generalizations, the following are particularly relevant for the problem discussed here or helpful to explain what is different in the new problem. To focus on the problem itself, we only discuss assumptions and real-world backgrounds rather than algorithmic developments.

- *Parallel stations:* Two or more stations are arranged in parallel to allow for increased local cycle times (if p parallel stations are installed, the local cycle time amounts to $p \cdot c$). Then, it is possible to assign tasks with operation times larger than the cycle time. Because each copy of the station performs the *same set of tasks* in an alternating manner, it has *exclusive access* to the workpiece currently within its station area (cf. Buxey 1974; Pinto et al. 1975, 1981; Sarker and Shanthikumar 1983; Bard 1989).
- *Assignment restrictions:* In practice, there are usually constraints which restrict the assignment of tasks to stations in addition to the cycle time constraint and the precedence relations (cf. Scholl 1999, ch. 1.3.4; Boysen et al. 2007; Scholl et al. 2008).
  Regularly, there are restrictions on assigning tasks to certain stations. Such *station restrictions* are particularly relevant when the line is already installed and should be rebalanced without rearranging all the equipment (cf. Tonge 1960; Boguschewski et al. 1990), only certain station types are capable of performing some task (Johnson 1983) or the workpiece is required in a certain position (Lapierre and Ruiz 2004).
  Furthermore, there are often *incompatible tasks* that must not be assigned to the same station, e.g. due to the danger of soiling the seats of a car if the same worker has to handle the seats and to lubricate movable parts (cf. Agnetis et al. 1995; Johnson 1983; Rachamadugu 1991). Another reason for setting tasks incompatible is due to different *mounting position*s at a large workpiece (like cars, trucks, washing machines). In order to reduce walking distances of workers, certain combinations of mounting position and, thus, the corresponding tasks are defined to be incompatible (cf. Johnson 1983).
- When large workpieces are assembled, it is possible that different operators work at the same product unit simultaneously (cf. Kilbridge and Wester 1962; Akagi et al. 1983). That is, *several workplaces* are installed *at the same station*. Each worker (workplace) gets an own set of tasks at individual mounting positions. It has to be ensured that the workers do not interfere with each other by exclusively assigning each mounting position required to a single workplace. In case of a *two-sided line* (2ALBP), each station consists of (up to) two workplaces, one at the right and one at the left side of the line (Bartholdi 1993; Kim et al. 2000; Lee et al. 2001; Lapierre and Ruiz 2004; Lapierre et al. 2006). All tasks that have to be performed at the

left (right) side of the workpiece must be assigned to a left (right) workplace. Remaining tasks can be assigned either to a left or a right workplace. Thus, the workpiece can be considered as being subdivided into two incompatible mounting positions (left and right), or alternatively, each pair of tasks which have to be performed on opposite sides is set incompatible. The two-sided line can be generalized to a *multi-sided line* (NALBP) by increasing the maximal number of workplaces per station (cf. Gehring and Boguschewski 1990), e.g. a car can be subdivided into 4 exclusive mounting positions (front left, front right, back left, back right).

In Section 2, we generalize the concept of stations with multiple workplaces in a flexible manner as to consider real-world conditions as accurately as possible. The resulting problem VWALBP is formulated as an integer linear program in Section 3. Lower bounds and reduction rules are developed in Section 4. A branch-and-bound procedure for optimally solving this problem, which can also be applied as a heuristic, is presented in Sections 5 and 6. Computational experiments that evaluate the performance of the solution methods are reported in Section 7. The paper ends with conclusions and remarks on future research challenges in Section 8.

2 The assembly line balancing problem with variable workplaces (VWALBP)

As mentioned before, many products manufactured on assembly lines are large enough to be worked at by several workers simultaneously. As a major example, we focus on the final assembly of cars in the automobile industry where up to five workplaces are installed within a single station on a paced assembly line. Other examples of products assembled in such a manner are trucks, busses, large machines and even helicopters (Bartholdi 1993; Lee et al. 2001).

2.1 Problem description

In the following, we define a new decision problem which is intended to model the situation of such car assembly lines (and related production systems as mentioned before) as realistically as possible. However, in order to define a basic problem, we do not include all conditions which might occur at a real line but focus on the most important ones giving structure to the problem. The new problem is called Assembly Line Balancing Problem with Variable Workplaces (VWALBP) and is an extension of SALBP characterized by the SALBP assumptions (S-1) to (S-5) and modifications of (S-6) to (S-9):

(V-6) *Task times:* Deterministic and integral[1] task times t_j some of which might exceed the cycle time. Such *extra-long tasks* (collected in set EL) occur, e.g., when mounting large pieces like cockpits or examining the electrical devices already installed (see (V-8)).

(V-7) *Assignment restrictions:* Station restrictions fixing certain tasks to a specific station (e.g. automatic station for installing the front windows) or a certain station interval (e.g., segment of the line) and two types of incompatibilities have to be considered. Tasks are called *station incompatible* to each other if they must not be assigned to the same station (e.g. due to different mounting conditions required). Tasks are called *workplace incompatible* to each other if they can be assigned to the same station but not to the same workplace (e.g.

1 From a practical point of view this is no relevant restriction, because non-integral task times can be easily transformed into integer values by changing the scale, e.g., from minutes to seconds.

incompatible working positions due to long walking distance). All pairs of station incompatible tasks are collected in the set SI, while the pairs of workplace incompatible tasks are contained in another set WI.[2] Furthermore, assignment restrictions implicitly result from the fact that each mounting position can only be assigned to a single workplace per station.

(V-8) *Task indivisibility:* Tasks are not split up between two or more stations except for extra-long tasks. Such a task $j \in EL$ is spread over successive stations but performed by the same workplace (Arcus 1966, Falkenauer 2005). Such a *multi-station workplace*, thus, covers $p_j = \lceil t_j / c \rceil$ stations and requires p_j operators which alternately assemble the workpieces, each for p_j successive cycles (Inman and Leon 1994). At the end of the last cycle involved, remaining idle time (if $p_j \cdot c - t_j > 0$) can be used to perform other tasks. That is, any extra-long task starts at the beginning of the cycle in its first station and is performed without break until it is terminated.

(V-9) *Station configuration and equipment:* Usually, each workplace covers a single station and is manned by one operator (for the only exception see V-8). Each station (segment of the line) might contain 0 to w_{max} parallel workplaces. Depending on the product and the conditions at the line, the maximum number of workplaces is typically restricted to $W \in [2, 5]$. Particular equipment fixed at certain (automated) stations leads to station restrictions as described in (V-7).

The modified assumptions describe most aspects of the new problem VWALBP. Only some further explanations are necessary which will be supported by examples:

Each task is uniquely assigned to a workplace, i.e., the set of tasks is partioned into disjoint subsets called *WP loads* such that the local cycle time of the workplace is observed (c for normal and $p_j \cdot c$ for multi-station workplaces that contain an extra-long task $j \in EL$). The WP loads and, thus, the workplaces are assigned to stations such that the precedence constraints and the maximal number of parallel workplaces are observed. Since all operators assigned to the workplaces installed at the same station *simultaneously perform different tasks at the same workpiece*, it must be avoided that they obstruct each other. This is achieved by subdividing the workpiece into mounting positions each of which can be used by a single workplace only.

Figure 1 shows a subdivision of some workpiece into Q mounting positions. Here, $Q = 24$ is chosen which is a typical setting for a car body in the automobile industry (cf. Hildebrand 2006).

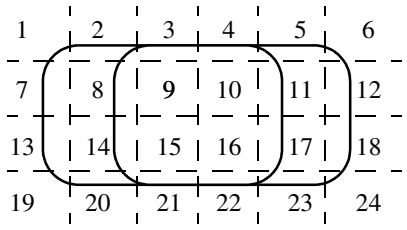

Figure 1.
Mounting positions at a car body

As is the case with SALBP, the objective consists of maximizing the line efficiency which has to be defined somewhat different for VWALBP, because stations, workplaces and operators do not mean the same anymore. Each station may contain no, one or many workplaces and each workplace is manned by one or many operators (normal and multi-station workplaces, respectively). Thus, the productive capacity provided by the line is neither defined by the number of stations nor the number of workplaces but the number of operators required. To differentiate between these terms, we now use the following notation:

2 By WI_j and SI_j we denote the set of tasks to which task j is workplace and station incompatible, respectively.

m number of operators (workers)

K maximal number of stations (restricted by available space, installed conveyor belt)

W maximal number of workplaces per station (restricted by size of workpiece, station space)

By analogy with SALBP, we get different versions of VWALBP as documented in Table 2 which additionally shows the classification tuples according to the scheme of Boysen et al. (2007). Extra-long tasks are the only aspect which cannot be modelled in the classification scheme in a direct manner. Thus, we include a further value $\alpha_5 = ch$ for the topic assignment restrictions which expresses that a *chain* of tasks must be assigned to a subsequence of successive stations (cf. Section 3). Notice that this chain restrictions could also be modelled in a more complicated way by minimal and maximal distances ($\alpha_5 = \min, \max$).

		cycle time c	
		given	minimize
no. m of operators	given	VWALBP-F [inc,fix,type,ch|pwork|]	VWALBP-2 [inc,fix,type,ch|pwork|c]
	mini-mize	VWALBP-1 [inc,fix,type,ch|pwork|m]	VWALBP-E [inc,fix,type,ch|pwork|E]

Table 2. Versions of VWALBP

In the following, we concentrate on VWALBP-1 (minimize the number of workers given the cycle time) because this problem version reflects the regular situation in automobile plants as the cycle time is usually deduced from required output quantities (actual orders and sales forecasts) and the manpower requirements have to be minimized. However, if other problem versions (VWALBP-F, -2, -E) are required in a certain problem situation, the results obtained when analyzing VWALBP-1 can easily be transferred following the well-known relationships between the corresponding SALBP problem types (cf. Scholl 1999, ch. 2.2.1.3, 4.2.2, and 4.3.2).

Three issues need further explanations: the assignment of mounting positions to workplaces, the detailed scheduling within stations and extra-long tasks causing multiple-station workplaces.

2.2 Assigning mounting positions to workplaces

Let Q_j denote the set of (compatible) mounting positions required to perform a task j. When assigning task j to a workplace w, also the mounting positions in Q_j are exclusively assigned to w (in all stations covered by w) in order to avoid mutual obstruction of workers. Concerning a WP load WL for w, we get the set Q(w) of positions occupied by workplace w: $Q(w) = \bigcup_{j \in WL} Q_j$.

Mounting position related tasks: As a consequence, tasks that require at least one joint mounting position have to be assigned to the same workplace provided that they should be performed in the same station. Pairs of such mounting position related tasks h and j are stored in the set MR.

Incompatible mounting positions: In assumption (V-7), it is mentioned that some pairs of mounting positions are incompatible with respect to a single workplace due to long walking distances. These pairs are stored in the set IMP. As a consequence, two tasks h and j are workplace incompatible if there exists a pair $(p_1, p_2) \in IMP$ such that $\{p_1, p_2\} \subseteq Q_h \cup Q_j$. Due to this relationship, incompatible mounting positions can be transferred into workplace incompatible task pairs, additionally to those already contained in WI, in a preprocessing step.

Figure 2 illustrates the different meanings of stations and workplaces and the assumptions on mounting positions in different problem types. In SALBP, each station consists of a single workplace each of which is manned by a single operator. All mounting positions are assumed to be available irrespective of walking distances.

In case of the two-sided line (2ALBP), each station contains up to two workplaces which are arranged at the left and/or the right side of the line. Each workplace is manned by one operator and restricted to left and right mounting positions, respectively, but walking distances or further reasons for incompatibility are not considered. In VWALBP, each station contains up to, e.g., W = 4 workplaces which are arranged in a flexible manner only constrained by the requirement to assign mounting positions uniquely and to observe incompatibilities. Each workplace is manned by one operator except for the multi-station workplace WP_3 which needs two workers to perform an extra-long task alternately. Notice that multi-station workplaces reserve all mounting positions required by the extra-long task in all stations covered but the positions required by additional (normal) tasks only in the last station (see Section 2.4).

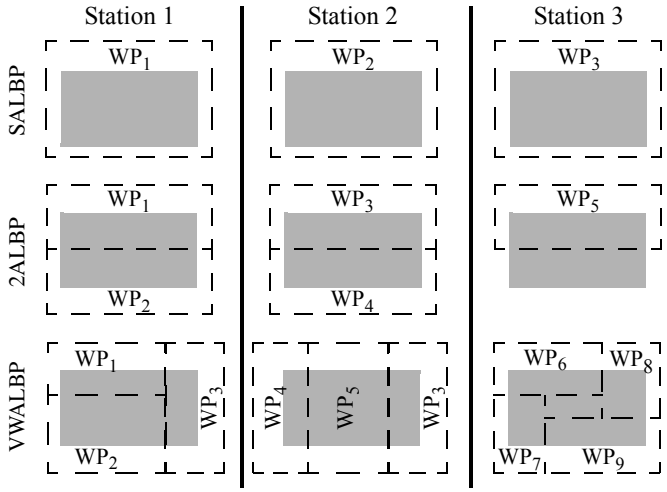

Figure 2. Stations and workplaces in SALBP, 2ALBP, VWALBP

2.3 Detailed scheduling

Besides the *assignment part* of the problem (tasks to workplaces, workplaces to stations), an additional subproblem needs to be solved due to the flexibility of arranging workplaces within a station. It consists of assigning actual starting times (relative to the start of the line) for all tasks assigned to a station, i.e., a *scheduling problem* has to be solved for each station (Falkenauer 2005).

Consider the example instance with nine tasks presented in Figure 3. In addition to the traditional precedence diagram, the set of mounting positions Q_j is specified for each task j. As presented, the workpieces are subdivided into eight mounting positions. Furthermore, the following set of incompatible mounting positions is to be considered: IMP = {(1,5), (1,6), (2,5), (2,6), (3,7), (3,8), (4,7), (4,8)}. Given a cycle time $c = 10$, we get an optimal solution with two stations and five workplaces and operators, respectively, as illustrated in the Gantt chart of Figure 3. In station 1, three workplaces are installed, while two workplaces are set into station 2. The mounting positions occupied by the workplaces are illustrated in the rightmost picture.

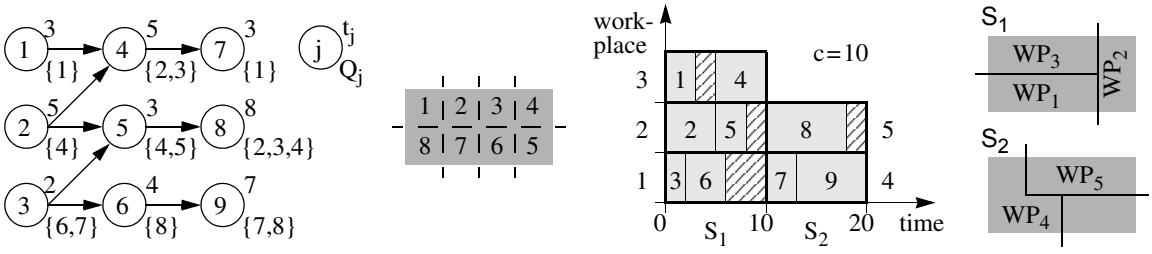

Figure 3. Example instance of VWALBP

While in SALBP it is easy to schedule tasks by assigning them earliest starting times in an arbitrary precedence-feasible sequence in each station (thus shifting idle time towards the end of the station), it is necessary to consider all workplaces simultaneously in VWALBP (and 2ALBP; cf. Bartholdi 1993). Here, task 4 in WP_3 has to wait until is predecessors 1 (also in WP_3) and 2 (in

WP_2) are finished. Because task 2 is longer than task 1, an idle time of $t_2 - t_1 = 2$ is necessary between tasks 1 and 4. This is why it is not ensured that a feasible solution exists for a certain station load and a given number of workplaces even if each WP load observes the cycle time individually. For this so-called station problem and its solution see Section 6.

2.4 Extra-long tasks and multiple-station workplaces

Figure 4 shows a problem instance containing an extra-long task 1 with time $t_1 = 15$ exceeding the cycle time $c = 10$. The Gantt chart illustrates the optimal solution with the two-station workplace 2 which has available twice the cycle time and requires two workers each of which has access to a workpiece for 20 time units. In the last station of such a multi-station workplace remaining idle time can be used to assign additional tasks. Here, task 6 is additionally performed in station 2 of workplace 1. So, the workers perform the WP loads {1,6} with remaining idle time of 1 time unit on successive workpieces in an alternating manner. In station 1, the multi-station workplace occupies only the mounting position 1, while it requires positions 1 and 2 in station 2 due to the additional task 6.

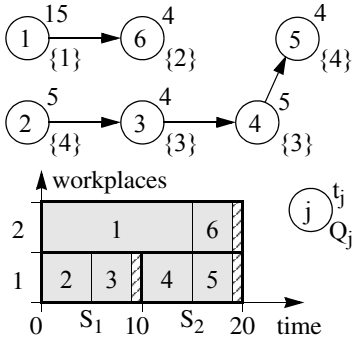

Figure 4. Extra-long task

3 Mathematical model

For SALBP, there is a number of integer linear programs available (cf. Scholl 1999, ch. 2.2; Peeters and Degraeve 2006). Following the basic lines of these models, we develop a mixed-integer program (MIP) for VWALBP-1 which reflects all modifications discussed in Section 2. Notice that the model can easily be adapted to the other problem versions (cf. Scholl 1999, ch. 2.2).

In order to do without explicitly considering stations, we number the potential workplaces in a consecutive manner from 1 to $\overline{w} = m \cdot W$. By an easy transformation, the station k(i) to which a particular workplace i belongs can be retrieved: $k(i) = \lceil i / W \rceil$ for $i = 1,...,\overline{w}$.

Each workpiece is launched down the line and enters station 1 at time $t = 0$ and leaves the last station at time $t = m \cdot c$. The workplaces $i = 1,...,\overline{w}$, belonging to station k(i), start operating each workpiece at time $ts(i) = (k(i)-1) \cdot c$ after having launched this workpiece. Figure 5 shows a profile of potential workplaces for $W = 4$ and $K = 4$. The first 4 workplaces are assigned to S_1, the next four to S_2 and so on. The binary variables y_i indicate whether or not workplace i is installed. Here, 12 workplaces are in use distributed over the stations as indicated by gray areas.

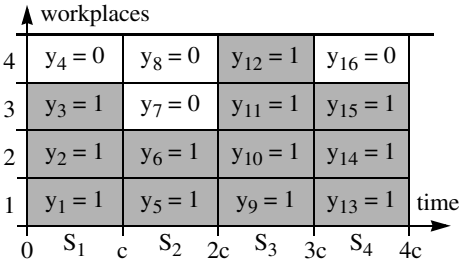

Figure 5. Profile of workplaces

In order to include extra-long tasks into the model without defining a more complex workplace profile, it is assumed that each task $j \in EL$ with $t_j > c$ is subdivided into a chain of tasks (each task connected to the next by an arc in the precedence graph) having task time c and a final task with task time t_j *modulo* c as described in Section 2. All tasks of the chain have the same mounting positions in set Q_j and share the same incompatibilities with other tasks. As a matter of convenience, the first task of the chain gets the original number j, while the others get yet unused numbers

increasing n accordingly. The final task of the resulting chain is referenced by f(j); the original task time is saved in $\bar{t}_j$. A further convention simplifies presentation: Multiple station workplaces are simply considered as a sequence of workplaces belonging to successive stations such that the numbers of installed workplaces and employed operators are equal. After having replaced all extra-long tasks by chains, the graph is reordered in a topological manner, i.e., each task gets a smaller number than each of ist successors.

Task 1 of the example in Figure 4 is replaced by the modified task 1 with $t_1 = 10$ and an additional task 7 with $t_7 = 15 - c = 5$. The predecessors of original j remain predecessors of the modified j, while the successors of original j become successors of f(j) now. Finally, the tasks 6 and 7 exchange numbers to get the topologically ordered graph in Figure 6.

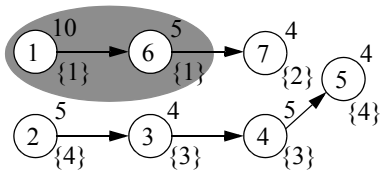

Figure 6. Transformation of extra-long task 1

Assuming that these preliminary steps have been done, we develop a mixed integer program based on the following decision variables (using the indices h,j = 1,...,n for tasks (including chains replacing extra-long tasks), i = 1,...,$\overline{w}$ for workplaces and p = 1,...,Q for mounting positions):

$x_{ij} = \begin{cases} 1 & \text{if task j is assigned to workplace i} \\ 0 & \text{otherwise} \end{cases}$ for all i and j

$y_i = \begin{cases} 1 & \text{if the potential workplace i is installed} \\ 0 & \text{otherwise} \end{cases}$ for all i

$g_{ijh} = \begin{cases} 1 & \text{if tasks j and h are jointly performed at workplace i} \\ 0 & \text{otherwise} \end{cases}$ for all $j \neq h$ and all i

$v_{jh} = \begin{cases} 1 & \text{if task j is executed before h at the same workplace} \\ 0 & \text{otherwise} \end{cases}$ for all $j \neq h$

s_j starting time of task j (relative to the launch time) for all j

The *mathematical model* for ALBP-VWP-1 is to

$$\text{minimize } F(\mathbf{x},\mathbf{y},\mathbf{g},\mathbf{v},\mathbf{s}) = m = \sum_{i=1}^{\overline{w}} y_i \tag{1}$$

such that the constraints (2)–(16) are fulfilled.

- *Operational workplaces:* Tasks can only be assigned to installed workplaces.

$$\sum_{j=1}^{n} x_{ij} \leq n \cdot y_i \qquad \text{for all } i = 1,...,\overline{w} \tag{2}$$

- *Station assignment:* Each task j must be started and finished in the time span covered by that station where the workplace is located to which j is assigned.

$$s_j \geq \sum_{i=1}^{\overline{w}} ts(i) \cdot x_{ij} \qquad \text{for all } j = 1,...,n \tag{3}$$

$$s_j + t_j \leq \sum_{i=1}^{\overline{w}} (ts(i)+c) \cdot x_{ij} \qquad \text{for all } j = 1,...,n \tag{4}$$

- *Precedence relations* require that a task j can only be started if all predecessors are finished:

$$s_h + t_h \leq s_j \qquad \text{for all } j = 1,...,n \text{ and } h \in P_j \tag{5}$$

- For each task j, a *station interval* $[E_j, L_j]$ is given explicitly if assignment is restricted to a fixed station ($\alpha_5 = fix$ with $E_j = L_j$) or a segment of the line ($\alpha_5 = type$) and implicitly due to precedence constraints based on earliest and latest starting times (see Section 4.2). In any case, a basic interval is given due to the inequality $1 \le E_j \le L_j \le K$. The station interval restricts the workplaces to which task j can be assigned in the following manner:[3]

$$\sum_{i=(E_j-1)\cdot W+1}^{L_j \cdot W} x_{ij} = 1 \qquad \text{for all } j = 1,...,n \tag{6}$$

- The binary variables g_{ijh} indicate whether or not two tasks h and j are *assigned to the same workplace* i. This is achieved by the following two restrictions:

$$g_{ijh} \ge x_{ij} + x_{ih} - 1 \qquad \text{for all } i = 1,...,\overline{w};\ j, h = 1,...,n \text{ and } j \ne h \tag{7}$$

$$g_{ijh} \le x_{ij} \qquad \text{for all } i = 1,...,\overline{w};\ j, h = 1,...,n \text{ and } j \ne h \tag{8}$$

- *Exclusive operation:* Two tasks j and h assigned to the same workplace i must not overlap. Either j must be finished before h is started ($v_{jh} = 1$) or vice versa ($v_{jh} = 0$). The disjunctive constraints (9) and (10) use a sufficiently large number $M \ge c \cdot m$ to ensure that one of the two cases is selected while the other is represented by a redundant constraint. Furthermore, both restrictions are automatically fulfilled when the tasks j and h are not jointly assigned to workplace i ($g_{ijh} = 0$). By (11) it is ensured that v_{jh} is zero if the tasks j and h do not share the same workplace.

$$s_j + t_j \le s_h + (2 - g_{ijh} - v_{jh}) \cdot M \qquad \text{for all } i = 1,...,\overline{w};\ j, h = 1,...,n \text{ and } j \ne h \tag{9}$$

$$s_h + t_h \le s_j + (1 - g_{ijh} + v_{jh}) \cdot M \qquad \text{for all } i = 1,...,\overline{w};\ j, h = 1,...,n \text{ and } j \ne h \tag{10}$$

$$v_{jh} \le \sum_{i=1}^{\overline{w}} g_{ijh} \qquad \text{for all } j, h = 1,...,n \tag{11}$$

- If two tasks j and h are *related by mounting position*, they must be assigned to the same workplace or to different stations:

$$\sum_{i=(k-1)\cdot W+1}^{k \cdot W} (x_{ij} + x_{ih} - g_{ijh}) \le 1 \qquad \text{for all } k = 1,...,K;\ (j,h) \in MR \tag{12}$$

- All pairs (j,h) of *workplace incompatible tasks* must not share the same workplace:

$$g_{ijh} = 0 \qquad \text{for all } i = 1,...,\overline{w};\ (j,h) \in WI \tag{13}$$

- All pairs (j,h) of *station incompatible tasks* must not share the same station:

$$\sum_{i=(k-1)\cdot W+1}^{k \cdot W} (x_{ij} + x_{ih}) \le 1 \qquad \text{for all } k = 1,...,K;\ (j,h) \in SI \tag{14}$$

- Chains for an *extra-long task* j must be executed without a break:

$$s_{f(j)} + t_{f(j)} - s_j = \bar{t}_j \qquad \text{for all } j \in EL \tag{15}$$

- Definition of *binary variables*:

$$y_i, x_{ij}, g_{ijh}, v_{jh} \in \{0;1\} \qquad \text{for all } i = 1,...,\overline{w};\ j, h = 1,...,n \tag{16}$$

Modelling is an important milestone in understanding a problem's structure and complexity. Furthermore, it always holds some capability for solving the problem directly. However, preliminary experiments show that standard optimization software is not able to solve instances of VWALBP-1 of real-world size in a satisfying manner. Sometimes, they do not find a feasible solution at all.

3 Notice that the variables x_{ij} with $i \le (E_j - 1) \cdot W$ and $i > L_j \cdot W$ can be set to zero or eliminated from the model.

This negative finding is due to the NP-hardness of VWALBP-1. Additionally, deciding whether or not a feasible solution exists (and finding a feasible solution) is NP-hard, too, because the number of stations is restricted from above by K and, thus, the number of workplaces must not exceed $\overline{w} = K \cdot W$. Due to this restriction, finding a feasible solution is as hard as solving VWALBP-F with prespecified $m = \overline{w}$ which is NP-hard as a generalization of SALBP-F.

4 Problem analysis and reduction procedures

Due to the complexity it is helpful for finding good or even optimal solutions to analyze the problem's structure and to reduce search effort by applying lower bound arguments and logical tests. In the following, we describe such approaches which can be applied to VWALBP-1 independent of the solution procedure used afterwards. In order to go without special cases for extra-long tasks, we assume that they are already transformed into chains as described in Section 3.

4.1 Lower bounds

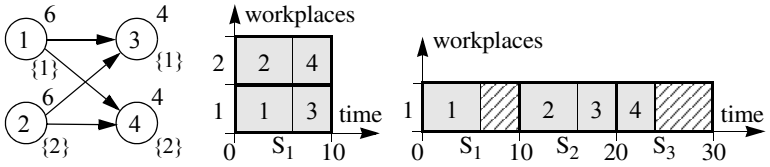

Figure 7. Saving a workplace by parallel work (c = 10)

Though SALBP-1 is a special case of the new problem VWALBP-1 (for $W = 1$ and no assignment restrictions), it is not a relaxation, because the assumption of strictly serial work in SALBP is an additional restriction. Thus, special bounds for SALBP-1 that are based on the property that tasks are never performed in parallel (e.g., the one-machine bound proposed by Johnson 1988) are not applicable to VWALBP-1 (and 2ALBP-1 as well; cf. Bartholdi 1993; Becker 2007, ch. 2.3.2.2). An example is given in Figure 7. The optimal VWALBP-1 solution in the left Gantt chart requires two workplaces while the optimal SALBP-1 solution in the right chart requires three workplaces due to task 4 having to wait until both predecessors (which cannot share a workplace) are finished.

Other bounds for SALBP-1 are based on the binpacking problem BPP-1 which is to minimize the number of equal-sized bins (stations) necessary to pack a set of items (tasks) with given sizes (task times) and, thus, constitutes a relaxation of SALBP-1 by omitting precedence constraints (cf. Scholl 1999, ch. 2.2.1.5). Binpacking based bounds are also valid for VWALBP-1, because BPP-1 relaxes the SALBP restriction of serial work.

We use only inexpensive binpacking bounds (for further bounds we refer to Martello and Toth 1990, Scholl et al. 1997, Fekete and Schepers 2001).

- The *capacity bound* LB_1 utilizes that the total task time t_{sum} must be subdivided among workplaces each of which has a capacity of c time units and that the number of workplaces is integral (cf. Baybars 1986; Martello and Toth 1990, pp. 224): $LB_1 = \lceil t_{sum} / c \rceil$
- The *counting bound* LB_2 counts the number of tasks j with $t_j > c/2$, because all of those tasks have to be assigned to different workplaces. LB_2 can be strengthened by adding half of the number of tasks (rounded up to the next integer if necessary) with task time c/2, because two of them may share one workplace (Johnson 1988; Scholl 1999, ch. 2.2.2.1).
- The *counting bound* LB_3 generalizes LB_2 with respect to thirds of c and is computed by adding up weights: All tasks j with $t_j > 2c/3$ are given the weight 1, because they cannot be combined with any other of the tasks considered. Tasks with $t_j \in (c/3, 2c/3)$ get the weight 1/2,

because two of them may share a station. The tasks with $t_j = c/3$ and $t_j = 2c/3$ are weighted with 1/3 and 2/3, respectively (Johnson 1988; Scholl 1999, ch. 2.2.2.1).

- *Extended bin packing bounds:* The logic behind the simple counting bounds is combined and extended in several manners in order to define more sophisticated bound arguments (cf. Martello and Toth 1990, Berger et al. 1992, Scholl et al. 1997).

4.2 Station intervals and time windows

Due to precedence relations, indivisibility of tasks and integrality of task times, the set of stations to which a task can be assigned is reduced in addition to pre-specified station restrictions. In the following, we describe how to compute earliest and latest stations based on earliest and latest starting times of tasks. In order to simplify description, we do without including station restrictions in the formulae presented, because they can be additionally considered by reducing the resulting intervals accordingly.

In case of SALBP-1, earliest stations E_j and latest stations L_j are computed by applying the logic of LB_1 (cf. Section 4.1) to predecessors and successors, respectively (cf. Scholl 1999, ch. 2.2.1.1):

$$E_j = \left\lceil \left(t_j + \sum_{h \in P_j^*} t_h \right) / c \right\rceil \quad \text{and} \quad L_j = K + 1 - \left\lceil \left(t_j + \sum_{h \in F_j^*} t_h \right) / c \right\rceil \qquad \text{for } j = 1,...,n$$

These values are not valid for VWALBP-1, because each station can have up to W workplaces and, thus, a maximal capacity of $W \cdot c$ time units. However, replacing c by $W \cdot c$ in the above formulae leads to very weak station intervals $[E_j, L_j]$ that do not account for the precedence relations among the predecessors and successors, respectively.

To get smaller station intervals, we apply the logic of critical path analysis from project scheduling (cf. Demeulemeester and Herroelen 2002, ch. 1.2.1; Klein 2000, ch. 2.2.1) to compute earliest starting times es_j and latest starting times ls_j which finally allow for computing earliest and latest stations. In order to compute tight values in a single run, the tasks are considered following their topological numbering, i.e., the earliest starting times es_h of all predecessors $h \in P_j^*$ are already known, when task j is considered. The procedure starts with $es_j = 0$ for all tasks with $P_j = \{\}$.

A first lower bound on the starting time of task j is based on the *critical path* of the precedence graph interpreted as a project network: $\tau_j^1 = \max\{0;\ es_h + t_h \,|\, h \in P_j\}$ for $j = 1,...,n$

Another bound utilizes the logic of the *capacity bound of the resource constrained project scheduling problem* (RCPSP; cf. Klein 2000, ch. 4.1.1.1). In each period of its execution, any task requires one unit of a renewable resource which is available with W units (workers) per period:

$$\tau_j^2 = \left\lceil \left(\sum_{h \in P_j^*} t_h \right) / W \right\rceil \qquad \text{for } j = 1,...,n$$

A third bound is based on *station incompatibilities*. If task j must not be assigned to the same station as any predecessor h, then it must start in a later station than h:

$$\tau_j^3 = \max\{0;\ E_h \cdot c \,|\, h \in (P_j^* \cap \{p \mid (p,j) \in SI\})\} \qquad \text{for } j = 1,...,n$$

If an earliest station E_j' is pre-specified due to station restrictions, an additional bound is given by $\tau_j^4 = (E_j' - 1) \cdot c$. If none is specified, then $E_j' = 1$ is to be set. The largest of these bounds defines an earliest starting time of task j: $\tau_j = \max\{\tau_j^1;\ \tau_j^2;\ \tau_j^3;\ \tau_j^4\}$ for $j = 1,...,n$

Additionally, it has to be considered that a task must be finished in the same station where it started. This results in a *feasible* earliest starting time and a corresponding earliest station (the left-hand side of the inequality indicates the station to which the starting time τ_j corresponds, while the right-hand side represents the station where j would end):[4]

$$es_j = \begin{cases} \left\lceil \frac{\tau_j}{c} \right\rceil \cdot c & \text{if } \left\lfloor \frac{\tau_j}{c} \right\rfloor + 1 < \left\lceil \frac{\tau_j + t_j}{c} \right\rceil \\ \tau_j & \text{otherwise} \end{cases}; \qquad E_j = \left\lfloor \frac{es_j}{c} \right\rfloor + 1 \qquad \text{for } j = 1,...,n \tag{17}$$

For the precedence graph in Figure 8 with $c = 10$ and $W = 3$, we get $es_j = 0$ for $j = 1,...,3$ and $es_4 = \tau_4 = \tau_4^1 = 5$. If at most $W = 2$ workplaces are available at any station, we have an increased earliest starting time $\tau_4 = \tau_4^2 = \lceil 6.5 \rceil = 7$ while τ_4^1 remains unchanged. From $\tau_4 + t_4 > c$ it follows that $es_4 = 10$ and $E_4 = 2$. The same results would be obtained if task 4 were station incompatible to one of the other tasks ($\tau_4^3 = 10$).

Latest starting times and stations are computed in an analogous manner. We get the following upper bounds on the latest starting time by applying the same principles as before (L'_j is a pre-specified latest station, if none is given, then $L'_j = K$):

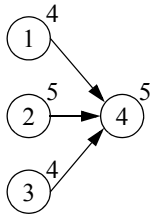

Figure 8. Example

$$\lambda_j^1 = \min \{c \cdot K;\ ls_h \mid h \in F_j\},\ \lambda_j^2 = m \cdot K - \sum\nolimits_{h \in F_j^*} t_h / W$$

$$\lambda_j^3 = \min\{c \cdot K;\ (L_h - 1) \cdot c \mid h \in (F_j^* \cap \{p \mid (p,j) \in SI\})\},\ \lambda_j^4 = L'_j \cdot c$$

The minimum value defines a valid upper bound on the latest starting time:

$$\lambda_j = \min\{\lambda_j^1; \lambda_j^2; \lambda_j^3; \lambda_j^4\} \qquad \text{for all } j = 1,...,n$$

A feasible latest station time and the latest station are computed as follows:

$$ls_j = \begin{cases} \left\lfloor \frac{\lambda_j}{c} \right\rfloor \cdot c - t_j & \text{if } \left\lfloor \frac{\lambda_j - t_j}{c} \right\rfloor + 1 < \left\lceil \frac{\lambda_j}{c} \right\rceil \\ \lambda_j - t_j & \text{otherwise} \end{cases}; \qquad L_j = \left\lfloor \frac{ls_j}{c} \right\rfloor + 1 \qquad \text{for } j = 1,...,n \tag{18}$$

A necessary condition for the existence of a feasible solution obviously is $E_j \le L_j$ for all tasks $j = 1,...,n$. If this condition is not fulfilled, the number of stations K, the cycle time c and/or the maximal number of parallel workplaces W are too restrictive and must be increased.

4.3 Strengthening incompatibilities

Station incompatibility: In addition to prespecified task pairs that must not be assigned to the same station due to technological or organizational reasons, a task pair (j, h) can be added to the set SI if at least one of the following easily checked conditions holds:

- The tasks are related by mounting position but do not jointly fit into one workplace, i.e., $(j,h) \in MR$ and $t_j + t_h > c$.
- Task j is a follower of h and the duration of the longest path from the start of h to the end of j exceeds the cycle time. This situation is illustrated in the right part of Figure 9, where the task pair

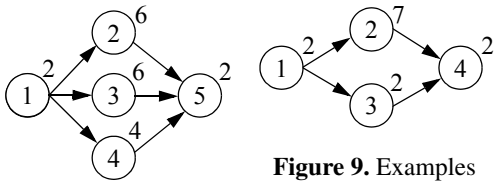

Figure 9. Examples

4 For all tasks with time 0 having predecessors the formula has to be changed to $E_j = \lceil es_j / c \rceil$. Similar modifications are necessary for latest stations but will be omitted to simplify presentation.

$(1,4)$ can be added to SI if $c \leq 10$. Notice that this condition is always true for all task pairs of a chain replacing an extra-long task.

- Task j is an indirect follower of task h and the maximal capacity of a station is not sufficient to perform h and j and all tasks in-between them. This is obviously true if the condition $t_h + t_j + \left(\sum_{i \in F_h^* \cap P_j^*} t_i\right) / W > c$ is fulfilled. An example is given by the left part of Figure 9, where tasks 1 and 5 are station incompatible in case of $c = 10$ and $W = 2$.
- Task j is an indirect follower of task h and on any path connecting them there is a task which is station incompatible to h or j.
- The tasks j and h are related by common mounting positions, i.e., $(j,h) \in MR$, but are also workplace incompatible, i.e., $(j,h) \in WI$, due to other incompatible positions. For example, consider two tasks 1 and 2 with $Q_1 = \{1,2\}$ and $Q_2 = \{2,3\}$ in case of $(1,3) \in IMP$.
- The station intervals of task h and j do not overlap, i.e., $[E_j, L_j] \cap [E_h, L_h] = [\ ]$.

Workplace incompatibility: A task pair (j,h) is in WI if it is not allowed to perform the tasks j and h at the same workplace. This is true if at least one of the following simple conditions holds:

- The task pair is station incompatible, i.e., $(j,h) \in SI$.
- The tasks jointly exceed the cycle time, i.e., $t_j + t_h > c$.
- The tasks are to be executed at incompatible mounting positions, i.e., there exist a $p_1 \in Q_h$ and a $p_2 \in Q_j$ such that $(p_1, p_2) \in IMP$.

If $W = 1$ as in case of SALBP, we get $WI = SI$. In general, it is obviously true that $SI \subseteq WI$.

4.4 Task time incrementing

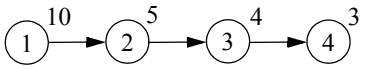

Figure 10. Incrementing times

In order to improve bound values and problem reduction, it is helpful to increase task times based on unavoidable idle times.

We adapt the *extended duration augmentation rule* (*EDAR*) proposed by Fleszar and Hindi (2003) for SALBP-1 which is shortly described as follows: For each task j, it computes a lower bound for the idle time LI_j unavoidable in any WP load which contains task j. This is achieved by solving a modified *subset sum problem* (SSP; cf. Martello and Toth 1990, ch. 4). This SSP consists of selecting a set of items (tasks that can be combined with task j, collected in a *set of compatible tasks* K_j) with maximal sum of task times such that the resulting sum does not exceed the (remaining) capacity $c - t_j$ and all tasks are compatible with each other. Though SSP is also an NP-hard optimization problem, it usually can be solved very quickly by a specialized branch&bound procedure (Fleszar and Hindi 2003). Figure 10 shows an example with $c = 10$. Task 2 can only be combined with task 3, i.e., $K_2 = \{3\}$. Thus, at least an idle time $LI_2 = 1$ occurs in each load containing task 2. For task 3, we get $K_3 = \{2,4\}$ and $LI_3 = 1$. Task 4 has $K_4 = \{3\}$ and $LI_4 = 3$.

Now, the time of some task j can be increased to $t_j' := t_j + LI_j$ without changing the set of feasible solutions. However, this increase can reduce unavoidable idle times of tasks in K_j which, thus, have to be corrected. For example, by increasing the time of task 4 to $t_4' = 3 + 3 = 6$, we get $LI_3 = 0$. The sequence of considering tasks for a possible increase of their task times influeneces the resulting times considerably. For example, starting with tasks 2 or 4, we get $t_2 = t_4 = 6$ and $t_3 = 4$, while $t_2 = t_3 = t_4 = 5$ results when task 3 is considered first. Fleszar and Hindi (2003) consider the tasks according to monotonically decreasing task times. Since determining the set of (re-

ally) compatible tasks K_j is non-trivial, Fleszar and Hindi (2003) include each task h with $t_h \le c - t_j$ in K_j, the time window of which does overlap with the time window of task j.

In order to extend EDAR to VWALBP-1, a difference has to be made between station incompatible and workplace incompatible task pairs as illustrated by an example. Figure 11 contains an instance of VWALBP-1 and its optimal solution for c = 10, K = 1, W = 2 and IMP = {1,2}.

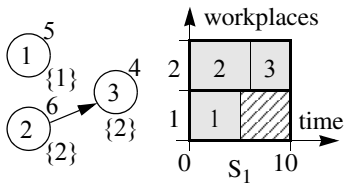

Figure 11.
Workplace incompatible tasks

Task 1 is workplace incompatible to both other tasks, i.e., WI = {(1,2), (1,3)}, while no station incompatible pairs exist, i.e., SI = { }. When task 1 is scheduled, an unavoidable idle time $LI_1 = 5$ time units occurs and $t_1' = 5 + 5 = 10$ is possible. If the additional precedence relation (1,3) is to be considered, the same optimal solution and the same unavoidable idle time $LI_1 = 5$ are obtained. Nevertheless, the task time t_1 must not be increased to 10, because this would exclude the optimal solution (task 3 could not be assigned to station 1 anymore). This example shows that it is not always feasible to increase task times by unavoidable idle times if they are caused by workplace incompatibility. However, if only station incompatible task pairs are responsible for an unavoidable idle time, the task time can be incremented feasibly in any case.

Let $KW_j = \{h \mid h \in V - (\{j\} \cup WI_j)\}$ and $KS_j = \{h \mid h \in V - (\{j\} \cup SI_j)\}$ be the task sets which are compatible to j concerning the same workplace and station, resp. ($KW_j \subseteq KS_j$). In Figure 11, we get $KW_1 = \{\ \}$ and $KS_1 = \{2,3\}$. To compute the *unavoidable idle time* LI_j of a task j, the modified SSP is solved using the set KW_j, while it is necessary to solve another instance of the modified SSP with the set KS_j to compute a lower bound on the *feasible time increment* $\Delta t_j \le LI_j$.

If a task j considered by EDAR is not related by precedence to any of the station compatible but workplace incompatible tasks, i.e., $h \notin (F_j^* \cup P_j^*)$ for all $h \in KS_j - KW_j$, then the task time is increased by the unavoidable idle time, i.e., $t_j' = t_j + LI_j$. If this condition is not fulfilled, then only the increase to $t_j' = t_j + \Delta t_j$ is feasible. If the task time has changed, the unavoidable idle times and minimal increments of station and workplace compatible tasks need an update. In the example, we get $LI_1 = 5$ and $\Delta t_1 = 1$. If the precedence relation (1,3) is valid, the task time may be increased only by Δt_1 to $t_1' = 6$. Without this relation, yet an increase by LI_1 to $t_1' = 10$ is feasible.

Since binpacking bounds ignore precedence relations, the difficulty mentioned above does not occur. So, *alternative* task times $t_j'' = t_j + LI_j$ can be used for all tasks j.

4.5 Relationships between related VWALBP-1 instances

Consider two instances A and B of VWALBP-1 which are identical except for the number of available stations with $K(A) > K(B)$. Because A is less restrictive, each feasible solution of B is also feasible for A. Thus, the minimal number of workplaces m*(A) provides a lower bound on the minimal number of workplaces m*(B). Furthermore, a known optimal solution of B can serve as a feasible start solution when solving A.

Figure 12 illustrates this relationship. Instance A with K(A)=2 requires two workplaces, while three workplaces are necessary for instance B with K(B)=1.

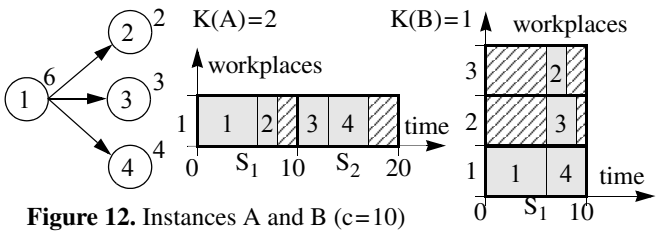

Figure 12. Instances A and B (c=10)

The same relationships are valid if A and B only differ in the maximal number of

parallel workplaces with W(A)>W(B). An example is given in Figure 7. These relationships might be useful as they facilitate solving problems differing only in one parameter in the context of sensitivity analyses.

5 Branch&bound procedure VWSolver

As the new problem VWALBP-1 cannot be solved efficiently by standard software, we develop a branch&bound procedure named VWSolver which might be applied in an optimum-seeking as well as an heuristic version.

5.1 Preprocessing

Before the enumeration is started, the following preprocessing steps have to be performed:

- A *dummy sink node* n (after increasing n by 1) is added to the graph which becomes successor of all tasks without real successors. This additional task gets task time zero, is set compatible to all tasks and does not require a mounting position. Whenever it is assigned to a station, a complete solution has been found.
- Initialize the sets MR (mounting position related task pairs), SI and WI (incompatible task pairs; Section 2 & 4.3). Compute earliest and latest starting times and stations (Section 4.2).
- Apply the modified EDAR to compute *unavoidable idle times, increased* and *alternative* task times (cf. Section 4.4).
- Based on alternative task times compute the *lower bounds* LB_1 to LB_3 (vgl. Section 4.1). The largest one is used as initial *global lower bound* LB.
- Compute potential dominances for the modified Jackson dominance rule (cf. Section 5.3).
- Renumber the tasks such that potentially dominating tasks get smaller numbers than dominated ones while maintaining a topological ordering. First level ties are broken in favor of the smallest value of the latest station, second level ties, in favor of the largest task time and third level ties, in favor of the smaller original number.
- For each task j and each station $k \in [E_j, L_j]$ determine the list of tasks that are compatible to j in station k. This list facilitates enumerating station loads.
- Set the global upper bound to $UB = W \cdot K + 1$ to indicate that no feasible solution is known.

5.2 Branching scheme and bounding

Branching: Similar to SALOME for SALBP-1, the branching is organized as a laser search (depth-first search) in a *stationoriented* manner (cf. Scholl and Klein 1997). In each branching step at a level k (k= 1,...,K) of the enumeration tree, a load S_k for station k is generated. Whenever the task n (dummy sink node) is assigned, the solution (task-station assignments) is examined and stored as incumbent solution with updated UB if the number of workplaces is lower than the former UB. In either case, a backward step is performed to consider another load for the current station k. If all loads for k (based on the current branching path on the levels $1,..., k-1$) have been examined, a backward step to level (station) $k-1$ is made. This is continued until all loads have been examined in level 1. Since then all solutions of the VWALBP-1 instance have been examined explicitly or implicitly, the search is finished and the current incumbent solution is optimal, i.e., UB is equal to the minimal number of workplaces.

Each node h (with h being a consecutive number following the depth-first search) at a level k of the enumeration tree represents a *partial solution* T_h with station loads $S_1, ..., S_{k-1}$ and a *residual problem* R_h which consists of the subgraph with all yet unassigned tasks and has the same structure as the original problem. Each arc from level k to k+1 represents a station load S_k generated by branching. The (locally optimal) objective function value of the partial solution $F(T_h)$ is given by the number of workplaces required up to station k. This value results from the (optimal) objective value $F(T_p)$ of the ancestor node p plus the minimal number of workplaces required for station k (see the *station problem* described below).

Bounding: For the residual problem R_h, a lower bound $LB(R_h)$ is computed by LB_1 based on alternative (increased) task times (cf. Section 4.1 and 4.4). In the first node 0 at level 1, the partial solution T_0 is empty and the residual problem R_0 is equal to the original overall problem, i.e., $LB(R_0) = LB$. In each node h, a local lower bound on the minimal number of workplaces required in a complete solution is given by $LB(T_h) = F(T_h) + LB(R_h)$ (cf. Scholl 1999, ch. 4.1.2.1).

Figure 13 shows an example instance and the resulting enumeration tree.[5] Above the nodes, the cumulative number $F(T_h)$ of workplaces required and the lower bound $LB(T_h)$ is noted. The load $S_1 = \{1,2,3,4\}$ leading to node 1 requires two workplaces with WP loads {1,3} and {2,4} (optimum of the station problem) such that $F(T_1) = F(T_0) + 2 = 2$. The residual problem R_1, consisting of task 5 only, requires (at least) $LB(R_1) = 1$ additional workplace(s). So, we get the lower bound $LB(T_1) = 2 + 1 = 3$.

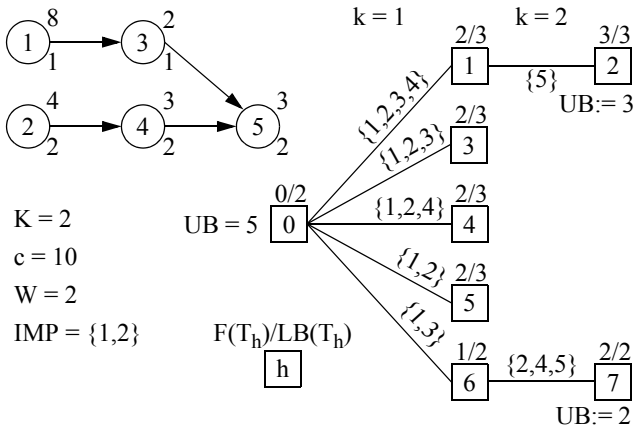

Figure 13. Enumeration tree

In node 2, a first feasible solution is found and stored as incumbent solution, because the initial $UB = 5$ is improved to $UB = 3$. Due to this improved global upper bound all nodes h with $LB(T_h) \geq UB$ are fathomed (nodes 3 to 5, here). In node 7, a further improvement to UB=2 is reached. Due to LB=UB, the complete procedure can be stopped with the optimal solution $S_1 = \{1,3\}$, $S_2 = \{2,4,5\}$ getting along with 2 workplaces (workers).

In order to describe how station loads S_k are built and analysed, the following issues have to be discussed in more detail:

A task j is *assignable* to a temporary load S_k if station k is in its station interval ($k \in [E_j, L_j]$), all predecessors are already assigned ($i \in \bigcup_{h=1}^{k} S_h$ for all $i \in P_j^*$), task j is not station incompatible to any task already assigned to station k ($(i,j) \notin SI$ for all $i \in S_k$), and there is sufficient capacity left ($t_j + \sum_{i \in S_k} t_i \leq W \cdot c$).

Feasibility of station loads: In contrast to SALBP, where successively including assignable tasks automatically leads to feasible station loads, this is not generally true for VWALBP-1 with $W \geq 2$. The aggregated capacity condition considered in the definition of "assignable" is necessary but not sufficient, because the cycle time must additionally be

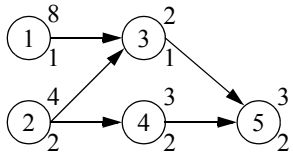

Figure 14. Station problem

5 All sets Q_j only have a single element. So, we specify these elements rather than the sets to ease presentation.

observed in each workplace (disaggregated condition $\sum_{i \in WL} t_i \leq c$ for each WP load WL in station k). However, even this is not sufficient for the existence of a feasible solution, because precedence constraints might require to introduce additional idle time into the schedule (cf. Section 2.3). For example, consider the instance in Figure 14 with $W = 2$ and $c = 10$. The station load $S_1 = \{1,2,3,4,5\}$ fulfills the aggregated and the WP loads $WP_1 = \{1,3\}$ and $WP_2 = \{2,4,5\}$ fulfill the disaggregated capacity conditions. Nevertheless, it is not possible to schedule these loads feasibly, because task 5 must wait for task 3 and, thus, cannot start before time 10. As a result, S_1 is not feasible for station 1.

The difficulty discussed before is called *station problem*. It consists of determining the minimal number of parallel workplaces required and the corresponding WP loads for a given station load S_k, if this is possible. If more than W workplaces are required, any task cannot be finished within the single station considered or station incompatible task pairs are contained in S_k, then the load S_k is identified as being infeasible. The station problem is an (NP-hard) optimization problem by itself and, thus, is solved by a suited exact (branch&bound) procedure described in Section 6.

Load generation: The loads S_k required for branching a node h, are generated by systematically combining all tasks that are assignable to station k. In a first step, all tasks j with $L_j = k$ are assigned (*prefixing*), because they must not be executed at a later station. In particular, this is true for elements of the chain representing an extra-long task. Whenever the first subtask has been assigned to a station, the subsequent subtasks are immediately prefixed to the successive stations without break. The prefixing is valid for all loads S_k emerging from the current node h of the tree. If already the prefixed load is identified to be infeasible (see above), node h is fathomed. Otherwise, the prefixed load is appended to form different station loads S_k by assigning assignable tasks in a lexicographic manner based on the renumbering (cf. Section 5.1). After adding each task, feasibility is checked. If the partial load is found to be infeasible, the task is removed and the next one examined. Otherwise, the load is extended further. This is repeated until no task can be assigned feasibly. The resulting load S_k is branched by an arc emerging from node h to built a new node in the tree which is considered next within the laser search. After having examined this node and its complete subtree (laser search), the task added to S_k the latest is removed and replaced by the next (higher-numbered) task and the enumeration process is continued as described above. This is repeated until no further tasks can replace a just removed task. Then, this reduced load is branched also. The tree in Figure 13 shows that by this procedure loads are built in a lexicographic order with subsets built later than their supersets. By doing so, the procedure tries to use as many as possible parallel workplaces first in order to find a feasible solution early and to use the advantages of parallelism as completely as possible.

5.3 Dominance rules

Dominance rules are used to reduce the size of the enumeration tree and speed up the solution procedure. A (currently considered) partial solution T_h is considered as being *dominated* by another (already or still not considered) partial solution T_i if it is guaranteed that solving the residual problem R_h does not lead to a better overall solution than solving the residual problem R_i. Then, the partial solution T_h and the complete subtree following the node h can be excluded from further search because only one optimal solution is searched for (cf. Scholl and Becker 2006). The following dominance rules, generalizing respective rules for SALBP-1, are applied in VWSolver:

Maximum load rule

For SALBP-1, Jackson (1956) proposed the so-called *maximum load rule* (furthermore, see Scholl 1999, ch. 2.2.2.2). A station load S_k is maximal if no unassigned task can be added feasibly. Each partial solution T_h that contains a non-maximal station load is dominated because there always exists another partial solution T_i that only contains maximal loads and requires the same or a lower number of stations.

In case of VWALBP-1, the definition of maximiality must be modified, because additionally assigning a task to a load might increase the number of workplaces required. Thus, a station load is defined to be maximal here, if no assignable task can be added without increasing the minimal number of workplaces required (cf. the station problem in Section 6). Non-maximal loads are excluded from branching.

In Figure 13, the nodes 3, 4, and 5 are fathomed due to bounding. Additionally, they are dominated according to the maximum load rule. Even if task 2 can be assigned to $S_1 = \{1,3\}$ (node 6) feasibly, this load is maximal, because the number of workplaces would increase from 1 to 2.

Feasible set dominance rule

Enumeration regularly generates partial solutions that contain the same set of tasks but differ in the way the tasks are assigned to stations. By construction, these sets are *feasible* in the respect that they also contain all predecessors of each task included. In case of SALBP-1, a partial solution T_h corresponding to a feasible set of tasks $J(T_h)$ which is equal to (or a subset of) the feasible set $J(T_i)$ of another partial solution T_i already considered is dominated if it does not require less stations. In order to apply this dominance concept, the already generated feasible sets and information on the best solution so far have to be stored. This can be done in an efficient manner by means of a tree structure as proposed by Nourie and Venta (1991). An extended version is described by Scholl and Klein (1999).

For VWALBP-1, the dominance definition must be modified to cope with additional constraints. The following conditions are required to constitute a dominance of partial solution T_i over T_h:

- All tasks already assigned in T_h are also assigned in T_i, i.e., the sets of assigned tasks have the subset relationship $J(T_h) \subseteq J(T_i)$.
- Partial solution T_i requires at most as many workplaces as T_h, i.e., $F(T_i) \leq F(T_h)$.
- The partial solution T_i does not distribute the workplaces among more stations than T_h, i.e., $K(T_i) \leq K(T_h)$ with $K(T_h)$ denoting the number of stations required for T_h.

While the first two conditions are also required for SALBP-1, the latter one is necessary due to the restricted number of stations K and the station restrictions in VWALBP-1. Consider the instance of Figure 15 with c = 10. The partial solutions T_1 and T_2 contain the same tasks and T_2 requires one workplace less. Nevertheless, there is no dominance relationship between both partial solutions, because it is not sure that the residual problem R_2 has a feasible solution.

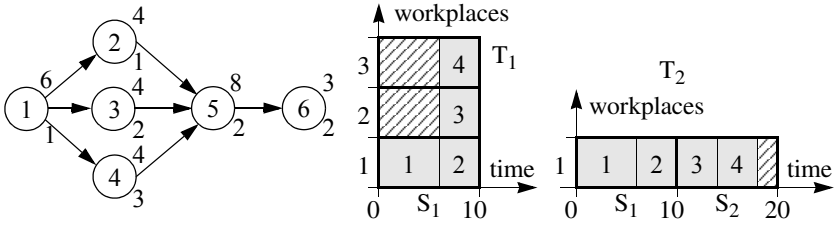

Figure 15. Efficient partial solutions

For example, assume that task 6 is fixed to station 3. Then, only T_1 leads to a feasible solution. So, there is a tradeoff between number of workplaces and number of stations such that partial solutions are *efficient* and, thus, not dominated even if additional workplaces are required provided that they are compensated by a reduced number of stations.

To apply the feasible set dominance rule in an efficient manner, the partial solutions are stored in a dynamic tree structure which is a generalization of the structure proposed by Nourie and Venta (1991) and the dominance relationships are reduced to the case $J(T_h)=J(T_i)$. For each set of assigned tasks, the tasks contained are encoded by a path in this dynamic tree (for details see Nourie and Venta 1991 as well as Becker 2007, ch. 4.1.3.2) which terminates in a memory node that stores the minimum number of workplaces and the corresponding number of stations required by the best partial solution with this task set so far. If several efficient combinations of both numbers exist (see above), further memory nodes are appended in a list.

Whenever a partial solution T_h has been built (before branching it), the tree structure is used to retrieve the memory node(s) which correspond to $J(T_h)$. If such a node does not exist, this task set has been considered first and a new memory node is inserted into the structure and filled with the information of T_h. If at least one such memory node already exists and one of the partial solutions T_i stored there dominates T_h, the partial solution T_h is not used for branching. Otherwise, if T_h dominates a stored T_i, then the information of T_i is overwritten by that of T_h in the respective memory node. If T_h represents an additional efficient solution for $J(T_h)$, a further memory node is appended in the list for $J(T_h)$.

Jackson dominance rule

The *Jackson dominance rule* (Jackson 1956, strengthened by Scholl and Klein 1997) is applied to reduce the number of alternative loads which have to be considered for a certain station k. It is based on potential dominance. In case of SALBP-1, a task h *potentially dominates* another task j that is not related to h by precedence, if $F_j \subseteq F_h^*$ and $t_j \leq t_h$ hold. The rule excludes a maximal station load S_k from being branched if at least one task $j \in S_k$ can be replaced feasibly by a still unassigned task h which potentially dominates j. This rule utilizes the fact that all successors of task j are successors of h as well and cannot start before h is finished. Hence, the sequence of j and h is not important for the successors of j such that replacing j by h does not exclude later task assignments which would be possible when j remained. The condition $t_j \leq t_h$ guarantees that the station utilization will not decrease if h replaces j in S_k.

For VWALBP-1, the definition of the *potential dominance* must be extended. In addition to the above-mentioned conditions, the following ones must be fulfilled to constitute a potential dominance of task h over task j:

- The latest station of h is not smaller than that of task j, i.e., $L_h \geq L_j$.
- The task j is also workplace compatible to all tasks to which h is workplace compatible, i.e., with respect to incompatibility the inclusion $WI_j \subseteq WI_h$ must hold.
- The task h is mounting position related to all tasks that are mounting position related to task j, i.e., $MR_j \subseteq MR_h$.

The modified rule excludes each station load which contains a task j that can be replaced feasibly by a potentially dominating task h without increasing the number of workplaces required. If all conditions are fulfilled as equations, the lower-numbered task is defined to be dominating.

Figure 16 shows an instance with unspecified mounting positions for task 2. If $Q_2=\{1\}$, task 1 potentially dominates task 2 and task 4. The station load $S_1=\{1,3,4\}$ (first chart) dominates the alternative load $\{2,3,4\}$ (second chart). Both completed solutions shown are optimal but only one must be considered. If $Q_2 = \{1,2\}$ is given, task 1 does not potentially dominate task 2 anymore, because task 2 is mounting position related to task 6 which is not the case for task 1. Now, the first solution based on $S_1 = \{1,3,4\}$ is infeasible such that dominating $\{2,3,4\}$ would exclude the optimal solution.

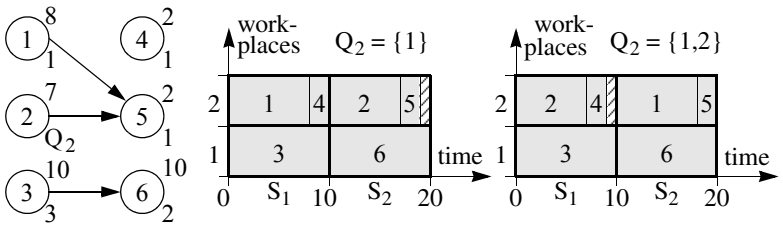

Figure 16. Example for Jackson dominance rule (c=10, K=2, IMP={1,3})

Empty-station rule

Due to pre-specified earliest stations, it can be necessary to leave stations empty, i.e., no workplace is installed at this segment of the line. This is typical in practical applications, because empty stations are systematically provided in order to have buffers for future extensions.

Consider the instance in Figure 17 with task 4 fixed to station 3, where the required immotile machine is installed. An optimal solution is given by $S_1=\{1,3\}$, $S_2=\{\}$, $S_3=\{2,4,5\}$ with two workplaces (workers). Thus, an empty load must be assigned to station 2 in the enumeration tree. In general, an empty station is always generated (as the last load considered) for branching S_k, if there exists an unassigned task j with $E_j > k$. Immediately after building an empty load S_k, the feasible set dominance rule must not be applied. In the example, this rule would fathom T_6 because it seems to be dominated by T_4.

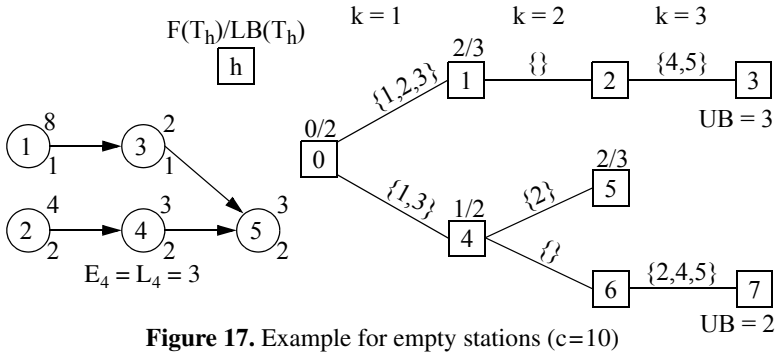

Figure 17. Example for empty stations (c=10)

5.4 Heuristic version of VWSolver

The branch&bound procedure VWSolver described above can easily be modified to a heuristic procedure (called VWSolv-H) by two simple modifications:

- The computation time for solving the station problem (as described in Section 6) is restricted to a limit of Δ_1 sec each. In case of a time-out, corresponding station loads S_k are considered infeasible and, thus, are not branched in VWSolver.
- Another time limit Δ_2 is set for the branching process of VWSolver. After having found an initial feasible solution, the procedure is enforced to (heuristically) fathom all nodes on the current branching path and to return to the root node of the tree always after Δ_2 sec. This enables the procedure to examine further loads for station 1 and all further stations in order to diversify the search. By doing so, a main disadvantage of laser search is overcome, because the search will never return to the first levels of the tree in case of large instances and restricted computation time.

6 Solving the station problem

To evaluate a station load S_k within the procedure VWSolver, the station problem mentioned in Section 5.2 has to be solved. This is done by the exact branch&bound procedure WPA (Work-Place Assignment) described in the following.

6.1 Refined problem statement and solution idea

The *station problem* consists of assigning the tasks j of a given load S_k to workplaces i=1,...,W and determining starting times s_j from the intervals $[0,\ c-t_j]$ such that the number of opened workplaces is minimized. If it is not possible to assign all tasks $j \in S_k$ to at most W workplaces in a feasible manner, the station problem has no feasible solution and the station load must not be used for branching in VWSolver. The station problem is a special case of VWALBP-1 with K = 1 and the task set S_k which is NP-hard, too. Thus, BPP-1 is a relaxation of the station problem and binpacking bounds (cf. Section 4.1) are valid also.

Furthermore, the station problem is equivalent to a special type of *resource constrained project scheduling* problem with given due date (equal to the cycle time c). Each task corresponds to a job of the project and each workplace represents a unit of a single renewable resource which is available in W units per period. The objective consists in minimizing the resource level required (maximal resource usage in any period) such that the due date is fulfilled. Due to this relationship to resource constraint project scheduling, the *serial scheduling scheme* (cf. Kelley 1963; Kolisch 1995, ch. 5.2.1) is adapted to the station problem in order to additionally observe workplace incompatible task pairs and mounting position related task pairs.

6.2 Preprocessing

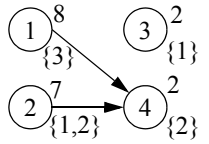

Figure 18. Task grouping

Before the enumeration starts, all tasks that are related by mounting position are *grouped* together. Initially, each task j defines a group G_j for ist own. In the next step, any two groups G_a and G_b are united if there is at least one task pair $(j,h) \in MR$ with $j \in G_a$ and $h \in G_b$. This process is repeated until no further grouping is possible and ends up with groups G_g (with consecutive group numbers g). Thus, if the sum of task times $t(G_g)$ of any group G_g exceeds the cycle time, no feasible solution exists for station load S_k and WPA terminates immediately. The same is true if a group contains tasks that are workplace incompatible to each other. In the example of Figure 18 with $S_k = \{1,2,3,4\}$ and $c = 10$, we get two groups $G_1 = \{1\}$ and $G_2 = \{2,3,4\}$. Though tasks 3 and 4 are not directly related they belong to the same group because they are both related by mounting position to task 2. The group G_2 has a time of $t(G_2) = 11$ time units such that no feasible solution exists.

Two *groups* G_g and G_p are *incompatible* to each other and must not be assigned to the same workplace if one of the following conditions is true:

- The total task time exceeds the cycle time, i.e., $t(G_g)+t(G_p)>c$.
- There exists a workplace incomptaible pair $(j,h) \in WI$ with $j \in G_g$ and $h \in G_p$.

All pairs of incompatible groups are computed and stored in GI.

Due to the fact that the workplaces are identical, at least one group can initially be assigned to workplace 1. Furthermore, it might be possible to assign a further group which is incompatible to the first one, to another workplace and so on. This technique is called *forward checking* and ex-

plained in Section 6.4. In particular, subtasks of an extra-long task (except for the last) are incompatible to each other task and require a workplace for their own. In Figure 19, we get the groups $G_1 = \{1,4\}$, $G_2 = \{2\}$ and $G_3 = \{3,5\}$. Initially, w.l.o.g., G_1 can be assigned to workplace 1 and G_3 to workplace 2, because these groups are incompatible to each other.

An *initial lower bound* on the number of workplaces is computed by means of LB_1. Another lower bound is given by the number of workplaces already installed by forward checking. In the example, both bound arguments result in two workplaces (workers).

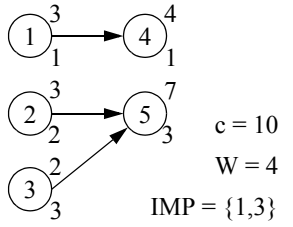

Figure 19. Forward checking

In the branching scheme of VWSolver the load S_k is built to extend a partial solution T_h to a partial solution T_p (see Section 5.2). If $F(T_h)+LB_1(S_k)+LB(R_p) \geq UB$ holds, it is not necessary to solve the station problem at all and WPA terminates immediately, because a solution improvement is not possible by branching the load S_k irrespective of the number of workplaces.

For each task, a time window $[es_j, ls_j] \subseteq [0, c-t_j]$ of starting times is computed based on the formulae in Section 4.2 applied to the set S_k ignoring all other tasks. In order to consider promising solutions first, the tasks are reindexed in non-decreasing order of earliest starting times es_j. Table 3 contains the time windows for the example of Figure 19, where the task numbering remains valid.

j	1	2	3	4	5
es_j	0	0	0	3	3
ls_j	3	0	1	6	3

Table 3. Time windows

Based on time windows, additional precedence relations may be deduced. If $ls_h + t_h \leq es_j$, the additional arc (h,j) can be included. Though additional arcs do not exclude feasible solutions they restrict the search effort. Furthermore, they allow for further strengthening of time windows. Thus, the procedure iterates on updating time windows and adding arcs until no additional arc can be found. In the example, the arcs (2,4) and (3,4) can be added.

Frequently, the station problem has no feasible solution. To identify such instances quickly, a logical test is applied to each group G_g with more than one task. This test is based on two polynomially solvable makespan-minimizing one-machine problems (cf. Lawler 1973). Both problems consider the members of G_g as jobs which have to be performed one after the other such that the makespan is not larger than the cycle time. In the first problem, *heads* $a_j = es_j$ are considered for all jobs $j \in G_g$. The minimal makespan is obtained by scheduling the jobs in non-decreasing order of heads as early as possible. In the second problem, *tails* $n_j = c - ls_j$ are considered instead of heads and the optimal solution is computed by scheduling the jobs in non-increasing order of tails as early as possible. If at least one of the minimal makespans exceeds the cycle time, no feasible solution exists for the station problem of S_k and this load is not used for branching in VWSolver.

Consider the example in Figure 20 with $c = 10$, $IMP = \{\}$ and $G_1 = \{1\}$, $G_2 = \{2\}$, $G_3 = \{3,4\}$. Due to $es_3 = 6$ and $es_4 = 5$, job 4 is scheduled at $s_4 = 5$ and job 3 at $s_3 = 5+3 = 8$. Since the makespan results to $8+4 = 12$, the station problem has no feasible solution.

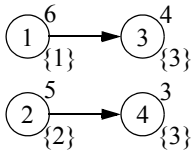

Figure 20. Example

6.3 Branching

WPA applies an adaptation of the serial scheduling scheme to all precedence-feasible task sequences and systematically enumerated task-workplace assignments. This is an extension of the approach of Patterson et al. (1989) for the resource constrained project scheduling problem.

In order to explain the branching process in more detail, two values have to be considered for each workplace, both initially set to zero:

tc_i blocked capacity of workplace i with $tc_i \in [0,c]$; capacity is used for executing assigned tasks, irrespective if a starting time is already fixed or not, and by unavoidable idle times

d_i current schedule time of workplace i (finishing time of the latest task scheduled for i so far); $d_i \in [0, c-t_{min}]$ with t_{min} denoting the shortest time of a task in S_k

In each branching step, an assignable task $j \in S_k$ (all predecessors are already scheduled) is assigned to a workplace $i \in \{1,\ldots,W\}$ to build a new subproblem (node). If j is a member of a group and another of the members has already been assigned to a workplace before, this decision is predetermined. If j is considered as the first member of its group G(j), its assignment increases the blocked capacity to $tc_i = tc_i + t(G(j))$.

Following the logic of the serial scheduling scheme, the just assigned task j is scheduled as early as possible at time $s_j := \max\{d_i, s_h + t_h \,|\, h \in P_j \cap S_k\}$. If $s_j > ls_j$, then task j cannot be scheduled feasibly and the subproblem is fathomed. Otherwise, if $s_j > d_i$, then an idle time results between j and the former last task in workplace i such that the additional capacity is blocked, i.e., $tc_i := tc_i + s_j - d_i$. The new schedule time is set to the ending time of j, i.e., $d_i := s_j + t_j$.

In each node of the enumeration tree, branching alternatives emerge from the set of assignable tasks and their feasible workplace assignments. These alternatives are enumerated by selecting the tasks in order of non-decreasing earliest starting times es_j. If the group G(j) containing the chosen task j is already assigned to a workplace, only one subproblem is built. Otherwise, several subproblems are constructed by alternatively assigning j and, thus, G(j) to all workplaces i (in increasing order of their numbers) where this is feasible with respect to the following *assignment rules*:

- The workplace i must not contain a group which is incompatible to G(j).
- The remaining capacity of i must be sufficient for the complete group, i.e., $tc_i + t(G(j)) \le c$.
- The current schedule time of i must not exceed the latest starting time of any task h in G(j), i.e., $d_i \le ls_h$ must hold for all $h \in G(j)$.
- If in a node more than one workplaces are still uninstalled (no task has been assigned so far), only one subproblem is built by assigning task j to the lowest-numbered one, because all (empty) workplaces are identical and, thus, still exchangeable.

Branching is organised as a laser search, i.e., in each node only one subproblem is built at a time and immediately followed. Further subproblems are generated each time the procedure returns.

Figure 22 repeats the instance of Figure 19 and the results of the preprocessing including additional arcs (dashed). At the arcs of the enumeration tree, branching decisions are noted. The nodes are numbered consecutively, node weights are the number w of already installed workplaces and the lower bound $LB_1(S_k)$. In the root node 0, the first subproblem (node 1) is obtained by scheduling task 1, which has been assigned to workplace 1 together with task 4 by preprocessing, at time 0. In node 1, forward checking (see Section 6.4) reveals that WP_3 is the only feasible workplace for task 2 (in WP_1 task 2 cannot be scheduled at its latest time ls_2 and $t(G_2) + tc_2 > c$ excludes WP_2). The remaining branching decisions in the nodes 2 to 4 are predetermined due to already having assigned all tasks to workplaces. The tasks are scheduled as early as possible in each case. In node 5, a first feasible solution with UB=3 workplaces is obtained (see the first Gantt chart). Since no alternative subproblems exist, the search traces back to node 0, where task 2 is to be con-

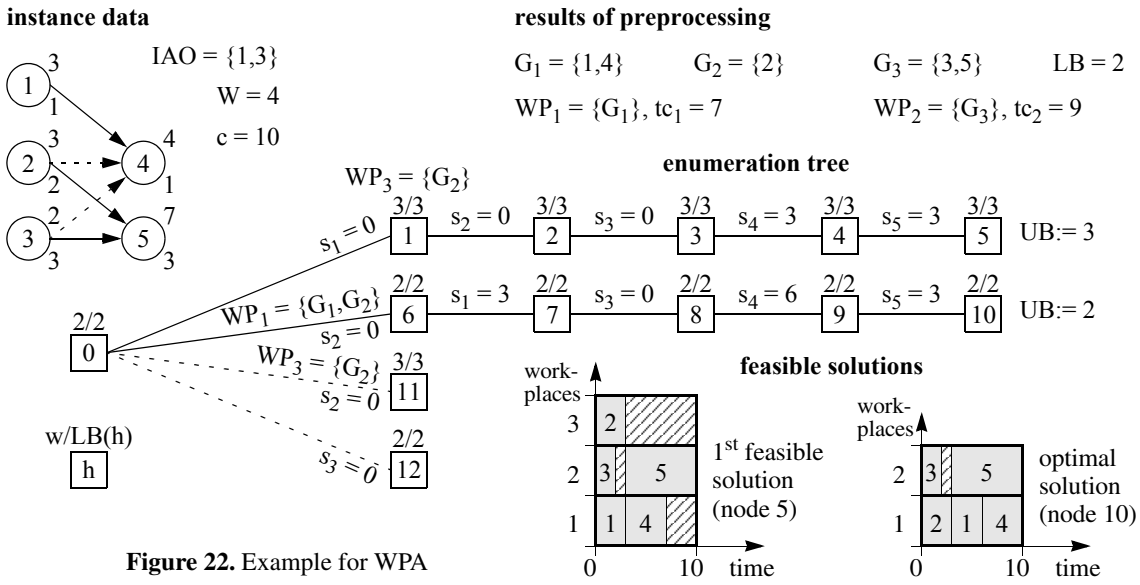

Figure 22. Example for WPA

sidered next. It can be assigned to WP_1 or WP_3 defining two subproblems while $t(G_2) + tc_2 > c$ excludes WP_2. Having task 2 assigned to WP_1 and scheduled at $s_2 = ls_2 = 0$ leads over further predetermined branching decisions to an improved feasible solution with UB=2 workplaces (second Gantt chart). After tracing back to node 0, the remaining subproblems (task 2 assigned to and scheduled in WP_3 and task 3 scheduled in WP_2) need not be built anymore, because UB is equal to the global lower bound LB(0)=2 and, thus, the second feasible solution proven to be optimal.

6.4 Logical tests and reduction rules

The effort of enumerating is drastically reduced by applying several logical tests and reduction rules (for details see Becker 2007, ch. 4.2).

- As already mentioned, **forward checking** is a projection approach which tries to reduce the domains of free variables in order to avoid unneccessary branching decisions which would not result in feasible or improved solutions (cf. Klein 2002).
  Always after scheduling a task, forward checking examines, if unassigned groups can be excluded from some of the workplaces now. To be more specific, a workplace i is removed from the set of possible workplaces of some unassigned group G_g, if at least one of the assignment rules described in Section 6.3 (with G_g replacing G(j)), is violated. If no possible workplace remains for any unassigned group, the current node is fathomed. If only one workplace remains for a group G_g, this group is immediately assigned to the respective workplace. This includes that if only uninstalled workplaces remain in the set, the group is fixed to the lowest-numbered one because those are still identical. Always after having fixed a task, forward checking is started again in order to possibly find further domain reductions.
  Furthermore, forward checking is used to examine whether already assigned but still unscheduled tasks j can be scheduled in a future step. This requires $d_i < ls_j$ to hold for each such task j. If this is not the case, the node is fathomed.
- **Reduction of double enumeration:** Applied to different task sequences, the serial scheme frequently leads to identical (partial) solutions (cf. Klein 2000, ch. 6.4.2). A simple mechanism to reduce (a part of) the resulting double enumeration works as follows: If task h immediately follows task j in the task sequence and the resulting start time s_h is lower than s_j (or $s_h = s_j$ and $h < j$), the node can be fathomed.

- **Simplified branching:** If all tasks in S_k are not related by precedence, only a single task sequence must be considered, the station problem reduces to a bin packing problem with item incompatibilities. If a feasible packing can be found, starting times can be assigned in an arbitrary manner. Thus, only a single task is considered in each stage of the enumeration tree, branching alternatives only stem from alternative workplace assignments of this task.
- **Storing solutions:** In order to avoid solving a station problem twice (in different nodes of VWSolver), the solutions of already obtained station loads S_k are stored together with the optimal number of workplaces $F^*(S_k)$. The storage is organized in a dynamic tree structure similar to the one used for the feasible set dominance rule (cf. Section 5.3). Whenever a station problem is considered for some load S_k, it is checked if a solution has already been found and stored. Only when it cannot be found, it has to be solved and the resulting solution has to be stored in the structure. If no feasible solution exists, this information is also stored. In case of extra-long task the mechanism can be strengthened because its parts except the last are identical in size and incompatibilities.
- **Extending solutions:** When generating station loads in VWSolver, tasks are added to the trial load step-by-step. To avoid generating infeasible loads, the corresponding station problem is solved in parallel. In some cases, the optimal solution obtained for a trial load S_k can easily be extended to an optimal solution for $S_k \cup \{j\}$ or utilized to prove that no feasible solution exists without resolving the enlarged station problem. If task j additionally fits into one of the already installed workplaces without rearranging the schedule, the optimal values $F^*(S_k)$ and $F^*(S_k \cup \{j\})$ are obviously identical. If the time sum of task j and all tasks that share mounting positions with j is larger than c it is clear that no feasible solution exists. If task j is the first part of an extra-long task, it requires a workplace for its own such that $F^*(S_k \cup \{j\}) = F^*(S_k)+1$ is known. The same result is obtained if $LB_1(S_k \cup \{j\}) > F^*(S_k)$ holds. In both cases this leads to the optimal solution for the enlarged problem or indicates infeasibility for $F^*(S_k)=W$. In all other cases, the station problem must be solved explicitly by applying WPA.

7 Computational experiments

We perform computational experiments to examine the effectivity of VWSolver in solving VWALBP and its special cases. Besides data sets from the literature, a real-world case study is used to define a test bed. The procedure VWSolver and its heuristic version VWSolv-H have been programmed as a console application with Borland Delphi 7.0. The used personal computer contains an Intel Pentium 4 processor with 3.2 GHz clock rate and 1 GB of RAM. The maximal computation time is set to 500 sec. If the optimal solution cannot be found by VWSolver within this time span, the procedure is terminated and the incumbent solution with objective function value UB is used as a heuristic solution. Within VWSolv-H the time limits are $\Delta_1 = 0.4$ and $\Delta_2 = 12$ sec (vgl. Section 5.4), because these limits gave the best results in a preliminary test series.

There is no benchmark data set available for VWALBP, but literature contains some instances of special problem types like 2ALBP and NALBP. Additionally, we have recorded a real-world problem instance from the automotive industry and vary it systematically. Table 4 summarizes all precedence graphs and assigns them to different problem versions. In each case, the number n of tasks, the number Q of mounting positions and the maximal task time are specified. Different instances are obtained by defining several cycle times per graph (see detailed results below).

The "Becker" graphs map a segment of a real assembly line of a German car maker. In the real setting, 45 workers perform the 115 tasks at 25 stations using a cycle time of 110 time units (TU). The maximal number of parallel workplaces is $W = 4$. Some tasks exceed the cycle time such that multi-station workplaces are to be installed. The remaining precedence graphs are taken from the literature.

name	n	Q	t_{max}	reference and description
VWALBP-1				
Becker1	115	24	290	original real-world instance
Becker2	50	24	290	tasks 1 to 50 from Becker1
NALBP with assignment restrictions				
Bogusch1	46	4	171	Gehring/Boguschewski (1990, V9)
2ALBP with assignment restrictions				
Bartholdi2	148	2	383	Bartholdi (1993)
Bartholdi3	148	2	170	Bartholdi2 modified by Kim et al. (2000)
Bogusch2	46	2	171	Gehring/Boguschewski (1990, V6)
Kim1	9	2	3	Kim et al. (2000, P9)
Kim2	12	2	3	Kim et al. (2000, P12, K = 3)
Kim3	12	2	3	Kim et al. (2000, P12, K = 4)
Kim4	24	2	9	Kim et al. (2000, P24)
Kim5	65	2	272	Kim et al. (2000, P65)
2ALBP without assignment restrictions				
Bartholdi1	148	2	383	Bartholdi (1993); Bartholdi2 without assignment restrictions
Bartholdi4	148	2	170	Bartholdi2 modified by Lee et al. (2001)
Lee1	16	2	9	Lee et al. (2001, figure 2)
Lee2	205	2	944	Lee et al. (2001, table A2)
SALBP				
well-known collection of 25 precedence graphs; Scholl (1999, ch. 7.2.2.1); www.assembly-line-balancing.de				

Table 4. Overview of used precedence graphs

Table 5 summarizes the results obtained for "Becker1" with different realistic cycle times c, maximal numbers K of stations and maximal number W of parallel workplaces. Neither procedure can solve any of these large instances to optimality within the time limit of 500 sec, considerable gaps between the resulting upper bound UB and the global lower bound LB_1 remain.[6]

The results for W=2 are best though the minimal total number of workplaces monotonically decreases with increasing W.

This is due to the limited computation time, because the solution space is smaller and a greater part can be considered for small W. Another reason is the branching process which is organized (in order to apply the maximum load rule efficiently) such that large station loads requiring W workplaces are considered before small ones with fewer parallel workplaces. This consumes a lot of time in early branching stages due to more complex station problems. Partly, this is overcome by the diversification mechanism of VWSolv-H leading to better solutions, in particular for W=4. Compared to the current real-world solution for c=110 and K=25 with 45 workplaces, a considerable improvement is obtained by solutions with 40 and 41 workplaces (highlighted).

			VWSolver			VWSolv-H		
			UB for W=			UB for W=		
c	K	LB_1	2	3	4	2	3	4
80	30	51+	55	58	59	55	57	59
90	30	45	49	52	54	49	51	52
100	30	41+	44	45	47	44	45	47
110	25	37	41	41	43	40	41	41
130	25	31	34	37	37	35	36	37
150	25	27	31	32	33	31	32	32
175	25	24+	26	28	30	27	27	27
200	25	21+	23	26	27	23	24	25

Table 5. Results for "Becker1"

Because the large instances could not be solved exactly, the modified precedence graph "Becker2" has been built by restricting to the first 50 tasks. Table 6 contains the results with F* denoting the minimal number of workplaces (computed by VWSolver in up to 7 hours) and cpu denoting the computation times in full seconds. Some instances could be solved within the time limit (boldfaced), mostly optimality is proven even by VWSolv-H due to UB=LB. In other cases, the optimum has been found (UB=F*) but could not be proven. For the remaining instances an additional workplace is required. The performance of VWSolver and VWSolv-H is very similar with reduced computation times for the latter.

6 The '+' indicates that the lower bound based on alternative task times is larger than the bound based on original task times. In each case, the bound is increased by one workplace.

			VWSolver						VWSolv-H					
		W	2		3		4		2		3		4	
c	LB_1	F*	UB	cpu	UB	cpu	UB	cpu	UB	cpu	UB	cpu	UB	cpu
80	27^+	27	**27**	27	**27**	121	**27**	286	**27**	65	**27**	49	**27**	82
90	24^+	24	**24**	117	**24**	207	25	500	**24**	121	25	500	25	500
100	22^+	22	**22**	0	**22**	0	**22**	0	**22**	0	**22**	0	**22**	0
110	20^+	20	21	500	21	500	21	500	21	494	21	500	21	500
130	17^+	18	18	500	18	500	19	500	18	292	18	500	18	500
150	15^+	16	16	500	16	500	16	500	16	412	16	500	16	500
175	14^+	14	**14**	0	**14**	10	**14**	369	**14**	0	**14**	11	**14**	251
200	12^+	13	13	500	13	500	13	500	13	500	13	500	13	500

Table 6. Results for "Becker2" (K=15)

Kim et al. (2000) applied their genetic algorithm (GA) for 2ALBP with assignment restrictions (special case of VWALBP) to the test instances of Table 7. As benchmark they used exact solutions for "Kim1" to "Kim3" computed by the model solver CPLEX (CP) and heuristic solutions from the first fit heuristic (FFH) of Bartholdi (1993) for the larger instances where CP failed. While the computation times of FFH were neligible, the times required for exactly solving the small instances with 12 tasks by CP were considerable (up to an hour).

Table 7 contrasts the results of those procedures and the new ones. The computation times specified for GA must be taken with care, because Kim et al. used a Pentium CPU with only 166 MHz. However, the comparison shows that the new procedures (designed for a more general problem) compute optimal solutions and outperform GA for small and medium-sized instances with low computation times. For three of the 148-task instances "Bartholdi2" with large cycle times VWSolver fails even in finding feasible solutions which is due to the complex station problems. VWSolv-H succeeds in finding feasible solutions which are, however, worse than those of GA. As a summary it can be stated that the new procedures perform well for small and medium-sized problems and are competitive for large-sized problems if cycle times are small.

Table 8 compares our new procedures with the ant algorithm (AA) of Baykasoglu and Dereli (2008) and the procedures of Lee et al. (2001) for 2ALBP instances with and without assignment restrictions (cf. Table 4). Lee et al. propose a traditional (Prio) and a group based (GP) priority rule based procedure. The latter one groups tasks in a first step and assigns the groups to stations in the second step.

instance			GA		CP/ FFH	VW-Solver		VW-Solv-H	
	c	K	UB	cpu	UB	UB	cpu	UB	cpu
Kim1	3	3	6	0	6	6	0	6	0
	4	3	5	0	5	5	0	5	0
	5	3	4	0	4	4	0	4	0
	6	3	3	0	3	3	0	3	0
Kim2	5	3	6	0	6	6	0	6	0
	6	3	5	0	5	5	0	5	0
	7	3	4	0	4	4	0	4	0
Kim3	4	4	8	0	8	8	0	8	0
	5	4	6	0	6	6	0	6	0
	6	4	5	0	5	5	0	5	0
	7	4	4	0	4	4	0	4	0
Kim4	20	4	8	4	8	8	0	8	0
	25	4	6	4	7	6	0	6	0
	30	4	5	4	7	5	0	5	0
	35	4	5	4	6	5	0	5	0
	40	4	4	4	6	4	0	4	0
Kim5	275	10	20	15	20	19	3	19	4
	300	10	18	15	19	18	0	18	0
	325	10	17	15	18	16	0	16	0
	350	10	15	15	16	15	0	15	0
	375	10	15	15	16	14	0	14	0
	400	10	13	15	16	13	2	13	2
	425	10	13	15	16	12	9	12	10
	450	10	12	15	15	12	0	12	0
	475	10	11	15	14	11	0	11	0
	500	10	11	15	14	11	0	11	0
Bartholdi3	171	16	31	60	-	30	4	30	5
	200	16	26	60	-	26	0	26	0
	225	16	24	60	-	23	1	23	1
	250	16	21	60	24	22	500	21	10
	275	16	19	60	22	19	11	19	8
	300	16	18	60	22	18	1	18	8
	325	16	16	60	20	18	500	16	25
	350	16	15	60	19	-	500	16	500
	375	16	14	60	18	-	500	16	500
	400	16	13	60	18	-	500	16	500

Table 7. Comparison to Kim et al. (2000)

instance			GP	Prio	AA		VW-Solver		VW-Solv-H	
	c	K	UB	UB	UB	cpu	UB	cpu	UB	cpu
Kim5	326	10	17	17	17	<1	16	0	16	0
	381	10	15	15	15	<1	14	0	14	0
	435	10	13	14	13	<1	12	0	12	0
	490	10	12	12	12	<1	11	0	11	0
	544	10	10	11	10	3	10	0	10	0
Bartholdi4	204	16	27	27	26	4	26	0	26	0
	255	16	21	22	21	16	21	49	21	6
	306	16	18	19	18	51	17	21	18	500
	357	16	15	16	15	4	15	10	15	8
	408	16	14	14	14	2	-	500	14	500
	459	16	13	12	12	181	-	500	12	16
	510	16	11	12	11	15	-	500	11	312
Lee2	1133	15	23	24	24	451	-	500	21	31
	1322	15	20	21	22	449	-	500	18	6
	1510	15	20	18	18	288	-	500	17	301
	1699	15	16	16	18	448	-	500	15	152
	1888	15	16	15	15	178	-	500	14	112
	2077	15	14	14	14	7	-	500	13	46
	2266	15	13	12	12	131	-	500	13	94
	2454	15	12	12	12	7	-	500	12	500
	2643	15	12	11	11	68	-	500	12	84
	2832	15	10	10	10	303	-	500	11	500

Table 8. Comparison to Lee et al. (2001) and Baykasoglu and Dereli (2008)

Table 8 shows similar results as before: VWSolver fails for large-sized problems at least for large cycle times, while VWSolv-H finds competitive solutions in all cases. In particular when cycle times are small, VWSolv-H computes better solutions than the reference procedures. Concerning the computation times, it has to be observed again that very different computers have been used (Lee et al. 2001: Pentium, 166 MHz; Baykasoglu and Dereli 2008: Pentium III, 1.3 GHz).

instance				VW-Solver		VW-Solv-H	
	c	K	LB_1	UB	cpu	UB	cpu
Bartholdi1	383	16	15	-	500	16	500
	400	16	15	-	500	15	7
	403	16	14	-	500	14	14
	434	16	13	-	500	14	500
	470	16	12	-	500	13	500
	513	16	11	-	500	12	500
	564	16	10	-	500	11	500
	626	16	9	-	500	10	500
	705	16	8	-	500	9	500
	805	16	7	-	500	8	500
Bartholdi2	459	34	13	-	500	15	500
	470	34	12	-	500	14	500
	513	34	11	-	500	14	500
	564	34	10	-	500	13	500
	626	34	9	-	500	12	500
	705	34	8	-	500	12	500
	805	34	7	-	500	11	500
Lee1	15	4	6	6	0	6	0
	18	4	5	5	0	5	0
	20	4	5	5	0	5	0
	22	4	4	4	0	4	0
Lee2	950	15	25	-	500	25	28
	1000	15	24	-	500	24	17
	1100	15	22	-	500	22	7

instance				VW-Solver		VW-Solv-H	
	c	K	LB_1	UB	cpu	UB	cpu
Bogusch1	171	12	14^+	15	500	15	396
	180	12	14^+	14	1	14	1
	190	12	13^+	14	500	14	500
	200	12	12	13	500	13	149
	210	12	12^+	13	500	13	500
	220	12	11	12	500	12	368
	230	12	11^+	12	500	12	500
	245	12	10^+	11	500	11	348
	275	12	9	10	141	10	155
	300	12	8	10	500	10	500
Bogusch2	171	12	14^+	15	500	15	500
	180	12	13	14	244	14	12
	190	12	13^+	13	0	13	0
	200	12	12	13	500	13	500
	210	12	11	11	0	11	0
	220	12	11	12	500	12	500
	230	12	10	10	0	10	0
	245	12	9	11	500	11	500
	275	12	9	9	0	9	0
	300	12	8	8	0	8	0

Table 9. Results for still untested instances

In Table 9, results of applying the new procedures to untested instances from literature (cf. Table 4) are summarized. Only for "Bogusch1" and "Bogusch2" with c=245, solutions with 14 and 10 workplaces, respectively, are reported by Gehring and Boguschewski (1990). Bartholdi (1993) provides a solution for "Bartholdi1" with c = 400 using 15 workplaces.

The results confirm the ones reported before. VWSolver is useful for rather small instances and fails for large ones, but the heuristic version always provides reasonable solutions.

In a final test, we apply the new procedures to the benchmark data set for SALBP-1 with 269 instances and compare to SALOME-1 (cf. Scholl and Klein 1997), one of the best procedures for SALBP-1. The results of SALOME-1 have been obtained at a similar computer also using a time limit of 500 sec and can be downloaded together with the data files from *www.assembly-line-balancing.de*. Table 10 summarizes the results. Taking into account that VWSolver is a procedure for a much more complicating problem and, thus, contains a lot of overhead, it performs rather well. It finds about 85% optimal solutions and on average only 1/6 superfluous workplace is planned. Concerning the number of proven optima the exact procedure outperforms the heuristic version as expected.

	SALOME-1	VWSolver	VWSolv-H
# found optima	260 (97 %)	225 (84 %)	228 (85 %)
# proven optima	257 (96 %)	218 (81 %)	175 (65 %)
av. abs. dev.	0.04	0.17	0.16

Table 10. Comparison for SALBP data set

8 Conclusions and future research

For the first time, we propose a description of an assembly line balancing problem typically for the automotive and other industries producing large products, formulate it as a mixed-integer program and develop an exact solution procedure based on the branch&bound principle and containing a lot of versatile bounding and dominance rules. In particular, the problem considers important real-world issues like parallel workplaces in a flexible manner, extra-long tasks exceeding the cycle time, mounting positions, and other assignment restrictions. In computational experiments the

new procedure VWSolver shows to be competitive to solve small and medium-sized problem instances to optimality, at least for small cycle times typical in the automotive industries. For large-sized instances with many tasks and/or large cycle times, a heuristic version of the procedure proofs to be superior. However, in real-world situations it is usually not necessary to balance an entire line with several hundreds or thousands of tasks but segments of the line with much smaller task numbers. So even applying the exact procedure can be a realistic alternative.

Based on these fundamental developments, it will be possible to give better decision support to assembly line planners in the future. Nevertheless, further developments are necessary. At first, it is required to include additional practical conditions and restrictions like different workpiece positions (up, down; upright, raked etc.) and further types of assignment restrictions. Second, better support for creating and maintaining precedence graphs is necessary, because planners are often afraid of the additional expense of acquiring and keeping up-to-date the required precedence information. Third, improved algorithms for large instances with large cycle times are should be developed and integrated into the standard software used by the firms. Finally, further test instances form practice and systematically constructed test beds are required.

References

Agnetis, A.; Ciancimino, A.; Lucertini, M.; Pizzichella, M. (1995): Balancing flexible lines for car components assembly. International Journal of Production Research 33, 333–350.

Akagi, F.; Osaki, H.; Kikuchi, S. (1983): A method for assembly line balancing with more than one worker in each station. International Journal of Production Research 21, 755–770.

Arcus, A.L. (1966): COMSOAL: A computer method of sequencing operations for assembly lines. International Journal of Production Research 4, 259–277.

Bard, J.F. (1989): Assembly line balancing with parallel workstations and dead time. International Journal of Production Research 27, 1005-1018.

Bartholdi, J. (1993): Balancing two-sided assembly lines: A case study. International Journal of Production Research 31, 2447-2461.

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

Baykasoglu, A.; Dereli, T. (2008): Two-sided assembly line balancing using an ant-colony-based heuristic. International Journal of Advanced Manufacturing Technology 36, 582-588.

Becker, C. (2007): Abstimmung flexibler Endmontagefließbänder in der Automobilindustrie. Books on Demand, Norderstedt.

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

Berger, I.; Bourjolly, J.; Laporte, G. (1992): Branch-and-bound algorithms for the multi-product assembly line balancing problem. European Journal of Operational Research 58, 215-222.

Boguschewski, A.; Gehring, H.; Köstermann, K. (1990): Montagelinien in der Pkw-Montage abstimmen. Die Arbeitsvorbereitung 27, 130-133.

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

Buxey, G. (1974): Assembly line balancing with multiple stations. Management Science 20, 1010-1021.

Demeulemeester, E.L.; Herroelen, W.S. (2002): Project scheduling: A research handbook. Kluwer, Boston.

Falkenauer, E. (2005): Line balancing in the real world. In: Proceedings of the International Conference on Product Lifecycle Management PLM 05, Lumiere University of Lyon, France, 2005 (on cd-rom).

Fekete, S.P.; Schepers, J. (2001): New classes of fast lower bounds for bin packing problems. Mathematical Programming 91, 11–31.

Fleszar, K.; Hindi, K.S. (2003): An enumerative heuristic and reduction methods for the assembly line balancing problem. European Journal of Operational Research 145, 606–620.

Gehring, H.; Boguschewski, A. (1990): Leistungsabstimmung von Montagelinien unter Berücksichtigung praktischer Restriktionen. Working Paper 3/1990, Faculty for Economics, Freie Universität Berlin.

Hildebrand, S. (2006): Interaktive Montagelinienaustaktung in der Automobilindustrie. OR News, 38-39.

Inman, R.R.; Leon, M. (1994): Scheduling duplicate serial stations in transfer lines. International Journal of Production Research 32, 2631–2644.

Jackson, J.R. (1956): A computing procedure for a line balancing problem. Management Science, 261-271.

Johnson, R.V. (1983): A branch and bound algorithm for assembly line balancing problems with formulation irregularities. Management Science 29, 1309-1324.

Johnson, R.V. (1988): Optimally balancing large assembly lines with "FABLE". Management Science 34, 240-253.

Kelley, J.E. (1963): The critical-path method: Resources planning and scheduling. In: Muth, J.F.; Thompson, G.L. (eds.): Industrial scheduling. Prentice Hall, Englewood Cliffs, 347-365.

Kilbridge, M.D.; Wester, L. (1962): A review of analytical systems of line balancing. Operations Research 10, 626-638.

Kim, Y.K.; Kim, Y.; Kim, Y.J. (2000): Two-sided assembly line balancing: A genetic algorithm approach. Production Planning & Control 11, 44-53.

Klein, R. (2000): Scheduling of resource-constrained projects. Kluwer, Boston.

Klein, R. (2002): Constraint programming. In: Stadtler, H. and Kilger, C. (eds.): Supply chain management and advanced planning. 2nd ed., Springer, Berlin, S. 411-418.

Kolisch, R. (1995): Project scheduling under resource constraints - Efficient heuristics for several problem classes. Physica, Heidelberg.

Lapierre, S.D.; Ruiz, A.B. (2004): Balancing assembly lines: An industrial case study. Journal of the Operational Research Society 55, 589-597.

Lapierre, S.D.; Ruiz, A.; Soriano, P. (2006): Balancing assembly lines with tabu search. European Journal of Operational Research 168, 826-837.

Lawler, E.L. (1973): Optimal sequencing of a single machine subject to precedence constraints, Management Science 19, 544-546.

Lee, T.O.; Kim, Y.; Kim, Y.K. (2001): Two-sided assembly line balancing to maximize work relatedness and slackness. Computers and Industrial Engineering 40, 273–292.

Martello, S.; Toth, P. (1990): Knapsack problems: Algorithms and computer implementations. Wiley, Chichester.

Nourie, F.J.; Venta, E.R. (1991): Finding optimal line balances with OptPack. Operations Research Letter 10, 165-171.

Patterson, J.H.; Slowinski, R.; Talbot, F.B.; Weglarz, J. (1989): An algorithm for a general class of precedence and resource constrained project scheduling problems. In: Slowinski, R.; Weglarz, J. (eds.): Advances in project scheduling. Elsevier, Amsterdam, 3-29.

Peeters, M.; Degraeve, Z. (2006): An linear programming based lower bound for the simple assembly line balancing problem. European Journal of Operational Research 168, 716-731.

Pinto, P.; Dannenbring, D.G.; Khumawala, B.M. (1975): A branch and bound algorithm for assembly line balancing with paralleling. International Journal of Production Research 13, 183-196.

Pinto, P.A.; Dannenbring, D.G.; Khumawala; B.M. (1981): Branch and bound and heuristic procedure for assembly line balancing with paralleling of stations. International Journal of Production Research 19, 565-576.

Rachamadugu, R. (1991): Assembly line design with incompatible task assignments. Journal of Operations Management 10, 469-487.

Sarker, B.R.; Shanthikumar, J.G. (1983): A generalized approach for serial or parallel line balancing. International Journal of Production Research 21, 109-133.

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

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

Scholl, A.; Fliedner, M.; Boysen, N. (2008): ABSALOM: Balancing assembly lines with assignment restrictions. Jena Research Papers in Business and Economics (JBE) 2/2008, FSU Jena.

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

Scholl, A.; Klein, R. (1999): Balancing assembly lines effectively - A computational comparison. European Journal of Operational Research 114, 50-58.

Scholl, A.; Klein, R.; Jürgens, C. (1997): BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Computers & Operations Research 24, 627-645.

Tonge, F.M. (1960): Summary of a heuristic line balancing procedure. Management Science 7, 21-39.

Wee, T.; Magazine, M. (1982): Assembly line balancing as generalized bin packing. Operations Research Letters 1, 56-58.