Kolisch, Rainer

Integrated scheduling, assembly area- and part-assignment for large scale, make-to-order assemblies

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 468

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 468

Integrated Scheduling, Assembly Area– and Part–Assignment for Large Scale, Make–to–Order Assemblies

R. Kolisch

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 468

Integrated Scheduling, Assembly Area– and Part–Assignment for Large Scale, Make–to–Order Assemblies

R. Kolisch

February 1998

Paper presented at the International Conference on Industrial Engineering and Production Management IEPM'97, Lyon/France, October 1997.

Dr. Rainer Kolisch
Lehrstuhl für Produktion und Logistik,
Institut für Betriebswirtschaftslehre,
Christian–Albrechts–Universität zu Kiel,
Olshausenstr. 40, 24098 Kiel, Germany
email: kolisch@bwl.uni–kiel.de
URL: http://www.wiso.uni–kiel.de/bwlinstitute/Prod/kolisch.html
ftp://ftp.wiso.uni–kiel.de/pub/operations–research

Abstract

The problem of scheduling multiple, large scale, make–to–order assemblies is considered. Beside "classical" precedence– and resource constraints as known from resource constrained project scheduling we take spatial resource and part availability constraints into account. The objective is to minimize the sum of the weighted tardiness. We propose a MIP–model of the problem which is a generalization of three allocation problems. Since the problem is $\mathcal{NP}$–hard, a list–scheduling heuristic is proposed and evaluated on a set of benchmark instances.

1 Introduction

1.1 Outline of the Problem

This paper is concerned with scheduling of large scale assemblies in a make–to–order environment. A practical application is a German company which manufactures customized palletising systems for the chemical and food industry. The company has 70 employees, 20 of them working in the final assembly. In 1997, 100 palletising systems have been delivered to customers, each of them with an average revenue of 200,000 German Mark. The time between the confirmation of an order and the delivery to the customer is on average 20 weeks and is made of order–specific construction, fabrication, and assembly. The major part of the fabrication is done by outside suppliers. Currently this causes long lead times of up to 12 weeks. The assembly of parts takes on average 3 weeks.

Within the final assembly, P end items are assembled according to assembly networks of the type given in Figure 1. Nodes represent assembly operations and arcs represent precedence relations between operations. Each operation has a given processing time, a resource requirement w.r.t. different types of assembly resources and a requirement of type A–parts which are assembled by that operation. The entire assembly has to take place on an assembly area, which is typically the shop floor. Since the shop floor area is limited, not more than $C^S \geq 0$ orders can be assembled at the same time. The assembly of an order $p = 1, \ldots, P$ begins as soon as the first operation belonging to the order is started and it commences until the last assembly of the order has been finished. During the entire time interval the order occupies the shop floor. Order preemption, i.e., the removal of a partly assembled product from the assembly area, is not allowed because of high set up costs and the risk of damage.

Assembly resources are discriminated in types such as welders, mechanical assemblers, electrical assemblers, and power tools. Overall there are $r = 1, ..., R^A$ different types. Resource type r has an availability of $C^A_{r,t} \geq 0$ units at time instant t. The time varying capacity has different reasons. First, is stems from the fact that planning is based on a rolling horizon, where resources may be tied to operations which have already been started and are still being processed. Second, time varying capacity is caused by planned off–time of workers (vacation, fluctuation) and planned down–times of machines (inspection, repair). Each operation j requires $c_{j,r} \geq 0$ units of resource type r during every period it is processed.

Parts which are build into the end item are distinguished in C– and B–parts as well as A–parts. C– and B– parts are, e.g., bolts, screws, hydraulic elements, and rollers which are common to most of the products and are available from stock. A–parts are components which have been fabricated or ordered specifically for planned orders. Delayed A–parts will disrupt or delay the assembly and hence need special management attention (cf . Vaart et al. [39] and Nof et al. [30]). We denote the different A–part types with $i = 1, \ldots, I$. For each part type i the quantity on hand is $n_{i,0} \geq 0$. Additionally, the number of parts which will become available at time instant t is $n_{i,t} \geq 0$. The cumulated number of parts which will become available *until* time instant t is

order	due date	weight	operation	processing time	capacity required	parts required
1	8	2	2	2	2	1
			3	3	1	2
			4	3	3	1
			5	2	2	
2	6	3	7	2	1	2
			8	1	2	1
			9	2	1	
			10	3	2	
3	7	4	11	3	2	1
			12	2	3	

Table 1: Orders and operations data

$N_{i,t} = \sum_{\tau=0}^{t} n_{i,\tau}$. The amount and timing of incoming parts are known from vendor contracts and production schedules of in–house part fabrication. Operation j requires $q_{j,i} \geq 0$ units of part type i. Before an assembly operation can be initiated, all required parts must be kitted. As noted by Nof et al. [30], kitting plays a central role in assembly.

The objective of the assembly scheduling problem is to place orders on assembly areas, to assign parts to operations, and to schedule operations subject to precedence and resource constraints such that the sum of the weighted tardiness of the orders is minimized. Note that in contrast to, e.g., Agrawal et al. [1], Chen and Wilhelm [8], and Faaland and Schmitt [12] we do not consider any earliness costs. This is due to the short–term character of our problem where holding costs of parts and availability costs of resource levels have already been determined.

Similar problems as the one given above appear in the assembly of machine tools (cf. Drexl and Kolisch [10]), ships (cf. Lee et al. [24]), and construction sites (cf. Moder et al. [26]).

Figure 1 and Table 1 give an example with 3 orders, each with an order specific weight, 10 operations, 1 resource type, and 1 part type. Operations are labelled from $2, \ldots, 11$ for reasons which will become apparent in Subsection 2.1.

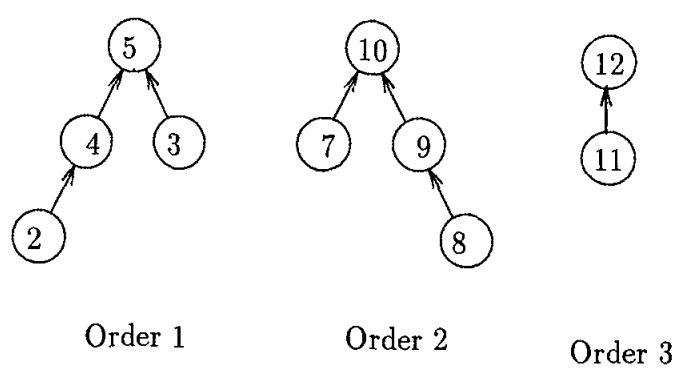

Figure 1: Operations sequences

1.2 Literature Review

The following discussion of the literature focuses on assembly scheduling and multi–project scheduling. Both these areas are relevant to the problem outlined above.

Assembly scheduling

Russell and Taylor [35] as well as Lalsare and Sen [22] investigate the efficiency of different priority rules in a dynamic fabrication/assembly shop where operations are either processed on fabrication or assembly resources.

Neumann and Schwindt [29] employ activity–on–node networks with minimal and maximal time lags in order to model scheduling problems of make–to–order–production. Amongst other topics they treat the 'assignment sequence problem' where the parts emerging from a fabrication operation have to be assigned to assembly operations such that the overall makespan is minimized.

Potts et al. [33] consider a deterministic two–stage assembly system, a generalization of the two–machine flowshop problem, where m fabrication machines at the first stage produce m different components which are assembled on a single assembly machine at the second stage. Several heuristics for the $\mathcal{NP}$–hard optimization problem are proposed.

Faaland and Schmitt [12, 13] report about an fabrication/assembly problem encountered in the manufacturing of electronic products. Operations have to be sequenced on machines such that earliness and tardiness costs are minimized. The method consists of two phases. The first phase performs a heuristic sequencing of the operations with the minimum slack time rule. The second phase optimizes for a given sequence the start times of the operations.

Chen and Wilhelm [7, 8] consider the kitting problem of multi–echelon electronical assembly. A set of customer orders consists of operations which are ordered in an assembly tree by precedence constraints. To be processed, an operations requires a part kit and a capacity of "1" in the shop where the operation is assembled. The availability of assembly area is not considered. The objective is to schedule operations such that the sum of order lateness and holding costs of operations is minimized. As solution procedures Chen and Wilhelm propose a heuristic in [7] and an optimal branch–and–bound approach in [8]. The latter calculates lower bounds by Lagrangian relaxation.

Agrawal et al. [1] present the problem of 'just–in–time' production of large–scale assemblies where final assemblies, each one consisting of a number of operations, have to be scheduled on work centers with identical machines. Each operation is processed on one machine of one work center. The objective is to minimize the maximum makespan of all final assemblies. A priority rule based scheduling heuristic is proposed which performs backward scheduling of the operations based on the latest start times. Anwar and Nagi [2] extend the approach of Agrawal et al. by integrating the scheduling decision with lot–sizing. First, they perform the scheduling as in Agrawar et al. Second, they goup multiple lots of the same parts into aggregate lots by merging operations. The scheduling phase is repeated on this new set of operations.

All of the above given papers do not consider scarce assembly area which is occupied during the entire duration of an end item assembly. The kitting problem and its variant the 'assignment sequence problem' is only taken into account by Chen and Wilhelm [7, 8] and Neumann and Schwind [29].

Multi project scheduling

The resource constrained multi–project scheduling problem (RCMPSP) arises when the kitting problem and the assignment of assembly areas are relaxed. What remains is a number of projects, which comprise precedence–related operations. When processed an operation requires certain amounts of capacitated resources. The RCMPSP with the objective of minimizing the sum of the weighted tardiness is treated, e.g., in Patterson [32], Kurtulus and Narula [21], Norbis and Smith

[31], Kim and Schniederjans [16], Moccellin [25], Mohanty and Siddiq [27, 28], as well as Lawrence and Morton [23]. De Boer et al. [9] treat the case where parts have already been assigned to operations. Then, the availability of kits can be modelled by release dates for assembly operations.

From the literature review it can be concluded, that the problem as outlined above has not yet been treated. Therefore, we will develop a mathematical programming formulation of the problem in Section 2. Section 3 is devoted to a suited list scheduling heuristic. The results of an experimental investigation will be reported in Section 4.

2 Problem Formulation

The above given problem can be modelled in different ways. For example, if $c_{j,r} \in \{0,1\}$ holds for each operation j we could extend the formulation given in Agrawal et al. [1] which explicitly assigns resources to operations and enforces conjunctive precedence constraints between operations which are processed on the same resource. Beside these two binary decision variables we would need two more types which indicate the allocation of parts to operations and the assignment of assembly area to orders. We depart from this approach of the explicitly assignment of resources, parts, and assembly area. Instead, we employ an approach which performs operation scheduling subject to aggregated availabilities of resources, assembly area, and parts. This is the same approach as done for the resource constraints in multi-project scheduling problems and for part availability constraints in the kitting problem given in Chen and Wilhelm [8].

2.1 Precedence Network Representation

Before we can write down the model, we depict the assembly scheduling problem as a precedence network where nodes represent operations and technological precedence relations are given by arcs. For all orders which do not have a unique start (end) operation, we insert an artificial start (end) operation (e.g., operations 1 and 6 for orders 1 and 2, respectively). Let J denote the number of operations. Each operation is assigned a unique number $j \in \{1,\ldots,J\}$ such that the operations of one order are consecutively numbered and for each upstream operation h of operation j, $h < j$ holds. We denote with $\mathcal{J}$ the set of all operations. Between each downstream operation h and its immediate upstream operation j, an arc with weight $t^{\min}_{h,j} = 0$ is introduced. Let denote $s(p)$ and $e(p)$ the unique start and end operation of order p. We introduce a dummy source 0 and a dummy sink $J+1$. Between all order end operations and the sink we introduce arcs with weight $t^{\min}_{e(p),J+1} = d^{\max} - d_p$ where $d^{\max}$ denotes the maximal due date of all orders. Figure 2 gives the precedence network which results from our example. Each node j represents an operation; the three numbers above the node give the processing time p_j, the capacity requirement c_j, and the part demand q_j. Since we have only one resource type, we omitted the index r. The numbers above the arcs give the minimal time lags; minimal time lags with value 0 have been omitted.

Based on the network we can calculate earliest start times ES_j and latest start times LS_j for all operations $j = 1,\ldots,J$ by forward recursion from $ES_0 = 0$ and backward recursion (cf. Elmaghraby [11]) from $LS_J = T$ where T denotes the latest time instant for all orders to be finished.

2.2 Model

We assume that events, i.e. the delivery of parts and the change of available capacity, occur at discrete multiples of a standard period length, e.g., a shift or half shift, and that the processing times are discrete multiples of the standard period length. In this case, all operation start times

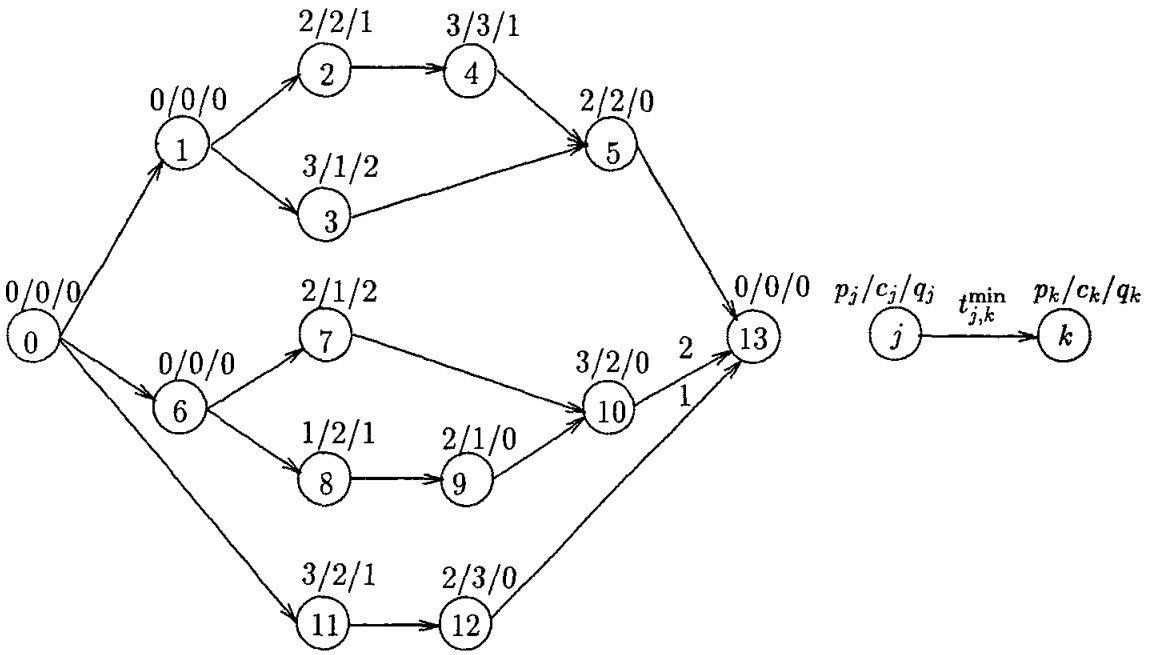

Figure 2: Precedence network

will also be discrete multiples of the standard period length. We employ two types of decision variables. First, the binary decision variable $x_{j,t} = 1$, if operation j is started at time instant t, and 0, otherwise (cf. Pritsker et al. [34]). That is, $x_{j,0} = 1$ denotes that operation j starts at time instant $t = 0$, ends at time instant $t = p_j$ and is processed during periods $t = 1, \ldots, p_j$. The second decision variable is $T_p \geq 0$, the time span order p is tardy. Denoting with w_p the weight of order p, the assembly scheduling problem (ASP) can now be modelled as follows:

$$\text{Min} \quad Z = \sum_{p=1}^{P} w_p \cdot T_p \tag{1}$$

s.t.

$$\sum_{t=ES_j}^{LS_j} x_{j,t} = 1 \qquad (j = 1, \ldots, J) \tag{2}$$

$$\sum_{t=ES_h}^{LS_h} (t + p_h)\, x_{h,t} - \sum_{t=ES_j}^{LS_j} t \cdot x_{j,t} \leq -t_{h,j}^{\min} \qquad (j = 1, \ldots, J;\ h \in \mathcal{P}_j) \tag{3}$$

$$\sum_{j=1}^{J} \sum_{\tau=\max\{0,t-p_j+1\}}^{t} c_{j,r} \cdot x_{j,\tau} \leq C_{r,t}^{A} \qquad (r = 1, \ldots, R^A;\ t = 0, \ldots, T) \tag{4}$$

$$\sum_{p=1}^{P} \sum_{\tau=0}^{t} \left(x_{s(p),\tau} - x_{e(p),\max\{0,\tau-p_{e(p)}\}} \right) \leq C^S \qquad (t = 0, \ldots, T) \tag{5}$$

$$\sum_{\tau=0}^{t}\sum_{j=1}^{J} q_{j,i}\cdot x_{j,\tau} \le N_{i,t} \qquad (i=1,\ldots,I;\ t=0,\ldots,T) \tag{6}$$

$$\sum_{t=ES_{e(p)}}^{LS_{e(p)}} \left(t+p_{e(p)}\right)\cdot x_{e(p),t} - T_p \le d_p \qquad (p=1,\ldots,P) \tag{7}$$

$$x_{j,t} \in \{0,1\} \qquad (j=1,\ldots,J;\ t=ES_j,\ldots,LS_j) \tag{8}$$

$$T_p \ge 0 \qquad (p=1,\ldots,P) \tag{9}$$

The objective function (1) minimizes the sum of the weighted tardiness. (2) forces each operation to be processed. (3) stipulates the technological precedence constraints between operations where $\mathcal{P}_j$ denotes the set of immediate upstream operations of assembly operation j. (4) guarantees that the capacity of each type of assembly resource is respected at every time instant. (5) ensures for every time instant the spatial capacity constraints imposed by the limited availability of the assembly area. (6) models the constraints imposed by the fact that for each part type i and each time instant t the sum of assembled units has to be less equal the sum of the parts which have become available until t. (7) links the continuous tardiness variable with the binary start variables of the order sink. Finally, (8) and (9) define the binary and continuous decision variables, respectively. Note, that one can model the problem with the $x_{j,t}$ variables solely. But employing the T_p variables makes the objective function more handy.

2.3 Model Properties

If the constraints (4) – (6) are relaxed, the optimal solution of the ASP is to start each job j at its precedence feasible earliest start time ES_j. If we add one of the constraints (4) – (6) to the relaxed problem (1) – (3), (7) – (9) we obtain each time an $\mathcal{NP}$–hard optimization problem, namely the resource allocation problem, the assembly area allocation problem, and the part allocation problem.

The resource allocation problem. If we add constraint (4) to problem (1) – (3), (7) – (9) we obtain the resource constrained multi–project scheduling problem, where for each resource r the available capacity at time instant t has to be allocated to the operations which are processed at t. The RCMPSP with the objective of minimizing the sum of the weighted tardiness is treated in Patterson [32], Kurtulus and Narula [21], Norbis and Smith [31], Kim and Schniederjans [16], Moccellin [25], Mohanty and Siddiq [27, 28], as well as Lawrence and Morton [23]. As a generalization of the classical job shop scheduling problem, the resource constrained multi–project scheduling problem is an $\mathcal{NP}$–hard optimization problem.

The assembly area allocation problem. If we add constraint (5) to problem (1) – (3), (7) – (9) we obtain the parallel machine scheduling problem $(C^S \mid\mid \sum w_p \cdot T_p)$. Here, each project p is treated as one "job" p with processing time $ES_{e(p)}$, weight w_p, and due date d_p. All $p=1,\ldots,P$ jobs have to be scheduled on C^S identical machines where each machine can process no more than one job at a time. The objective is to schedule the jobs such that the sum of the weighted tardiness $\sum w_p \cdot T_p$ is minimized. Problem $(C^S \mid\mid \sum w_p \cdot T_p)$ is $\mathcal{NP}$–hard in the strong sense because it is a generalization of the parallel machine scheduling problem with the objective to minimize the weighted finish times (cf. van den Akker et al. [38]).

The part allocation problem. If we add constraint (6) to problem (1) – (3) and (7) – (9) we obtain the part allocation problem (cf. Balakrishnan et al. [3]). Part allocation is concerned with

the allocation of currently available and incoming parts to operations. Carlier and Rinnooy Kan [6] show that the part allocation problem can be solved with a polynomially bounded algorithm if $P = 1$ but that the problem becomes $\mathcal{NP}$–hard for $P > 1$.

Since each of the allocation problems is $\mathcal{NP}$–hard, ASP is $\mathcal{NP}$–hard, too. Hence, optimal algorithms are not applicable for industrial applications, where hundreds to thousands of operations have to be scheduled within minutes of CPU–time. We therefore, will device a robust heuristic in the next Section. Before, let us look at an important property of ASP.

Definition 1: *Left–regular* (cf. Sprecher et al. [37]). Consider two feasible solutions (schedules) $S = (S_1, \ldots, S_J)$ and $S' = (S'_1, \ldots, S'_J)$ for ASP. S_j denotes the start time of operation j. Let $S_j \leq S'_j$ for a single operation j and $S_i = S'_i$ for $i = 1, \ldots, j-1, j+1, \ldots, J$. The objective function of a minimization problem is left–regular, if $Z(S) \leq Z(S')$ holds, where $Z(S)$ denotes the objective function value associated with the solution S.

Theorem 1. *The objective function (1) is left–regular.*

Proof. Consider a feasible solution S where a non–sink operation j has the start time S_j. Changing, c.p., the start time to $S_j - 1$ cannot increase the objective function value since the latter is only effected by the start times of the sink operations. Consider now that the start time $S_{e(p)}$ of sink $e(p)$ is, c.p., decreased by one period to $S_{e(p)} - 1$. If $S_{e(p)} \leq d_p$ holds, the objective function value does not change while for $S_{e(p)} > d_p$, it is decreased by w_p units.□

3 A List Scheduling Heuristic

A very popular approach to solve scheduling problems is list scheduling (cf. Schutten [36]) which works as follows: First, the operations of the scheduling problem are sequentially ordered in a list. Second, in the order given by the list, the operations are scheduled at their earliest feasible start times. In what follows, we will first show how a list can be transformed into a schedule, afterwards we turn to the generation of so–called feasible lists, i.e., lists which will bring forth a feasible schedule.

3.1 Schedule Generation

Let $\pi = [j_1, j_2, \ldots, j_J]$ be a list of the J operations belonging to problem ASP. j_g is the operation at position g. Let us assume for now that we have such a list π and want to transform it into a feasible schedule $S(\pi)$. A feasible schedule gives a start time S_j for each operation j such that precedence relations, part availability, assembly resource, and spatial resource constraints are obeyed. In order to schedule the g–th operation on the list, we need to know the material availability $\tilde{N}_{i,t}(g)$, the assembly capacity $\tilde{C}^A_{r,t}(g)$, and the spatial resource capacity $\tilde{C}^S_t(g)$ after the first $g-1$ operations have been scheduled. This can be easily calculated with the following recursions for $g = 2, \ldots, J$:

After initializing $\tilde{N}_{i,t}(1) = N_{i,t}$ for $i = 1, \ldots, I$ and $t = 0, \ldots, T$, we have

$$\tilde{N}_{i,t}(g) = N_{i,t}(g-1) - q_{j_{g-1},i} \quad (i = 1, \ldots, I;\ t = S_{j_{g-1}}, \ldots, T). \tag{10}$$

Similar, for the capacity of assembly resources we set $\tilde{C}^A_{r,t}(1) = C^A_{r,t}$ for $r = 1, \ldots, R$, $t = 0, \ldots, T$ and update the available capacity as follows

$$\tilde{C}^A_{r,t}(g) = \tilde{C}^A_{r,t}(g-1) - c_{j_{g-1},r} \quad (r = 1, \ldots, R;\ t = S_{j_{g-1}}, \ldots, S_{j_{g-1}} + p_{j_{g-1}} - 1). \tag{11}$$

Finally, for the capacity of spatial resources, we set $\tilde{C}_t^S(1) = C^S$ for $t = 0, \ldots, T$. If we denote with $\mathcal{S} = \{s(p) \mid p = 1, \ldots, P\}$ and $\mathcal{E} = \{e(p) \mid p = 1, \ldots, P\}$ the set of start and end operations of all orders, updating can be done by

$$\tilde{C}_t^S(g) = \tilde{C}_t^S(g-1) + \begin{cases} 1 & \text{, if } j_g \in \mathcal{E} \quad (t = S_{j_{g-1}} + p_{j_{g-1}}, \ldots, T) \\ -1 & \text{, if } j_g \in \mathcal{S} \quad (t = S_{j_{g-1}}, \ldots, T) \end{cases} \tag{12}$$

Now, assume all list predecessors $h = j_1, \ldots, j_{g-1}$ of operation $j = j_g$ have been scheduled and we want to start j as early as possible. If we only take into account precedence constraints (PC) we obtain

$$S_j^{PC} = \max_{h \in \mathcal{P}_j} \left\{ S_h + p_h + t_{h,j}^{\min} \right\}. \tag{13}$$

With respect to the part availability (PA) we get

$$S_j^{PA} = \min_{t=ES_j}^{LS_j} \left\{ t \mid \tilde{N}_{i,\tau}(g) \geq q_{j,i} : \; i = 1, \ldots, I; \; \tau = t, \ldots, T \right\}. \tag{14}$$

Considering only the assembly capacity (AC) and the dynamic earliest start time ES_j', the start time is

$$S_j^{AC}(ES_j') = \min_{t=ES_j'}^{LS_j} \left\{ t \mid \tilde{C}_{r,\tau}(g) \geq c_{j,r} : \; r = 1, \ldots, R^A; \; \tau = t, \ldots, t + p_j - 1 \right\}. \tag{15}$$

Taking only the spatial capacity (SC) into account the start time is

$$S_j^{SC} = \min_{t=ES_j}^{LS_j} \left\{ t \mid \tilde{C}_t^S \geq 1 \right\}. \tag{16}$$

We now can proceed to the algorithm.

Algorithm 1: Schedule Generation(π)
Initialization: $S_0 = 0$.
For $g = 1$ **to** J **do**

(1) Calculate $\tilde{N}_{i,t}(g)$, $\tilde{C}_{r,t}^A(g)$, and $\tilde{C}_t^S(g)$

(2) Take the next job from the list: $j = j_g$

(3) Determine the dynamic earliest start time ES_j' of j w.r.t. precedence, part availability, and spatial constraints:
`If` $j \in \mathcal{S}$ `then` $ES_j' = \max\left\{S_j^{PC}, S_j^{PA}, S_j^{SC}\right\}$, `else` $ES_j' = \max\left\{S_j^{PC}, S_j^{PA}\right\}$

(4) Determine the earliest resource feasible start time of j which is $\geq ES_j'$:
$S_j = S_j^{AC}(ES_j')$.

Step (1) updates the part availability and the resource availability of assembly and spatial resources according to (10), (11), and (12). After step (2) has selected the next operation from the list, its dynamic earliest start time w.r.t. precedence constraints, part availability constraints, and spatial constraints is calculated in step (3). Note that for non–start operations, no spatial constraints have to be taken into account. Step (4) determines the earliest resource feasible start time within time window $\left[ES_j', \ldots, LS_j\right]$.

Table 2 reports the schedule generation for the example problem and the list $\pi =< 6, 8, 11, 7, 9, 12, 1, 10, 2, 3, 4, 5]$. The resulting schedule $S = (6, 6, 9, 8, 12, 0, 3, 0, 1, 5, 0, 3)$ is pictured in Figure 3 as Gantt chart. Order 1 is 6 periods tardy, order 2 is 2 periods tardy, and order 3 is on time. Hence, the objective function value is $Z = 2 \cdot 6 + 3 \cdot 2 + 4 \cdot 0 = 18$ which happens to be the optimal objective function value for this problem instance.

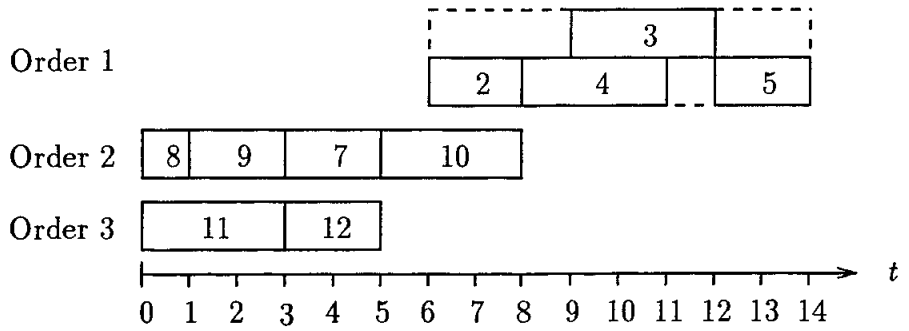

Figure 3: Schedule

Definition 2: *Active schedules* (cf. Sprecher et al. [37]). An active schedule is a feasible schedule where none of the operations can be started earlier without delaying some other activity.

Theorem 2. *Algorithm 1 generates feasible schedules which are active in the case of* $C^S \geq P$ *and which might be non-active for* $C^S < P$.

Proof: For $C^S \geq P$, Algorithm 1 generates active schedules by construction. For $C^S < P$ consider the following example with $C^S = 2$, $I = 1$, $R^A = 0$, $P = 3$. Each order comprises two precedence related operations. All operations $j = 1, \ldots, 6$ have a processing time of $p_j = 1$. Operation 4 requires a single unit of part type 1 which becomes available at $t = 4$. List $\pi = \langle 3, 1, 4, 5, 6, 2]$ results in the feasible but not active schedule given in Figure 4. Note that an active schedule will be obtained with list $\pi = \langle 3, 1, 2, 5, 6, 4]$.

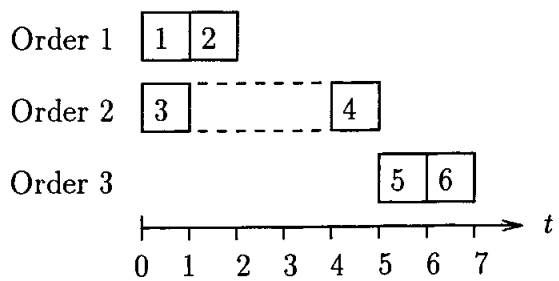

Figure 4: Non active schedule

g	$\tilde{N}_t(g)$ $\tilde{C}_t^A(g)$ $(t = 0, \ldots, 20)$ $\tilde{C}_t^S(g)$	j_g	$S_{j_g}^{PC}$ $S_{j_g}^{PA}$ $S_{j_g}^{SC}$	$S_{j_g} = S_{j_g}^{AC}$
1	(2,2,2,4,4,4,6,6,6,8,8,8,8,8,8,8,8,8,8,8,8) (4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4) (2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2)	6	0 0 0	0
2	(2,2,2,4,4,4,6,6,6,8,8,8,8,8,8,8,8,8,8,8,8) (4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4) (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)	8	0 0 –	0
3	(1,1,1,3,3,3,5,5,5,7,7,7,7,7,7,7,7,7,7,7,7) (2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4) (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)	11	0 0 0	0
4	(0,0,0,2,2,2,4,4,4,6,6,6,6,6,6,6,6,6,6,6,6) (0,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4) (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)	7	0 3 –	3
5	(0,0,0,0,0,0,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4) (0,2,2,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4) (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)	9	1 0 –	1
6	(0,0,0,0,0,0,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4) (0,1,1,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4) (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)	12	3 0 –	3
7	(0,0,0,0,0,0,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4) (0,1,1,0,0,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4) (0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)	1	0 0 5	5
8	(0,0,0,0,0,0,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4) (0,1,1,0,0,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4) (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)	10	5 0 –	5
9	(0,0,0,0,0,0,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4) (0,1,1,0,0,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4) (0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1)	2	0 6 –	6
10	(0,0,0,0,0,0,1,1,1,3,3,3,3,3,3,3,3,3,3,3,3) (0,1,1,0,0,2,0,0,4,4,4,4,4,4,4,4,4,4,4,4,4) (0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1)	3	0 9 –	9
11	(0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1) (0,1,1,0,0,2,0,0,4,3,3,3,4,4,4,4,4,4,4,4,4) (0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1)	4	8 6 –	8
12	(0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0) (0,1,1,0,0,2,0,0,1,0,0,3,4,4,4,4,4,4,4,4,4) (0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1)	5	11 0 –	12

Table 2: Schedule generation

3.2 List Generation

We now turn to the problem of generating a list $\pi = \langle j_1, j_2, \ldots, j_J]$. In order to transform a list π into a feasible schedule $S = (S_1, \ldots, S_J)$, two properties corresponding with constraints (3) and (5) have to be met. The first property corresponds to the precedence constraints in (3). It requires that the list position of an operation j must be greater than the list position of each of its direct and indirect predecessors (cf., e.g., Hartmann [15])

$$\mathcal{P}_{j_g} \subseteq \{j_1, \ldots, j_{g-1}\} \quad (g = 2, \ldots, J). \tag{17}$$

The second property concerns the spatial resource constraints given in (5). Consider the available spatial capacity given in (12). Whenever an order start operation $s(p)$ is scheduled, the available spatial capacity is reduced by 1 from the start of that operation to the end of the planning horizon. Scheduling an order end operation adds 1 capacity unit from the finish time of the operation to the end of the planning horizon. Hence, starting with an available capacity of C^S at iteration 1 of the Algorithm 1, no more than C^S order sources can be on list positions $1, \ldots, g$ without an order sink in–between. Let $\mathcal{L} = \{j_1, \ldots, j_{g-1}\}$ denote the set of operations which have been assigned to list positions $1, \ldots, g-1$. $\tilde{C}^S$, the spatial capacity available at the g–th position of the list, is

$$\tilde{C}^S = C^S - \mid \mathcal{L} \cap \mathcal{S} \mid + \mid \mathcal{L} \cap \mathcal{E} \mid . \tag{18}$$

Now, $\tilde{C}^S \geq 0$ has to hold for each list position $g = 1, \ldots, J$.

Let $\mathcal{A}$ be the set of all available operations which can be put at list position g. $\mathcal{A}$ is the union of two disjunct sets of operations. Start operations (SO) of orders $\mathcal{A}_{SO} = \{j \in \mathcal{S} \mid j \notin \mathcal{L}\}$ and remaining operations (RO) $\mathcal{A}_{RO} = \{j \in \mathcal{J} \setminus \mathcal{S} \mid j \notin \mathcal{L},\ \mathcal{P}_j \subseteq \mathcal{L}\}$. Both, start and remaining operations respect the precedence constraint property (17). The former because they do not have any (non–dummy) predecessors, the latter by definition. Denoting with v_j a priority value associated with operation j we can give the list generation algorithm as follows:

Algorithm 2: List Generation
For $g = 1$ to J do

(1) Update $\mathcal{L}$, $\tilde{C}^S$, $\mathcal{A}_{RO}$, and $\mathcal{A}_{SO}$

(2) `If` $\tilde{C}^S \geq 1$ `then` $\mathcal{A} = \mathcal{A}_{SO} \cup \mathcal{A}_{RO}$ `else` $\mathcal{A} = \mathcal{A}_{RO}$

(3) Choose $j_g \in \mathcal{A}$ with $v_{j_g} = \min_{i \in \mathcal{A}} \{v_i\}$

The set of available operations is defined in step (1). If there is spatial capacity, i.e. $\tilde{C}^S \geq 1$, then the set comprises start and remaining operations, otherwise, i.e. $\tilde{C}^S = 0$, the set comprises remaining operations only. In step (2) we select one operation j from the set of available operations with smallest priority value v_j. Afterwards, we update the set of start and remaining operations in step (3). Table 3 reports the list generation for the example problem when the priority values as given in Table 5 are employed.

Table 4 lists different priority rules which have been successfully utilized for multi–project scheduling problems (cf. Lawrence and Morton [23]). $p(j)$ denotes the project operation j belongs to. The latest finish and start times, LF_j and LS_j, are derived by backward recursion from the project specific due dates. The two rules SPT and RAND have been added for comparison purposes. In case of ties, we have employed two tie breaking rules. The first one is "first come first serve" (FCFS) where the operation is selected which has been first in the set $\mathcal{A}$. As second tie breaker we use the operation number.

g	π	$\tilde{C}^S$	$\mathcal{A}$	j_g
1	$\langle\,]$	2	$\{1,6,11\}$	6
2	$\langle 6]$	1	$\{1,11,7,8\}$	8
3	$\langle 6,8]$	1	$\{1,11,7,9\}$	11
4	$\langle 6,8,11]$	0	$\{7,9,12\}$	7
5	$\langle 6,8,11,7]$	0	$\{9,12\}$	9
6	$\langle 6,8,11,7,9]$	0	$\{12,10\}$	12
7	$\langle 6,8,11,7,9,12]$	1	$\{1,10\}$	1
8	$\langle 6,8,11,7,9,12,1]$	0	$\{10,2,3\}$	10
9	$\langle 6,8,11,7,9,12,1,10]$	1	$\{2,3\}$	2
10	$\langle 6,8,11,7,9,12,1,10,2]$	1	$\{3,4\}$	3
11	$\langle 6,8,11,7,9,12,1,10,2,3]$	1	$\{4\}$	4
12	$\langle 6,8,11,7,9,12,1,10,2,3,4]$	1	$\{5\}$	5
$\pi = \langle 6,8,11,7,9,12,1,10,2,3,4,5]$				

Table 3: List generation

Acronym	Priority Rule	Definition
EDD	Earliest Due Date	$v_j = d_{p(j)}$
LFT	Minimum Latest Finish Time	$v_j = LF_j$
SLK	Minimum Slack	$v_j = LS_j - ES_j$
RAND	Random	$v_j \in [1, \mid \mathcal{A} \mid]$
SPT	Shortest Processing Time	$v_j = p_j$
WEDD	Weighted Earliest Due Date	$v_j = d_{p(j)}/w_{p(j)}$
WLFT	Weighted Minimum Latest Finish Time	$v_j = LF_j/w_{p(j)}$
WSLK	Weighted Minimum Slack	$v_j = (LS_j - ES_j)/w_{p(j)}$
WSPT	Weighted Shortest Processing Time	$v_j = p_j/w_{p(j)}$

Table 4: Priority Rules

j	1	2	3	4	5	6	7	8	9	10	11	12
v_j	1.5	2.5	3.5	3.5	4.5	0.3	1.3	0.6	1.3	2.3	1	1.5

Table 5: Priority values

3.3 Resource and Part Assignment

As pointed out in Section 2, we have modelled the problem such that the allocation of resources and parts is done in an aggregated fashion. For each time instant the total number of assembly and spatial resources as well as part units used does not exceed the available amount. For a successfull implementation of a schedule on the shop floor we need to know precisely which of the $r = 1, \ldots, R^A$ resources are assigned to operation j during the processing interval $S_j, \ldots, S_j + p_j$, which of the parts are assigned to operation j, and on which of the C^S assembly areas order p is assembled. Altogether, there are $R^A + I + 1$ independent assignment problems. Compared to the $\mathcal{NP}$–hard allocation problems treated in Section 2, each of the assignment problems is easy because the scheduling of the operations has already been performed. In what follows, we will outline how to do the assignment.

The **resource assignment** shall be clarified by Table 6 which reports the assignment for the example problem. As input data we need the operation start times S_j in increasing order as given in the first row of Table 6, i.e., $t \in \{0, 1, 3, 5, 6, 8, 9, 12\}$. Associated with each start time t we calculate $\mathcal{A}_t = \{j \mid S_j \leq t < S_j + p_j\}$ the set of operations which are processed (active) at t. The resource assignment begins in the first start period, which is for our example $t = 1$, and assigns the resources $k = 1, \ldots, c_{h,r}$ to operation h which is the operation in $\mathcal{A}_t$ with the lowest operation number. In the example, resources $k = 1$ and 2 are assigned to operation 8. Next, resources $k = c_{h,r} + 1, \ldots, c_{h,r} + c_{j,r}$ are assigned to the operation j with the second smallest label etc. When resources have been assigned to all active operations in the current period, we proceed to the next start period.

If the assembly resources have a time constant capacity, the algorithm can be refined in order to assure that to each operation the same resources are assigned for the entire processing time. Whenever a new start period is considered, operations which start in a prior period and are still active in the current period are considered first. To each of them, the same resources as in the prior period are assigned. Table 6 reports the resource assignment for all 8 start periods and Figure 6 visualizes the resulting assignment.

Note that the assignment of the same resource for the entire processing time of an operation cannot be assured in the case of variable resource capacity as modelled in (1) - (9). Figure 5 with $C^A_{r,0} = C^A_{r,2} = 1$ and $C^A_{r,1} = 2$ illustrates this. Operation 2 is processed by resource 2 in period 2 and by resource 1 in period 3.

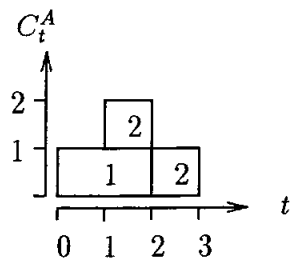

Figure 5: Assembly resource assignment in the case of variable capacity

The **assembly area assignment** can be treated as a resource assignment problem with constant resource capacity C^S and $p = 1, \ldots, P$ operations with start times $S_{s(p)}$. Using the above outlined algorithm we obtain the assembly area assignment given in Figure 7.

Finally, the **part assignment** can be viewed as a transportation problem where each delivery

t		0	1	3	5	6	8	9	12
$\mathcal{A}_t$		{8,11}	{9,11}	{7,12}	{10}	{2,10}	{4}	{3,4}	{5}
k	1	8	9	7	10	10	4	4	5
	2	8		12	10	10	4	4	5
	3	11	11	12		2	4	4	
	4	11	11	12		2		3	

Table 6: Calculation of the resource assignment

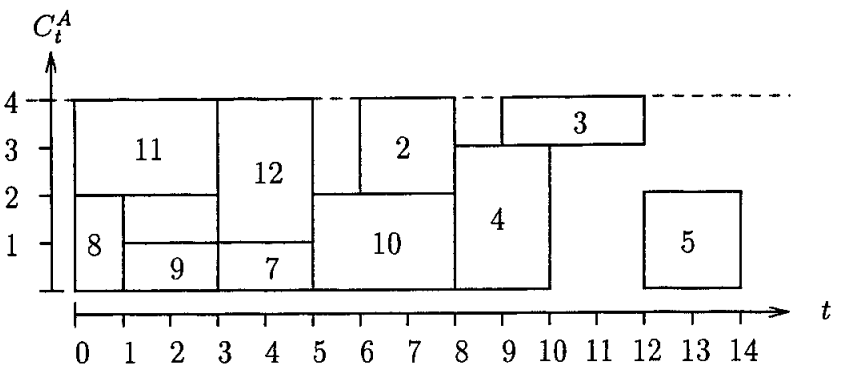

Figure 6: Assembly resource assignment

time t with $n_{i,t} > 0$ is a supply point with $n_{i,t}$ units of supply and each operation j with $q_{j,i} > 0$ is a demand point with demand $q_{j,i}$. Supplying parts from time t to an operation j which is scheduled in $S_j < t$ is forbidden and hence penalized with costs of ∞ per unit. All other costs are set to 1 per unit. Any feasible solution of the transportation problem will provide an assignment. Figure 8 gives the (only) part assignment for the schedule of the example problem.

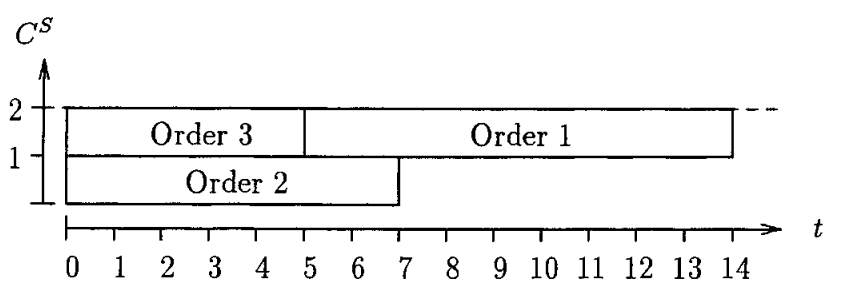

Figure 7: Spatial resource assignment

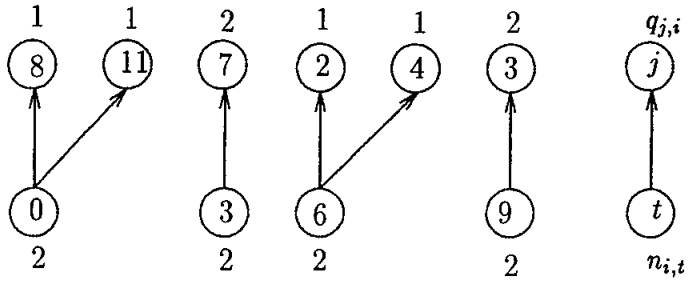

Figure 8: Part assignment

4 Experimental Evaluation

4.1 Test Instances

For evaluation purposes, a set of 270 problem instances was generated with a parameter controlled instance generator for assembly type problems which builds up on ProGen (cf. Kolisch et al. [20]). The instance set can be divided w.r.t. the size into small and large instances. A full factorial experimental design with three independent problem parameters was employed for both sets. Each of the independent problem parameters corresponds with one of the 'hard' constraints (4), (5), and (6) of ASP, respectively. All other parameters were randomly drawn from the given intervals.

For the small test instances, there is $P = 3$, $J_p \in [3, 5]$, $R^A = 1$, $RF^A = 1$, $I = 1$, $PF = 0.7$, and $r_p \in [0, 5]$. J_p denotes the count of operations belonging to an order, RF^A denotes the resource factor of all operations. It measures the density of constraint type (4) (cf. Kolisch et al. [20]). Correspondingly, PF denotes the part factor which measures the density of constraint type (6). $RF^A = 1$ and $PF = 0.7$ express the fact that for all non-dummy operations there is $c_{j,r} > 0$ and for 7 out of 10 operations we have $q_{j,r} > 0$. The following parameters were randomly drawn from the specified intervals: $p_j \in [1, 3]$, $c_{j,r} \in [1, 3]$, $w_p \in [1, 5]$, $q_{j,i} \in [1, 2]$. The due date d_p was calculated as follows: $d_p = r_p + ES_{e(p)}$ with r_p, the release date of order p.

The following parameter values were set differently for the large test instances: $P = 10$, $J_p \in [5, 10]$, $R^A = 2$, $I = 2$, and $r_p \in [0, 20]$.

The three independent problem parameters are the assembly resource strength, the spatial resource strength, and the part strength. The assembly resource strength (RS^A) measures the scarcity of the assembly resource capacity given in (4). For $RS^A = 0$, the capacity for each resource type $r = 1, \ldots, R^A$ equals the minimum capacity needed when no two operations are processed in parallel. For $RS^A = 1$, the capacity for each resource type r suffices to realize the schedule $S = (ES_1, \ldots, ES_J)$. RS^A was set to 0.1, 0.3, and 0.5 for the small instances and to 0.1, 0.2 and 0.3 for the large instances, respectively. The spatial resource strength (RS^S) measures the available capacity of the assembly area. For $RS^S = 0$ there is just one assemble area available, while for $RS^S = 1$, there is enough capacity to assemble all orders in parallel. RS^S was set to 0.1. 0.5, and 1.0 for the small instances and to 0.3, 0.5, and 1.0 for the large instances, respectively. The part strength (PS) measures the timing, parts are made available for assembly. For $PS = 1$ each part type is made available such that the schedule $S = (ES_1, \ldots, ES_J)$ can be realized while for $PS = 0$, due to constraint type (6), operations cannot be started earlier than given in the schedule $S = (LS_1, \ldots, LS_J)$. Latest start times LS_j are calculated by backward recursion from

	lower bound		upper bound	
Rule	small	large	small	large
WEDD	19.25	61.93	19.04	3.01
WLFT	31.58	88.22	31.36	19.20
WSLK	50.04	90.34	49.82	20.81
EDD	54.47	109.74	54.21	32.59
WSPT	57.11	122.91	56.90	42.61
SLK	70.79	128.60	70.56	44.55
LFT	71.35	138.77	71.09	50.94
RAND(1)	78.17	169.34	77.92	71.01
SPT	84.37	174.55	84.09	74.16
MPR	6.08	57.65	5.88	0.45
RAND(100)	3.73	107.75	3.53	32.33
RAND(1000)	0.66	94.65	0.49	24.27

Table 7: Priority Rules — Experimental Results

the upper bound $T = \sum_{j=1}^{J} p_j$. For the small instances, PS was set to 0.7, 0.8, and 0.9, while for the large instances, it was set to 0.8, 0.9, and 1, respectively. Additionally, some variance was added to the part arrival times by using the part variability (PV). For $PV = 0$ there is no variation and parts arrive as calculated by the part strength. For $PV = 1$ and $PS = 0.5$ part arrival is randomly drawn out of the interval defined by the earliest and latest start times of the part requesting operations. PV was set to 0.3 for the small instances and to 0.4 for the large instances, respectively. Realizing a full factorial design with 5 replications for each combination of the independent parameter levels, $5 \cdot 3^3 = 135$ instances were generated for the small and the large instance set, respectively.

4.2 Results

Table 7 reports the average deviation from a lower and an upper bound of the objective function value. Thereby a distinction is made w.r.t. small and large instances. The lower bound has been obtained by solving for each small instance the MIP–model (1) – (9) and for each large instance the LP–relaxation of (1) – (9) with CPLEX (cf. Bixby and Boyed [4]). The modelling part was done with AMPL (cf. Fourer et al. [14]). Note that for the small instances the term "lower bound" coincides with the optimum. The upper bounds were calculated for each instance by taking the best objective function value derived with one of the 9 priority rules as well as RAND(100) and RAND(1000). RAND(x) denotes a simple random sampling schemes (cf. Kolisch and Hartmann [18]) where x solutions are generated by arbitrarily selecting one of the available operations in each stage of the list generation algorithm.

The first 9 rows of Table 7 report the results for the priority rules given in Table 4. All priority rules with the exception of SPT perform better than RAN (denoted in Table 7 as RAN(1)). Rules which take into account the order weight perform at the 0% level of confidence significantly better than their counterparts which are based on operation information only. Testing was done with the nonparametric Wilcoxon matched–pairs signed–rank test (cf. Kolisch [17]).

The last 3 rows give the results of multi–pass heuristics which employ not only one but several lists in order to generate multiple schedules. MPR denotes a multi–priority rule heuristic which

Parameter	Level	WEDD	MPR	RAND(1000)
	0.1	82.94	77.60	123.84
RS^A	0.2	56.76	52.73	86.50
	0.3	46.07	42.59	73.59
		(0.00)	(0.00)	(0.00)
	0.3	62.45	59.08	66.39
RS^S	0.5	61.51	56.03	84.94
	1.0	61.81	57.82	123.59
		(0.85)	(0.79)	(0.00)
	0.8	41.63	38.48	70.85
PS	0.9	55.00	50.42	87.28
	1.0	89.14	84.03	125.79
		(0.00)	(0.00)	(0.00)

Table 8: Effect of the Problem Parameters (Large Instances — Average Deviation from the Lower Bounds)

simply uses all priority rules except RAN in order to generate eight solutions (cf. Boctor [5]). All multi–pass heuristics show better results than the single–pass heuristics for the small instances while only MPR is better than the single–pass heuristics for the large instances. The performance of the random sampling schemes deteriorates dramatically when the problem size and hence the solution space increases. This effect is well known from other problem classes, e.g. the resource–constrained project scheduling problem (cf. Kolisch and Hartmann [18]).

Table 8 reports the effect of the independent problem parameters on the best single–pass heuristic WEDD, and the two multi–pass heuristics MPR and RAND(1000). The value in parenthesis gives the level of confidence when testing with the nonparametric Kruskal–Wallis test whether the problem parameter has a significant influence on the solution quality of a heuristic. The effect of the assembly resource strength on the performance of all three heuristics is significant at the 0% level of confidence. The solution quality of all three solution procedures deteriorates when the RS^A level is lowered, i.e. capacity becomes scarce. A significant effect of the spatial resource strength is only noticeable for RAND(1000). The solution quality deteriorates with increasing level of RS^S, i.e. with growing spatial capacity. The explanation is that a larger spatial capacity causes a greater number of available operations because orders can be processed in parallel. The greater number of available operations in turn lowers the selection probability of the operations which would lead to the best schedule. Finally, the part strength shows a significant effect on the solution quality of all three solution procedures. A lower part strength, i.e. a late delivery of parts, gives way to better results of all three heuristics.

5 Summary and Conclusions

We have introduced the assembly scheduling problem which is concerned with the scheduling of assembly operations belonging to different orders subject to assembly resource, spatial resource and part availability constraints. The objective is to minimize the sum of the weighted tardiness. A MIP–formulation and a simple list scheduling heuristic have been developed. Both handle the resource and part availability constraints in an aggregated fashion. Once a feasible schedule has been determined, the allocation of resources and parts to operations is done afterwards. In an

experimental investigation we evaluated different priority rules, a multi–priority rule heuristic and random sampling approaches. Simple single–pass priority rules show a satisfactory, the multi–pass multi–priority rule method shows a good performance; the performance of the random sampling scheme deteriorates for large problem instances. Further efforts are headed towards more sophisticated metaheuristic methods to solve the assembly scheduling problem (cf. Kolisch and Heß [19]).

Acknowledgement. I am thankful to Karsten Heß and Stephan Nardello for coding the algorithms and performing the computational experiments as well as to Andreas Drexl for his continuous support.

References

[1] A. Agrawal, G. Harhalakis, I. Minis, and R. Nagi. "Just–in–time" production of large assemblies. *IIE Transactions*, 28:653–667, 1996.

[2] M.G. Anwar and R. Nagi. Integrated lot–sizing and scheduling for just–in–time production of complex assemblies with finite set–ups. *International Journal of Production Research*, 35(5):1447–1470, 1997.

[3] A. Balakrishnan, R.L. Francis, and S.J. Grotzinger. Bottleneck resource allocation in manufacturing. *Management Science*, 42(11):1611–1625, 1996.

[4] N. Bixby and E. Boyed. *Using the CPLEX callable library.* CPLEX Optimization Inc., Houston, 1996.

[5] F.F. Boctor. Some efficient multi–heuristic procedures for resource–constrained project scheduling. *European Journal of Operational Research*, 49:3–13, 1990.

[6] J. Carlier and A.H.G. Rinnooy Kan. Scheduling subject to nonrenewable–resource constraints. *Operations Research Letters*, 1(2):52–55, 1982.

[7] J.F. Chen and W.E. Wilhelm. An evaluation of heuristics for allocating components to kits in small–lot, multi–echelon assembly systems. *International Journal of Production Research*, 31(12):2835–2856, 1993.

[8] J.F. Chen and W.E. Wilhelm. Optimizing the allocation of components to kits in small–lot, multi–echelon assembly systems. *Naval Research Logistics*, 41:229–256, 1994.

[9] R. de Boer, A. van Harten, and W.H.M Zijm. Scheduling multi–processor tasks with precedence constraints and release and due dates. Technical report, Faculty of Mechanical Engineering, University of Twente, 1997.

[10] A. Drexl and R. Kolisch. Assembly management in machine tool manufacturing and the PRISMA–Leitstand. *Production and Inventory Management Journal*, 37(4):55–57, 1996.

[11] S.E. Elmaghraby. *Activity networks: Project planning and control by network models.* Wiley, New York, 1977.

[12] B. Faaland and T. Schmitt. Scheduling tasks with due dates in a fabrication/assembly process. *Operations Research*, 35(3):378–381, 1987.

[13] B. Faaland and T. Schmitt. Cost–based scheduling of workers and equipment in a fabrication and assembly shop. *Operations Research*, 41(2):253–268, 1993.

[14] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL A modeling language for mathematical programming.* boyd & fraser, Danvers, 1993.

[15] S. Hartmann. A competitive genetic algorithm for resource–constrained project scheduling. Technical Report 451, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, 1997.

[16] S.O. Kim and M.J. Schniederjans. Heuristic framework for the resource constrained multi–project scheduling problem. *Computers & Operations Research*, 16(6):541–556, 1989.

[17] R. Kolisch. Efficient priority rules for the resource–constrained project scheduling problem. *Journal of Operations Management*, 14(3):179–192, 1996.

[18] R. Kolisch and S. Hartmann. Heuristic algorithms for the resource–constrained project scheduling problem: Classification and computational analysis. In J. Weglarz, editor, *Handbook on recent advances in project scheduling*, Kluwer, Amsterdam, 1998.

[19] R. Kolisch and K. Heß. Efficient methods for scheduling make–to–order assemblies under resource, assembly area, and part availability constraints. Technical report, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, Kiel.

[20] R. Kolisch, A. Sprecher, and A. Drexl. Characterization and generation of a general class of resource–constrained project scheduling problems. *Management Science*, 41(10):1693–1703, 1995.

[21] I.S. Kurtulus and S.C. Narula. Multi–project scheduling: Analysis of project performance. *IIE Transactions*, 17:58–66, 1985.

[22] P. Lalsare and S. Sen. Evaluating backward scheduling and sequencing rules for an assembly shop environment. *Production and Inventory Management Journal*, 36(4):71–77, 1995.

[23] S.R. Lawrence and T.E. Morton. Resource–constrained multi–project scheduling with tardy costs: Comparing myopic, bottleneck, and resource pricing heuristics. *European Journal of Operational Research*, 64:168–187, 1993.

[24] J.K. Lee, K.J. Lee, H.K. Park, J.S. Hong, and J.S. Lee. Developing scheduling systems for Daewoo shipbuilding: DAS project. *European Journal of Operational Research*, 97:380–395, 1997.

[25] J.V. Moccellin. A two–stage approach to resource–constrained multi–project scheduling problems. *Policy and Information*, 13(1):1–18, 1989.

[26] J.J. Moder, C.R. Phillips, and E.W. Davis. *Project management with CPM, PERT and precedence diagramming.* Van Nostrand Reinhold, New York, 3. edition, 1983.

[27] R.P. Mohanty and M.K. Siddiq. Multiple projects — Multiple resource–constrained scheduling: A multi–objective analysis. *Engineering Costs and Production Economics*, 18:83–92, 1989.

[28] R.P. Mohanty and M.K. Siddiq. Multiple projects — multiple resource–constrained scheduling: Some studies. *International Journal of Production Research*, 27(2):261–280, 1989.

[29] K. Neumann and C. Schwindt. Activity–on–node networks with minimal and maximal time lags and their application to make–to–order production. *OR Spektrum*, 19(3):205–217, 1997.

[30] S.Y. Nof, W.E. Wilhelm, and H.-J. Warnecke. *Industrial Assembly.* Chapman & Hall, London, 1997.

[31] M.I. Norbis and J.M. Smith. Two level heuristic for the resource constrained scheduling problem. *International Journal of Production Research*, 24(5):1203–1219, 1989.

[32] J.H. Patterson. Project scheduling: The effects of problem structure on heuristic performance. *Naval Research Logistics Quarterly*, 20:95–123, 1976.

[33] C.N. Potts, S.V. Sevastjanov, V.A. Strusevich, L.N. Van Wassenhove, and C.M. Zwaneveld. The two–stage assembly scheduling problem: Complexity and approximation. *Operations Research*, 43(2):346–355, 1995.

[34] A.A.B. Pritsker, L.J. Watters, and P.M. Wolfe. Multiproject scheduling with limited resources: A zero–one programming approach. *Management Science*, 16:93–107, 1969.

[35] R.S. Russell and B.W. Taylor III. An evaluation of sequencing rules for an assembly shop. *Decision Sciences*, 16(1):196–212, 1985.

[36] J.M.J. Schutten. List scheduling revisited. *Operations Research Letters*, 18:167–170, 1996.

[37] A. Sprecher, R. Kolisch, and A. Drexl. Semi–active, active, and non–delay schedules for the resource–constrained project scheduling problem. *European Journal of Operational Research*, 80:94–102, 1995.

[38] J.M. van den Akker, J.A. Hoogeveen, and S.L. van de Velde. Parallel machine scheduling by column generation. Technical report, Center for Operations Research and Econometrics, Catholic University of Louvain, 1995.

[39] J.T. van der Vaart, J. de Vries, and J. Wijngaard. Complexity and uncertainty of materials procurement in assembly situations. *International Journal of Production and Economics*, 46–47:137–152, 1996.