Jena Research Papers in Business and Economics

Level Scheduling of Mixed-Model Assembly Lines under Storage Constraints

Nils Boysen, Malte Fliedner, Armin Scholl

09/2007

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

Level Scheduling of Mixed-Model Assembly Lines under Storage Constraints

Nils Boysen[a,*], Malte Fliedner[a], Armin Scholl[b]

[a] *Universität Hamburg, Institut für Industrielles Management, Von-Melle-Park 5, D-20146 Hamburg,* {boysen,fliedner}@econ.uni-hamburg.de

[*] *Corresponding author, phone +49 40 42838-4640.*

[b] *Friedrich-Schiller-Universität Jena, Lehrstuhl für Betriebswirtschaftliche Entscheidungsanalyse, Carl-Zeiß-Straße 3, D-07743 Jena,* a.scholl@wiwi.uni-jena.de

Abstract
In a mixed-model assembly line different models of a common base product can be manufactured in intermixed production sequences. A famous solution approach for the resulting short-term sequencing problem is the so called level scheduling problem, which aims at evenly smoothing the material requirements over time in order to facilitate a just-in-time supply. However, if materials are delivered in discrete quantities, the resulting spreading of material usages implies that issued cargo carriers of a respective material remain at a station for a longer period of time. In practical applications with plenty materials required per station, this procedure might lead to bottlenecks with respect to the scarce storage space at stations. This paper investigates level scheduling under the constraint that the induced part usage patterns may not violate given storage constraints. The resulting sequencing problem is formalized and solved by suited exact and heuristic solution approaches.

Keywords: Mixed-Model Assembly Lines; Sequencing; Dynamic Programming; Simulated Annealing

1 Introduction

In mixed-model assembly systems different product models are jointly manufactured on the same line. The use of modern production technologies allows the reduction of setup

times and costs between models to such an extent that they can be ignored and facultative, intermixed model sequences can be assembled (lot-size one). Typically, all models are variations of the same base product and only differ in specific customizable product attributes, which results to an assembly of different parts or modules. Although the application of flexible machinery and cross-trained workers makes facultative model sequences technically possible, the sequence of models has nevertheless extensive economic impacts. Thus, the sequencing of mixed-model assembly lines has attracted the attention of research, which aims to support model sequencing with suited optimization approaches for decades. A recent general survey on model sequencing is provided by Boysen et al. (2007a).

The most wide-spread attention in research (see the surveys of Kubiak, 1993; Dahmala and Kubiak, 2005; Boysen et al., 2006) and practical applications (e.g. Monden, 1998; Duplaga et al., 1996; Mane et al., 2002) alike, received the so called "level scheduling", which constitutes a major cornerstone of the famous "Toyota Production System" (see Monden, 1998). This approach aims at evenly smoothing the material requirements induced by the model sequence over time, so that a just-in-time (JIT) supply of material is facilitated and safety stocks are minimized. For that purpose, each material receives a (theoretical) target consumption rate, which is determined by distributing its overall demand evenly over the planning horizon. Hence, a sequence is sought where actual consumption rates of materials are as close as possible to target rates. Kubiak (1993) refers to this material oriented level scheduling as the Output Rate Variation (ORV) problem, because materials constitute the outputs of preceding production levels, whose actual demand rates are to be leveled.

However, pilot studies regarding the application of level scheduling for sequencing car models at major German manufacturers brought forward that the resulting model sequences and their associated material usage patterns often violate storage space restrictions. The space available at the stations can be extremely scarce, as new assembly lines are often constructed in already existing factory buildings. Due to the decreasing vertical integration, a large number of parts have to be delivered to and temporarily stored near the line. Furthermore, additional product features are often introduced during the lifetime of the assembly system, so that the total amount of required material tends to steadily grow over time. The tremendous variety of car models offered by some manufacturers, e.g. 10^{32} theoretically possible models at German manufacturer BMW (Meyr, 2004), requires parts and modules to be handled in considerable numbers and versions causing large space requirements. This general scarcity of storage space near the line (limited space, but many parts to be stored) is, however, not an individual problem of car manufacturers, but a general problem arising in many assembly systems (see Klampfl et al., 2006).

Under scarce storage space, the part consumption patterns induced by level scheduling can lead to conflicts as they do not consider how the material supply is typically organized. Parts are usually stored and moved in some kinds of logistic handling units, referred to as cargo carriers, and not unit by unit. An in-house or even a third party logistic provider (3PL) cyclically delivers these carriers (e.g. a euro-pallet or special container) to the line, so that parts arrive in discrete quantities larger than one. In the case

of a 3PL, parts of a wide range of suppliers are consolidated and stored in a consignment stock adjacent to the line, from which they are issued carrier by carrier by the manufacturer as soon as his own intermediate storage at the station is depleted and the part is required again. Thus, parts once transferred from the 3PL's consignment stock are to be stored directly near the line until all units are assembled and a new cargo carrier arrives. But also if an in-house department organizes the supply, parts usually arrive in fixed quantities as a unit by unit supply would simply require too many tours or expensive transportation and handling equipment.

These two prerequisites: (i) scarce storage space at stations and (ii) a wide variety of parts per station issued in discrete quantities larger than one, lead to the following problem when applying level scheduling. The ORV spreads the part usage evenly over the planning horizon, so that cycles in which the same part is assembled tend to be shifted apart. As a consequence, the time span until all parts of a cargo carrier are completely consumed is extended and average inventory per production cycle is increased, so that the storage space per station might not be sufficient to maintain the plenty parts required not to mention increased capital cost. In such a setting, manufacturers which aim at model sequences in accordance with the JIT philosophy need to additionally observe the limited storage space.

The remainder of the paper is organized as follows. In Section 2 a suited mathematical program is formulated which accounts for the aforementioned phenomenon. Then, Sections 3 and 4 provide suited exact (Dynamic Programming) and heuristic solution approaches (Simulated Annealing). A computational study, which evaluates the performance of the proposed approaches is presented in Section 5. Finally, Section 6 concludes the paper with a general discussion of level scheduling under these conditions.

2 Mathematical model

We formulate the mathematical program as an extension of the traditional ORV problem, which can be described as follows: Consider a set M of models each of which having a demand d_m for copies of model m to be produced during a specific period (e.g. one day or shift), which is further divided into T production cycles with $\sum_{m \in M} d_m = T$. Each model m consists of different parts p (with $p \in P$). The production coefficients a_{pm} specify the number of units of part p required for the assembly of one unit of model m. The matrix of coefficients $A = (a_{pm})$ is usually referred to as bill of material. By means of the total demand for part p required by all copies of all models m throughout the planning horizon, the target demand rates per production cycle r_p are calculated as follows:

$$r_p = \frac{\sum_{m \in M} d_m \cdot a_{pm}}{T} \qquad \forall p \in P \tag{1}$$

Together with the integer variables x_{mt}, which represent the total cumulative production quantity of model m up to cycle t, the well-known ORV problem can be modeled as follows (Joo and Wilhelm, 1993; Monden, 1998; Bautista et al., 1996):

$$\text{(ORV) Minimize } Z = \sum_{t=1}^{T} \sum_{p \in P} Z_{pt} \tag{2}$$

$$0 \leq x_{mt} - x_{m,t-1} \leq 1 \qquad \forall\, m \in M; t = 2, ..., T \tag{3}$$

$$\sum_{m \in M} x_{mt} = t \qquad \forall\, t = 1, ..., T \tag{4}$$

$$Z_{pt} = \left(\sum_{m \in M} x_{mt} \cdot a_{pm} - t \cdot r_p \right)^2 \qquad \forall\, p \in P;\, t = 1, \ldots, T \tag{5}$$

$$x_{m0} = 0;\, x_{mT} = d_m \qquad \forall\, m \in M \tag{6}$$

$$x_{mt} \in \mathbb{N}^0 \qquad \forall\, m \in M;\, t = 1, \ldots, T-1 \tag{7}$$

Objective function (2) minimizes the squared deviation of actual material demands from the equally distributed target demands accumulated over all cycles t and parts p. In (5), the auxiliary variables Z_{pt} are defined separately as they are used later on. Constraints (3) ensure that cumulative production quantities increase monotonically throughout the planning horizon. The production of exactly one copy of a single model in each cycle t is ensured by constraints (4), whereas constraints (6) force the models to be produced in the required quantities.

We restrict our discussion to the "sum of squared deviations"-case although other deviation functions discussed in the literature, e.g. maximum of absolute deviations or sum of Euclidean deviations (see Boysen et al., 2006, 2007a), could be utilized in a similar fashion. As both the model formulation and all solution procedures presented simply require a substitution of the respective objective function our argumentation holds for these cases as well.

This basic model is now extended by storage constraints. To exactly cover the inventories l_{pt} of a part p during production cycle t, the following assumptions with regard to the delivery of parts are introduced:

- The manufacturer issues a cargo carrier for a part p as soon as the model copy requires a unit of p and his own inventory directly at the line is empty. Note, that as the sequence is communicated to the logistics provider in advance, it is not difficult to exactly match this point in time in principle. If a model copy in cycle t requires a part p and a cargo carrier with G_p units of part p is issued, it is immediately accessible at the beginning of cycle t, so that G_p units reduced by the amount of parts directly consumed by the current model copy remain in the part inventory in cycle t.

- In our discussion we only consider parts which are issued in part-specific cargo carriers of size $G_p > 1$, which turn out to be the vast majority of parts in real-world applications. Parts which are made available unit-by-unit have no impact on the inventory near the line and are thus not subject to the space restrictions. Such parts can, however, be considered in the objective function.

M	set of models (index m)
T	number of production cycles (index t)
S	set of stations (index s)
P	set of materials (index p)
P_s	set of parts required for assembly operations in station s
a_{pm}	demand coefficient: number of units of part p required for producing one unit of model m
d_m	demand for units of model m
r_p	target demand rate with respect to part p
C_s	storage space available at station s
c_p	storage space required by a unit of part p
G_p	capacity of the cargo carrier for part p
L_p	quantity of part p initially stored in stock
y_{pt}	integer variable: number of cargo carriers for part p issued up to cycle t
x_{mt}	integer variable: number of scheduled copies of model m up to cycle t
l_{pt}	continuous variable: quantity of part p stored during cycle t
Z_{pt}	squared deviation of cumulated actual demand and target demand of part type p in cycle t

Table 1: Notation

- As usual or even necessary in practice due to organizational and distance considerations, each cargo carrier is exclusively assigned to the station where a respective part is required. To simplify the model we additionally assume that the same part is not required at multiple stations. Nevertheless, the model and the presented algorithms could be readily extended to also cover this more general case.

- We assume that the available storage space is limited in only one dimension, as is the standard assumption of most inventory-oriented sequencing and scheduling models. This assumption is sufficient in many real-world assembly lines facing scarce storage space. In order to avoid a repacking of parts, manufacturers generally aim at storing parts in the cargo carriers parts arrive in at the stations (Boysen et al., 2007b). However, if the scarce storage space inhibits that all cargo carriers of the stations' parts can be stored simultaneously near the line, this policy would considerably restrict the model portfolio of the line. A model containing all parts of a station could not be assembled, as this would require a cargo carrier of any kind to be stored at the same time. As this is unacceptable, parts typically need to be removed from their carriers and repacked into a mixed carrier like some kind of rack or shelf next to the line. Thus, factor c_p represents the shelf space occupied by a unit of part p.

With these premises on hand, the part inventory l_{pt} of a part p in cycle t resulting from a model sequence can be exactly determined. Part inventory l_{pt} can then be weighted with the demand for storage space c_p required of a unit of type p and compared to the total

storage space C_s available at a station $s \in S$. The storage constrained level scheduling problem ORV-S can now be formulated on the basis of the ORV by introducing the following additional constraints:

$$\sum_{m \in M} x_{mt} \cdot a_{pm} + l_{pt} = y_{pt} \cdot G_p + L_p \qquad \forall\, p \in P;\, t = 1, \ldots, T \tag{8}$$

$$\sum_{p \in P_s} l_{pt} \cdot c_p \leq C_s \qquad \forall\, s \in S;\, t = 1, \ldots, T \tag{9}$$

$$0 \leq y_{pt} - y_{pt-1} \leq 1 \qquad \forall\, p \in P;\, t = 2, \ldots, T \tag{10}$$

$$y_{pt} \in \mathbb{N}^0;\, l_{pt} \geq 0 \qquad \forall\, p \in P;\, t = 1, \ldots, T \tag{11}$$

The balance equations (8) define the quantity l_{pt} stored per part p and cycle t as the overall number of issued units (number of issued carriers y_{pt} times carrier size G_p) plus initial stock L_p minus the cumulative consumption of the part through previously scheduled model copies. Constraints (9) ensure that the available storage space C_s per station s is not violated by part inventories l_{pt} weighted with the respective demand for storage space c_p in neither cycle t. Constraints (10) enforce the integer variables y_{pt}, which represent the number of cargo carriers for part type p issued up to cycle t, to monotonically increase over time.

As the original ORV problem is well-known to be NP-hard (c.f. Zhu and Ding, 2000), the modified ORV-S problem, which contains the ORV as a subproblem, is also NP-hard. Moreover, the storage constraints can complicate the model's solution considerably, as the retrieval of a feasible solution is not trivial anymore and even deciding whether or not a feasible solution exists might be hard.

Example: The conflicting principles of leveling part usage rates and observing storage constraints shall be clarified by an example with the help of the data given in Table 2. Three models $m = 1, 2, 3$ with demands d_m are produced on the line. The coefficients a_{pm} specify the number of units of two parts $p = 1, 2$ required to produce one unit of model m. Both parts are assembled at the one and only station of the assembly line ($S = \{1\}$), whose storage space is assumed to be $C_s = 3$. The parts are supplied in containers each of which contains $G_p = 3$ units. For storing a unit of part p, a storage space $c_p = 1$ is required for each part p. It is further supposed that the stock is completely empty at the beginning of the shift ($L_1 = L_2 = 0$).

Table 2 displays two alternative model sequences π along with the resulting deviations $Z_{pt}(\pi) = \left(\sum_{\tau=1}^{t} a_{\pi_\tau p} - t \cdot r_p\right)^2$ for all $t = 1, \ldots, T$, $p \in P$. Additionally, the quantities l_{pt} of part p stored in each cycle $t = 1, \ldots, 5$ are given. Solution A is the optimal ORV solution, which leads to an objective value of $Z(\pi) = 0.8$, but violates the storage constraint in the first cycle ($l_{11} + l_{21} = 4 > 3$). Solution B is the optimal ORV-S solution getting by with the storage space available and an objective function value of $Z(\pi) = 1.0$.

		m				
	a_{pm}	1	2	3	G_p	c_p
p	1	1	1	0	3	1
	2	1	0	1	3	1
	d_m	2	1	2		

Table 2: Example data

t	1	2	3	4	5	1	2	3	4	5
π	1	3	2	3	1	3	1	2	3	1
$a_{1\pi_t}$	1	0	1	0	1	0	1	1	0	1
$a_{2\pi_t}$	1	1	0	1	1	1	1	0	1	1
l_{1t}	2	2	1	1	0	0	2	1	1	0
l_{2t}	2	1	1	0	2	2	1	1	0	2
$Z_{1t}(\pi)$	0.16	0.04	0.04	0.16	0	0.36	0.04	0.04	0.16	0
$Z_{2t}(\pi)$	0.04	0.16	0.16	0.04	0	0.04	0.16	0.16	0.04	0
	solution A					solution B				

Table 3: Impact of the model sequence on inventory and part usages

3 Dynamic Programming approach

The first solution approach presented, is an exact Dynamic Programming (DP) approach and an extension of the DP approach provided by Bautista et al. (1996) for the basic ORV problem. Like their approach our modified procedure is based on an acyclic digraph $G = (V, E, r)$ with a node set V divided into $T+1$ *stages*, a set E of arcs connecting nodes of adjacent stages and a node weighting function $r : V \to \mathbb{R}$. Each sequence position t is represented by a stage which contains a subset $V_t \subset V$ of nodes representing *states* of the production system in cycle t. Additionally, a start level 0 is introduced. Each index $i \in V_t$ identifies a state (t, i) defined by the vector $\mathbf{X}_{ti}$ of cumulated quantities X_{tim} of all models $m \in M$ produced up to cycle t. It is sufficient to store the cumulated quantities instead of the partial sequence up to cycle t, because the objective function is separable with respect to cycles. The values $Z_{p,t+1}$ only depend on the cumulated production quantities X_{tim} and the model produced in $t + 1$ (cf. (5)).

The following conditions define all *feasible* states to be represented as nodes of the graph:

$$\sum_{m \in M} X_{tim} = t \qquad \forall\, t = 0, \ldots, T;\ i \in V_t \tag{12}$$

$$0 \le X_{tim} \le d_m \qquad \forall\, m \in M;\ t = 0, \ldots, T;\ i \in V_t \tag{13}$$

Obviously, the node set V_0 contains only a single node (initial state $(0, 1)$) corresponding to the vector $\mathbf{X}_{01} = [0, 0, \ldots, 0]$. Similarly, the node set V_T contains a single node (final state $(T, 1)$) with $\mathbf{X}_{T1} = [d_1, d_2, \ldots, d_{|M|}]$. The remaining stages have a variable number of nodes depending on the number of feasible model vectors $\mathbf{X}_{ti}$.

Two nodes (t,i) and $(t+1,j)$ of two consecutive stages t and $t+1$ are connected by an arc if the associated vectors $\mathbf{X}_{ti}$ and $\mathbf{X}_{t+1j}$ differ only in one element, i.e., a copy of exactly one model is additionally produced in cycle $t+1$. This is true if $X_{tim} \leq X_{t+1jm}$ holds for all $m \in M$, because both states are feasible according to (12) and (13). The overall arc set is defined as follows:

$$E = \{((t,i),(t+1,j)) \mid t = 0,\ldots,T-1; i \in V_t; j \in V_{t+1} \text{ and } X_{tim} \leq X_{t+1jm}\ \forall m \in M\} \tag{14}$$

The produced quantities of all models up to cycle t in a state (t,i) directly determine the cumulative demands D_{tip} for all parts p of the respective partial schedule:

$$D_{tip} = \sum_{m \in M} X_{tim} \cdot a_{pm} \qquad \forall\, p \in P \tag{15}$$

Now, to each node corresponding to a state (t,i) a unique node weight r_{ti} is assigned, which is equal to the sum of squared deviations from the target rates in cycle t and calculated as follows:

$$r_{ti} = \sum_{p \in P} (D_{tip} - t \cdot r_p)^2 \qquad \forall\, t = 0,\ldots,T;\ i \in V_t \tag{16}$$

With this graph on hand, the optimal solution of the ORV problem reduces to finding the shortest path from the unique source node at level 0 to the unique sink node at level T, where the length of the path is given by the sum of weights of the nodes contained. The length of the shortest path is equal to the minimum sum of squared deviations of the material demands induced by the optimal model sequence. The corresponding model sequence π can be deduced by considering each arc $((t,i),(t+1,j))$ with $t = 0,...,T-1$ on the shortest path SP. The model to be assigned at sequence position $t+1$ is the only one for which $X_{t+1jm} - X_{tim} = 1$ holds.

In order to account for the storage constraints of the modified ORV-S problem in this basic DP approach, each single state can be examined to determine whether or not the induced part inventories violate storage constraints. For this purpose it is to be checked if the actual stock level at a station weighted with its respective space coefficients c_p exceeds the given storage capacity C_s. The inventories I_{tip} of the parts $p \in P$ during a cycle t in state (t,i) are easily derived by (17), because they are either units from initial stock L_p not consumed by cumulated demand D_{tip} or residual units out of newly issued cargo carriers of size G_p. The special case $I_{tip} = 0$ arises when the carrier has been emptied at the beginning of t or was already empty and no unit of p has been required in cycle t.

$$I_{tip} = \begin{cases} L_p - D_{tip}, & \text{if } L_p \geq D_{tip} \\ 0, & \text{else if } (D_{tip} - L_p) \bmod G_p = 0 \\ G_p - (D_{tip} - L_p) \bmod G_p, & \text{otherwise} \end{cases} \qquad \forall\, p \in P \tag{17}$$

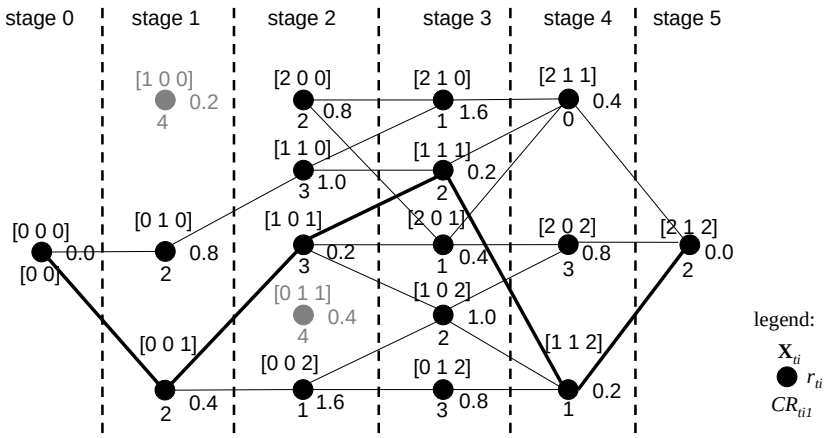

Figure 1: Example graph of the Dynamic Programming procedure

By defining the required storage capacity per station $CR_{tis} = \sum_{p \in P_s} I_{tip} \cdot c_p$, the state (t, i) can be defined as being *feasible*, iff the following condition holds:

$$CR_{tis} \leq C_s \qquad \forall\, s \in S \tag{18}$$

If this condition does not hold for a station s, then the state is infeasible and the corresponding node is excluded from the graph.

Instead of constructing the complete graph before computing the shortest path, the more efficient DP approach consists of determining the shortest path from the initial state to each node stage-by-stage ($t = 0, ..., T-1$). In order to do so, only two stages of the graph have to be stored simultaneously, because the shortest path to a node $(t+1, j)$ in stage $t+1$ is composed of a shortest path to a node (t, i) in stage t (already determined and stored) and the connecting arc $((t, i), (t+1, j))$. Among all such paths to $(t+1, j)$ one with minimal sum of node weights (length of path to (t, i) plus r_{t+1j}) is to be selected. The length-minimizing node (t, i) is stored as the predecessor in the shortest path to $(t+1, j)$ together with the length of this path. After reaching the final state $(T, 1)$ in stage T, the optimal path can be retrieved in backward direction stage-by-stage using the stored predecessor nodes.

Example (cont.): The resulting DP-graph along with the (bold-faced) shortest path for our example is depicted in Figure 1. This path corresponds to the optimal model sequence $\pi = \{3, 1, 2, 3, 1\}$ with minimal total deviation $Z = 1.0$. Nodes representing infeasible states are colored light grey.

4 A Simulated Annealing approach

As within the DP approach the number of states to be inspected raises exponentially with the number of cycles T and models $|M|$, problem instances of real-world size are barely solvable to optimality. Hence, a Simulated Annealing (SA) approach is presented in the following. SA is a stochastic local search meta-heuristic, which bases the acceptance of a modified neighboring solution on a probabilistic scheme inspired by thermal processes for obtaining low-energy states in heat baths (e.g. Kirkpatrick et al., 1983; Aarts et al., 1997).

Preliminary tests with simple priority rule based start heuristics, where the model sequence is simply filled from left to right and actual choices are guided by a priority value, revealed that those approaches predominantly lead to infeasible solutions. Thus, a meta-heuristic seems much more promising to guide the search into feasible regions of the solution space. Although other meta-heuristics like tabu search are possible, we decided for SA as it is a quite simple yet powerful approach, which is successfully applied to real-world sequencing problems, e.g. at the French car manufacturer Renault (see Solnon et al., 2006).

Our SA approach operates on a vector π with elements π_t (with $t = 1, \ldots, T$), storing the model actually assigned to the respective sequence position t. As a neighborhood-function we apply a simple swapping move, where two models at randomly determined sequence positions are interchanged. The initial solution vector is randomly filled with models m in accordance with their demands d_m. For a given sequence vector π the respective objective function value can be determined as follows:

$$Z(\pi) = \sum_{t=1}^{T} \sum_{p \in P} Z_{pt}(\pi) \tag{19}$$

If the acceptance decision for a solution in the neighborhood is taken on the basis of this value only, the procedure might return too many infeasible solutions. As simple repair mechanisms are not on hand, we apply the idea of penalty costs to penalize solutions, which violate the storage constraints in proportion to the degree of violation.
Thus, a modified objective function $Z^{SA}(\pi)$ is applied to guide the acceptance of neighborhood solutions on the basis of a given penalty value PV:

$$Z^{SA}(\pi) = Z(\pi) + PV \cdot \left(\sum_{s \in S} \sum_{t=1}^{T} \max \left\{ \sum_{p \in P_s} I_{pt} \cdot c_p - C_s; 0 \right\} \right) \tag{20}$$

Each unit of the actual inventory I_{pt} of all parts assigned to a station in excess of the available storage capacity C_s is weighted with the penalty value PV. The actual inventory I_{pt} of part p in cycle t induced by sequence π is calculated in accordance with equation (17).

A proper determination of the penalty value PV turns out to be a very critical task with regard to the performance of the SA. On the one hand, it is to be avoided that the SA gets stuck in less promising regions of the solution space (feasible solutions with

poor objective values or even infeasible solutions surrounded by neighboring solutions violating even more storage constraints), which cannot be left due to only marginal acceptance probabilities of infeasible solutions. On the other hand, local optima might be missed if the search enters infeasible regions of the solution space too readily. It thus seems recommendable to employ variable penalty values in terms of a diversification-intensification strategy. For this purpose, penalty value PV is initialized with $PV^{start} = Z(\pi)^{start} \cdot 10$, where $Z(\pi)^{start}$ is the objective value of the first random solution (may it be feasible or not). Then, with each improvement of the modified objective value $Z^{SA}(\pi)$ the search towards a local optimum is intensified by increasing the penalty value by $PV := PV \cdot 1.2$. If within the last 30 swap moves (in direct succession) no improvement of $Z^{SA}(\pi)$ is obtained, PV is reduced by $PV := PV \cdot 0.5$ to direct the search into other regions of the solution space (diversification).

A neighborhood solution π' obtained by a swap move is accepted to replace the actual solution π as the starting point for the next iteration on the basis of the following traditional probability scheme (e.g. Aarts et al., 1997):

$$Prob(\pi' \text{ replacing } \pi) = \begin{cases} 1, & \text{if } Z^{SA}(\pi') \leq Z^{SA}(\pi) \\ exp\left(\frac{Z^{SA}(\pi) - Z^{SA}(\pi')}{C}\right), & \text{otherwise} \end{cases} \tag{21}$$

Our SA is guided by a simple static cooling schedule (see Kirkpatrick et al., 1983). The initial value for control parameter $C = PV^{start} \cdot 10$ is chosen, which is continuously decreased in the course of the procedure by multiplying it with factor 0.995 in each iteration. If the penalty value PV falls below $\frac{PV^{start}}{10}$, the procedure is restarted with a new random sequence and a re-initialized control parameter C. A total of 10,000 neighboring solutions are evaluated by our SA approach and the feasible solution (if obtained) with the minimum objective function value $Z(\pi)$ is returned. Within our computational study, we only report results for the values of the control parameters described above, as preliminary studies indicated these parameter values to be most promising.

5 Computational study

As no established test-bed is available for a computational study, we first elaborate on the instance generation. Then, a sensitivity analysis is presented in order to examine the influence of storage constraints on the ability of finding leveled sequences. Finally, numerical results on the performance of the proposed algorithms are discussed.

5.1 Instance generation

In our computational study, we distinguish between two classes of test instances: case A (sensitivity analysis) and case B (algorithmic performance) instances.

To derive these instance classes the input parameters listed in Table 4 are used to produce the demand coefficients for parts a_{pm}, model demands d_m, sizes of cargo carriers G_p, and storage space per station C_s defining an ORV-S instance.

symbol	description	values case A	values case B
T	number of production cycles	15	10, 15, 20, 25
$\|M\|$	number of models	5, 7, 9	
$\|P\|$	number of parts	4, 6, 8	4, 6, 8, 10
$\|S\|$	number of stations	2	
c_p	demand for storage space per part unit	1	
MSC	maximum size of cargo carrier (in part units)	2, 3,. . . , 9	3, 5, 7
$PROB$	probability of a model m containing part p	0.3, 0.5, 0.7	
B	parameter to fix storage space per station	0.4, 0.45,. . . , 0.9	0.5, 0.7, 0.9

Table 4: Parameters for instance generation

Within each test case, the parameters are combined in a full-factorial design, so that 2376 (1296) different case A (case B) instances were obtained. On the basis of a given set of parameters each single instance is generated as follows:

- *Demand coefficient matrix:* For each individual demand coefficient a_{pm} a $[0,1]$-random number rnd is drawn and compared to the probability $PROB$ of a model containing the respective part, so that coefficients can be fixed with regard to the following formula:

$$a_{pm} = \begin{cases} 1, \text{ if } rnd \leq PROB \\ 0, \text{ otherwise} \end{cases} \quad \forall\, m \in M;\, p \in P \tag{22}$$

- *Demand for models:* At first, each model demand d_m is initialized to one unit. Then, demands of randomly drawn (equally distributed) models are increased by one unit, until the overall model demand ($\sum_{m \in M} d_m$) equals the given number of production cycles T.

- *Size of cargo carrier:* The number of units G_p of parts p to be stocked on a cargo carrier are randomly drawn by an equally distributed integer random number out of the interval $[2, MSC]$.

- *Material-station-assignment:* To derive the set of parts P_s assigned to a station s, each station receives $\frac{P}{|S|}$ arbitrarily chosen distinct parts.

- *Storage space:* The storage space constraints at the stations $s \in S$ are generated by summing up the sizes of the cargo carriers G_p of assigned parts $p \in P_s$ times parameter B:

$$C_s = \sum_{p \in P_s} G_p \cdot B \quad \forall\, s \in S \tag{23}$$

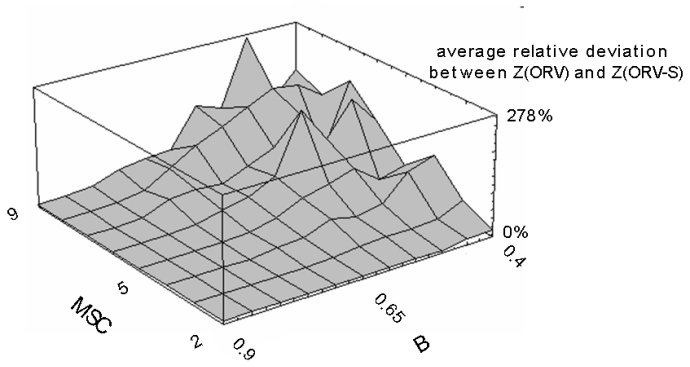

Figure 2: Relative deviation between ORV and ORV-S in dependency of restrictiveness of storage constraints of case A instances

Thus, parameter B is an indicator for the scarcity of storage space at the stations.

All generated instances can be downloaded from the internet (www.assembly-line-balancing.de).

5.2 Computational results

The methods described above have been implemented in Visual Basic.NET (Visual Studio 2003) and run on a Pentium IV, 1800 MHz PC, with 512 MB of memory. First, case A instances are evaluated to explore to what extent storage constraints (ORV-S) restrict smooth part consumption (ORV) and, vice versa, how often storage constraints (ORV-S) are violated by leveled ORV solutions. Therefore, case A instances are solved to optimality (i) by our DP-approach to the ORV-S and (ii) by the same DP-approach but with infinite storage spaces, which leads to the optimal ORV solution. The results in dependency of the restrictiveness of storage constraints, which is mainly influenced by parameter B and the amount of storage demand evoked by the size of the cargo carriers (parameter MSC), are depicted in Figure 2 and listed in Table 5.

Figure 2 relates the average relative deviation of both optimal solutions (measured by $\frac{Z(ORV-S)-Z(ORV)}{Z(ORV)}$) to the parameters used for instance generation MSC and B. On average, storage requirements grow with increasing MSC, as the average size of cargo carriers and, thus, the space required to store the parts contained in the carrier is increased. Parameter B is applied to calculate available storage capacity. Smaller values of B indicate less storage capacity at the stations and, hence, more restrictive capacity situations.

Figure 2 reveals that with increased restrictiveness deviations of ORV and ORV-S solutions increase considerably. In less restricted problems (smaller MSC and/or higher B), the optimal ORV solution often violates no storage constraint at all and is, hence, optimal for the ORV-S as well. With increasing restrictiveness (higher MSC and/or smaller B) optimal ORV and ORV-S solutions more and more deviate up to a maximum average relative deviation of 278% between ORV-S and ORV objective function values

	MSC								
B	2	3	4	5	6	7	8	9	avg.
0.4	9/58	45/83	119/88	58/67	135/100	202/63	131/100	164/89	82/76
0.45	7/52	17/72	97/94	42/73	205/100	42/43	173/100	102/83	65/75
0.5	0/0	14/70	36/88	87/67	124/91	153/75	182/89	278/86	84/64
0.55	0/0	13/69	24/95	68/78	211/90	86/77	153/100	115/85	74/70
0.6	0/0	10/41	38/80	51/68	67/94	57/78	119/95	138/94	54/64
0.65	0/0	4/23	12/62	21/54	35/96	32/80	90/86	45/89	28/58
0.7	0/0	0/0	8/44	6/46	23/64	35/61	52/77	41/83	19/45
0.75	0/0	0/0	0/0	2/19	6/44	31/54	15/39	31/67	10/27
0.8	0/0	0/0	0/0	0/0	2/19	3/30	5/37	8/54	2/17
0.85	0/0	0/0	0/0	0/0	0/0	0/0	0/7	1/19	0/3
0.9	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
avg.	1/9	8/30	24/45	24/53	53/53	39/45	56/54	62/60	31/40

legend: average relative deviation $\frac{Z(ORV-S)-Z(ORV)}{Z(ORV)}$ [percent]/fraction of infeasible ORV-S solutions by ORV [percent]

Table 5: Comparison of ORV and ORV-S solutions in dependency of restrictiveness of storage constraints of case A instances

and an infeasibility of optimal ORV solutions for ORV-S of 100% (see Table 5). Hence, to meet storage constraints in our problem setting, it gets the more important to apply the ORV-S approach for model sequencing, the more part units are contained in cargo carriers and the less space is available at the stations. On the other hand, very scarce storage space at the line results to model sequences which fail to reflect the JIT philosophy of leveled part usages.

In the following, the case B instances are evaluated to explore the performance of algorithms (DP and SA) in solving the ORV-S problem. Average results are summarized in Table 6. The computational test reveals that out of the 1296 instances generated only 1140 are feasible, so that the test bed is reduced accordingly. The SA is able to find feasible solutions for all of these instances except for two. Moreover, 479 instances are solved to optimality. Overall, the SA approach yields an average relative deviation of 12.1% (measured by: $\frac{Z(SA)-Z(DP)}{Z(DP)}$), which mainly stems from highly constrained instances (see Table 7). With regard to computation times SA clearly outperforms DP (see Table 6) and can, hence, be appropriately applied to larger problem instances of real-world size.

Finally, the performance of algorithms is evaluated according to the restrictiveness of the storage constraints. Table 7 lists the results for the SA-approach. With increasing restrictiveness (indicated by smaller values of parameter B and higher values of parameter MSC) average relative deviations of the SA-approach compared to optimal solutions increase considerably. This is explained by the fact that with increased restrictiveness, promising solutions, i.e. feasible solutions with near optimal objective function value, are more likely separated by regions of infeasibility, which cannot be bridged by the swap moves of the SA directly. In spite of the dynamic penalty values, the SA still

measure	DP	SA
number of feasible instances	1140	1140
number of feasible solutions found	1140	1138
number of optimal solutions	1140	479
average relative deviation from optimum in %	0	12.1
maximum relative deviation from optimum in %	0	994
average absolute deviation from optimum	0	3.0
maximum absolute deviation from optimum	0	544
average CPU-seconds	28.7	1.7

Table 6: Performance of algorithms aggregated over all case B-instances

	MSC			
B	3	5	7	avg.
0.5	13.2	27.5	53.7	27.4
0.7	3.0	14.7	12.0	9.7
0.9	3.8	3.6	3.9	3.8
avg.	6.5	13.6	17.4	21.1

Table 7: Average relative deviation from optimum in % by SA for case B-instances in dependency of the difficulty to find feasible solutions

considerably punishes moves from a feasible to an infeasible solution in the modified objective function value Z^{SA}. Thus, long trajectories of moves through infeasible regions receive low acceptance probabilities, so that the SA tends to get stuck in feasible but often unleveled solutions and the solution performance is low.

The opposite finding holds for the exact DP approach. Here, increasing difficulty to find feasible solutions considerably accelerates the determination of optimal solutions (see Table 8). Increasing difficulty results in an enlarged number of infeasible states, which can be excluded from the procedure. The graph contains less nodes and optimal solutions are found considerably faster.

	MSC			
B	3	5	7	avg.
0.5	30.3	19.8	14.5	21.5
0.7	33.1	32.0	29.2	31.4
0.9	33.1	33.2	33.1	33.2
avg.	32.2	28.3	25.6	28.7

Table 8: Average CPU-seconds required by DP for case B-instances in dependency of the difficulty to find feasible solutions

6 Discussion

The paper on hand presents solutions approaches which aim at leveled part usages in accordance with the JIT principle. At the same time, storage constraints are observed which can considerably restrict the smoothing of part usages induced by traditional level scheduling at the stations of the line. For the solution of this novel sequencing problem an exact Dynamic Programming approach and a heuristic Simulated Annealing procedure are presented.

The computational study shows that with increasing restrictiveness, storage constraints more and more enforce deviations from leveled part usages. Ultimately, highly constraint storage space leads to totally unleveled solutions. This observation is based on the fact that smoothed part usages (ORV) are directly opposed to a fast consumption of part inventories to meet storage constraints (ORV-S). This coherency questions the idea of level scheduling for assembly lines facing storage constraints. However, if parts are jointly delivered in cargo carriers in discrete time intervals, which turns out to be the majority of parts at least in the automobile industry (c.f. Boysen et al., 2007a), the whole idea of leveling part usages over the whole planning horizon seems questionable. The ORV seems especially adequate whenever material demands are directly pulled throughout the whole production system. This assumption is generally fulfilled if preceding production levels are located in immediate vicinity of the final assembly and are directly coupled via a Kanban system or feeder lines. If parts are delivered in discrete time intervals, not a leveling of all part usage seems relevant, but first and foremost a leveling of delivery quantities and intervals (c.f. Pleschberger and Hutomi, 1993; Aigbedo, 2004). Of course, a smooth part usage also allows the construction of more or less equal delivery quantities and intervals. Nevertheless, the objective of traditional level scheduling models can be far to restrictive. Only the sum of part consumptions between the deliveries within the planning horizon is actually to be leveled, whereas the succession of part usages within a delivery interval is irrelevant (at least if storage space is not scarce). With these additional degrees of freedom on hand, storage constraints can be met much easier without jeopardizing the JIT principle.

To derive a model for such a modified level scheduling supporting discrete part deliveries, the target demand rate r_p is to be modified as follows:

$$r'_p = \frac{\left\lceil \frac{\sum_{m \in M} d_m \cdot a_{pm}}{G_p} \right\rceil}{T} \tag{24}$$

Here, the total number of deliveries necessary per part is evenly spread over the planning horizon. With this modified target demand rate r'_p on hand, a level scheduling model for discrete part deliveries is obtained by substituting equations (5) with:

$$Z_{pt} = \left(y_{pt} - t \cdot r'_p\right)^2 \quad \forall p \in P;\, t = 1, \ldots, T \tag{25}$$

The resulting model evenly distributes the occurrence of part deliveries over time. Our solutions approaches presented could be easily modified to account for this delivery oriented level scheduling model. It remains up to future research to empirically test the

efficiency of such a new level scheduling model to ease JIT supply in real-world settings with discrete part deliveries and scarce storage space.

References

[1] Aarts, E.H.L., Korst, J.H.M., van Laarhoven, J.M., 1997. Simulated Annealing, In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local search in combinatorial optimization, Chichester et al., 91–120.

[2] Aigbedo, H., 2004. Analysis of parts requirements variance for a JIT supply chain, International Journal of Production Research 42, 417–430.

[3] Bautista, J., Companys, R., Corominas, A., 1996. Heuristics and exact algorithms for solving the Monden problem, European Journal of Operational Research 88, 101–113.

[4] Boysen, N., Fliedner, M., Scholl, A., (2006): Level-Scheduling bei Variantenfließfertigung: Literaturüberblick, Klassifikation und Modellkritik, Journal für Betriebswirtschaft (to appear).

[5] Boysen, N., Fliedner, M., Scholl, A., 2007a. Sequencing mixed-model assembly lines: Survey, Classification and Model Critique. Jena Research Papers in Business and Economics (JBE) 02/2007, FSU Jena.

[6] Boysen, N., Fliedner, M., Scholl, A., 2007b. Sequencing mixed-model assembly lines to minimize part inventory cost. Jena Research Papers in Business and Economics (JBE) 03/2007, FSU Jena.

[7] Dhamala, T.N., Kubiak, W., 2005. A brief survey of just-in-time sequencing for mixed-model systems, International Journal of Operations Research 2, 38–47.

[8] Duplaga, E.A., Hahn, C.K., Hur, D., 1996. Mixed-model assembly line sequencing at Hyundai Motor Company, Production and Inventory Management Journal 37, 20–26.

[9] Joo, S.-H., Wilhelm, W.E., 1993. A review of quantitative approaches in just-in-time manufacturing, Production Planning & Control 4, 207–222.

[10] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing, Science 220, 671–680.

[11] Klampfl, E., Gusikhin, O., Rossi, G., 2006. Optimization of workcell layouts in a mixed-model assebly line environment, International Journal of Flexible Manufacturing Systems 17, 277–299.

[12] Kubiak, W., 1993. Minimizing variation of production rates in just-in-time systems: A survey, European Journal of Production Research 66, 259–271.

[13] Mane, A., Nahavandi, S., Zhang, J., 2002. Sequencing production on an assembly line using goal chasing and user defined algorithm, In: Yücesan, E., Chen, C.-H., Snowdom, J.L., Charnes, J.M. (eds.) Proceedings of the 2002 Winter Simulation Conference (on cd-rom).

[14] Meyr, H., 2004. Supply chain planning in the German automotive industry, OR Spectrum 26, 447–470.

[15] Monden, Y., 1998. Toyota Production System: An integrated approach to just-in-time, 3rd edition, Norcross.

[16] Pleschberger, T.E., Hutomi, K., 1993. Flexible final-assembly sequencing method for a JIT manufacturing environment, International Journal of Production Research 31, 1189–1199.

[17] Solnon, C., Cung, V.D., Nguyen, A., Artigues, C., 2006. The car sequencing problem: Overview of the state-of-the art methods and industrial case-study of the ROADEF'2005 challenge problem, European Journal of Operational Research (to appear).

[18] Zhu, J., Ding, F.-Y., 2000. A transformed two-stage method for reducing the part-usage variation and a comparison of the product-level and part-level solutions in sequencing mixed-model assembly lines, European Journal of Operational Research 127, 203–216.