Jena Research Papers in Business and Economics

Level Scheduling for batched JIT supply

Nils Boysen, Malte Fliedner, A. Scholl

01/2009

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 for batched JIT supply

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

[a] *Friedrich-Schiller-Universität Jena, Lehrstuhl für Operations Management, Carl-Zeiß-Straße 3, D-07743 Jena, Germany,* {nils.boysen,malte.fliedner}@uni-jena.de

[*] *Corresponding author, phone +49 3641 943100.*

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

Abstract
A mixed-model assembly line requires the solution of a short-term sequencing problem which decides on the succession of different models launched down the line. A famous solution approach stemming from the Toyota Production System is the so called Level Scheduling, which aims at distributing the part consumption induced by the model sequence evenly over time. Traditional Level Scheduling seeks to closely approximate target demand rates at every production cycle, however, such a strict leveling is only required if parts are directly pulled from a connected feeder line. In real-world assembly lines, parts are predominately delivered in (small) batches at certain points in time. In such a situation, a Just-in-Time supply is already facilitated whenever the cumulative consumption is leveled in accordance with each part's delivery schedule, while the exact consumption pattern between two delivery points seems irrelevant. The paper on hand provides new Level Scheduling models, proves complexity, presents exact and heuristic solution procedures and shows inferiority of traditional Level Scheduling for such a batched JIT-supply of parts.

Keywords: Mixed-model assembly line; Just-in-Time; Sequencing; Level Scheduling; Batched part supply

1 Introduction

In a mixed-model assembly line different models of a common base product are assembled in intermixed production sequences (lot size one). Although (almost) any succession of models is technically possible, the chosen product sequence very well influences different economic values. In addition to work overloads induced by the direct succession of work intensive models at a station (see Tsai, 1995), from a supply view, demand peaks for parts need to be avoided. Peaks

in part consumption resulting from an "unleveled" model sequence hinder a cost efficient JIT-supply, because enlarged safety stocks are required to ensure a reliable part supply. Consequently, the so-called "Level Scheduling" was developed as part of the famous Toyota Production System to evenly spread the part consumption over the planning horizon (see Monden, 1998). A detailed discussion of Level Scheduling and alternative sequencing procedures is provided in a latest review paper by Boysen et al. (2009).

The traditional Level Scheduling, which was initially formalized by Monden (1998), aims at a leveling of each part's consumption pattern. Kubiak (1993) refers to this case of Level Scheduling as Output Rate Variation (ORV) problem, because materials constitute the outputs of preceding production levels, whose actual demand rates are to be leveled. Within the ORV problem each part type receives a target demand rate, which is determined by distributing the material's overall demand evenly over the planning horizon. Then, a sequence is sought where actual demand rates for parts are as close as possible to the ideal target rates in every production cycle. Figure 1 gives a schematic representation of the basic idea of Level Scheduling for a single part.

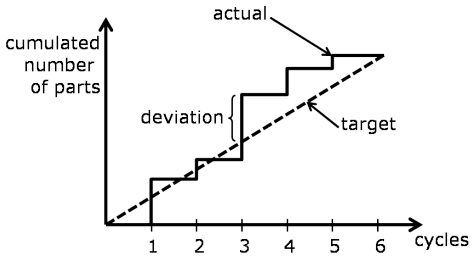

Figure 1: Schematic representation of Level Scheduling for a single part

As the ORV aims at a steady demand rate in every production cycle, it seems especially adequate whenever material demands are directly pulled throughout the whole production system (Boysen et al., 2009). This assumption is generally fulfilled if preceding production levels are located in immediate vicinity of the final assembly and are, e.g., directly coupled via feeder lines. Today's trend of reducing vertical integration, however, leads to a dramatic decrease in the number of parts produced in-house.

In automotive industry, final assembly lines are typically supplied with parts utilizing the following delivery paths (percentage ranges given represent the situation at different final assembly lines of a major German car manufacturer):

- Especially, bulky parts are delivered directly to the respective line segment in large load carriers, often by truck via a dock door near to the assembly area (about 10 to 30%).
- Smaller parts are stacked in small load carriers of given capacity which are delivered by forklifts or automated guided vehicles from the central store (about 30 to 60%).
- An increasing number of parts (15 to 30%) are delivered from decentralized "supermarkets" where parts are intermediately stored and commissioned in smaller carriers. From the

supermarket, tugger vehicles supply parts typically on the basis of a fixed delivery schedule (with cycles of 30 minutes to two hours).

- Some car specific parts (especially smaller ones) are pre-packaged in special boxes which accompany their respective car model directly on the conveyor belt (about 10 to 15%).
- Only several parts are delivered in a continuous manner to the line, e.g., drive chains, which are pre-assembled in-house (in a factory-in-factory setting) directly connected with the final assembly via a feeder line (see Boysen et al., 2008).

This overview of delivery paths reveals that more than 80% of parts (first three paths) are delivered batch-wise, whereas the fraction of parts delivered in a (quasi-)continuous manner is much smaller (last two paths). Thus, a modification of Level Scheduling for batched deliveries seems essential to better support real-world part supply.

Note that even if parts are supplied just-in-sequence (JIS), delivery usually takes place in a batched manner using some type of JIS-trolley. This situation simplifies the leveling problem because it is, from the logistics point of view, not anymore necessary to differentiate between several versions of the same part, because the required part versions are arranged at the JIS-trolley as defined by the final model sequence. Instead, product models are only subdivided into those having the option that requires a version of the part and the others which do not have the option and, thus, do not need the part at all.

Under batched deliveries, a leveling of part consumption is still a valuable objective for model sequencing, because demand peaks are avoided and, thus, emergency deliveries and safety stocks are reduced. However, as the production stages are more loosely coupled, an adjustment towards an ideal production rate in all cycles seems of minor relevance. Instead, the aggregated part consumption, cumulated over all production cycles between two delivery points, needs to be leveled over the whole delivery schedule (all delivery points), whereas the detailed demand pattern between two delivery points is irrelevant from a JIT point of view.

A leveling of cumulated part consumption over the delivery points can be executed in two different ways depending on the delivery policy applied. First, if a *fixed-schedule-policy* is applied, delivery schedules are determined upfront, so that fixed delivery points are specified for a part, whereas the delivered part quantity is adjusted according to the actual demand induced by the model sequence up to the next delivery point. In this case, a leveling of cumulated part consumption over all delivery points should be aimed at. Alternatively, if part supply is organized according to a *fixed-quantity-policy*, the delivered quantity per part is fixed, e.g., by a fixed carrier capacity, and a replenishment is initiated whenever the inventory near the line is exhausted and the respective part is needed again. By sequencing, part demands are to be adjusted, so that batched deliveries per part are evenly distributed over the planning horizon such that a regular delivery schedule can be derived. It is the aim of the paper to adopt the idea of traditional Level Scheduling for these two JIT-replenishment policies typical for real-world mixed-model assembly systems.

M	set of models/products (index m)
T	number of production cycles (index t)
P	set of parts/materials (index p)
D_p	set of fixed delivery points for part p (index τ)
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
q_p	fixed delivery quantity of part p
s_p	initial inventory of part p at the beginning of the planning horizon
y_{pt}	binary variable: 1, if part p is delivered in cycle t; 0, otherwise
x_{mt}	binary variable: 1, if model m is scheduled at cycle t; 0, otherwise

Table 1: Notation

The remainder of this paper is organized as follows. Section 2 briefly summarizes existing Level Scheduling literature. In Section 3, two novel Level Scheduling models adopted to the aforementioned replenishment situations are described in detail. Sections 3.1 and 3.2 consider the fixed-schedule-policy, where the delivery quantities for a given schedule are to be leveled, and the fixed-quantity-policy, where intervals between part deliveries are to be leveled for given delivery quantities, respectively. Section 4 presents exact (Dynamic Programming) and heuristic (Simulated Annealing) solution procedures. A comprehensive computational study in Section 5 investigates algorithmic performance of the solution procedures (Section 5.2) and shows that traditional Level Scheduling not sufficiently supports batched JIT-supply (Section 5.3). Finally, Section 6 concludes the paper.

2 Literature review

The traditional ORV problem was first formalized by Monden (1998), who also introduced the famous "Goal Chasing Methods" for solving ORV. Further prominent contributions stem from Kubiak (1993), Bautista et al. (1996), Kubiak et al. (1997) as well as Zhu and Ding (2000). Detailed reviews are provided by Kubiak (1993), Dhamala and Kubiak (2005) as well as Boysen et al. (2009). With the notation of Table 1 the ORV problem can be formalized as follows.

Consider a set M of product models each of which having a demand d_m for copies of this model m to be produced during a specific period (e.g. one day or shift) 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 needed in the assembly of one unit of product m. The matrix of coefficients $A = (a_{pm})$ is called "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 rate r_p per production cycle is calculated as follows:

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

Together with the binary variables x_{mt}, which indicate whether product m is produced in cycle t ($x_{mt} = 1$) or not ($x_{mt} = 0$), and the notation of Table 1 the ORV problem can be modeled as follows (Joo and Wilhelm, 1993; Monden, 1998; Bautista et al., 1996):

$$(ORV)\ \text{Minimize}\ Z = \sum_{p \in P} \sum_{t=1}^{T} \left(\sum_{m \in M} \sum_{\tau=1}^{t} x_{m\tau} \cdot a_{pm} - t \cdot r_p \right)^2 \tag{2}$$

subject to

$$\sum_{t=1}^{T} x_{mt} = d_m \qquad \forall\, m \in M \tag{3}$$

$$x_{mt} \in \{0, 1\} \qquad \forall\, m \in M;\, t = 1, \ldots, T \tag{4}$$

Objective function (2) considers deviations of actual from ideal cumulative demands per production cycle t and part p. These deviations are weighted by a penalty function $(\cdot)^2$ to quantify the respective distances. The separate deviations for all t and m are aggregated to a global objective value by an aggregation function, which is to be minimized. See Boysen et al. (2009) for a detailed description of this and other penalty and aggregation functions considered in the literature. We restrict our investigation to the "sum of squared"-deviations case, although models and algorithms can be easily adopted to alternative formulations. Constraints (3) enforce the products to be produced in the demanded quantities and (4) define binary variables x_{mt}. Additional constraints might be required, if the storage space at the stations of the line is scarce. This extension is investigated by Boysen et al. (2007a).

Simplified Level Scheduling approaches (labeled "Product Rate Variation Problem" (PRV)), which only level the different product models over time are discussed, e.g., by Miltenburg (1989), Kubiak and Sethi (1991), Steiner and Yeomans (1993) as well as Inman and Bulfin (1991). Recently, Boysen et al. (2007b) showed that these simplified PRV procedures are not sufficient to cope with the high product variety of modern mixed-model assembly lines.

Some alterations of Level Scheduling required whenever parts are not steadily pulled from preceding production stages via feeder lines but delivered in batches are already discussed by Aigbedo (2004). However, Aigbedo investigates the fixed-schedule-policy from a different point of view. He does not introduce an adopted sequencing approach but three measures (so-called "variance metrics"), which were developed to asses the suitability of given sequences dor batched JIT-supply. He generates sequences by a two-stage ORV heuristic and concludes that ORV sequences are also able to level the part consumption over given delivery points. This conclusion is not astounding as a sequence which levels part consumption over the complete planning horizon should also lead to a (more or less) level consumption over a subset of cycles, i.e., the given set of

delivery points. However, he does not compare the ORV solutions with optimal solutions gained for the actual (modified Level Scheduling) problem with batched delivery.

Such a comparison is part of our computational study and our results show that ORV solutions are much less suited than procedures which directly treat the actual problem. Moreover, if part consumption is leveled in every cycle (ORV) although a leveling is merely required at a few delivery points, the degrees of freedom for model sequencing are unnecessarily reduced. This property is especially undesired whenever a multi-objective scheduling approach, e.g., additionally considering work overloads, is applied like it is often done in real-world sequencing procedures (see Boysen et al., 2009). In this case, the number of "good" sequences according to the leveling objective is considerably reduced, because demand rates need to be leveled in each single cycle. Consequently, solutions also fulfilling the aims of the alternative sequencing objectives, i.e., as a compromise solution, become more rare in the solution space. Thus, it seems recommendable to introduce a novel Level Scheduling approach, which levels demand exclusively over those points in time a balancing is actually aimed at and keeps the degrees of freedom at the desired level. Moreover, we also investigate the fixed-quantity-policy, which is not yet covered in literature in spite of its practical importance.

3 Level Scheduling models for batched part deliveries

In this section two novel Level Scheduling problems are described in detail. First, we introduce the approach designated to a fixed-schedule-policy for part replenishment. Then, Section 3.2 deals with the alternative approach for a fixed-quantity-policy.

3.1 Fixed-schedule-policy

A fixed-schedule-policy assumes a delivery schedule is determined upfront, so that cdelivery points for each part are fixed, e.g., some part might be delivered every two hours, another twice a shift. These delivery points are represented by the sets D_p containing the production cycles at which deliveries are scheduled per part p. In order to get a manageable overall delivery schedule, delivery points are often arranged equidistantly but might also be given in a less regular manner. The delivered quantities of parts are variable and amount to the cumulated consumption from the respective delivery point up to the next (or the last production cycle, if no succeeding delivery point exists). Consequently, the cumulated part consumption of each part p is to be leveled over all given delivery points D_p, so that our modified Level Scheduling can be formalized as follows:

$$(LS^S) \text{ Minimize } Z = \sum_{p \in P} \sum_{\tau \in D_p} \left(\sum_{m \in M} \sum_{t=1}^{\tau} x_{mt} \cdot a_{pm} - \tau \cdot r_p \right)^2 \tag{5}$$

$$\text{subject to (3)–(4)}$$

Model LS^S represents a Level Scheduling where deviations between actual and ideal part

consumption are merely measured at the given delivery points D_p. This modified version of Level Scheduling is an NP-hard optimization problem, as can be easily shown on the basis of Kubiak's (1993) NP-hardness proof for the min-sum ORV. As any ORV-instance corresponds to an instance of LS^S with a part delivery in each cycle, i.e. $|D_p| = T \,\forall\, p \in P$, the reduction employed in Kubiak (1993) is equally valid for LS^S.

3.2 Fixed-quantity-policy

In a fixed-quantity-policy, the delivery quantity q_p of each part $p \in P$ is given by the capacity of the container used for this part. Then, we aim at distributing the sequence dependent delivery points evenly over time, such that a regular delivery scheme can be realized. Consequently, a modified target rate r'_p is to be defined, which becomes a so-called *target delivery rate* by dividing the overall number of deliveries by the number of cycles T. To compute the number of required deliveries, the total demand $\sum_{m \in M} d_m \cdot a_{pm}$ for part p is reduced by the initial inventory s_p and divided by the delivery quantity q_p and rounded up (this might result in an oversupply which constitutes the initial inventory for the next shift or day, respectively):

$$r'_p = \frac{\left\lceil \frac{\sum_{m \in M} d_m \cdot a_{pm} - s_p}{q_p} \right\rceil}{T} \tag{6}$$

Example: Consider a shift consisting of 400 production cycles, where 175 sunroofs ($p = 1$) are to be installed at car models. If these sunroofs are delivered in fixed quantities of $q_1 = 50$ and the initial inventory s_1 is not greater than 24, then four deliveries are required and the target rate amounts to $r'_1 = \frac{1}{100}$. If $s_1 \in [25, 74]$, then three deliveries are sufficient and the target rate amounts to $r'_1 = \frac{3}{400}$.

With the help of this adopted target rate and additional variables y_{pt} and l_{pt} defining whether a delivery of part p occurs in cycle t ($y_{pt} = 1$) or not ($y_{pt} = 0$) and the inventory per part p in cycle t, respectively, the model LS^Q can be formalized as follows:

$$(LS^Q) \text{ Minimize } Z = \sum_{p \in P} \sum_{t=1}^{T} \left(\sum_{\tau=1}^{t} y_{pt} - t \cdot r'_p \right)^2 \tag{7}$$

subject to (3)–(4) and

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

$$0 \leq l_{pt} \leq q_p \qquad \forall\, p \in P;\, t = 1, \ldots, T \tag{9}$$

$$y_{pt} \leq \sum_{m \in M} x_{mt} \cdot a_{pm} \qquad \forall\, p \in P;\, t = 1, \ldots, T \tag{10}$$

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

In objective function (7) deviations per part p and cycle t are defined as the squared difference between the actual and intended number of deliveries up to t. The balance equations (8) define the inventory l_{pt} of part p at (the end of) cycle t by summing up initial inventory s_p and cumulated delivery up to t and subtracting cumulated part demand including t. Constraints (9) enforce that deliveries are executed not before a part's inventory is exhausted. Inequalities (10) assure that deliveries are executed just-in-time, i.e., only in cycles t where the respective part p is actually required for assembly. Together with constraints (9) this way it is assured that a delivery is executed exactly in that cycle the inventory level is zero and the part is required again for assembly. Finally, constraints (11) define binary variables y_{pt}.

Remarks:

- As expected, the problem LS^Q is also NP-hard as is proven in the appendix.
- Constraints (9) represent a *one-bin-policy* usually applied if space is very scarce, i.e., only (at most) one bin with a part p is located near the line at a time. Another typical setting is the *two-bin-policy* which means that always a second full container is available and exchanged with the first whenever the latter has been emptied. At the next delivery point, the empty one is exchanged by a new full one. The two-bin policy can also be handled in the model by changing (9) to $0 \leq l_{pt} \leq 2 \cdot q_p$ for all p and t.
- The JIT-constraints (10) claim that no delivery should be made before a part is needed again. In order to ensure that no shortage occurs, actual delivery will usually consider uncertain lead times in practice.
- Both models could be easily extended by additionally considering space restrictions at the stations as proposed by Boysen et al. (2007a).

4 Solution procedures

In this section, exact and heuristic algorithms for our novel Level Scheduling problems are presented. In Section 4.1, we present a Dynamic Programming (DP) approach, which is an adoption of the procedure of Bautista et al. (1996) for the ORV problem. Then, Section 4.2 presents a simple heuristic simulated annealing approach, which can be applied to larger instances of real-world size.

4.1 A Dynamic Programming procedure

The DP approach is based on an acyclic digraph $G = (V, E, c)$ 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 $c : V \rightarrow \mathbb{R}$. Each sequence position t is represented by a stage which contains a subset $V_t \subset V$

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

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

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

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 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 $p \in P$ of the respective partial schedule:

$$D_{tip} = \sum_{m \in M} X_{tim} \cdot a_{pm} \qquad \forall\, p \in P;\ t = 0, \ldots, T;\ i \in V_t \tag{15}$$

Then, to each node corresponding to a state (t, i) a unique node weight c_{ti} is assigned, which depends on the Level Scheduling approach to be solved. For the LS^S problem with given delivery schedules D_p the sum of squared deviations of the actual cumulated demand D_{tip} from the part-specific target rate are to be added whenever a part delivery is scheduled in production cycle $t \in D_p$. Consequently, node weight c^S_{ti} for a fixed-schedule-policy and model LS^S amounts to:

$$c^S_{ti} = \sum_{p \in P} \begin{cases} (D_{tip} - t \cdot r_p)^2, & \text{if } t \in D_p \\ 0, & \text{otherwise} \end{cases} \qquad \forall\, t = 0, \ldots, T;\ i \in V_t \tag{16}$$

Alternatively, if a fixed-quantity-policy is applied and model LS^Q is to be solved, node weight c^Q_{ti} is calculated as follows:

$$c_{ti}^{Q} = \sum_{p \in P} \left(\left\lceil \frac{\max\{0;\, D_{tip} - s_p\}}{q_p} \right\rceil - t \cdot r'_p \right)^2 \qquad \forall\, t = 0, \ldots, T;\; i \in V_t \tag{17}$$

Here, for each part p the actual number of deliveries to satisfy cumulated demand D_{tip} of the respective node (minus initial stock s_p) is to be calculated, so that the squared deviation from the intended number of deliveries can be computed.

With this graph on hand, determining optimal solution 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 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.

4.2 A Simulated Annealing procedure

As both problems were shown to be NP-hard the aforementioned exact DP procedure will not be able to solve instances of real-world size. Thus, efficient heuristic procedures are required. We now present a straight forward Simulated Annealing (SA) procedure for this setting.

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). 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., 2008). 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 $Z(\pi)$ depends on the Level Scheduling problem to be solved. For model LS^S with a given delivery schedule, objective value $Z^S(\pi)$ can be determined as follows, where π_t represents the model scheduled at position t of sequence π:

$$Z^S(\pi) = \sum_{p \in P} \sum_{\tau \in D_p} \left(\sum_{t=1}^{\tau} a_{p\pi_t} - \tau \cdot r_p \right)^2 \tag{18}$$

For the alternative problem LS^Q and a fixed-quantity-policy objective value $Z^Q(\pi)$ is calcu-

lated as follows:

$$Z^Q(\pi) = \sum_{p \in P} \sum_{t=1}^{T} \left(\left\lceil \frac{\max\{0; \sum_{\tau=1}^{t} a_{p\pi_\tau} - s_p\}}{q_p} \right\rceil - t \cdot r'_p \right)^2 \tag{19}$$

With these objective values on hand 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):

$$Prob(\pi' \text{ replacing } \pi) = \begin{cases} 1, & \text{if } Z^{\cdot}(\pi') \leq Z^{\cdot}(\pi) \\ exp\left(\frac{Z^{\cdot}(\pi) - Z^{\cdot}(\pi')}{C}\right), & \text{otherwise} \end{cases} \tag{20}$$

Our SA is guided by a simple static cooling schedule (see Kirkpatrick et al., 1983). The initial value for control parameter $C = Z^{\cdot}(\pi^*) \cdot 10$ is chosen, where π^* is a randomly drawn model sequence. Control parameter C is continuously decreased in the course of the procedure by multiplying it with factor 0.9995 in each iteration. A total of 50,000 potential moves to neighboring solutions are evaluated by our SA approach and a solution with minimum objective function value $Z^{\cdot}(\pi)$ is returned. Within our computational study, we only report results for the values of the control parameters described above, as preliminary studies indicated that these parameter values deliver a reasonable compromise between runtime and solution quality.

Remark: Note that the solution value of a neighboring solution remains unchanged if the swapped positions are not separated by at least one part's delivery point. This fact is considered when generating neighboring solutions and might considerably reduce the size of the neighborhood in each iteration. As a consequence, search will be accelerated compared to ORV. This will be particularly useful for developing specialized exact and heuristic solution procedures. For example, within a branch & bound or bounded DP algorithm dominance relationships can be used to reduce the number of nodes to be examined. This potential reduction of solution effort comes along with enlarged degrees of freedom and relevance for practice as argued at the end of Section 2.

5 Computational study

As no established test-bed is available for a computational study, we first elaborate on the instance generation. Then, algorithmic performance of our solution procedures is investigated. Finally, we compare the results of our novel approaches with traditional Level Scheduling.

5.1 Instance generation

In our computational study, we distinguish between two classes of test instances: case A instances, which are small enough to be solved to optimality, and case B instances, which represent instances

of real-world size and, thus, need to be solved heuristically. To generate instances of our modified Level Scheduling models LS^S and LS^Q, respectively, the input parameters listed in Table 2 are used to produce the demand coefficients for parts a_{pm}, model demands d_m, delivery schedules D_p, and delivery quantities q_p.

		values	
symbol	description	case A	case B
T	number of production cycles	10, 15, 20	50, 100, 150
$\|M\|$	number of models	5, 7, 9	5, 10, 15
$\|P\|$	number of parts	4, 6, 8	
$PROB$	probability of a model m containing part p	0.3, 0.5, 0.7	
s_p	initial inventory of part p	0	
N	number of deliveries per part	2, 3	2, 3, 4, 5

Table 2: Parameters for instance generation

Within each test case, the parameters are combined in a full-factorial design, so that 324 (648) 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 uniform $[0, 1]$-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:

$$a_{pm} = \begin{cases} 1, \text{ if } rnd \leq PROB \\ 0, \text{ otherwise} \end{cases} \quad \forall\, m \in M;\, p \in P \tag{21}$$

- *Demand for models:* At first, each model demand d_m is initialized to one unit. Then, demands of randomly drawn (uniformly 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.

- *Delivery schedules:* For an instance of LS^S the sets D_p of delivery points per part p are determined by dividing the number of cycles T by the number of deliveries N: $t^d = \lfloor \frac{T}{N} \rfloor$. Then, approximately any t^d cycles a delivery point is added as follows: $D_p = \{n \cdot t^d + rnd : n = 1, \ldots, N-1\}\, \forall\, p \in P$, where rnd is an equally distributed random integer drawn from the interval $[-1, +1]$. Additionally, the first production cycle is chosen as a delivery point for any part due to having set the initial inventory to zero.

- The *delivery quantities* required for an LS^Q-instance are generated by dividing the overall part demand by the number of deliveries N: $q_p = \left\lceil \frac{\sum_{m \in M} a_{pm} \cdot d_m}{N} \right\rceil \; \forall\, p \in P$.

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

		LS^S (fixed delivery schedule)					LS^Q (fixed delivery quantity)				
		DP	SA				DP	SA			
T	$\|M\|$	cpu [sec]	avg gap [%]	max gap [%]	#opt	cpu [sec]	cpu [sec]	avg gap [%]	max gap [%]	#opt	cpu [sec]
10	5	<0.1	0.0	0.0	18	0.177	<0.1	0.0	0.0	18	0.196
	7	<0.1	0.0	0.0	18	0.179	<0.1	0.0	0.0	18	0.198
	9	<0.1	0.0	0.0	18	0.180	<0.1	0.3	5.4	17	0.200
15	5	<0.1	0.0	0.0	18	0.182	<0.1	0.8	12.1	15	0.208
	7	0.1	0.0	0.0	18	0.185	0.1	0.8	7.7	15	0.209
	9	0.4	0.5	8.7	17	0.184	0.5	0.3	2.4	15	0.211
20	5	0.1	0.0	0.0	18	0.194	0.1	0.4	3.0	14	0.218
	7	1.0	0.0	0.0	18	0.194	1.1	1.3	10.3	14	0.221
	9	7.2	1.0	10.3	16	0.194	7.3	1.4	8.1	12	0.223
	total	1.0	0.2	10.3	159	0.185	1.0	0.6	12.1	138	0.209

Table 3: Algorithmic performance: Results for case A instances

5.2 Algorithmic performance

The methods described above have been implemented in C#.NET (Visual Studio 2008) and run on a 2.1 GHz PC, with 2 GB of memory. First, case A instance are evaluated to investigate the algorithmic performance of exact DP and heuristic SA. Table 3 summarizes the results in dependency of the number of cycles T and number of model $|M|$, where "avg gap" and "max gap" denote the average and maximum deviation of our SA procedure from optimum (DP) with each single deviation measured by: $\frac{Z(SA)-Z(DP)}{Z(DP)} \cdot 100\%$. Additionally, "#opt" counts the number of instances solved to optimality by SA. The solution time is reported by "cpu", which denotes the CPU-seconds averaged over all 18 instances per parameter constellation.

The exact solution procedure DP solves all 324 instances to optimality without notable difference (with regard to the runtime) whether problem LS^S or LS^Q is solved. However, the results indicate an exponential increase of runtime with increasing number of cycles T and number of models $|M|$. This raises the question for an upper limit of these parameters up to which DP can still be reasonably applied. In an additional experiment which increases the number of cycles stepwise (and leaves any other parameter unchanged) these limits were determined as follows: with $|M| = 5$ models all instances with $T = 53$ could still solved to optimality within a given time frame of 300 CPU seconds (cpu=241). For $|M| = 7$ and $|M| = 9$ this limit of 300 CPU seconds is reached at $T = 31$ (cpu=234) and $T = 23$ (cpu=279) cycles, respectively.

Furthermore the question for the solution performance of SA is to be answered. With regard to problem LS^S and a fixed delivery schedule the performance of SA seems very promising. SA solves 98% of the 162 case A instances for LS^S to optimality with an average gap of merely 0.2%. Solely a little worse SA performs when solving LS^Q instances with fixed order sizes. Here, 85% of 162 instances are solved to optimality with an average gap of merely 0.6%. Moreover, with this size of test instances solution times are negligible with less than 1 CPU second solution time

for any test instance.

Of course, algorithmic performance could be further improved by applying more sophisticated procedures. However, as our main intention is to adopt Level Scheduling to discrete deliveries and to question the performance of traditional Level Scheduling in these modified settings, the solution performance of DP and SA seem satisfactorily enough, so that they can be substantially applied within the next section to investigate our main research question.

5.3 Comparison with traditional Level Scheduling

To evaluate traditional Level Scheduling (ORV) within our modified setting with batched part deliveries, we extract the ORV specific part of our test instances and solve these instances with an ORV procedure. As an exact procedure basing on Bounded Dynamic Programming the approach by Bautista et al. (1996) is utilized. Additionally, we apply the aforementioned SA procedure with the only adoption of evaluating each solution vector with the traditional ORV objective function instead of our modified ones to get comparable conditions. The resulting model sequences of these ORV approaches are then evaluated with the modified objective functions of LS^S and LS^Q by utilizing formulas (18) and (19) seen as the more realistic measures in case of batched deliveries. These solutions gained by traditional Level Scheduling can then be compared to our procedures DP and SA. This way, the question can be investigated whether or not traditional Level Scheduling is also suited for batched deliveries.

First, Table 4 lists the aggregated results for case A instances. These instances are small enough to be solved with the exact DP procedures, so that solution performance of the ORV approaches can be evaluated against optimal solutions of the problems LS^S and LS^Q. In addition to the performance measures mentioned above we report the absolute deviation averaged over all 162 instances per problem (labeled "avg abs") and the maximum absolute deviation (labeled "max abs"). These measures in relation to optimal solutions are listed for both problems LS^S (left columns) and LS^Q (right columns). For each problem, the solutions gained by the exact ORV procedure (labeled DP(ORV)) and heuristic ORV procedure (labeled SA(ORV)) are evaluated.

	LS^S (fixed delivery schedule)		LS^Q (fixed delivery quantity)	
measure	DP(ORV)	SA(ORV)	DP(ORV)	SA(ORV)
avg gap [%]	20	24	24	23
max gap [%]	110	90	97	66
avg abs	0.3	0.4	2.1	2.2
max abs	2.7	2	10.5	9.2
# opt	59	47	28	23

Table 4: Comparison with traditional Level Scheduling: Results for case A instances

The results reveal that traditional Level Scheduling produces considerable deviations when applied to the modified, more realistic problems. With an average gap of 20% and more the ORV models and procedures show considerably inferior to those directly treating batched deliveries.

Interestingly, for problem LS^Q, the heuristic ORV procedure SA(ORV) produces even better results than its exact counterpart DP(ORV), which highlights the limited suitability of ORV procedures for batched part deliveries. This is further underlined by considering number of instances where the optimal solutions of ORV is also optimal for LS^S or LS^Q. In case of LS^S, this is achieved for 59 out 162 instances, while the situation is even worse for LS^Q with only 28 out of 162 instances. This shows that the restrictiveness of ORV (considering irrelevant deviations in non-delivery cycles) reduces solution quality (only depending on deviations in delivery cycles).

Analogously, Table 5 summarizes the aggregated results for case B instances. Here, optimal solutions can not be gained, so that merely the heuristic SA procedures can be compared. The performance measures report relative and absolute deviations of the heuristic ORV procedure (SA(ORV)) from heuristic SA procedures solving the problems LS^S (left column) and LS^Q (right column). The figures show consistent results compared to case A instances (see Table 4) as the deviations lie in a comparable range with an average gap of more than 20%.

measure	SA(LS^S) vs. SA(ORV)	SA(LS^Q) vs. SA(ORV)
avg gap [%]	25	21
max gap [%]	188	88
avg abs	0.5	12
max abs	4	76

Table 5: Comparison with traditional Level Scheduling: Results for case B instances

Additional conclusions can be drawn from more detailed results. Figure 2 displays the average gap of the heuristic ORV procedure (SA(ORV)) from the heuristic SA procedures (SA(LS^S); solid line) and SA(LS^Q); dashed line) in dependency of the parameters of instance generation.

Whereas parameter $PROB$, which specifies the probability of a model containing a specific part, seems to have no special influence, from the other four parameters the following conclusions can be drawn:

- With increasing size of the instances the disadvantage of traditional Level Scheduling increases. This coherency is indicated by an increase of the average gap with the number of cycles T, models $|M|$ and parts $|P|$. This seems plausible as with increasing instance size more parts need to be coordinated which accumulate deviations along an increasing number of delivery points.

- The more frequently parts are supplied (increasing N), the lower amounts the average gap for traditional Level Scheduling. This coherency is in line with intuition, because with an increasing number of deliveries our modified Level Scheduling for discrete deliveries converges to traditional Level Scheduling. In the last, when deliveries occur in every production cycle both problem types become identical and no gap occurs.

- There exist no significant differences with regard to these two findings between the problems LS^S and LS^Q and the delivery policies they represent.

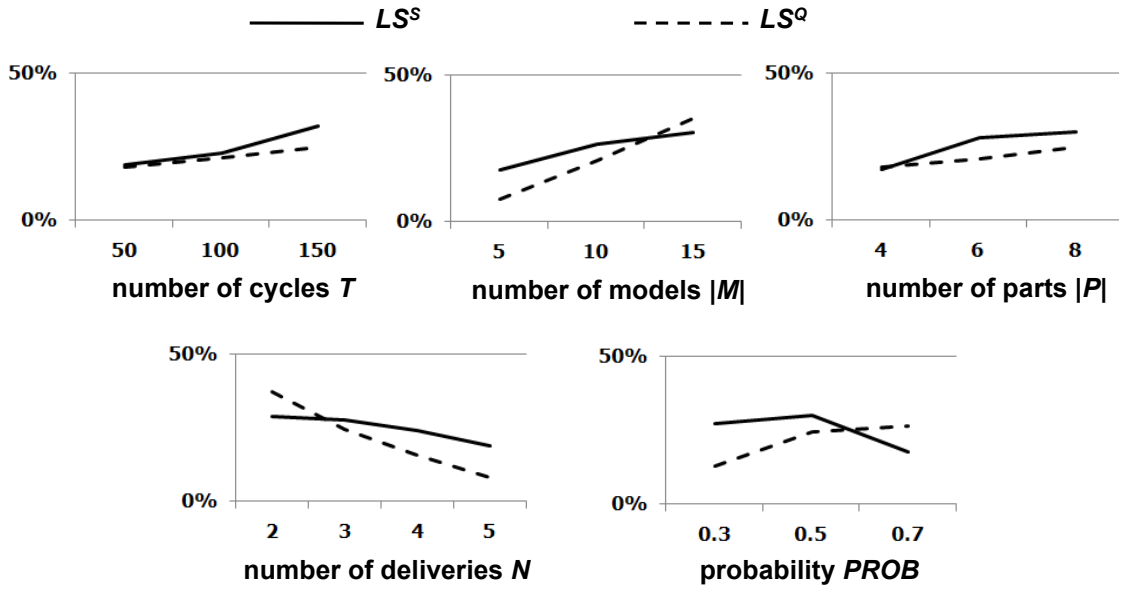

Figure 2: Average gap of traditional Level Scheduling per parameter of instance generation (case B instances)

Thus, it can be stated that traditional Level Scheduling when applied to realistic batched part deliveries is especially misleading and should be replaced by one of our modified Level Scheduling problems if problems of real-world size with hundreds of cycles, models and parts need to be solved. Such a replacement of solution procedures becomes the more important the fewer deliveries are scheduled for JIT part supply.

6 Conclusion

The paper on hand shows how to adopt traditional Level Scheduling to batched deliveries. Two novel models are introduced where the first aims at a part leveling for a given delivery schedule and the second tries to evenly spread delivery events over the planning horizon for given delivery quantities (container sizes). For these modified Level Scheduling problems, exact (Dynamic Programming) and heuristic (Simulated Annealing) solution procedures are introduced and tested with regard to their solution performance in a comprehensive computational study. The main finding of this paper is that traditional Level Scheduling is not as suited as our novel approaches if parts are supplied in a batched manner. In such a setting, the solution procedures presented outperform traditional Level Scheduling. However, there remain plenty challenges for future research:

- First of all, more sophisticated solution procedures could be developed. For instance, the best performing exact (Bounded Dynamic Programming; Fliedner et al., 2008) and heuristic (Beam Search; Sabuncuoglu et al., 2008) procedures for the ORV could be adopted

to modified Level Scheduling. This would require the introduction of efficient bounding procedures.

- There might be a mixture of policies, i.e., for some parts delivery quantities and for others delivery schedules are given, or even both parameters might be fixed. For such a mixed situation, an integrating model could be developed.

- The modified Level Scheduling problems for batched part supply are by themselves merely surrogate models for the underlying economic factors, as a leveled distribution of part deliveries does not necessarily yield a direct economic value. It is nevertheless said to facilitate a JIT-supply, as the need for costly safety-stocks and flexible capacities is reduced. This raises the question for a detailed model of part supply and its related costs. Only such a model or representative simulation studies of real-world situations could ultimately answer the question about suitability of the presented Level Scheduling models.

With decreasing vertical integration more and more parts need to be coordinated and delivered Just-in-Time, so that part supply becomes one of the greatest challenges for operating assembly systems. Consequently the answer to these research questions would be a valuable contribution to further streamline modern mixed-model assembly lines.

Appendix: NP-Hardness Proof for LS^Q

We prove NP-hardness for LS^Q by a reduction from a specific Set Partition problem. The Balanced Set Partion problem is NP-Hard (see Garey and Johnson, 1979) and can be stated as follows:

Balanced Set Partition Problem: Given $2n$ positive integers a_i $(i = 1, ..., 2n)$ with $\sum_{i=1}^{2n} a_i = 2B$ does there exist a partition of the set $\{1, 2, ..., 2n\}$ into two sets $\{A_1, A_2\}$ of equal cardinality $|A_1| = |A_2| = n$ such that $\sum_{i \in A_1} a_i = \sum_{i \in A_2} a_i = B$?

Reduction from Balanced Set Partition to LS^Q: Consider an instance of LS^Q with four parts $P = \{1, 2, 3, 4\}$, $T = 2n + 1$ production cycles and models $M = \{1, 2, \ldots, 2n, 2n + 1\}$ with demands $d_m = 1 \quad \forall m \in M$. Let the delivery quantities be $q_1 = B$, $q_2 = (n-1)B$, $q_3 = 2n + 1$ and $q_4 = 1$ and initial inventories be $s_1 = B$, $s_2 = (n-1)B$, $s_3 = n$ and $s_4 = 2n^2 + n$. The part consumption for the first $2n$ models is:

$$\begin{aligned} a_{1m} &= a_m \\ a_{2m} &= B - a_m \\ a_{3m} &= 1 \\ a_{4m} &= m \end{aligned} \qquad \forall m \in M \backslash \{2n + 1\} \tag{22}$$

The part coefficients of the last model are $a_{1,2n+1} = a_{2,2n+1} = a_{4,2n+1} = 0$ and $a_{3,2n+1} = n + 1$. Such an LS^Q-instance can be derived from any instance of Balanced Set Partition polynomially in n, where the first $2n$ models correspond to the $2n$ integer values of Balanced Set Partition.

Notice that due to the definition of parameter values the total numbers of deliveries for the first three parts are equal to $Y_1 = \frac{2B-B}{B} = Y_2 = \frac{(2n-2)B-(n-1)B}{(n-1)B} = Y_3 = \frac{3n+1-n}{2n+1} = 1$, where $Y_p = \sum_{t=1}^{T} y_{pt}$. Part 4 was merely introduced to allow a direct matching of integer values from Set Partition. Its initial inventory is sufficient for satisfying the total part consumption, so that $Y_4 = \sum_{t=1}^{T} y_{t4} = \frac{2n^2+n-2n^2-n}{1} = 0$ and no delivery is required. As a consequence, part 4 does not influence the quality of solution sequences and we will concentrate on the first three parts in the following.

Kubiak and Sethi (1991) show for the PRV Problem (Miltenburg, 1989) that a number of copies of a model family can be evenly distributed in an assembly sequence by computing "ideal due dates" for each copy. If all copies can be assigned to their respective due date, then the deviation from the target rate is minimal for this model family. We can use this insight to derive a sufficient optimality condition for LS^Q. Let D_p^* denote the set of ideal delivery points for part p. This set is computed by

$$D_p^* = \left\{ \left\lceil \frac{2k-1}{2r'_p} \right\rceil : k = 1, ..., Y_p \right\} \tag{23}$$

As the first three parts all require exactly one delivery, ideal due dates according to (23) are all equal: $D_1^* = D_2^* = D_3^* = \{n + 1\}$. In other words, the delivery should ideally be scheduled at the middle position of the production sequence for all three parts. Notice that we can use this insight to compute a lower bound for an LS^Q-instance which we will denote as LB. Kubiak and Sethi (1991) further show that any deviation from this middle position leads to additional deviation penalties, so that for LS^Q-instances of the described form $Z = LB$ only holds if $y_{1,n+1} = y_{2,n+1} = y_{3,n+1} = 1$.

We will show that the decision problem which answers the question of whether there exists a solution to such an LS^Q-instance with $Z \leq LB$ is at least as hard as Balanced Set Partition.

The solution of any YES-instance of Balanced Set Partition can be transformed to a solution of LS^Q by simply arranging the sets A_1 and A_2 and the additional last model as follows:

$$\pi :=< A_1 \quad A_2 \quad 2n + 1 >,$$

where the internal order of members in sets A_1 and A_2 can be determined arbitrarily. It can be easily verified that such a solution sequence always allows that delivery points are set to their ideal due date for all relevant parts. As a consequence, the resulting objective value of such a solution will be LB, so that it provides a certificate for a YES-instance of LS^Q.

We further established that for any YES-instance of LS^Q it has to hold that deliveries occur

at the ideal middle position for the first three parts. Any early or late delivery would lead to increased deviation penalties. From restrictions (8)-(10) of the LS^Q model we can deduce two necessary conditions which allow such a timely delivery. It has to hold that:

$$\sum_{t=1}^{n} \sum_{m \in M} x_{mt} a_{pm} \leq s_p \wedge \sum_{t=1}^{n+1} \sum_{m \in M} x_{mt} a_{pm} > s_p \quad \forall p \in P \backslash \{4\} \tag{24}$$

If the first condition is violated, part delivery needs to be scheduled earlier than $n+1$, if the second condition is violated, deliveries need to be scheduled later. It immediately follows from (24) that model $2n+1$ cannot be scheduled in the first n slots of the sequence, since $a_{3,2n+1} = n+1 > s_p = n$. Instead, the first n slots of the sequence need to consist of a subset of size n of the first $2n$ models. Due to the definition of part coefficients, it holds for any subset M' of M that the cumulated part consumptions of the first two parts are $\sum_{m \in M'} a_{1m} = |M'| \cdot B - \sum_{m \in M'} a_{2m} \quad \forall M' \subset M$. As a consequence, it holds for a subset M^* of size n, that if $\sum_{m \in M^*} a_{1m} < B$ then $\sum_{m \in M^*} a_{2m} > (n-1)B$. In other words, if the total consumption of part 1 up to cycle n is strictly smaller than its initial inventory B, then the total consumption of part 2 violates (24) and vice versa. It follows for a certificate of a YES-instance of LS^Q that $\sum_{t=1}^{n} \sum_{m \in M} x_{mt} a_{pm} = B$, which directly yields the required balanced partition. We can conclude that an instance of Balanced Set Partition is a YES-instance, if and only if the corresponding LS^Q-instance is a YES-instance, which completes the proof. $\square$

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, 101–131.

[4] Boysen, N., Fliedner, M., Scholl, A., 2007a. Level scheduling under storage constraints, International Journal of Production Research (to appear).

[5] Boysen, N., Fliedner, M., Scholl, A., 2007b. The product rate variation problem and its relevance in real world mixed-model assembly lines, European Journal of Operational Research (to appear).

[6] Boysen, N., Fliedner, M., Scholl, A., 2008. Sequencing mixed-model assembly lines to minimize part inventory cost, OR Spectrum 30, 611–633.

[7] Boysen, N., Fliedner, M., Scholl, A., 2009. Sequencing mixed-model assembly lines: Survey, Classification and Model Critique, European Journal of Operational Research 192, 349–373.

[8] 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.

[9] Fliedner, M., Boysen, N., Scholl, A., 2008. Solving symmetric mixed-model multi-level just-in-time scheduling problems, Jena Research Papers in Business and Economics (JBE) 18/2008, FSU Jena, Germany.

[10] Garey, M.R., Johnson, D.S., 1979. Computers and intractability: A guide to the theory of NP-completeness, Freeman, New York.

[11] Inman, R.R., Bulfin, R.L., 1991. Sequencing JIT mixed model assembly lines, Management Science 37, 901–904.

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

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

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

[15] Kubiak, W., Sethi, S.P., 1991. A note on "level schedules for mixed-model assembly lines in just-in-time production systems", Management Science 37, 121–122.

[16] Kubiak, W., Steiner, G., Yeomans, J.S., 1997. Optimal level schedules for mixed-model, multi-level just-in-time assenbly systems, Annals of Operations Research 69, 241–259.

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

[18] Miltenburg, J., 1989. Level Schedules for mixed-model assembly lines in just-in-time production systems, Management Science 35, 192–207.

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

[20] Sabuncuoglu, I., Gocgun, Y., Erel, E., 2008. Backtracking and exchange of information: Methods to enhance a beam search algorithm for assembly line scheduling, European Journal of Operational Research 186, 915–930.

[21] 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 191, 912–927.

[22] Steiner, G., Yeomans, J.S., 1993. Level schedules for mixed-model, just-in-time processes, Management Science 39, 728–735.

[23] Tsai, L.-H., 1995. Mixed-model sequencing to minimize utility work and the risk of conveyor stoppage, Management Science 41, 485–495.

[24] 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.