Jena Research Papers in Business and Economics

Solving symmetric mixed-model multi-level just-in-time scheduling problems

Malte Fliedner, Nils Boysen, Armin Scholl

18/2008

Jenaer Schriften zur Wirtschaftswissenschaft

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

ISSN 1864-3108

Solving symmetric mixed-model multi-level just-in-time scheduling problems

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

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

Abstract The generation of leveled production schedules is of high importance for mixed-model assembly lines whose parts and materials are supplied just-in-time by multi-level production processes. The Output Rate Variaton problem is the standard mathematical representation of this complex level scheduling problem and has been extensively studied by research thus far. This work identifies novel symmetries in solution sequences of this problem class and shows how these insights can be used to improve exact solution procedures presented in the literature. The effectiveness of the modifications is evaluated by a computational study.

Keywords: Mixed-Model Assembly Lines, Level Scheduling, Output Rate Variation Problem, Dynamic Programming

1 Introduction

Mixed-model assembly systems are employed to produce a huge variety of different models of a common base product on the same assembly line. Their high technical flexibility allows facultative production sequences at negligible setup times and cost. Due to the high demand of diverse parts and materials, mixed-model assembly systems are often supplied just-in-time by upstream production processes. In order to facilitate just-in-time deliveries and keep safety stocks as small as possible, the actual output rates for all products, parts and materials should be kept close to constant, as this avoids overproportionally high demand peaks which might not be served timely. As part of the famous

Toyota Production System [1], it is thus proposed to find a final assembly sequence which levels demand rates for all required materials and production processes. This sequencing problem is referred to as level scheduling and has received widespread attention in research and practice alike and is still vividly discussed up to now (e.g., see [2], [3], [4], [5]). A recent survey on this and other mixed-model sequencing approaches is provided by Boysen et al. [6].

In the literature two fundamental optimization problems for level scheduling have been proposed. The so called Product Rate Variation problem focuses on the final assembly stage and aims at evenly spreading the production rates of different product models. Since in just-in-time production systems, the final model sequence directly pulls the material demand of upstream production processes, such a leveled sequence should also lead to leveled material demands whenever products require approximately the same mix and number of parts (e.g., [7]). However, this assumption seems to be unjustified for the vast majority of real-world assembly systems (see [8]).

The more detailed Output Rate Variation (ORV) problem takes the actual bills of material of each product into account. The problem can be summarized as follows: A set of P product models (or products for short) is to be sequenced on an assembly line. Each product $p \in P$ is demanded in discrete quantities D_p, where each copy of a product is assigned to exactly one distinct production cycle t, so that in total $T = \sum_{p \in P} D_p$ slots are available. The final assembly stage requires a number of subassemblies, parts and materials which are supplied by a set K of preceding production processes. Each process can be thought of as a preceding production level which supplies a set M_k of outputs to the final level. For each output $m \in M_k$ of a process $k \in K$ a constant target rate is calculated, which corresponds to the ideal fraction of the total demand at which a particular output m should be required at any given production cycle in an ideally leveled schedule.

Target rates are usually generated in one of the two following fashions. Let a_{pmk} denote the demand coefficient of a product p for output m at process k and $A_{mk} = \sum_{p \in P} a_{pmk}$ be the total demand for output m at process k. This total demand is either leveled over time, so that the target rate amounts to $\frac{A_{mk}}{T}$ (see [9]) or it is leveled with respect to the total demand for all outputs of process k, so that the target rate becomes $\frac{A_{mk}}{\sum_{m' \in M_k} A_{m'k}}$ (see [10]). As the insights generated in this work are equally valid for both types of target rates, we will instead more generally refer to the constant target quantity l_{pmk} of output m at process k which a product p should ideally demand to allow a completely leveled schedule. If demand is leveled over time, then the target quantity is exactly equal to the target rate, so that $l_{pmk} = \frac{A_{mk}}{T}$, and furthermore identical for all products $p \in P$. If demand is leveled with respect to all other outputs of a process, then the corresponding target rate is weighted with the total demand for all outputs of a process k for each product individually, so that $l_{pmk} = \frac{A_{mk}}{\sum_{m' \in M_k} A_{m'k}} \cdot \sum_{m' \in M_k} a_{pm'k} \quad \forall p \in P$. The following example clarifies the calculation of target rates and target quantities.

Example: Three products $P = \{1, 2, 3\}$ are to be sequenced on an assembly line in the quantities $D_1 = 2$ and $D_2 = D_3 = 1$. The assembly is served by a single production

T	number of production cycles (index t)
P	set of products (index p)
K	set of production processes (index k)
M_k	set of materials produced by process k (index m)
X_{pt}	cumulated production quantity of product p up to cycle t
x_{pt}	a copy of product p is assigned to cycle t ($x_{pt} = 1$) or not ($x_{pt} = 0$)
D_p	total demand for product p
a_{pmk}	quantity of output m demanded by product p at process k
A_{mkt}	cumulated quantity of output m at process k induced by all product units up to cycle t
l_{pmk}	target quantity of output m at process k induced by product p
L_{mkt}	cumulated target quantity of output m at process k induced by all product units assigned up to cycle t
δ_{mkt}	deviation from target quantity of material m for process k at cycle t

Table 1: Notation

process $|K| = 1$ which provides two outputs $|M_1| = 2$. Output demands are $a_{111} = 1$ and $a_{121} = 2$ for product 1, $a_{211} = 3$ and $a_{221} = 0$ for product 2 and $a_{311} = a_{321} = 1$ for product 3, respectively. The total demand for output 1 over all products is thus $A_{11} = 2 \cdot 1 + 1 \cdot 3 + 1 \cdot 1 = 6$ and for output 2 $A_{21} = 2 \cdot 2 + 1 \cdot 0 + 1 \cdot 1 = 5$.

If output demand is leveled over time then the target quantitities amount to $l_{111} = l_{211} = l_{311} = \frac{6}{4} = 1.5$ and $l_{121} = l_{221} = l_{321} = \frac{5}{4} = 1.25$. This means that in the ideal schedule leveled over time, a unit of any product should require 1.5 units of output 1 and 1.25 units of output 2.

If demand is leveled with respect to all outputs of the process, the target rates for the two outputs are $\frac{6}{11}$ and $\frac{5}{11}$. Target quantities are identical for products 1 and 2, i.e., $l_{111} = l_{211} = \frac{6}{11} \cdot 3 = 1.64$ and $l_{121} = l_{221} = \frac{5}{11} \cdot 3 = 1.36$, since both products demand the same total quantity of outputs. They differ for product 3, i.e., $l_{311} = \frac{6}{11} \cdot 2 = 1.09$ and $l_{321} = \frac{5}{11} \cdot 2 = 0.91$. It follows that in the ideal level schedule with respect to the total output demand, any unit of products 1 and 2 should require 1.64 and 1.36 units of outputs 1 and 2, while product 3 should instead demand 1.09 and 0.91 units, respectively.

Apparently the actual output demands a_{pmk} of products in the example differ from the ideal target quantities. As a consequence any feasible production schedule will deviate from the ideal level schedule to a certain extent. The objective of the ORV problem is hence to minimize the total deviations between actual and target demands. The ORV problem is usually stated on the basis of cumulated quantities. Let X_{pt} be the cumulated production quantity of product p up to cycle t then the cumulated demands for outputs induced by the final assembly sequence up to cycle t amount to $A_{mkt} = \sum_{p \in P} a_{pmk} X_{pt}$ and the cumulated target quantities $L_{mkt} = \sum_{p \in P} l_{pmk} X_{pt}$, respectively.
Making use of the additional notation of Table 1 the general ORV problem can be modeled

as follows:

$$
\begin{aligned}
[P_1] \quad & \text{Minimize } H\left(F\left(Dev\left(A_{mkt}-L_{mkt}\right)\right)\right) && && (1)\\
& A_{mkt} = \sum_{p\in P} a_{pmk} X_{pt} && \forall k \in K, m \in M_k, t = 1,\ldots,T && (2)\\
& L_{mkt} = \sum_{p\in P} l_{pmk} X_{pt} && \forall k \in K, m \in M_k, t = 1,\ldots,T && (3)\\
& X_{pT} = D_p && \forall p \in P && (4)\\
& 0 \leq X_{pt} - X_{p,t-1} \leq 1 && \forall p \in P, t = 2,\ldots,T && (5)\\
& \sum_{p\in P} X_{pt} = t && \forall t = 1,\ldots,T && (6)\\
& X_{pt} \in \mathbb{N}^0 && \forall p \in P, t = 1,\ldots,T && (7)
\end{aligned}
$$

Equations (2) and (3) define the cumulated output demands and target quantities as a result of the final assembly sequence. Equations (4) ensure that the final product demand D_p is met for each product. Constraints (5) force the cumulated production to monotonically increase over time. Together with (6) and (7) they additionally guarantee that one and only one copy of a product model is assigned to each cycle t. Note that if demand is leveled over time, l_{pmk} is the same for all products p and due to (6) cumulated target quantities could more readily calculated by $L_{mkt} = \frac{A_{mk}}{T} \cdot t$.

Objective function (1) evaluates and aggregates all deviations to yield a single objective value per production schedule. We decompose the total objective into three separate functions. First, positive and negative deviations per output m, process k and cycle t are consolidated by deviation function $Dev(\cdot)$. The typical deviation functions proposed in the literature are unimodal, convex and axisymmetric around zero, so that $Dev(a) = Dev(-a)$, since positive deviations are usually considered as being as unwanted as negative ones. In the vast majority of cases either absolute $Dev(\cdot) = |\cdot|$ or squared $Dev(\cdot) = (\cdot)^2$ deviations are considered.

The consolidated deviations per output m, process k and cycle t are in a next step aggregated over all outputs and processes by aggregation function $F(\cdot)$. Usually, the sum of deviations $F(\cdot) = \sum_{k\in K}\sum_{m\in M_k}(\cdot)$ or the maximum deviations per output and process $F(\cdot) = \max_{k\in K, m\in M_k}\{\cdot\}$ are considered, however, also a combination such as $F(\cdot) = \max_{k\in K}\sum_{m\in M_k}\{\cdot\}$ is possible. $F(\cdot)$ can additionally consider output or process specific weights to normalize deviations or express the relative importance of parts (see [10]).

The resulting deviations per cycle are finally aggregated by $H(\cdot)$. Since deviations are usually not weighted differently with regard to the time slot they occur, $H(\cdot)$ is a commutative function, so that $H(a,b) = H(b,a)$, such as the sum over all cycles $H(\cdot) = \sum_{t=1}^{T}(\cdot)$ or the maximum deviation per cycle $H(\cdot) = \max_{t=1,\ldots,T}\{\cdot\}$. On the basis of these functions, we can line out the scope of this paper according to the following definition.

Definition: The Symmetric Output Rate Variation (SORV) problem is the problem defined by $[P_1]$, with an axisymmetric deviation function $Dev = (\cdot)$, so that $Dev(a) = Dev(-a)$ and a commutative function $H = (\cdot)$, so that $H(a,b) = H(b,a)$, which aggregates deviations over all production cycles.

As was mentioned above, the vast majority of (multi-level) mixed-model just-in-time sequencing problems covered in literature and practice is in line with the above definition, so that the insights generated in this paper are highly relevant for all reported applications.

In the remainder of the work we will analyze the structure of SORV problems and show how these structural insights can strengthen exact solution algorithms. This potential was already mentioned by Bautista et al. in [9] for a special version of the ORV problem, but has not been considered in subsequent research (e.g. [4], [10]). We generalize and proof their conjecture and show how to implement the necessary extensions for different objectives. For this purpose the paper is organized as follows. In Section 2 we will investigate important subproblems of SORV and identify symmetries in its solution sequences. Section 3 will demonstrate how this knowlendge can be used to improve exact solution methods. Section 4 hence evaluates the approach by a computational experiment.

2 Symmetric Output Rate Variation Problem

2.1 Formalization of Subproblems

Before we investigate the solution structure of SORV instances more closely, we will first differentiate two relevant subproblems which will be useful in the subsequent discussion. In contrast to the integer program $[P_1]$ based on cumulated quantities (X_{pt}), we will chose an equivalent binary representation where variables x_{pt} decide whether product p is assigned to cycle t ($x_{pt} = 1$) or not ($x_{pt} = 0$). Let $D_p^* \leq D_p \quad \forall p \in P$ be a partial production plan of a corresponding instance of $[P_1]$. Assume that these products have to be assigned to the first $t^* = \sum_{p \in P} D_p^*$ production cycles, then the optimal partial production schedule can be computed as follows

$$[P_2] \quad \text{Minimize } H^0\left(F\left(Dev\left(\delta_{mkt}\right)\right)\right) \tag{8}$$

$$\delta_{mkt} = \delta_{mk,t-1} + \sum_{p \in P} (a_{pmk} - l_{pmk}) x_{pt} \qquad \forall k \in K, m \in M_k, t = 1, \ldots, t^* \tag{9}$$

$$\delta_{mk0} = 0 \qquad \forall k \in K, m \in M_k \tag{10}$$

$$\sum_{t=1}^{t^*} x_{pt} = D_p^* \qquad \forall p \in P \tag{11}$$

$$\sum_{p \in P} x_{pt} = 1 \qquad \forall t = 1, \ldots, t^* \tag{12}$$

$$x_{pt} \in \{0,1\} \qquad \forall p \in P, t = 1, \ldots, t^* \tag{13}$$

where δ_{mkt} is the deviation caused by material m of process k from the target quantity at cycle t. Equations (9) define this deviation at cycle t as equal to the deviation of the previous cycle $t-1$ increased by the difference between the demand coefficient and the target demand induced by the model p assigned at cycle t. Equations (11) make sure that the product demand is met, while constraints (12) and (13) enforce that only a single product model is assigned to each cycle t.

This formalization requires the definition of an initial deviation δ_{mk0} for all outputs and processes. Aggregation function $H(\cdot)$ is thus extended to $H^0(\cdot)$, which is supposed to consider all deviations from cycle 0 to t^*, e.g. $H^0(\cdot) = \sum_{t=0}^{t^*}(\cdot)$, if the sum of deviations is considered. Due to (10), this extension does not affect the objective value of $[P_2]$, so that by recursively inserting for $\delta_{mk,t-1}$ in equations (9), it can be readily shown that the calculated deviations are identical for $[P_1]$ and $[P_2]$ if $D_p^* = D_p \quad \forall p \in P$ and that both mathematical programs are equivalent formalizations of the same ORV problem in that case.

On the basis of $[P_2]$, we can further introduce a more general problem where deviations δ_{mk0} at cycle 0 of each output and process are allowed to vary:

$$\begin{aligned} [P_3] \quad & \text{Minimize } (8) \\ & (9), (11) - (13) \\ & \delta_{mk0} = S_{mk} \qquad \forall k \in K, m \in M_k \end{aligned} \tag{14}$$

where S_{mk} denotes the initial deviation of output m at process k.

As we will see in Section 3 instances of $[P_2]$ and $[P_3]$ are of special relevance in the solution process of exact procedures for the ORV problem.

2.2 Symmetries of Solution Sequences

On the basis of the mathematical programs presented in Sections 1 and 2.1, we will investigate structural aspects of SORV problems in this section. Instead of representing the production schedule by integer or binary variables, we will make use of a sequence representation, where $\pi = (p_1, p_2, ..., p_T)$ denotes the production sequence up to slot T and $\pi(t) = p_t$ yields the product $p_t \in P$ which is assigned to time slot t.

Let $A^\pi_{mkt} = \sum_{\tau=1}^{t} a_{\pi(\tau),mk}$ be the cumulated demand for output m of process k induced by sequence π up to slot t and $L^\pi_{mkt} \sum_{\tau=1}^{t} l_{\pi(\tau),mk}$ be the respective cumulated target quantity, the actual deviations from target quantities at each slot t of sequence π can be calculated as follows

$$\begin{aligned} \delta^\pi_{mkt} &= \delta^\pi_{mk,t-1} + a_{\pi(t),mk} - l_{\pi(t),mk} \\ &= \delta^\pi_{mk0} + A^\pi_{mkt} - L^\pi_{mkt} \qquad \forall k \in K, m \in M_k, t = 1, \dots, T \end{aligned} \tag{15}$$

where initial deviations δ^π_{mk0} are zero for all outputs in problems $[P_1]$ and $[P_2]$ and can vary for $[P_3]$.

We will start out with some simple observations on the structure of a solution sequence to the ORV problem. The following statements (16) to (19) are true

$$\begin{aligned}
A^{\pi}_{mkt} &= A_{mk} - \sum_{\tau=t+1}^{T} a_{\pi(\tau),mk} \quad \forall k \in K, m \in M_k, t = 1, \dots, T-1 && (16)\\
L^{\pi}_{mkt} &= A_{mk} - \sum_{\tau=t+1}^{T} l_{\pi(\tau),mk} \quad \forall k \in K, m \in M_k, t = 1, \dots, T-1 && (17)\\
A^{\pi}_{mkT} &= A_{mk} \qquad \forall k \in K, m \in M_k && (18)\\
L^{\pi}_{mkT} &= A_{mk} \qquad \forall k \in K, m \in M_k && (19)
\end{aligned}$$

which follows directly from the definition of the actual and targeted cumulated demands A^{π}_{mkt} and L^{π}_{mkt}. We can directly conclude from (18) and (19) that for any solution sequence π the observed deviation is zero at the last slot T

$$\delta^{\pi}_{mkT} = \delta^{\pi}_{km0} + A^{\pi}_{mk} - L^{\pi}_{mk} = 0 \quad \forall k \in K, m \in M_k \tag{20}$$

We can use these statements to proof the following two lemmas:

Lemma 1: For any solution sequence π and its inverted sequence π', with $\pi(t) = \pi'(T-t+1)$ for $t = 1, \dots, T$ it holds that $-\delta^{\pi}_{mkt} = \delta^{\pi'}_{mk,T-t} \quad \forall k \in K, m \in M_k, t = 0, \dots, T$.

Proof: By considering (20), the equation certainly holds for $t = 0$ and $t = T$ since $-\delta^{\pi}_{mk0} = \delta^{\pi'}_{mk0} = -\delta^{\pi}_{mkT} = \delta^{\pi'}_{mkT} = 0 \quad \forall k \in K, m \in M_k$, but it also holds for all other cases as is shown by insertion:

$$\begin{aligned}
\delta^{\pi'}_{mk,T-t} &= \delta^{\pi'}_{mk0} + A^{\pi'}_{mk,T-t} - L^{\pi'}_{mk,T-t}\\
&= 0 + (A_{mk} - A^{\pi}_{mkt}) - (A_{mk} - L^{\pi}_{mkt})\\
&= -A^{\pi}_{mkt} + L^{\pi}_{mkt}\\
&= -\delta^{\pi}_{mkt}
\end{aligned} \tag{21}$$

Lemma 2: Any solution sequence π to an instance of the SORV problem can be inverted to yield a sequence π', so that $\pi(t) = \pi'(T-t+1)$ for $t = 1, \dots, T$, which has the same objective value.

Proof: By Lemma 1 it holds that $-\delta^{\pi}_{mkt} = \delta^{\pi'}_{mk,T-t} \quad \forall k \in K, m \in M_k, t = 0, \dots, T$. Since $Dev(\cdot)$ is axisymmetric around zero it follows that $Dev(-\delta^{\pi}_{mkt}) = Dev(\delta^{\pi'}_{mk,T-t})$ and thus $F(Dev(-\delta^{\pi}_{mkt})) = F(Dev(\delta^{\pi'}_{mk,T-t})) \quad \forall t = 0, \dots, T$. It finally follows from commutativity of H that the production cycle at which the deviation occurs does not affect the objective, so that both sequences yield the same value.

This directly leads to Theorem 1.

Theorem 1: Any optimal solution sequence π to an instance of problem $[P_1]$ can be inverted to a solution sequence π', so that $\pi(t) = \pi'(T-t+1) \quad \forall t = 1, \dots, T$, which is

	π					π'				
t	0	1	2	3	4	0	1	2	3	4
p_t		1	2	1	3		3	1	2	1
a_{p11}		1	3	1	1		1	1	3	1
l_{p11}		1.5	1.5	1.5	1.5		1.5	1.5	1.5	1.5
δ_{11t}	0	-0.5	1.0	0.5	0	0	-0.5	-1.0	0.5	0
a_{p21}		2	0	2	1		1	2	0	2
l_{p21}		1.25	1.25	1.25	1.25		1.25	1.25	1.25	1.25
δ_{21t}	0	0.75	-0.5	0.25	0	0	-0.25	0.5	-0.75	0
$\sum_{k\in K}\sum_{m\in M_k}\lvert\delta_{mkt}\rvert$	0	1.25	1.5	0.75	0	0	0.75	1.5	1.25	0

Table 2: Comparison of SORV Solution Sequences

also optimal for the instance of $[P_1]$.

Proof: Follows immediately from Lemma 2. Since any inverted sequence π' yields the same objective value as π, π' needs to be optimal if π is and vice versa.

The following example depicts this relationship.

Example (cont.): Consider sequence $\pi = (1, 2, 1, 3)$, which is an optimal solution to the problem instance introduced in Section 1 if the objective is to level the sum of absolute deviations over time. Table 2 displays the progression of deviations for both required outputs as compared to its inverted sequence $\pi' = (3, 1, 2, 1)$.
Note that the deviations of both solution sequences are symmetric for the outputs and as a result also the progression of the aggregated deviation is symmetric. It follows that both sequences have the same objective value of 3.5, so that since π is an optimal solution, so is π'.

We will now extend the discussion to partial sequences, where we refer to a partial sequence as any sequence of length $t^* < T$ to which not all product copies have been assigned yet. Such partial sequences are typically generated in the process of iterative solution algorithms which start from the first cycle $t = 1$ and then successively assign products to subsequent production cycles on the basis of some selection scheme until a solution sequence has been reached. In analogy to the concepts introduced above, we will show that the identified symmetry also holds for partial sequences. For convenience we will use the same symbols π and π' for partial sequences.

Consider a partial sequence π with length t^* to which D^*_p copies of product p have been assigned, so that $\sum_{p\in P} D^*_p = t^*$. Irrespective of the exact order of models, the deviation at the last slot t^* of this partial sequence will amount to:

$$\begin{aligned} \delta^{\pi}_{mkt^*} &= \delta^{\pi}_{mk0} + A^{\pi}_{mkt^*} - L^{\pi}_{mkt^*} \\ &= \delta^{\pi}_{mk0} + \textstyle\sum_{p\in P} D^*_p \left(a_{pmk} - l_{pmk}\right) \quad \forall k \in K, m \in M_k \end{aligned} \tag{22}$$

Note that we can interpret any partial sequence for an instance of $[P_1]$ as a feasible solution to an instance of subproblem $[P_2]$ with product demands D_p^* and initial deviations are zero or likewise as a feasible solution to an instance of $[P_3]$, where initial deviations are allowed to vary.

In analogy to Lemmas 1 and 2 for complete solution sequences, we can prove the following two lemmas for partial sequences.

Lemma 3: It holds for any partial sequence π with length t^* to which D_p^* copies of product $p \in P$ have been assigned and its inverted partial sequence π', with $\pi(t) = \pi'(t^* - t + 1) \quad \text{for } t = 1, \ldots, t^*$, that $-\delta^{\pi}_{mkt} = \delta^{\pi'}_{mk,t^*-t} \quad \forall k \in K, m \in M_k, t = 0, \ldots, t^*$ with $\delta^{\pi}_{mk0} = 0$ and $\delta^{\pi'}_{mk0} = \sum_{p \ inP} D_p^* \left(l_{pmk} - a_{pmk}\right) \quad \forall k \in K, m \in M_k$.

Proof: Due to (22) and the definition of initial deviations for π', the relationship obviously holds for $t = 0$ and $t = t^*$, but likewise holds for all intermediate slots as

$$\begin{aligned}
\delta^{\pi'}_{mk,t^*-t} &= \delta^{\pi'}_{mk0} + A^{\pi'}_{mk,t^*-t} - L^{\pi'}_{mk,t^*-t} \\
&= \textstyle\sum_{p \ inP} D_p^* \left(l_{pmk} - a_{pmk}\right) + \left(A^{\pi}_{mk,t^*} - A^{\pi}_{mkt}\right) - \left(L^{\pi}_{mk,t^*} - L^{\pi}_{mkt}\right) \\
&= \textstyle\sum_{p \ inP} D_p^* \left(l_{pmk} - a_{pmk}\right) + \sum_{p \ inP} D_p^* a_{pmk} - A^{\pi}_{mkt} \\
&\quad - \textstyle\sum_{p \ inP} D_p^* l_{pmk} + L^{\pi}_{mkt} \\
&= -A^{\pi}_{mkt} + L^{\pi}_{mkt} \\
&= -\delta^{\pi}_{mkt}
\end{aligned} \tag{23}$$

Lemma 4: Any feasible solution sequence π to an instance of problem $[P_2]$ can be inverted to sequence π', with $\pi(t) = \pi'(t^* - t + 1) \quad \text{for } t = 1, \ldots, t^*$, so that π' is a feasible solution to the corresponding instance of $[P_3]$ with $S_{mk} = \sum_{p \ inP} D_p^*(l_{pmk} - a_{pmk}) \quad \forall k \in K, m \in M_k$ and the objective values of π in $[P_2]$ and π' in $[P_3]$ are exactly equal. Likewise any feasible solution sequence to such a $[P_3]$-instance can be inverted to a solution sequence of $[P_2]$, so that both have the same obejctive value.

Proof: In analogy to Lemma 2. If π' is a solution to such an instance of $[P_3]$, its initial deviation is set to $\delta^{\pi'}_{mk0} = S_{mk} = \sum_{p \ inP} D_p^*(l_{pmk} - a_{pmk}) \quad \forall k \ inK, m \in M_k$. It follows from Lemma 3 that $-\delta^{\pi}_{mkt} = \delta^{\pi'}_{mk,t^*-t} \quad \forall k \in K, m \in M_k, t = 0, \ldots, t^*$, so that for an axisymmetric deviation function $Dev(\cdot)$ and a commutative aggregation function $H^0(\cdot)$ the objective value of π in $[P_2]$ is equal to the objective value of π' in $[P_3]$.

There apparently is a duality between problems $[P_2]$ and $[P_3]$, in the sense that for any solution to an instance of $[P_2]$ we can find a corresponding solution, i.e., its inversion, to a respective instance of $[P_3]$ which yields the exact same objective value and vice versa. This leads to the following theorem:

Theorem 2: Any optimal solution π to an instance of $[P_2]$ can be inverted to a sequence π', with $\pi(t) = \pi'(t^* - t + 1) \quad \text{for } t = 1, \ldots, t^*$, so that π' is optimal for the corresponding instance of $[P_3]$ with $S_{mk} = \sum_{p \ inP} D_p^* \left(l_{pmk} - a_{pmk}\right) \quad \forall k \in K, m \in M_k$.

	π			π'		
t	0	1	2	0	1	2
p_t		3	1		1	3
a_{p11}		1	1		1	1
l_{p11}		1.5	1.5		1.5	1.5
δ_{11t}	0	-0.5	-1.0	1.0	0.5	0
a_{p21}		1	2		2	1
l_{p21}		1.25	1.25		1.25	1.25
δ_{21t}	0	-0.25	0.5	-0.5	0.25	0
$\sum_{k\in K}\sum_{m\in M_k}\lvert\delta_{mkt}\rvert$	0	0.75	1.5	1.5	0.75	0

Table 3: Comparison of Partial Sequences

Proof: By contradiction. Let π be the optimal solution sequence of an instance of $[P_2]$, then due to Lemma 4 it can be inverted to form a solution π' for the corresponding instance of $[P_3]$ with the same objective value. If there was a better solution to the $[P_3]$-instance, it could in turn be inverted to generate a second solution to the $[P_2]$-instance, which due to Lemma 4 had the same objective value, so that π could not be optimal.

The following example displays two partial sequences and clarifies the relationships between them.

Example (cont.) Consider the partial sequence $\pi = (3, 1)$, which constitutes an optimal solution for the instance of $[P_2]$ with $t^* = 2$, $D^*_1 = D^*_3 = 1$ and $D^*_2 = 0$ and compare it to sequence $\pi' = (1, 3)$, which is an optimal solution for the corresponding instance of $[P_3]$ with $S_{11} = 1.5 - 1 + 1.5 - 1 = 1$ and $S_{21} = 1.25 - 1 + 1.25 - 2 = -0.5$. Table 3 displays the progression of deviations, if the sum of absolute deviations is to be leveled over time. It shows that both sequences have symmetric deviations, if initial deviations for sequence π' are chosen accordingly and thus also yield the same objective value of 2.25.

3 Consequences for Solution Methods

3.1 Description of the Dynamic Programming Algorithm

In this section we will demonstrate how to exploit these structural insights in order to improve solution procedures. The best exact solution methods for the ORV problem are Dynamic Programming (DP) algorithms which are based on an implicit enumeration scheme of final assembly sequences (see [4], [9], [10]). In these approaches the solution space of the problem is represented by an acyclic digraph $G = (V, E, w)$ with a node set V comprising $T+1$ *stages*, a set E of arcs connecting nodes of adjacent stages and a node weighting function $w : V \to \mathbb{R}$. A stage t of the digraph contains a set $V_t = \{1, 2, \ldots\}$ of vertex numbers which represent all feasible *states* of the production system in cycle t. We will identify a node/state (t, i) by its stage number t, i.e., the production cycle up to which model copies have already been assigned, and its vertex number $i \in V_t$. Stored with each state is a vector $\mathbf{X}_t^i$ of length $|P|$ which stores the cumulated quantities X_{tp}^i of all models $p \in P$ produced up to cycle t.

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

$$\sum_{p \in P} X_{tp}^i = t \qquad \forall\, t = 0, \ldots, T,\ i \in V_t \tag{24}$$

$$0 \le X_{tp}^i \le D_p \qquad \forall\, p \in P,\ t = 0, \ldots, T,\ i \in V_t \tag{25}$$

The node set V_0 contains a single additional starting node (initial state $(0,1)$) corresponding to the vector $\mathbf{X}_0^1 = [0, 0, \ldots, 0]$. Similarly, the node set V_T contains a single node (final state $(T,1)$) with $\mathbf{X}_T^1 = [D_1, D_2, \ldots, D_{|P|}]$.

Two nodes (t,i) and $(t+1,j)$ of two adjacent stages t and $t+1$ are connected by an arc if the associated vectors $\mathbf{X}_t^i$ and $\mathbf{X}_{t+1}^j$ differ in only one element, i.e., a copy of exactly one model is additionally produced in cycle $t+1$. Due to (24) and (25) this holds whenever $X_{tp}^i \le X_{t+1,p}^j \forall p \in P$. The overall arc set is defined as follows:

$$E = \{((t,i),(t+1,j)) \mid t = 0, \ldots, T-1, i \in V_t, j \in V_{t+1} \text{ and } X_{tp}^i \le X_{t+1,p}^j\ \forall p \in P\} \tag{26}$$

The produced quantities of all models up to cycle t in a state (t,i) directly determine the actual cumulated output demands A_{tmk}^i and target demands L_{tmk}^i for all outputs and processes:

$$A_{tmk}^i = \sum_{p \in P} X_{tp}^i \cdot a_{pmk} \qquad \forall\, k \in K, m \in M_k \tag{27}$$

$$L_{tmk}^i = \sum_{p \in P} X_{tp}^i \cdot l_{pmk} \qquad \forall\, k \in K, m \in M_k \tag{28}$$

As a consequence, a node weight w_t^i can be determined for each state (t,i) on the basis of the employed deviation function $Dev(\cdot)$ and aggregation function $F(\cdot)$. If, for instance, the sum of absolute deviations $F(Dev(\cdot)) = \sum_{k \in K} \sum_{m \in M_k} |\cdot|$ is considered, node weights are determined by:

$$w_t^i = \sum_{k \in K} \sum_{m \in M_k} |A_{tmk}^i - L_{tmk}^i| \qquad \forall\, t = 1, \ldots, T;\ i \in V_t \tag{29}$$

If instead the maximum squared deviation per output $F(Dev(\cdot)) = \max_{k \in K, m \in M_k}\{(\cdot)^2\}$ is to be minimized weights are calculated according to:

$$w_t^i = \max_{k \in K, m \in M_k} (A_{tmk}^i - L_{tmk}^i)^2 \qquad \forall\, t = 1, \ldots, T;\ i \in V_t \tag{30}$$

The weight of the initial node at $(0,1)$ is set to $w_0^1 = 0$. On the basis of this graph, the optimal solution of the ORV problem reduces to finding the shortest path from source node $(0,1)$ to sink node $(T,1)$.

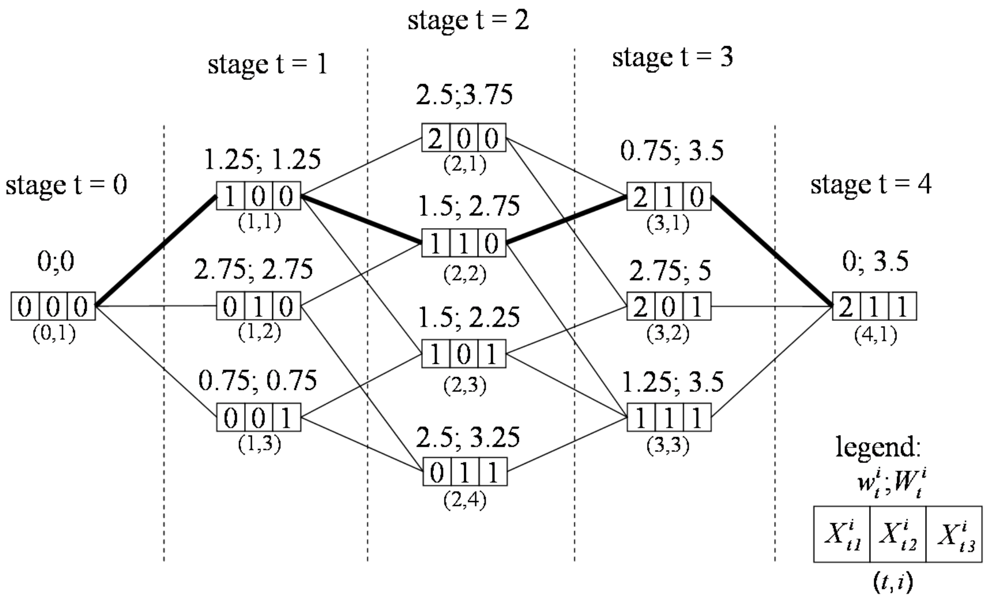

Figure 1: Example graph for the sum of absolute deviations

This path can be easily determined during the stage-wise construction of the graph by aggregating node weights w_t^i on the path up to stage t according to aggregation function $H(\cdot)$. If deviatons are summed up over all cycles by $H(\cdot) = \sum_{t=1}^{T}(\cdot)$, the aggregated weight (minimum path length) W_t^i of node i at stage t is calculated by the following recursion

$$W_t^i = \left\{\min_{(t-1,h)\in P_t^i}\{W_{t-1}^h\} + w_t^i\right\} \quad \forall t = 1,\ldots,T;\, i \in V_t \tag{31}$$

where P_t^i denotes the set of direct predecessors of node (t,i). If the maximum deviation is considered the recursion becomes

$$W_t^i = \max\left\{\min_{(t-1,h)\in P_t^i}\{W_{t-1}^h\};\, w_t^i\right\} \quad \forall\, t = 1,\ldots,T;\, i \in V_t \tag{32}$$

The best predecessor, i.e., the node of stage $t-1$ linked to (t,i) with the minimum aggregated weight, is stored for each state, so that the corresponding optimal sequence can be retrieved by tracing back all predecessor nodes along the shortest path to (t,i). As a consequence the DP-approach does not need to store the complete graph, but only the nodes at the last two stages and those nodes which are currently part of a shortest path. The optimal objective value is finally given by W_T^1 at sink node $(T,1)$.

Example (cont.): For the problem instance introduced in Section 1 the resulting graph for the sum of absolute deviations over time is displayed in Figure 1. For each node the cumulated production quantities are provided along with the node weight and the aggregated weight. The shortest path is sketched in bold, so that the optimal production sequence is $\pi = (1, 2, 1, 3)$ resulting to a minimum total deviation of 3.5. Note that just as predicted by Theorem 1, a second path with the same length can be identified, which results to the alternative optimal sequence of $\pi' = (3, 1, 2, 1)$.

In comparison to the total number of production sequences, the implicit enumeration procedure reduces the number of investigated states considerable. Nevertheless, the total number of nodes can be calculated by

$$\Pi_{p\in P}(D_p + 1) \leq \left\lceil \frac{D_1 + D_2 + \ldots + D_{|P|} + |P|}{|P|} \right\rceil^{|P|} = \left\lceil \frac{T + |P|}{|P|} \right\rceil^{|P|} \tag{33}$$

and thus increases exponentially in the problem size (see [10]).

In order to further reduce the number of nodes, Bautista et al. [9] and Kubiak et al. [10] propose to employ upper bound filters. For this purpose, a first upper bound is calculated by a heuristic procedure prior to the execution of the DP approach. During the node generation, lower bounds are computed for each state, so that a state can be fathomed whenever its lower bound is equal to or greater than the upper bound. We will briefly outline how to compute lower bounds for ORV problems.

Let δ_p^{min} denote the minimum contribution to the total deviation which is at least caused by scheduling a copy of product p. On the basis of these values, a global lower bound LB can be determined with regard to aggregation function $H(\cdot)$. If the sum of deviations is considered, D_p is summed up over all product copies:

$$LB = \sum_{p\in P} \delta_p^{min} \cdot D_p \tag{34}$$

if the sum of deviations is considered or to

$$LB = \max_{p\in P}\{\delta_p^{min}\} \tag{35}$$

if the maximum deviation per time slot is minimized. In order to determine δ_p^{min} we can make use of the following observation. The contribution of scheduling a product p at a given slot t in a sequence π apparently depends on the deviations for each material and production level at the prior slot $\delta^{\pi}_{mk,t-1}$. For convex, axissymmetric deviation functions and the vast majority of employed aggregation functions it can be shown that the deviations caused by scheduling a product p at a slot t become minimal whenever $\delta^{\pi}_{mk,t-1} = \frac{l_{pmk}-a_{pmk}}{2} \forall k \in K, m \in M_k$, which in turn means that $\delta^{\pi}_{mk,t-1} = -\delta^{\pi}_{mk,t}$ (e.g., [9]). The minimum deviation per product can thus be determined by

$$\delta_p^{min} = F(Dev(\frac{l_{pmk} - a_{pmk}}{2})) \tag{36}$$

This information can not only be used to calculate global lower bounds, but also for local lower bounds for each node in the dynamic programming algorithm by only considering the product copies which have not been scheduled yet (see [9]). For special objective functions LB can be further tightened, in order not to lose generality of the approach we will however abstain from such modifications.

3.2 Extended Dynamic Programming Approach

The structural insights of Section 2.2 can be used to improve the solution method introduced in the previous section, if the ORV problem to be solved is symmetric, i.e. covered by the SORV problem classs. Recall that after a stage t has been fully evaluated by the algorithm, for any state (t, i) in this stage the following information is available:

(i) the total production quantities $\mathbf{X}_t^i$ up to cycle t

(ii) the objective value W_t^i of the optimal partial sequence as the length of the shortest path to the corresponding node

(iii) by recursion the optimal partial production sequence π of length t which leads to the current state

A state (t, i) thus directly provides the optimal solution to a problem instance of $[P_2]$ with $D_p^* = X_{pt}^i \quad \forall p \in P$. The remaining subproblem apparently consists of optimally assigning the remaining model copies to the remaining number of $t^* = T - t$ production cycles, while considering the current deviation for each output and process at state (t, i). Note that this subproblem is actually an instance of $[P_3]$ with $D_p^* = D_p - X_{pt}^i \quad \forall p \in P$ and $S_{mk} = A_{tmk}^i - L_{tmk}^i \quad \forall k \in K, m \in M_k$. If this instance is solved to optimality, then the partial production sequence π associated with state (t, i) can be appended by the optimal solution sequence of the $[P_3]$-instance. The result is a feasible solution to the ORV problem, which furthermore constitutes the best possible solution sequence under the condition that cumulated production quantities of $X_{pt}^i \quad \forall p \in P$ are to be assigned to the first t cycles. It follows that if we had the solution to the $[P_3]$-instance, we could readily determine the best feasible solution to the ORV problem to which the partial schedule of (t, i) would result and the state could hence be fathomed. In the following we will show how the symmetries identified in Section 2.2 can be exploited to determine such a solution.

Assume that the DP approach has just evaluated stage $t^1 = \lceil \frac{T}{2} \rceil$. We select an arbitrary node of this stage (t^1, i) with cumulated production quantities of $X_{pt^1}^i \forall p \in P$. We are now interested in finding the optimal solution of the $[P_3]$-instance with $D_p^* = D_p - X_{pt^1}^i \forall p \in P$ and $S_{mk} = A_{t^1mk}^i - L_{t^1mk}^i = D_p^*(l_{pmk} - a_{pmk}) \quad \forall k \in K, m \in M_k$. Recall from Theorem 2 that the optimal solution can be retrieved by solving a problem instance of $[P_2]$ with $D_p^* = D_p - X_{pt^1}^i \forall p \in P$ and $S_{mk} = 0 \quad \forall k \in K, m \in M_k$ and invert the obtained solution sequence. As a matter of fact the Dynamic Programming algorithm has already found this solution up to stage t^1. As all partial production schedules are generated and evaluated by the algorithm there has to be an already evaluated node (t^2, j) with $X_{pt^2}^j = D_p - X_{pt^1}^i \forall p \in P$ at stage $t^2 = T - t^1$. Since $t^1 = \lceil \frac{T}{2} \rceil$, it either holds that $t^2 = t^1$ for even values of T, so that this node has to be in the same stage, or $t^2 = t^1 - 1$ for uneven T, so that the node lies in the previous stage.

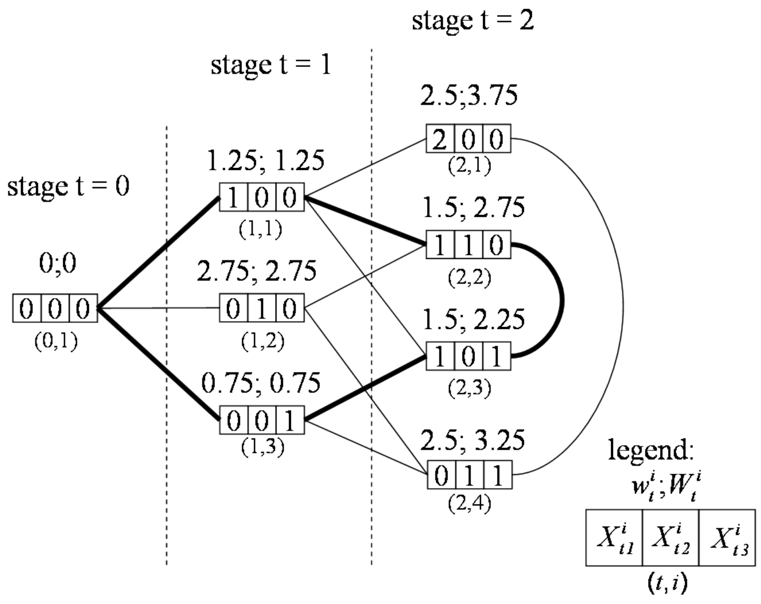

Figure 2: Example graph of extended dynamic programmin algorithm

If this complementary node (t^2, j) is retrieved, its optimal partial sequence π^2 can be inverted and appended to the partial sequence π^1 of state (t^1, i) to yield the best possible solution $\pi = (\pi^1(1), \pi^1(2), \ldots, \pi^1(t^1), \pi^2(t^2), \pi^2(t^2-1), \ldots, \pi^2(1))$ which state (t^1, i) could ever be extended to. Note that the determination of the objective value of this solution sequence again depends on aggregation function $H(\cdot)$. If the sum of deviations is considered the objective value of sequence π results to $W^i_{t^1} + W^j_{t^2} - w^i_{t^1}$, since deviations $w^i_{t^1} = w^j_{t^2}$ at slot t^1 and t^2 have been counted twice in total, once in the generation of π^1 and once to determine π^2. If maximum deviations are considered the objective value can be determined by $\max\{W^i_{t^1}, W^j_{t^2}\}$.

Example (cont): Consider the graph of Figure 2, which comprises all states of the example up to stage $t^1 = \lceil \frac{T}{2} \rceil = 2$, where four nodes have been generated and evaluated. Consider the second node of stage 2 $(2,2)$. It has cumulated production quantities $X^2_{21} = X^2_{22} = 1$ and $X^1_{23} = 0$ and an optimal partial sequence $\pi^2 = (1,2)$ on the shortest path to $(2,2)$ sketched in bold. Its complementary state is represented by node $(2,3)$ with cumulated production quantities of $X^3_{21} = X^3_{23} = 1$ and $X^3_{22} = 0$ and an optimal partial production sequence of $\pi^3 = (3,1)$. Note that the weights of both nodes are identical, i.e., $w^2_2 = w^3_2 = 1.5$ and that they have been considered in the calculation of both aggregated weights W^2_2 and W^3_2, respectively. The optimal solution to the overall problem is retrieved by combining π^2 and the inverse of π^3 to $\pi = (1,2,1,3)$ with an optimal objective value of $W^2_2 + W^3_2 - w^2_2 = 2.75 + 2.25 - 1.5 = 3.5$. The complementary node of state $(2,1)$ is $(2,4)$, however the corresponding objective value of $3.75 + 3.25 - 2.5 = 4.5$ is inferior.

For each node (t^1, i) exactly one complementary node (t^2, j) exists. If $X^i_{t^1p} = X^j_{t^2p} \forall p \in P$ then node (t^1, i) is its own complementary node (t^2, j). Note that stages t^1 and t^2 and the nodes therein can be considered in an arbitrary order, since due to Theorem 1 each obtained solution can be inverted to yield a second optimal solution.

We can conclude that for every SORV problem, the node generation process of the DP algorithm can be stopped at stage t^1. Instead of continuing the generation of nodes until T, we can rather make use of the optimal partial sequences already generated to find the global optimum solution. This can reduce the number of states up to approximately half of the total number (especially for large values of T). Note that the necessary methods of identifying and retrieving nodes at a stage on the basis of their cumulated production quantities is a standard requirement of the DP algorithm, so that the additional effort for implementation is fairly low. It can further be combined with pruning techniques, such as the upper bound filters discussed in Section (3.1). If a complementary node has already been fathomed on the basis of an upper bound and thus cannot be retrieved, then obviously also the actual node on hand can be discarded, since its final objective value needs to be at least as high as the objective value of its complementary node. In the following we will evaluate the actual run time performance of the considered extension by a computational evaluation.

4 Computational Evaluation

In the computational study we seek to evaluate the effectiveness of the algorithmic extension introduced in the previous section. As was shown the number of states which need to be investigated by a full implicit generation of sequences can be approximately halved. Since the identification of complementary nodes in the same or adjacent stage might, however, consume additional computational time, the actual run time performance can only be measured by experiment.

We thus test the performance of the Extended Dynamic Programming (EDP) algorithm compared to the original version DP on a set of generated test instances. The test bed is based on a study of Kubiak et al. [10], who consider a facility with four production levels $|K| = 4$, where the first production level represents the final assembly of products. According to the the formalization of $[P_1]$, we equivalently set $M_1 = P$ with $a_{pm1} = 1$, if $p = m$ and $a_{pm1} = 0$ otherwise. The number of products is varied during the experiment, so that $|P| \in \{8, 10, 12\}$. Production levels 2 to 4 provide a fixed number of parts with $|M_2| = 25$, $|M_3| = 50$ and $|M_4| = 75$, which should reflect typical relative differences between preceeding production processes. Part coefficients of these lower level outputs are randomly selcted out of a set of given intervals, which Kubiak et al. vary during the computational evaluation. Since they find that these variations have no impact on solution times, we will randomly select part coefficients out of a uniform distribution over the same interval $[0, 20]$. Instead we will vary the number of production cycles $T \in \{15, 20, 25, 30\}$, in order to better evaluate the performance with regard to the actual problem size. The demands for each product are also randomly determined

		DP		*EDP*	
$\|P\|$	T	time	states	time	states
8	15	0.17	3,568	0.10	2,375
		[0.13 0.20]	[2,880 4,374]	[0.08 0.13]	[1,898 2,937]
8	20	0.89	16,384	0.48	9,309
		[0.66 1.25]	[12,960 20,736]	[0.36 0.61]	[7,326 11,826]
8	25	3.22	57055	1.88	34,921
		[2.69 4.58]	[48,000 76,800]	[1.55 2.67]	[29,342 47,365]
8	30	10.90	171,023	5.80	93,814
		[7.80 14.86]	[129,600 225,000]	[4.19 7.88]	[70,926 123,887]
10	15	0.44	6,874	0.27	4,630
		[0.30 0.52]	[4,608 7,776]	[0.20 0.31]	[3,015 5,268]
10	20	2.75	37,901	1.53	21,670
		[1.77 3.52]	[25,920 46,656]	[1.00 1.94]	[14,700 26,768]
10	25	14.43	168,579	8.35	104,413
		[9.63 18.19]	[120,960 207,360]	[5.63 10.50]	[74,222 129,091]
10	30	58.88	555,408	30.62	306,484
		[37.28 95.50]	[381,024 777,600]	[19.05 48.42]	[209,312 430,344]
12	15	1.15	13,517	0.71	9,238
		[1.03 1.23]	[12,288 13,824]	[0.63 0.78]	[8,355 9,459]
12	20	8.10	81,339	4.38	46,802
		[4.69 11.22]	[51,200 104,976]	[2.53 5.95]	[29,209 60,657]
12	25	59.19	437,933	33.93	274,257
		[42.23 92.13]	[331,776 559,872]	[23.63 51.45]	[206,309 352,439]
12	30	-	-	199.41	968,840
		[218.25 -]	[1,244,160 -]	[108.97 296.19]	[688,371 1,298,234]

Table 4: Results of *DP* and *EDP* in first experiment

observing that the total demand for products is equal to the number of production cycles available, i.e., $\sum_{p \in P} D_p = T$.

In a first experiment we seek to investigate whether the algorithmic extension results in a considerable improvement in solution times. We will thus not use additional filtering methods which could bias the results. Note that the performance in this experiment hardly depends on the employed objective function, since all states need to be generated in any case and the evaluation of different objectives usually takes very similar computational effort. The results are thus merely presented for the sum of absolute deviations leveled over time. Algorithms *DP* and *EDP* were both implemented in C# and run on a Pentium IV, 1800 MHz PC, with 512 MB of memory and an upper limit on solution time of 300 cpu seconds was imposed. Table 4 displays minimum, average and maximum solution times and the number of evaluated states $\left({}_{[min.}\overset{avg.}{}{}_{max.]}\right)$ for each problem instance.

As an obvious and expected result the average solution times and the number of evaluated states increase overproportionally with the number of products and production cycles for both algorithms. In comparison, *EDP* clearly performs better than *DP* and is able to solve all instances to optimality within 300 cpu seconds. *DP* only solves one out of 10 replications for $|P| = 12$ and $T = 30$, so that merely minimum times and states are provided for this size.

We further find that solution times and the number of states vary considerably per problem size for both algorithms. This is not surprising, however, since due to (33) the total number of states depends on the actual demand values for products, which can differ among replications of the same size. Interestingly, the relative differences in the number of states between EDP and DP are rather constant per size. It further generally increases for larger values of T, but is considerably higher for even values of T, since uneven values of T require the generation of an additional stage $\lfloor \frac{T}{2} \rfloor + 1$.

Finally, the relative difference in computational time is consistently higher than the actual reduction in states. This finding might seem somewhat surprising on first look, since all algorithmic operations are performed per state only. However, this phenomenon can be explained on the basis of two considerations. First and foremost, the actual evaluation of nodes at stage $\lceil \frac{T}{2} \rceil$ will require less computational time for EDP than for DP. While EDP searches for a single complementary node for each element of this stage, DP needs to generate and consolidate all successive nodes for each element, which will typically take more computational effort. Secondly, the computer implementation might cause additional overhead in the course of the optimization, since memory needs to be frequently allocated, released and reallocated, which tends to consume more time the longer the procedure runs.

We can conclude from this experiment that while the maximum relative difference in the number of states is strictly bounded from above by 2, the difference in computational time can be significantly higher and actually exceed a factor of 2.

The considered setting is nevertheless somewhat stylized, since there is no apparent reason why we would choose not to employ additional filtering techniques to speed up the procedure. If an upper bound is able to fathom nodes early in the search, this can lead to a massive reduction of evaluated nodes at later stages. We can thus conjecture that the traditional DP approach will benefit more from such an upper bound filter as compared to EDP. In a second experiment we will therefore determine an upper bound prior to the exact solution, which is hence used to fathom nodes whose local lower bound is not lower than this upper bound. We consider two simple upper bound methods, a One-Stage heuristic, which always selects and assigns the product model p at slot t that minimizes the total deviations at this slot, and a Two-Stage heuristic, which chooses the model p at slot t that minimizes total deviations for slots t and $t+1$ (see [10]). Since their solution times are insignificant, both heuristics are run for each instance and the best objective value obtained is passed as an upper bound to the exact solution procedures.

Lower bounds are determined as described in Section 3.1. Since the tightness of such a lower bound can depend on the employed objective function, we will further consider the following four well-known objectives:

1. [SAD]: $H(F(Dev(\cdot))) = \sum_{t=1}^{T} \sum_{k \in K} \sum_{m \in M_k} |\cdot|$
2. [SSD]: $H(F(Dev(\cdot))) = \sum_{t=1}^{T} \sum_{k \in K} \sum_{m \in M_k} (\cdot)^2$
3. [MAD]: $H(F(Dev(\cdot))) = \text{Max}_{\ t,k,m}\{|\cdot|\}$

4. [MSD]: $H(F(Dev(\cdot))) = \text{Max}_{\ t,k,m}\{(\cdot)^2\}$

where deviations are leveled over time in all cases. 10 replications are computed for each problem size and objective function, so that in total 480 problem instances are solved to optimality. Tables 5 and 6 summarize the results of the second experiment for both algorithms separately.

Apparently, solution times and the number of evaluated states depend heavily on the considered objective function for both algorithms. While all instances of MAD and MSD are solved in less than a second on average, SSD and especially SAD take much more computational effort. This massive difference can be directly explained by the effectiveness of the employed upper bound filters. If deviations are summed up over time, the local lower bound of a node at an early stage of the DP approach will have hardly any predictive power with regard to the final objective of a solution. As a consequence, nodes can only be fathomed towards the end of the procedure. If maximum deviations per cycle are minimized, the upper bound can be used to exclude undesirable assignments already at early stages, which in turn results to a massive performance increase. In fact the performance for MAD and MSD is almost identical, which suggests that the bounding rules are tight for both objectives. In contrast to that the performance of SAD and SSD still differs considerably. This can be explained by the larger variations in the sum of squared deviations, which allow an earlier identification of undesirable assignments. Especially the DP procedure can exploit this in the later stages of its graph, while EDP does not profit as much. Therefore the relative improvement is the smallest for SSD. It nevertheless holds for all objectives, that the relative improvement tends to be the higher the larger the problem size.

Overall it can be stated that EDP performs significantly better for all instances and that the absolute improvement is especially large whenever deviations are summed over time. This is also underlined by a comparison to the results of Miltenburg [4], who introduces a "Reaching" Dynamic Programming (RDP) algorithm as a variation of the DP procedure employing a specific list enumeration method in order to identify and consolidate the nodes at each stage. In his work performance results for SAD and SSD are only presented for a comparatively small instance of Bautista et al. [9] ($|P| = 5$, $T = 20$, $|K| = 1$, $|M_1| = 4$), but the difference is remarkable. While the implementation of RDP with an upper bound of the Two-Stage heuristic run on a comparable system (Notebook with 1.8 GHz processor and 1 GB RAM) optimized SAD in 144.94 and SSD in 45.22 seconds, EDP takes only 2.18 seconds for SAD and 0.68 for SSD, respectively. Although a closer investigation is necessary to analyse this large discrepancy, it seems to be fair to say that EDP constitutes the best known exact solution method for symmetric versions of the ORV problem.

5 Conclusion

In this work we investigated the structure of solution sequences of the well-known ORV problem and identified symmetries which were subsequently exploited to strengthen exact solution methods. The findings are especially interesting since they are equally valid

		SAD		SSD		MAD		MSD	
$\|P\|$	T	times	states	times	states	times	states	times	states
8	15	0.13	3,010	0.10	2,608	0.00	277	0.01	277
		[0.09 0.17]	[2,361 3,938]	[0.08 0.16]	[2,036 3,574]	[0.00 0.02]	[71 598]	[0.00 0.03]	[71 598]
8	20	0.48	10,386	0.29	7,573	0.02	539	0.02	539
		[0.33 0.63]	[7,699 12,910]	[0.20 0.42]	[5,435 10,061]	[0.00 0.05]	[71 1,212]	[0.00 0.05]	[71 1,212]
8	25	1.09	24,954	0.56	15,074	0.02	562	0.02	562
		[0.83 1.56]	[19,867 34,541]	[0.42 0.80]	[11,986 20,823]	[0.00 0.03]	[85 1,037]	[0.00 0.05]	[85 1,037]
8	30	2.41	53,007	1.05	27,492	0.02	681	0.02	681
		[1.61 3.05]	[37,527 64,761]	[0.69 1.34]	[18,679 33,763]	[0.00 0.05]	[145 1,193]	[0.00 0.05]	[145 1,193]
10	15	0.37	6,455	0.33	5,955	0.03	757	0.03	757
		[0.23 0.44]	[4,382 7,369]	[0.22 0.36]	[4,117 6,782]	[0.02 0.06]	[311 1,419]	[0.02 0.06]	[311 1,419]
10	20	1.85	29,658	1.23	22,314	0.04	878	0.03	878
		[1.22 2.44]	[20,590 37,401]	[0.88 1.61]	[15,995 28,213]	[0.00 0.06]	[308 1,379]	[0.02 0.05]	[308 1,379]
10	25	7.12	103,492	3.64	63,302	0.07	1,629	0.07	1,629
		[4.36 9.47]	[68,607 132,766]	[2.08 5.52]	[39,337 90,393]	[0.02 0.17]	[324 4,127]	[0.02 0.17]	[324 4,127]
10	30	18.45	249,167	7.00	118,968	0.06	1,404	0.06	1,404
		[13.52 27.64]	[191,806 349,284]	[5.47 9.58]	[95,862 156,554]	[0.02 0.14]	[420 3,621]	[0.02 0.14]	[420 3,621]
12	15	1.11	13,372	1.05	13,105	0.09	1,775	0.08	1,775
		[1.00 1.22]	[12,175 13,717]	[0.98 1.09]	[12,061 13,540]	[0.05 0.14]	[1,148 2,725]	[0.05 0.14]	[1,148 2,725]
12	20	6.57	73,315	4.78	60,339	0.09	1,876	0.09	1,876
		[4.11 9.47]	[48,257 97,196]	[3.38 6.69]	[43,073 81,224]	[0.02 0.25]	[560 5,221]	[0.03 0.23]	[560 5,221]
12	25	38.01	336,363	19.03	218,049	0.13	3,173	0.14	3,173
		[27.39 61.00]	[260,187 449,548]	[14.56 29.22]	[173,636 306,171]	[0.05 0.34]	[1,213 7,653]	[0.05 0.34]	[1,213 7,653]
12	30	166.14	1,045,921	51.14	516,682	0.21	4,561	0.20	4,561
		[101.45 258.94]	[769,052 1,408,917]	[34.48 73.98]	[379,846 662,528]	[0.06 0.59]	[1,315 12,688]	[0.05 0.59]	[1,315 12,688]

Table 5: Results of DP with Heuristic Upper Bound

		SAD		SSD		MAD		MSD	
$\|P\|$	T	times	states	times	states	times	states	times	states
8	15	0.09	2,287	0.08	2,046	0.00	193	0.00	193
		[0.06 0.13]	[1,791 2,930]	[0.05 0.17]	[1,555 2,804]	[0.00 0.02]	[71 380]	[0.00 0.02]	[71 380]
8	20	0.35	7,835	0.23	5,806	0.01	365	0.01	365
		[0.25 0.47]	[5,909 10,104]	[0.16 0.30]	[4,153 7,483]	[0.00 0.03]	[71 739]	[0.00 0.02]	[71 739]
8	25	0.89	20,576	0.45	12,168	0.01	349	0.01	349
		[0.67 1.30]	[16,303 28,767]	[0.34 0.63]	[9,709 16,679]	[0.00 0.02]	[85 600]	[0.00 0.02]	[85 600]
8	30	1.98	42,844	0.85	21,704	0.02	399	0.02	399
		[1.34 2.48]	[30,258 51,887]	[0.56 1.11]	[14,896 26,403]	[0.00 0.05]	[145 696]	[0.00 0.05]	[145 696]
10	15	0.25	4,623	0.23	4,476	0.02	562	0.03	562
		[0.16 0.30]	[2,978 5,268]	[0.14 0.28]	[2,869 5,177]	[0.00 0.05]	[225 938]	[0.02 0.05]	[225 938]
10	20	1.30	20,790	0.95	16,769	0.02	631	0.03	631
		[0.83 1.67]	[13,772 26,239]	[0.63 1.25]	[11,358 21,510]	[0.00 0.03]	[308 1,061]	[0.02 0.05]	[308 1,061]
10	25	5.68	83,246	3.03	52,126	0.04	998	0.04	998
		[3.55 7.69]	[55,813 107,570]	[1.75 4.64]	[32,674 74,038]	[0.00 0.08]	[324 2,373]	[0.02 0.09]	[324 2,373]
10	30	14.76	197,872	5.71	96,403	0.03	888	0.03	888
		[10.61 22.72]	[146,724 282,160]	[4.53 8.14]	[77,410 129,909]	[0.00 0.09]	[344 2,323]	[0.02 0.09]	[344 2,323]
12	15	0.68	9,238	0.69	9,237	0.06	1,392	0.07	1,392
		[0.59 0.78]	[8,355 9,459]	[0.59 0.83]	[8,352 9,459]	[0.05 0.09]	[1,017 2,021]	[0.05 0.09]	[1,017 2,021]
12	20	4.12	46,734	3.47	42,928	0.07	1,442	0.06	1,442
		[2.39 5.72]	[29,185 60,654]	[2.13 4.97]	[27,308 58,220]	[0.02 0.16]	[470 3,584]	[0.02 0.16]	[470 3,584]
12	25	29.01	255,661	15.58	178,754	0.09	2,053	0.09	2,053
		[20.20 47.42]	[192,259 340,492]	[11.75 24.06]	[138,500 252,184]	[0.03 0.20]	[657 4,466]	[0.02 0.20]	[657 4,466]
12	30	125.54	794,922	41.73	419,704	0.13	2,898	0.13	2,898
		[75.56 204.67]	[566,714 1,089,500]	[27.66 60.22]	[303,548 557,028]	[0.05 0.36]	[1,120 7,925]	[0.05 0.36]	[1,120 7,925]

Table 6: Results of *EDP* with Heuristic Upper Bound

for a wide range of objective functions and thus include the vast majority of multi-level, just-in-time scheduling problems discussed in the literature. They further deepen the understanding of the considered problem and underline the fundamental symmetric structure of level scheduling problems, which has already been exploited to solve the related Product Rate Variation problem. The experimental analysis showed that the extended algorithm constitutes the state-of-the-art exact solution method for SORV problems. In order to further improve the performance of the procedure, there is an imperative need for tight lower bounds especially for problems where deviations are aggregated as the sum over time. Progress in this direction is likely to speed up the search considerably and is thus the logical next step for future research.

References

[1] Y. Monden, Toyota Production System, third ed., Industrial Engineering and Management Press, Institute of Industrial Engineers: Norcross, GA, 1998.

[2] A. Corominas, W. Kubiak, N.M. Palli, Response time variability, Journal of Scheduling 10 (2007) 97–110.

[3] V. Lebacque, V. Jost, N. Brauner, Simultaneous optimization of classical objectives in JIT scheduling, European Journal of Operational Research 182 (2007) 29–39.

[4] J. Miltenburg, Level schedules for JIT mixed-model production lines: characteristics of the largest instances that can be solved optimally, International Journal of Production Research 45 (2007) 3555–3577.

[5] N. Boysen, M. Fliedner, A. Scholl, Level scheduling of mixed–model assembly lines under storage constraints, International Journal of Production Research (2008) DOI: 10.1080/00207540701725067.

[6] N. Boysen, M. Fliedner, A. Scholl, Sequencing mixed-model assembly lines: Survey, classification and model critique, European Journal of Operational Research (2009) 349–373.

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

[8] N. Boysen, M. Fliedner, A. Scholl, The Product Rate Variation Problem and its relevance in real world mixed-model assembly lines, European Journal of Operational Research, 192 (2009) 349-373.

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

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