Diskrete Mathematik und Optimierung

Th. Epping, W. Hochstättler, R. Nickel, P. Oertel:

Order sequencing in the automobile industry

Technical Report feu-dmo001.05

Contact: thomas.epping@ifbag.com, winfried.hochstaettler@fernuni-hagen.de, robert.nickel@fernuni-hagen.de, poertel@ford.com

FernUniversität in Hagen
Lehrgebiet Mathematik
Lehrstuhl für Diskrete Mathematik und Optimierung
D – 58084 Hagen

2000 Mathematics Subject Classification: 90B35,90C10
Keywords: Project Management and Scheduling, Integer Programming, Sequencing

Order sequencing in the automobile industry

Th. Epping

ifb AG

Cologne, Germany

Email: thomas.epping@ifbag.com

and

W. Hochstättler and R. Nickel[*]

Department of Mathematics,

FernUniversität in Hagen,

D-58084 Hagen, Germany

Email: {winfried.hochstaettler, robert.nickel}@fernuni-hagen.de

and

P. Oertel

Ford Motor Company

Cologne, Germany

Email: poertel@ford.com

Abstract

We describe a new solution approach for order sequencing which reflects the build-to-order strategy of the European automobile industry. Therefore, it has to cope with a variety of customer orders that change daily.

Each order consists of a set of commodities which are implemented while the order sequence passes through a body shop, a paint shop, and an assembly shop. Existing solution approaches usually focus on the computation of an order sequence optimized to fit the needs of the assembly shop where the most significant savings can be achieved. We propose a solution approach that generalizes known approaches and covers the complete production process.

The resulting rule-based solution algorithm has been implemented in cooperation with Ford Motor Company and is currently successfully used in all plants across Europe. We illustrate its efficiency by computational results on real-world data.

[*]corresponding author (phone:+49-2331-9872658, fax:+49-02331-9874269)

1 Motivation

There exists a huge variety of order sequencing problems in practice and, therefore, just as much literature on this topic. However, each problem variant has its specific background that often prevents a generalization of ideas and algorithms and successful solution approaches are often kept as company secrets (see [1] for an overview of accessible reports).

Frequently, solution approaches to an automobile order sequencing problem are based on local search strategies (see [4, 6]). Other approaches employ constraint programming (see [1]), while a classical approach is the well-known goal chasing algorithm (see [5]). The basic idea, however, which all approaches have in common is the support of just-in-time production. The automobile industry is a leading user of this concept (see [7]) that focuses on a smooth production flow.

Although plants have a more complex structure, known approaches concentrate on a single part of the production process only, which is usually the assembly shop. Figure 1 shows a sketch of a typical automobile production plant.

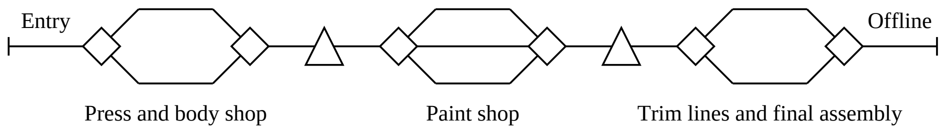


Figure 1: Sketch of an automobile production plant. Squares denote split points of the production sequence, while triangles denote interim storage systems.

We present an extension of these known models by dealing with the entire production plant and a larger variety of requirements. In particular, we provide means for the reduction of the number of color changes within the enamel booths of the paint shop to respect the required properties of a sequence. The restrictions are formulated in terms of rules and a production sequence that violates as few of these rules as possible is computed. Note that, as the number of rules is usually large, it is in general not possible to arrange the orders in a production sequence that incurs no violation of rules at all.

2 Framework and basic concepts

The production process is essentially a probabilistic process. The main reason for this are lots of eventualities for manufacturing errors which make the production process difficult to control. Our major goal is to develop concepts that model the production process in a deterministic and thereby controllable way. These concepts include a suitable representation of the plant, a sensible handling of manufacturing errors, and a suitable set of rules for the evaluation of the quality of a production sequence. For convenience, Appendix A lists the meaning of the most frequently used abbreviations.

2.1 Orders, commodities, and combinations

We assume that each customer request is specified by a set of commodities.

Definition 2.1 *Let $\mathcal{C}$ be a finite set. We refer to the elements of $\mathcal{C}$ as* commodities. *A subset of $\mathcal{C}$ is called an* order *and the set of orders for a specific production day is denoted by $\mathcal{O}$. We assume that $|\mathcal{O}| = n$ and $|\mathcal{C}| = m$.*

Each order $o \in \mathcal{O}$ can be interpreted as a vector $o \in \{0, 1\}^m$, where

$$o_i := \begin{cases} 1 & \text{if } o \text{ has commodity } i \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, \ldots, m$. For convenience, we say that $c \in o$ or *o has commodity c* if $o_c = 1$.

The interpretation of orders as binary vectors assigns identical vectors to orders with identical commodity sets and yields a straightforward partition of $\mathcal{O}$.

Definition 2.2 *Let $C \subseteq \mathcal{C}$ be a not necessarily non-empty set of commodities. We call two orders $a, b \in \mathcal{O}$ equivalent, if and only if $a_c = b_c$ for all $c \in C$. For a commodity set $C \subseteq \mathcal{C}$ we denote by $\mathcal{O}(C)$ the set of orders that have the commodities C. The corresponding partition is denoted by*

$$\mathcal{P}(C) := \dot{\bigcup_{C' \subseteq C}} \mathcal{O}(C').$$

We call each set $O := \mathcal{O}(C') \in \mathcal{P}(C)$ a commodity combination. *Thus, orders of the same commodity combination correspond with respect to all commodities of C. We will write $c \in O = \mathcal{O}(C)$ if $c \in C$.*

2.2 Plant layout

A plant consists of consecutive production shops and each production shop is split up into several lines which we call *zones*. Each zone, again, is split into *stations*. At least one commodity of some order is installed at each zone and exactly one commodity is applied at each station. Stations, however, are ignored in our model of the plant layout as the production process within a zone can not be influenced in any way.

Definition 2.3 *We denote the set of zones of the plant by $\mathcal{Z}$. The first zone of the plant is called* master zone. *We denote the master zone by $z_m \in \mathcal{Z}$ and call the order sequence in the master zone (which is the production sequence that has to be computed) the* master sequence. *Each position in the master sequence is an integer $p \in \{1, \ldots, n\}$.*

A suitable representation of a plant is given by a directed graph $G = (\mathcal{Z}, A)$ which is a directed path with a "multiplied" stable set of vertices (see [3]). Each zone corresponds to a vertex and arcs correspond to the production flow between zones (see Figure 2).

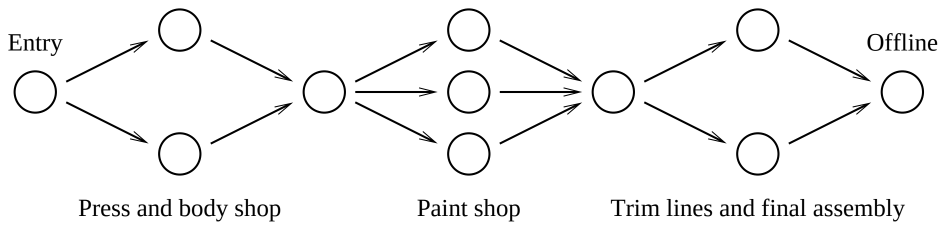


Figure 2: Representation of the plant of Figure 1 as a directed graph.

We assume that $\mathcal{Z}$ contains exactly one zone without predecessor (the master zone) and exactly one zone without successor (the last zone of the plant). If two zones have the same in-neighbor, they are called *parallel*. A zone which is not parallel is called *non-parallel*. Parallel zones of a plant may differ in their length and speed. Therefore, each zone may have another proportion of production, which we call the *quota* of the zone.

Definition 2.4 *We denote the* quota *of a zone $z \in \mathcal{Z}$ by*

$$q(z) \in \{k/n : 1 \leq k \leq n\}$$

and extend our plant representation to a vertex-weighted directed graph $G = (\mathcal{Z}, A; p)$, where $q(z) = 1$ if z is non-parallel. For any maximal set Z of parallel zones $\sum_{z \in Z} q(z) = 1$ must hold.

Zone quotas of parallel zones are used for the generation of deterministic patterns for the storage and retrieval of orders in the zones. Given a maximal set Z of parallel zones, a *pattern* is a sequence of integers $p_i \geq 1$, where each entry p_i is interpreted as the storage resp. retrieval of p_i orders on and from zone $i \bmod |Z|$. Consequently, we have $\sum_i p_i = n$ for each pattern.

Example 2.1 *Let $Z = \{z_1, z_2, z_3\}$ be a maximal set of parallel zones. Both the pattern $P_1 = (1, 1, 1, 1, 1, 2, \ldots)$ and the pattern $P_2 = (2, 2, 3, 2, 2, 3, \ldots)$ describe the storage and retrieval of orders for zone quotas $q(z_1) = q(z_2) = 2/7$ and $q(z_3) = 3/7$. The proportion of production is therefore $2n/7$ for both z_1 and z_2, and $3n/7$ for z_3. In practice, P_1 is preferred over P_2 as it distributes orders more evenly on the zones and supports a smooth production flow in a better way.*

2.3 Commodity delays

Following our goal to model the production process in a deterministic way, we emphasize a correlation between commodities and manufacturing errors, i. e. some commodities are more susceptible to manufacturing errors than others. Each manufacturing error leads to a production re-run that delays an order within the production sequence with respect to its original position. We model (and anticipate) the occurrence of manufacturing errors by the introduction of *commodity delays*. A commodity delay can be imposed on each zone of the plant.

Definition 2.5 *A* commodity delay *is a mapping $d : \mathcal{C} \times \mathcal{Z} \to \mathbb{N}_{\geq 0}$, and the delay of a commodity $c \in \mathcal{C}$ in a zone $z \in \mathcal{Z}$ is denoted by $d(c, z)$.*

Note that commodity delays may cause gaps in the order sequence of zones, which can be represented by dummy orders that act as wild-cards and can, for example, be filled by orders of preceding or succeeding production days.

To our knowledge, there does not exist any other solution approach for an order sequencing problem that employs commodity delays.

2.4 Sequence index routing

Zones and patterns as introduced in Section 2.2 are used to compute a routing of master sequence order positions through the plant. The routing assigns to each order position i of the master sequence a set of zones through which an order passes if sequenced to position i. Likewise, the routing assigns to each zone z of the plant a set of order positions that pass through z. Note that this yields either a permutation (for non-parallel zones) or a partition into sequences (for parallel zones) of the order positions in the master sequence.

Definition 2.6 *A* routing *is a mapping $R : \{1, \dots, n\} \to 2^{\mathcal{Z}}$ that satisfies $|R(i) \cap Z| = 1$ for each maximal set of parallel zones and $z \in R(i)$ for any non-parallel zone z. $R(i) \subseteq \mathcal{Z}$ denotes the set of zones through which an order passes if sequenced to position i of the master sequence, and $R^{-1}(z) := \{i \in \{1, \dots, n\} : z \in R(i)\}$ denotes the set of master sequence order positions that pass through a zone $z \in \mathcal{Z}$.*

We assume that the set $R(i)$ is not affected by commodity delays, i. e. commodity delays do not change the set of zones through which an order passes. Consequently the delay of an order (induced by the delay of its commodities) in a zone can be calculated as the sum of its maximal commodity delays in preceding zones. Note that zone quotas have to be respected and that two different orders can be delayed to the same position in a zone.

Definition 2.7 *Let $o \in \mathcal{O}$ be sequenced to position i of the master sequence. We denote by $P(i, z_0)$ the position of the master sequence position i in zone z_0 as given by the routing. Then with increasing i the position of o in any zone z_0 succeeding the master zone z_m is given by*

$$P(o, i, z_0) := P(i, z_0) + \sum_{z \in Z_p} \lfloor \max_{c \in o} \{d(c, z)\} \cdot q(z) \rceil + \Delta,$$

where $\Delta \in \mathbb{N}_{\geq 0}$ denotes a shift to the next position not already used by an order with smaller master sequence index. $Z_p \subseteq R(i)$ denotes the set of zones preceding z_0 and $\lfloor x \rceil$ is the value of x rounded to an integer.

2.5 Sequence slots

Despite of the impossibility to control the production process entirely, interim storage systems can provide a production sequence that is at least similar to the intended one. However, even a small perturbation of the original production sequence generally leads to a significant increase of the number of color changes in the paint shop. It is not sensible to focus on large subsequences of orders with the same enamel color before they reach the paint shop. Instead, we try to cope with unavoidable perturbations of the original production sequence by clustering enamel colors. The idea is to increase the probability that identical colors are sequenced close to each other by allowing a specified color to occur in specified intervalls of the sequence only. This in turn increases the probability that, by the use of interim storage systems prior and next to the paint shop for short-term order interchanges, larger blocks of the same color can be built.

The master zone and, with respect to zone quotas, all other zones of the plant are therefore divided into *slots* of equal size, i. e. position i of the master sequence is contained in slot s if and only if position i is contained in slot s of any zone $z \in R(i)$. Note, however, that sufficiently large commodity delays may cause an order sequenced to slot s of the master zone to appear in a slot $s' > s$ of a succeeding zone.

The introduction of slots is one of the key concepts for enamel color clustering, as enamel colors can be advised to appear in a specified number of slots only. Furthermore, the master sequence is computed slot by slot which significantly decreases the running time of our solution algorithm. In practice, each slot usually corresponds to a working shift.

We denote the number of slots of the master zone by S and assume that the size $\frac{n}{S}$ of each slot is an integer.

2.6 Rules

Each zone of the plant accepts rules. Each rule must state the zone, the commodity it applies to, and a priority. The priority of a rule is an integer $p \in \{1, \dots, 10\}$, where $p = 1$ denotes the highest and $p = 10$ the lowest priority.

Each rule specifies a constraint on a zone of the plant and the quality of a sequencing of an order o to a position i of the master sequence is rated by the number of resulting rule breaks. Note that, as we use a deterministic routing of order positions through the plant, it is in principle possible to evaluate each rule in the master zone instead of the zone it applies to.

The suggested distinction between paint shop and assembly shop rules is not strict. Rather, any rule can apply to other shops as well. For a given commodity $c \in \mathcal{C}$ and a zone $z \in \mathcal{Z}$ any of the following rules can be defined:

Paint shop rules	
Clustering:	c is allowed to occur in a specified number of slots only.

Assembly shop rules	
Banning:	c is not allowed to occur in a specified interval of the zone.
Ratio:	At most x out of y consecutive orders should have commodity c.
Spacing:	Orders with commodity c should occur with a minimum distance.
Grouping:	Orders with commodity c should occur in groups of given minimum and maximum size, where two groups must keep a specified minimum distance.
Spreading:	Orders with commodity c should be evenly spread.

Table 1: Possible rules for a commodity c on zone z

Most of the rules presented above are typical of automobile production. While they can be used to support a smooth production flow and just-in-time production, they provide opportunities to respect particular technical restrictions of plants as well. To our knowledge, the clustering rule is a new supplement in this context.

We may assume that the commodity set $\mathcal{C}$ contains only commodities for which at least one rule holds. Note that there is no emphasis of any of the rules to be an objective function, i. e. a high-quality order sequence is any order sequence that incurs a minimal violation of these rules with respect to their priorities.

In the following sections we present a two-stage solution approach that covers the entire rule set for any zone, no matter which shop it belongs to. The solution approach divides into two main stages:

1. Compute an order clustering that supports the subsequent computation of a master sequence slot by slot.

2. Compute a master sequence that violates as few rules as possible.

The computation of an order clustering will be considered in Section 4. At first we will deal with the construction of a master sequence, assuming that an order clustering has been computed already.

3 Master sequence construction

After each order is assigned to a slot, the master sequence is being constructed slot by slot in a straightforward greedy fashion. We iterate through all sequence positions $i = 1, \dots, n$ of the master sequence and temporarily assign each unsequenced order to the current position i. The effect on the quality of the master sequence arising from the current assignment is rated by an evaluation of the rule set with respect to the commodities of the order (see Section 3.1). The order whose assignment is rated best is sequenced to position i and we proceed to position $i + 1$. Note that it is sufficient to evaluate rules only for one order of each commodity combination and that commodity combinations may change from slot to slot.

3.1 Rule evaluation

Due to the deterministic model of the production process described by the framework in Section 2, the master sequence determines the order sequence in all zones of the plant. Whenever an order is assigned to a position of the master sequence, we can therefore check the quality of the assignment with respect to the given rule set. Note that we rate only a single assignment rather than a set of assignments.

Definition 3.1 *Let $o \in \mathcal{O}$ be an order sequenced to position i of the master sequence. The resulting* numbers of rule breaches *are collected in a vector $b = b(o, i) \in \mathbb{N}^{10}_{\geq 0}$, where $b_p = r$ if and only if the sequencing of order o to position i of the master sequence causes r rules of priority p to be broken. Likewise, a vector $v = v(o, i) \in \mathbb{R}^{10}$ of* penalty values *is created, where $v_p = r$ if and only if the sequencing of order o to position i of the master sequence causes the resulting penalty values for all rules of priority p to sum up to the value r.*

Each rule can be broken at most once for each commodity. For each broken rule of priority p, either the number of rule breaches b_p or the penalty value v_p or both are updated.

In the following, we specify all rules in more detail and describe how to obtain the number of rule breaches and penalty values resulting from broken rules, assuming that an order $o \in \mathcal{O}$ is sequenced to position $i \in \{1, \dots, n\}$ in slot $s \in \{1, \dots, S\}$ of the master sequence and that $c \in \mathcal{C}$ resp. $z \in \mathcal{Z}$ denotes the commodity resp. the zone affected by a rule of priority p. Recall the notation introduced in Section 2.6 and that $P(o, i, z)$ denotes the position of o in z (see Definition 2.7). We express rule breaks and penalty values in terms of commodity distances to keep them comparable.

The clustering rule is not evaluated, as the order clustering computed in Section 4 is not allowed to be changed. The rule specification is therefore moved to Section 4.

Banning rule Commodities may be banned from parts of zones. Therefore, it is possible to specify at most one interval, i. e. a set

$$B(c, z) = \{l_{c,z}, \dots, u_{c,z}\} \subseteq \{1, \dots, n\}$$

of consecutive integers, for any commodity $c \in \mathcal{C}$ and any zone $z \in \mathcal{Z}$, meaning that an order with commodity c should not occur at any position $i \in B(c, z)$ in zone z. Using the deterministic sequence index routing described in Section 2.4, we can deduce banned order sequence positions for each slot of the master zone. If $c \in o$ and $l_{c,z} \leq P(o, i, z) \leq u_{c,z}$, the number of rule breaches b_p is increased by 1 and the penalty value is increased by

$$v_p = 1 + \min\{P(o, i, z) - l_{c,z}, u_{c,z} - P(o, i, z)\}.$$

Ratio rule The maximal number of occurrences of a commodity $c \in \mathcal{C}$ within a consecutive subsequence of the production sequence can be specified by a rational number $R(c, z) = x/y \in \mathbb{Q}_{\geq 0}$ which means that at most x occurrences of c are allowed within any consecutive subsequence of length y of the order sequence in zone z. Note that $R(c, z) = x/y$ and $R(c, z) = (kx)/(ky)$ for $k \in \mathbb{N}_{>0}$ do not formulate identical constraints in general. Let

$$x' := |\{j \in \{0, \dots, y-1\} : P(o, i, z) - j \text{ holds an order } o' \text{ with } c \in o'\}|$$

denote the number of orders within a consecutive subsequence of orders of length y in z that have commodity c. If $x' > x$, the number of rule breaches b_p is increased by 1 and the penalty value is increased by $v_p = (x' - x)^2$.

Spacing rule A minimal distance between orders with a commodity $c \in \mathcal{C}$ can be specified by an integer $D(c, z) = d \leq 1$ which means that between any two orders in the order sequence in zone $z \in \mathcal{Z}$ that have commodity c there should be at least d orders that do not have commodity c. Let

$$d := \min\{j : P(o, i, z) - j \text{ holds an order } o' \text{ with } c \in o'\}$$

denote the minimal distance of o to an order that has commodity c. If $c \in o$ and $d < D(c,z)$, the number of rule breaches b_p is increased by 1 and the penalty value is increased by $v_p = (D(c,z) - d)^2$.

Grouping rule Integers $g_{\min}(c,z) \geq 2$ and $g_{\max}(c,z) \geq g_{\min}(c,z)$ can be used to specify that a commodity $c \in \mathcal{C}$ should occur in zone z in a consecutive subsequence which has a minimal length $g_{\min}(c,z)$ and a maximal length $g_{\max}(c,z)$. Moreover, it is possible to specify a minimal distance $D(c,z) \geq 1$ between consecutive subsequences of orders that contain c. Let

$$g := \max\{j : P(o,i,z) - k \text{ holds an order } o' \text{ with } c \in o' \text{ for all } k = 1, \ldots, j\}$$

denote the maximal length of a consecutive sequence of orders in z that have commodity c, starting from $P(o,i,z) - 1$. The rule is assumed to be broken if

$$\begin{array}{rcl} g+1 & > & g_{\max}(c,z) \text{ (for } g > 0 \text{ and } c \in o) \\ g & < & g_{\min}(c,z) \text{ (for } g > 0 \text{ and } c \notin o) \\ d & < & D(c,z) \text{ (for } g = 0 \text{ and } c \in o) \end{array}$$

where

$$d := \min\{j : P(o,i,z) - j \text{ holds an order } o' \text{ with } c \in o'\}$$

denotes the minimal distance of o in z to an order that has commodity c.

For any rule break, the number of rule breaches b_p remains unchanged and the penalty value is increased by $v_p = (g + 1 - g_{\max}(c,z))^2$, or $v_p = (g - g_{\min}(c,z))^2$, or $v_p = (D(c,z) - d)^2$, respectively. Note that the grouping rule is the only rule which is evaluated regardless whether $c \in o$ or $c \notin o$.

Spreading rule An even spreading of a commodity $c \in \mathcal{C}$ within the order sequence of a zone $z \in \mathcal{Z}$ can be specified. Evenly spread commodities are essential for a smooth production flow. Given a clustering of orders, we can easily deduce the number of occurrences $s(c)$ of any commodity $c \in \mathcal{C}$ within each slot $s \in \{1, \ldots, S\}$ of the master zone, which we use to prepare the evaluation of spreading rules. Recall that we may always evaluate spreading rules in the master zone instead of the zone specified by the spreading rule.

Definition 3.2 *We denote the* desired average spreading distance *for a commodity $c \in \mathcal{C}_{\mathrm{Spr}}$ in the master zone by $A(c, z_m; s) := n/(S \cdot s(c))$ for each slot $s = 1, \ldots, S$.*

Note that $A(c, z_m; s)$ is in general not an integer. Therefore, both $\lfloor A(c, z_m; s) \rfloor$ and $\lceil A(c, z_m; s) \rceil$ are the actually desired spreading distances. Let

$$p := |\{j : P(o, i, z_m) - j \text{ is contained in slot } s\}|$$

denote the number of orders in slot s up to position i and let $s_e := p/A(c, z_m; s)$ denote the expected number of orders with commodity c in slot s up to position i. Furthermore, let

$$s_f := |\{j : P(o, i, z_m) - j \text{ holds an order } o' \text{ in slot } s \text{ with } c \in o'\}|$$

denote the number of orders with commodity c in slot s up to position i. If $c \in o$ and $s_e \neq s_f$, the number of rule breaches b_p remains unchanged and the penalty value is increased by $v_p = s_f - s_e$.

A similar way of spreading rule evaluation is also used in the well-known goal chasing algorithm (see [5]), which is a standard algorithm for smooth production sequencing. Note that there is only little information for a sensible evaluation of the spreading rule at the start of each slot. This drawback can be overcome and the transition between slots can be improved by introducing suitable distance adjustments.

We emphasize that grouping and spreading rules do not affect the number of rule breaches because both rules are very restrictive and their violation would dominate other rules. Moreover, the intention of these rules is to tendentiously move the orders to spread positions or into groups.

Note that it is sufficient to consider only orders sequenced to positions $p \in \{1, \ldots, P(o, i, z)\}$ resp. $p \in \{1, \ldots, P(o, i, z_m)\}$ for all rule evaluations, as the master sequence is constructed order by order within each slot.

We complete the description of the master sequence construction algorithm by a description of our strategy to rate the quality of two different order assignments to the same master sequence position, using the resulting number of rule breaches and penalty values.

Definition 3.3 *Let o_1 resp. o_2 be orders sequenced to position i of the master sequence and let $b_1 = b(o_1, i)$ and $b_2 = b(o_2, i)$ resp. $v_1 = v(o_1, i)$ and $v_2 = v(o_2, i)$ denote the vector of numbers of rule breaches resp. penalty values resulting from the evaluation of all rules that affect o_1 and o_2.*

1. *If $b_1 < b_2$ with respect to lexicographical ordering, o_1 is rated better than o_2.*

2. *If $b_1 = b_2$, o_1 is rated better than o_2 if $\sum_{p=1}^{10}(11-p)(v_{1p} - v_{2p}) < 0$ holds.*

3. *If o_1 and o_2 are still rated equally, we rate o_1 better than o_2 if more rules apply to o_1 than to o_2. Otherwise, we resolve the tie between o_1 and o_2 arbitrarily.*

4 Order clustering

We now turn to the computation of an order clustering which has to be done prior to constructiong the master sequence as described in the previous section. At first we have to specify the clustering rule as it has been outlined in Section 2.6.

Clustering rule For each commodity $c \in \mathcal{C}$ can be specified by an integer $S(c,z) \geq 1$ that c should occur in at most $S(c,z)$ slots in zone $z \in \mathcal{Z}$. However, if $\mathcal{C}_{\text{Clu}} \subseteq \mathcal{C}$ is the set of commodities which are affected by a clustering rule, any order $o \in \mathcal{O}$ is allowed to have at most one commodity $c \in \mathcal{C}_{\text{Clu}}$. Furthermore, if z is a parallel zone for which a clustering rule holds, we assume that the clustering rule holds for all zones which are parallel to z. Although this rule is intended to support enamel color clustering only, it can in principle be applied to other commodities as well.

Since the master sequence is computed slot by slot, we assign each order to exactly one slot of the master sequence. Additionally, we have to focus on an even distribution of non-clustering commodities among the slots, so that the orders of each slot can be treated as a smaller copy of the original order set.

The order clustering step is divided into two parts.

1.1 Compute a preferred slot of the master zone for each order.

1.2 Assign each order to a slot of the master zone.

Rules which basically can prevent an even spreading of commodities among the slots of the master zone are clustering and banning rules. We therefore consider banning, clustering, and spreading rules during the order clustering step.

Definition 4.1 *We denote the set of commodities affected by a banning, clustering, spreading rule by $\mathcal{C}_{\mathrm{Ban}}, \mathcal{C}_{\mathrm{Clu}}, \mathcal{C}_{\mathrm{Spr}}$, respectively.*

4.1 Computation of preferred order slots

Our aim is to support an even spreading of commodities with respect to clustering and banning rules. We therefore try to distribute weighted proportions of commodity combinations representing these rules on each slot in a suitable way, i. e. the sum of weighted proportions assigned to each slot should approximate the desired slot contents as well as possible. As identical commodity combinations may appear in different slots and each commodity combination corresponds to a set of orders, this yields a set of preferred order slots for each order.

The problem of computing preferred order slots is related to a multidimensional version of the well-known bin packing problem (see [2]). In our case, however, the number of bins is fixed, the desired bin content is known, and bins are thus allowed to be overloaded.

Problem 4.1 BEST VECTOR PACKING INTO BINS PROBLEM (BVPBP)

Instance *A set B of bins, a desired bin content $w_b^* \in \mathbb{R}^d_{\geq 0}$ for each $b \in B$, a set I and a vector $w_i \in \mathbb{R}^d_{\geq 0}$ for each $i \in I$ for which $\sum_{b \in B} w_b^* = \sum_{i \in I} w_i$ holds.*

Task *Approximate each w_b^* with elements of $\{w_i : i \in I\}$ in an optimal way, i. e. find a packing $P : I \to B$ such that $\sum_{b \in B} ||w_b^* - \sum_{i \in P^{-1}(b)} w_i||^2$ is minimized.*

To our knowledge, the BVPBP is a new combinatorial problem which has not been discussed in the literature yet. Due to both a tight project schedule and running time reasons, we had to do without an exact solution algorithm for the BVPBP. We applied a greedy heuristic instead, which chooses a vector and a bin at each step so that the vector improves on the bins contents as much as possible.

An instance of the BVPBP in the context of order clustering consists of the set of slots $B = \{1, \ldots, S\}$ and vectors for the desired slot contents which respect banning rules as well as vectors whose components arise from commodity combinations which respect clustering and spreading rules. We consider the partition $\mathcal{P}(\mathcal{C}_{\text{Clu}}) = O_1\dot{\cup}\ldots\dot{\cup}O_u$ of the order set $\mathcal{O}$ and set $r := |\mathcal{C}_{\text{Spr}}|$ and introduce a vector for each commodity combination in $\mathcal{P}(\mathcal{C}_{\text{Clu}})$.

Definition 4.2 *Let $O = \mathcal{O}(C) \in \mathcal{P}(\mathcal{C}_{\text{Clu}})$. The vector $w_i = (w_{i,1}, \ldots, w_{i,r}, w_{i,r+1}) \in \mathbb{N}_{\geq 0}^{r+1}$ consists of weights defined by*

$$w_{i,j} := \begin{cases} |O_i \cap \mathcal{O}(c_j)| & \text{for } j = 1, \ldots, r \\ |O_i| & \text{for } j = r+1 \end{cases} , \quad c_j \in \mathcal{C}_{\text{Spr}}$$

As $\mathcal{P}(\mathcal{C}_{\text{Clu}})$ is a partition, it holds

$$\sum_{O_i \in \mathcal{P}(\mathcal{C}_{\text{Clu}})} w_{i,j} = |\mathcal{O}(c_j)| \text{ for } j = 1, \ldots, r.$$

Recall that if $\mathcal{O}(C) \neq \emptyset$ there is at most one commodity $c \in C \cap \mathcal{C}_{\text{Clu}}$ (see Section 2.6). If a clustering rule affects a commodity combination O_i and allows its contents to be clustered in $s > 1$ slots, the corresponding vector w_i is replaced by s new vectors with weights $w_{i,j}/s$. For example, the vector that corresponds to the commodity combination for which no clustering rule holds is always split into S vectors, i. e. into as many vectors as there are slots of the master zone.

Definition 4.3 *We denote the set of banned order sequence positions for slot s of the master zone by $B(c, z_m; s)$. The corresponding* banning factor *is given by*

$$b(c, z_m; s) := \frac{n - S \cdot |B(c, z_m; s)|}{n - |B(c, z_m)|}$$

for $s = 1, \ldots, S$ and indicates the banning proportion for each slot s of the master zone z_m.

Note that $B(c, z_m) = \bigcup_{s=1}^{S} B(c, z_m; s)$ holds. If $B(c, z_m) = \emptyset$ or each slot s has the same number $|B(c, z_m; s)|$ of banned order sequence positions, we have $b(c, z_m; s) = 1$. Otherwise, we have $0 \leq b(c, z_m; s) < 1$ if the banning in slot s is below average and $b(c, z_m; s) > 1$ if the banning in slot s is above average. In any case, $\sum_{s=1}^{S} b(c, z_m; s) = S$ holds.

Definition 4.4 *The* desired slot contents $w_s^* = (w_{s,1}^*, \dots, w_{s,r}^*, w_{s,r+1}^*) \in \mathbb{N}_{\geq 0}^{r+1}$ *of slot s of the master zone consists of weights defined by*

$$w_{s,j}^* := \begin{cases} b(c_j, z_m; s) \cdot |\mathcal{O}(c_j)|/S & \text{for } j = 1, \dots, r \\ n/S & \text{for } j = r+1 \end{cases}$$

for $s = 1, \dots, S$ and $c_j \in \mathcal{C}_{\text{Spr}}$.

Note that $\sum_{i=1}^{u} w_{i,r+1} = \sum_{s=1}^{S} w_{s,r+1}^* = n$ holds, i. e. we introduce the $(r+1)$-th component of each vector to achieve evenly occupied slots. As each vector w_i corresponds to a commodity combination $O_i \in \mathcal{P}(\mathcal{C}_{\text{Clu}})$ which in turn corresponds to a set of orders, a solution to the BVPBP, if applied to the set of slots and vectors as defined in Definition 4.2 and Definition 4.4, is an ambiguous suggestion how to assign orders to slots of the master sequence. Let $A(s)$ denote the set of vectors w_i assigned to slot s.

Definition 4.5 *We denote the set of preferred slots for a commodity combination $O_i \subseteq \mathcal{O}$ by*

$$S^*(O_i) = \{s \in \{1, \dots, S\} : w_i \in A(s)\}.$$

4.2 Assignment of orders to slots

The preferred slots of a commodity combination yield no usable assignment of orders to slots in general (as there is no guarantee that all slots are filled with the same number of orders and there may be more than one preferred slot for an order), but can be used to compute an actual assignment of orders to slots by a linear programming approach.

We consider the partition of $\mathcal{O}$ induced by the set of banning, clustering, and spreading commodities $\tilde{\mathcal{C}} := \mathcal{C}_{\text{Ban}} \cup \mathcal{C}_{\text{Clu}} \cup \mathcal{C}_{\text{Spr}}$ and assume that $\mathcal{P}(\tilde{\mathcal{C}}) = O_1 \cup \ldots \cup O_q$.

Definition 4.6 *We denote the set of commodity combinations affected by a commodity $c \in \tilde{\mathcal{C}}$ by $P_c(\tilde{\mathcal{C}}) := \{O \in \mathcal{P}(\tilde{\mathcal{C}}) : c \in O\}$.*

We introduce a variable x_{is} for each commodity combination $i = 1, \dots, q$ and each slot $s = 1, \dots, S$ that counts the number of orders in O_i to be assigned to slot s. The value of each x_{is} has to respect the following constraints and objectives.

Slot size constraint The number of orders in each slot must clearly not exceed the maximal slot size. Therefore,

$$\sum_{i=1}^{q} x_{is} = n/S$$

must hold for all slots $s = 1, \ldots, S$.

Banning constraint Recall that $B(c, z_m; s)$ denotes the set of banned sequence positions for a commodity $c \in \mathcal{C}$ from the master zone $z_m \in \mathcal{Z}$ in slot s. As we must not exceed the number of allowed sequence positions for c in the master zone z_m in slot s,

$$\sum_{O_i \in P_c(\tilde{\mathcal{C}})} x_{is} \leq n/S - |B(c, z_m; s)|$$

must hold for all slots $s = 1, \ldots, S$ and commodities $c \in \mathcal{C}_{\text{Ban}}$.

Clustering objective The occurrence of an order $o \in O_i$ within a slot $s \notin S^*(O_i)$ has to be avoided. Therefore, we demand

$$\sum_{\substack{O_i \in P_c(\tilde{\mathcal{C}}) \\ s \notin S^*(O_i)}} x_{is} \geq 0$$

for all slots $s = 1, \ldots, S$ and commodities $c \in \mathcal{C}_{\text{Clu}}$. Our objective is to reach equality for these constraints.

Spreading objective The spreading of commodities requires the consideration of banning rules. As the number of orders with a spreading commodity within each slot must therefore be scaled with the banning factor,

$$\sum_{O_i \in P_c(\tilde{\mathcal{C}})} x_{is} = \mathcal{O}(c)/S \cdot b(c, z_m; s)$$

must hold for all slots $s = 1, \ldots, S$ and commodities $c \in \mathcal{C}_{\text{Spr}}$. Our objective is to keep equality for these constraints.

We derive an integer program from these constraints and objectives by the introduction of slack variables and solve its relaxation. Note that we assume that $\mathcal{C}_{\text{Clu}} \cap \mathcal{C}_{\text{Spr}} = \emptyset$ holds, as the restrictions arising from clustering and spreading objectives are inconsistent otherwise. We solve the linear program successively for all rules from priority $p = 1$ up to priority $p = 10$ to achieve a strict separation of rules of different priorities. The first run considers only rules of priority $p = 1$ and its solution yields an initial set of constraints that are taken over to the next run. The second run includes rules of priority $p = 2$ as well and refines the solution of the first run, as the consideration of additional commodities yields a refinement of commodity combinations. Again, additional constraints are taken over to the next run, until all rules and a variety of accumulated constraints of previous runs are considered in the final run.

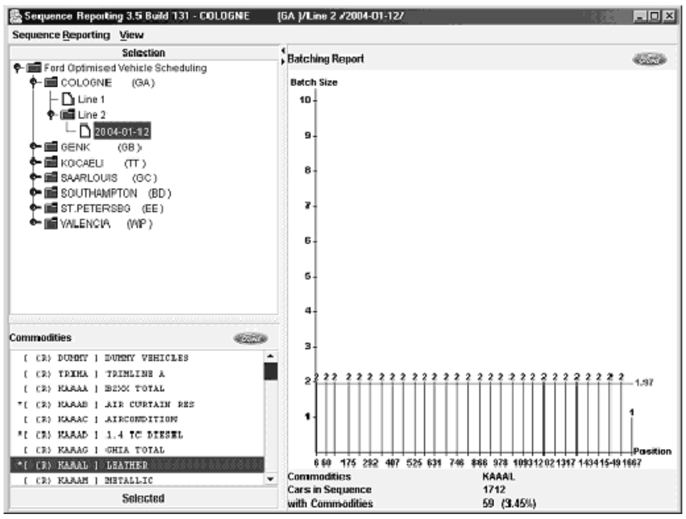
The result of these iterations is a set of values x_{is} that suggests to assign x_{is} orders of O_i to slot s. The actual assignment is in the end computed by a straightforward greedy picking algorithm that decides which orders of a commodity combination are assigned to a slot and additionally resolves the problem of rounding the values x_{is} to suitable integer values. Furthermore, it has to provide a strategy to cope with situations in which the linear program is not feasible, although we can almost always assume a reasonable input for the linear program in practice.

5 Computational results

Our solution approach is currently used for order sequencing in all automobile plants of Ford Motor Company across Europe. The implementation demands a careful balancing and monitoring of the rules. But this has led to significant increases in color batch size. The results vary depending on the order mix to be sequenced and external factors in the plants, but in one plant specifically a 50% increase in batch size on average has been achieved. Results for other European plants vary but have also been significant.

The algorithms presented in Section 3 and Section 4 are embedded in a graphical user interface provided by Ford Motor Company, which offers an easy way to specify rules for a forthcoming production day. Additionally, it includes diagnostic tools for past production days, which in particular allow the rating of the quality of the corresponding production sequences with respect to the specified rules. Figures 3 – 4 illustrate the efficiency of our approach by screen-shots of this interface.

The screen-shots are taken from different sequencer runs on real-world data for the plant in Cologne. The order set $\mathcal{O}$ contains $n = 1712$ orders. The average running time for a complete sequencer run on 1500 orders is about 3 minutes on a computer with a Sun E4500 processor with 450 MHz and 1 GB memory. Figure 3 and Figure 4 illustrate that grouping, spreading, clustering, and banning rules can be expected to work efficiently for high priorities. Figure 4(d) shows the effect of Definition 3.3, which neglects commodities with low priorities.

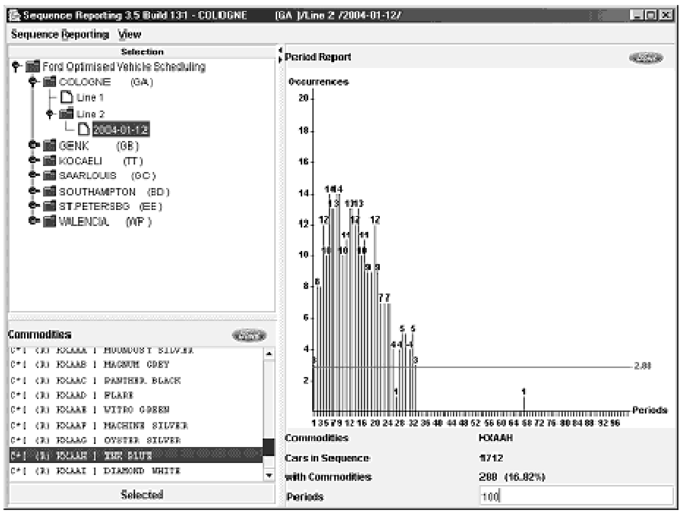


Commodities

```
 [ (R) DUMMY | DUMMY VEHICLES
 [ (R) TRIMA | TRIMLINE A
 [ (R) KAAAA | B2XX TOTAL
*[ (R) KAAAB | AIR CURTAIN RES
 [ (R) KAAAC | AIRCONDITION
*[ (R) KAAAD | 1.4 TC DIESEL
 [ (R) KAAAG | GHIA TOTAL
*[ (R) KAAAL | LEATHER
 [ (R) KAAAM | METALLIC
```

Commodities	KAAAL
Cars in Sequence	1712
with Commodities	59 (3.45%)

(a) Report showing how groups of size 2 of commodity KAAAL are spread equally within the master sequence.



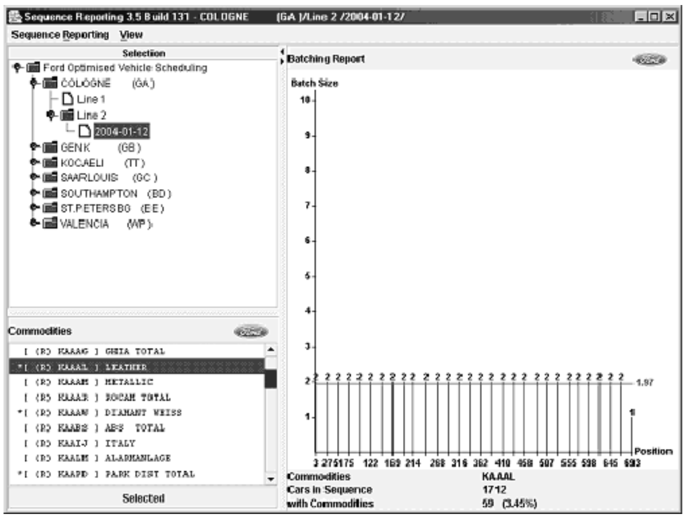
Commodities

```
C*[ (R) KXAAA | MOONDUST SILVER
C*[ (R) KXAAB | MAGNUM GREY
C*[ (R) KXAAC | PANTHER BLACK
C*[ (R) KXAAD | FLARE
C*[ (R) KXAAE | VITRO GREEN
C*[ (R) KXAAF | MACHINE SILVER
C*[ (R) KXAAG | OYSTER SILVER
C*[ (R) KXAAH | INK BLUE
C*[ (R) KXAAI | DIAMOND WHITE
```

Commodities	KXAAH
Cars in Sequence	1712
with Commodities	288 (16.82%)
Periods	100

(b) Report showing how commodity KXAAH (an enamel color) is clustered into the first slot of the master sequence.

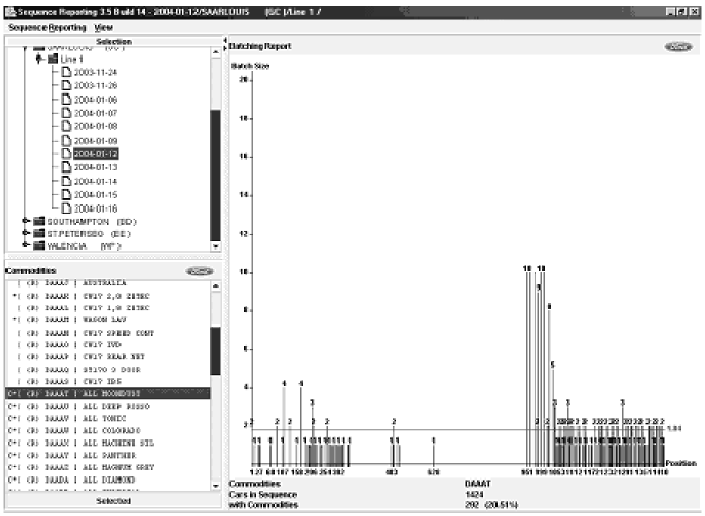
Figure 3: Screen-shots (1/2).



```
 [ (R) KAAAG ] GHIA TOTAL
*[ (R) KAAAL ] LEATHER
 [ (R) KAAAM ] METALLIC
 [ (R) KAAAR ] ROCAM TOTAL
*[ (R) KAAAW ] DIAMANT WEISS
 [ (R) KAABS ] ABS  TOTAL
 [ (R) KAAIJ ] ITALY
 [ (R) KAALM ] ALARMANLAGE
*[ (R) KAAPD ] PARK DIST TOTAL
```

```
Commodities            KAAAL
Cars in Sequence       1712
with Commodities       59  (3.45%)
```

(c) Report showing how commodity KAAAL is being banned from the last 1000 positions of the master sequence. The last occurrence is at position 693.



```
 [ (R) DAAAJ | AUSTRALIA
*[ (R) DAAAK | CW17 2,0 ZETEC
 [ (R) DAAAL | CW17 1,8 ZETEC
*[ (R) DAAAM | WAGON LAV
 [ (R) DAAAN | CW17 SPEED CONT
 [ (R) DAAAO | CW17 DVD
 [ (R) DAAAP | CW17 REAR NET
 [ (R) DAAAQ | ST170 3 DOOR
 [ (R) DAAAS | CW17 IDS
C*[ (R) DAAAT | ALL MOONDUST
C*[ (R) DAAAU | ALL DEEP ROSSO
C*[ (R) DAAAV | ALL TONIC
C*[ (R) DAAAW | ALL COLORADO
C*[ (R) DAAAX | ALL MACHINE STL
C*[ (R) DAAAY | ALL PANTHER
C*[ (R) DAAAZ | ALL MAGNUM GREY
C*[ (R) DAADA | ALL DIAMOND
```

```
Commodities            DAAAT
Cars in Sequence       1424
with Commodities       292  (20.51%)
```

(d) Report showing how groups of size 10 of commodity DAAAT are spread within the master sequence. Rules of higher priority prevent the grouping rule from functioning well.

Figure 4: Screen-shots (2/2).

A List of abbreviations

Notation	Meaning
$\mathcal{O}, \mathcal{C}$	Order set of size n and commodity set of size m
$\mathcal{P}(C)$	Partition of the order set with respect to $C \subseteq \mathcal{C}$
$O = \mathcal{O}(C)$	Set of orders which contain each commodity in $C \subseteq \mathcal{C}$
$\mathcal{Z}$	Set of zones of the plant
$q(z)$	Quota (proportion of production) of a zone $z \in \mathcal{Z}$
z_m	Master zone of the plant
$R(i)$	Routed zones of master sequence position i
$P(o, i, z)$	Position of $o \in \mathcal{O}$ in $z \in \mathcal{Z}$ if sequenced to position i in z_m
S	Number of slots of the master zone, each of size n/S
$S^*(O_i)$	Set of preferred slots for a commodity combination $O_i \subseteq \mathcal{O}$
$d(c, z)$	Delay of commodity $c \in \mathcal{C}$ in zone $z \in \mathcal{Z}$
$b(c, z_m; s)$	Banning factor for $c \in \mathcal{C}$ in $z_m \in \mathcal{Z}$ in slot s
$B(c, z_m; s)$	Banned positions for $c \in \mathcal{C}$ in $z_m \in \mathcal{Z}$ in slot s
$A(c, z_m; s)$	Desired average spreading distance for $c \in \mathcal{C}$ in $z_m \in \mathcal{Z}$ in slot s

References

[1] Michael E. Bergen. *Constraint-based assembly line sequencing.* PhD thesis, University of Alberta, 2001.

[2] Michael R. Garey and David S. Johnson. *Computers and intractability: A guide to the theory of $\mathcal{NP}$-completeness.* Freeman, 1979.

[3] Martin C. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, 1980.

[4] Rainer Hackl. *Optimierung von Reihenfolgeproblemen mit Hilfe Genetischer Algorithmen.* PhD thesis, Universität Regensburg, 2000.

[5] Yasuhiro Monden. *Toyota production system: An integrated approach to just-in-time.* Engineering & Management Press, 1998.

[6] Markus Puchta and Jens Gottlieb. Solving car sequencing problems by local optimization. In Stefano Cagnoni et al., editors, *EvoWorkshops 2002*, pages 132–142, Berlin Heidelberg, 2002. Springer.

[7] Horst Wildemann. *Das Just-in-Time Konzept: Produktion und Zulieferung auf Abruf.* TCW, 2001.