Title
Models for Concurrent Product and Process Design

Authors
Ahmadi, R. H.
Roemer, T. A.

Publication Date
2009-05-01

Models for Concurrent Product and Process Design

Thomas A Roemer

The Rady School of Management

University of California, San Diego

La Jolla, CA 92093-0093

troemer@ucsd.edu

Reza Ahmadi

Anderson School of Management

University of California at Los Angeles

Los Angeles, CA 90095

rahmadi@anderson.ucla.edu

May 6, 2009

Abstract

We propose procedures to address product design and manufacturing process configurations concurrently in environments characterized by large degrees of product proliferation. Exploiting the intrinsic flexibility of product and process design, we present two approaches that synchronize production flows through the manufacturing system. These approaches integrate product and manufacturing system design decisions with operational concerns and provide powerful means for managing production in environments characterized by a proliferation of products. Experimental results show that the propsed methods can substantially reduce manufacturing lead times, work in process (WIP), and overall system complexity.

KEYWORDS: Design Flexibility, Process Flow, AND-OR Trees, Heuristics.

1 Introduction

Over the last two decades a significant shift has taken place in the source of competitive advantage for manufacturing companies. Traditionally, firms made use of economies of scale to produce highly standardized products to satisfy massive and homogeneous markets. Nowadays, to stay competitive, firms need the capability to produce a broad variety of high quality products, and must exhibit rapid responsiveness to dynamic and increasingly fragmented markets by introducing new products frequently at short lead times (c.f. Blackburn, 1992 or Pine, 1999).

While standardized products were usually produced on dedicated assembly lines, under product variety most manufacturing systems are operated and scheduled as general job shops where the product types move independently along their own routes through the manufacturing system. Most commonly, such systems are controlled by myopic scheduling rules and require large queues of Work-In-Process (WIP) inventory to maintain high workstation utilization (Stecke, 1985). Moreover, scheduling is a time consuming process that requires highly skilled personnel (Lin and Solberg, 1991) and even sophisticated methods cannot guarantee acceptable WIP levels (Lee and Johnson, 1991). Together with WIP, manufacturing lead times also increase, thus countervailing and possibly jeopardizing the responsiveness that is so important in dynamic markets. In most manufacturing plants, the largest part of the manufacturing lead time is spent waiting in queues (Umble and Srikanth, 1990); in some instances over 90%, as demonstrated by Karmarkar et al. (1985).

These drawbacks vanish if the products can follow a *Synchronized Flow* (Ahmadi and Wurgaft, 1994a). In synchronized flows all products follow the same linear flow through the machines, thus moving smoothly and continuously from one operation to the next, with no waiting in queues. Small and medium size products can migrate through the system on pallets containing the entire product mix according to the ratios of their total demands. If, in addition, setup times are negligible, as is often the case if flexible manufacturing systems (FMSs) are employed, then batch sizes should be as small as possible and each pallet should contain a *Minimal Part Set* (MPS), which is the smallest set of products that satisfies the ratios of the total product demands. Many modern manufacturing systems, such as those for auto parts, computer boards and mechanical tools operate under these conditions. Throughout this paper we will assume an environment where these conditions hold and where each machine can be staged with c different operations.

A synchronized manufacturing system thus closely resembles the classical assembly line and bears all of its advantages, yet accommodates product variety. The MPSs are repetitively produced, a completed MPS leaves the system at the throughput rate and a new MPS is released into the system. Production scheduling and routing are thus easily accomplished and errors are unlikely. No queues can develop at the machines, keeping WIP low, lead times short and throughput high. Additional savings arise because the linear flow

only requires inexpensive equipment such as conveyors or stationary robots, rather than expensive labor or automatically guided vehicles, common in FMSs. The simplicity of the flow and the low WIP levels are also instrumental in detecting product defects early, thus further enhancing the system's responsiveness.

Ahmadi and Wurgaft (1994b) show how to partition product lines and design manufacturing processes to achieve synchronized product flows. However, attainability of this goal and its effectiveness depend largely on the product designs, in particular on the operations required and the structure of the manufacturing processes. In this paper we propose to utilize the inherent flexibility of product design to develop products that are amenable to a synchronized flow manufacturing system.

To model the flexibility in product design, we adopt the concept of AND-OR trees (Marple, 1961) to represent different product design choices. In particular, we assume that the different design choices for each product can be represented by an acyclic AND-OR digraph, whose nodes represent operations and whose arcs present precedence constraints between the operations. The core question in this paper is how product designs should be selected from their AND-OR tree representations, so that they can conform to the same linear flow, a property that we will refer to as *process compatibility*. More specifically, we will introduce algorithms that search through AND-OR trees to find process compatible designs while considering system capacity and throughput.

The example in Figure 1 illustrates how the choice of product designs impacts the production flow: Two products must be produced simultaneously and for each product one of two different designs, represented by the graphs[1] in Figure 1, must be chosen. Each of five available machines can perform exactly one operation, so each operation, represented by nodes, has to be assigned to a different machine. Notice that regardless of the design choice for product 1, choosing G_2^2 will require the performance of eight different operations for the manufacturing of both products, which exceeds the capacity of the five available machines. Figure 1a shows the assignment of operations to the machines and the production flow for the case when designs G_1^1 and G_2^1 are selected. In this case, the pallets have to revisit the machines where operations 3 and 4 are assigned and contentions are likely. Figure 1b shows the case when design G_1^2 is chosen for product 1. In this case, both products can conform to the same linear flow and each MPS can move from one machine to the next without conflicts.

Clearly, given sufficiently large capacity, it is always feasible to obtain a synchronized flow. However, unless product demands are quite high, this will inevitably lead to underutilized capacities. It is therefore imperative to find designs that are similar in the operations they require, yet do not cause reentry on

[1] In the next section we show how a single AND-OR tree can embed all design choices for a given product, thus obliterating the need to graph each design choice individually as in Figure 1.

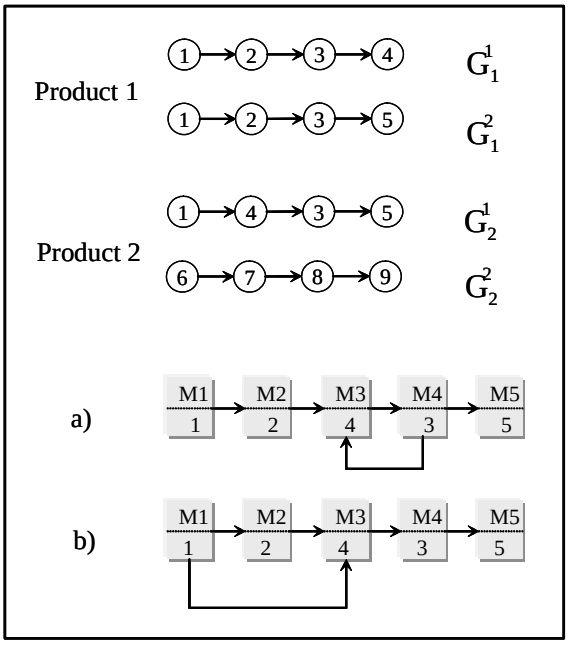


Figure 1: Effect of Product Design Choices

machines so that all products can follow the same linear unidirectional flow.

The remainder of the paper is organized as follows. In Section 2, we briefly discuss the concept of AND-OR design trees that are commonly employed to express flexibility in product design choices. In Section 3, we introduce a model and a solution procedure that selects process compatible designs under capacity constraints to yield well balanced lines with short lead times. A generalization of this model is presented in section 4, where we segment the product set into several process compatible product groups and evaluate the trade-offs between capacity requirements, WIP and lead times that arise in this model. Before we conclude the paper, we illustrate our procedure with the help of a 5-product example and benchmark their performance under a variety of different scenarios.

2 Product Design Flexibility

Product design can be regarded as a process that translates broad functional objectives successively into more and more detailed engineering specifications which eventually determine the final product in its entirety and provide all the necessary information for the product to be manufactured (Eastman, 1988). Inevitably, during this process, alternatives that are quite similar in terms of costs and quality have to be evaluated,

so that different design teams, when confronted with the same design tasks, usually come up with different solutions that satisfy the stated objectives (Stoll, 1990). Unfortunately, design teams all too often disregard the impact of their decisions on the manufacturing process, in particular when several products are designed concurrently, so that much of the potential flexibility remains unexploited (Whitney, 1988).

A common procedure for dealing with the complexity of design problems is decomposition (Hubka and Eder, 1988; Ulrich and Eppinger, 1995). The design problem is broken down into a series of subproblems such that the solution of all or some of them results in the solution of the original problem. These subproblems can be broken down further into smaller problems until the remaining problems are easily solved. The decomposition of complex design problems, in connection with the evaluation of alternative design choices gives rise to the concept of AND-OR tree representations (Marple, 1961; Eastman, 1988). AND-OR trees formally summarize all design choices in one graph and provide thus the opportunity to make an appropriate choice in the context of all design alternatives and the product's environment, such as existing product lines and manufacturing processes.

Formally an AND-OR tree is like any tree, but the vertices are partitioned into two distinct subsets, AND- and OR-nodes. The AND nodes define sets of components and technologies that are necessary in the design. The OR vertices represent alternative technologies and configurations.

Definition 1 *A subset of vertices of an AND-OR tree is said to be a* ***solution*** *if it satisfies the following three properties:*

- *The root vertex belongs to the solution.*
- *If a non-terminal AND-vertex belongs to the solution, then all of its children belong to the solution.*
- *If a non-terminal OR-vertex belongs to the solution, then exactly one of its children belongs to the solution.*

Figure 2 shows an example of such an AND-OR tree. Throughout the paper, we adopt the convention that OR-nodes are represented by squares, AND-nodes by circles and all nodes with less than two children are considered AND-nodes. The shaded nodes indicate one particular solution to this tree. The levels of the vertices in an AND-OR tree define hierarchical relations among the vertices. The higher the level of a vertex, the greater the level of detailed information for a part or component it contains. The vertices at the lower levels of the AND-OR tree represent alternative system or subsystem designs, encompassing many of the product's components and functions.

Lin and Solberg (1991) take the concept of AND-OR trees one level of detail further by translating the functional requirements into detailed engineering descriptions that specify all the necessary operation descriptions such as tooling, cutting positions, processing times and precedence relations among the operations.

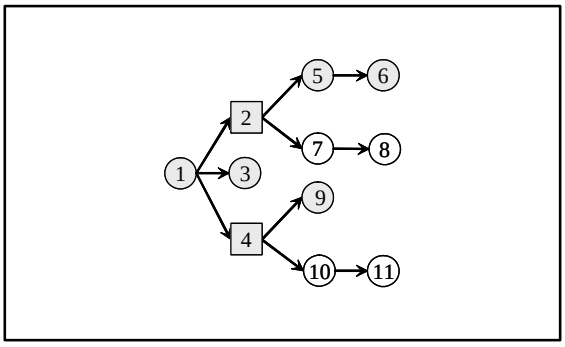


Figure 2: AND/OR Tree

Incorporating engineering descriptions into AND-OR trees adds additional flexibility to the design process since different operation types may yield the same results. For example, the same surface may alternatively be obtained by turning, slotting or planing (Victor, 2005) or a slot can be milled in one pass with a wider cutter or in multiple passes with a narrower tool (Rembold et al., 1985). The resulting structure from this translation is an AND-OR digraph representation of the different product designs in regard to their operations. In this representation, the nodes represent operations and the arcs represent precedence relations on the operations. Henceforth we will refer to such product representations as *Process Trees* (PTs). To preserve the tree structure, duplication of nodes as well as *dummy nodes*, representing logical constraints but not operations, are also allowed. A solution to a PT provides sufficient detail for the product to be manufactured and hence for the conditions for synchronized flow to be considered. For a more detailed description of AND-OR digraphs representing alternative process relations we refer the reader to Lin and Solberg (1991). Figure 3 shows the Process Tree representation for the two different design choices in Figure 1 where unnumbered nodes represent dummy nodes.

In the remainder of the paper, we assume a number of products are given, and for each product the PT has been obtained. We also assume that the alternative designs for the parts, given by the AND-OR trees, are similar in terms of general cost and conformance quality, that is inferior solutions have been deleted from the AND-OR trees of the products before obtaining the PTs. Finally, we assume without much loss of generality, that no individual design requires the same operation more than once.

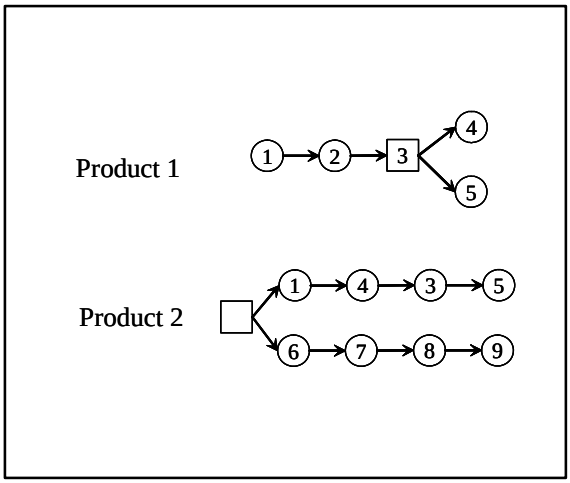


Figure 3: Process Tree Representation

3 Product Design Selection for Synchronized Flow

In this section, we study how a process compatible design selection can be determined from the process tree representations for the different products. However, searching merely for a compatible solution does not necessarily warrant satisfactory capacity requirements, WIP, lead times and throughput rates. We thus need to incorporate these aspects in our search for a "good" design selection. Notice that these four performance criteria are intimately linked in synchronized flows. The number of machines M (capacity) is equal to the number of $MPSs$ (WIP) in the system, since there is one MPS at each machine. The lead time in turn is the product of the number of machines and the maximum workload of the machines, which is the inverse of the throughput rate. Thus it is sufficient to address capacity and workload concerns in the search for a process compatible solution. We will address the former by requiring that the total number of operations required by a process compatible solution does not exceed the total staging capacity of a given number of machines.

To obtain a high throughput solution we will choose the designs that minimize the maximum processing time of the operations. If the machines are not flexible ($c = 1$), then this solution indeed maximizes the throughput. Otherwise, the maximum workload of the machines, and thus the throughput, depends not only on the chosen designs but also on how the operations are assigned to the machines. Given a design selection, a throughput maximizing assignment can be determined by solving the classical line balancing problem with the distinction of the additional constraint for the staging capacity. Hackman et al. (1989) concluded that

solutions to the line balancing problem improve quickly when the maximum processing time of the operations decreases. Based on this result, we select the designs with the smallest maximum processing time, and thus avoid the difficulty of addressing design selection and line balancing simultaneously.

Modeling this step is central to the next section where we introduce the *Design Selection Problem* (DSP). If the DSP is feasible, then its solution yields a process compatible solution that complies with the capacity requirements. In a second step, any of the heuristics suggested in the literature can be applied to this solution to solve the line balancing problem. The line balancing problem has been studied for at least six decades; a recent search on Google Scholar yielded 1620 results. Discussions and overviews of existing approaches include the research by Salveson (1955), Ignal (1965), Mastor (1970), Dar-El and Cother (1975), Baybars (1986), Hackman et al. (1989), Hofmann (1992), Berger et al. (1992) and Boctor (1995), Erel and Sarin (1998), McMullen and Tarasewich (2003), Becker and Scholl (2006), Dolgui et al. (2006), or Bock (2008).

3.1 Design Selection Problem

The notation and formulation of the DSP follow:

Parameters

P	Set of different products.
N	Set of different operation types.
D_p	Set of different designs for product p.
$s^i_{p,d}$	Contribution of product p to the processing time of operation i if design d is chosen.
$a^i_{p,d}$	$\begin{cases} 1, & \text{if design } d \text{ for product } p \text{ requires operation } i. \\ 0, & \text{otherwise.} \end{cases}$
M	Number of available machines.
c	Staging capacity of the machines.

Indices

p	Product indices, $p = 1, ..., \|P\|$.
d	Design indices, $d = 1, ..., \|Dp\|$.
i	Operation indices, $i = 1, ..., \|N\|$.

Variables

W	Maximum processing time.
$y_{p,d}$	$\begin{cases} 1, & \text{if design } d \text{ for product } p \text{ is chosen.} \\ 0, & \text{otherwise.} \end{cases}$
x_i	$\begin{cases} 1, & \text{if operation } i \text{ is performed.} \\ 0, & \text{otherwise.} \end{cases}$

DSP Minimize W *s.t.*

$$\sum_{p \in P} \sum_{d \in D_p} s^i_{p,d} \cdot y_{p,d} \leq W \quad \forall i \tag{1}$$

$$x_i \geq a^i_{p,d} \cdot y_{p,d} \quad \forall i, p, d \tag{2}$$

$$\sum_{i \in N} x_i \leq M \cdot c \tag{3}$$

$$\sum_{d \in D_p} y_{p,d} = 1 \quad \forall p \tag{4}$$

The selected designs must be compatible (5)

$$y_{p,d} \in \{0, 1\} \quad \forall p, d \tag{6}$$

$$x_i \in \{0, 1\} \quad \forall i \tag{7}$$

Constraints (1) define the total processing times for the operation. The total processing time of an operation depends on the designs requiring this operation and on the volume of the products corresponding to those designs. The objective function minimizes the largest operation processing time, so that the line balancing problem yields a well balanced line. Constraints (2) indicate whether an operation i is required by any of the selected designs. Constraint (3) imposes the total staging capacity of the machines and, since capacity is closely linked to WIP and lead times, also checks these two performance criteria. Constraints (4) indicate that exactly one design must be chosen for each product. Constraint (5) is the compatibility constraint which require the superimposition of the selected designs to be acyclic. If the superimposed graph[2] has no cycles, then all the parts can be processed in a single pass through the production system.

Note that in this formulation of the problem no operation can be assigned to more than one machine. However, Ahmadi and Wurgaft (1994b) have shown that duplication of operations can enhance the implementation of synchronized flows. At the price of additional complexity, the following modifications of constraints 3 and 5 permit duplication of operations:

$$\sum_{i=1}^{|N|} x_i + \alpha = M \cdot c \tag{8}$$

The selected designs must be α-compatible (9)

where we define designs to be α-*compatible*, if at most α operations must be repeated for the designs to be process compatible. In essence, this modification, which we will refer to as the *Relaxed Design Selection Problem* ($RDSP$), utilizes slack in Constraint (3) by assigning operations to more than one machine. The

[2]Let $G_i = (V_i, E_i)$ be the digraphs associated with a given set of design selections for the $|P|$ products. Digraph $G = (\cup_{i \in P} V_i, \cup_{i \in P} E_i)$ is said to be the *superimposed graph* for these selected designs.

following proposition and its corollary establish the complexity of DSP and $RDSP$.

Proposition 1 *DSP is unary NP-hard.*

Proof. We reduce DSP to the 3-Partition problem (c.f. Lenstra et al. 1977): Given nonnegative integers $a_1, a_2, ...a_{3t}, b$, such that each a_i satisfies $\frac{b}{4} < a_i < \frac{b}{2}$ and such that $\sum_{i=1}^{3t} a_i = tb$, does there exist a partition $(T_1, T_2, ..., T_t)$ of $T = \{1, ..., 3t\}$, such that $|T_j| = 3$ and $\sum_{i \in T_j} a_i = b$ for $j = 1, 2, .., t$? Consider the following instance of DSP: $c = 1$, $|N| = M = |D_P| = t$, $|P| = 3t$, $s^i_{p,d} = a_p$ if $d = i$, otherwise $s^i_{p,d} = 0$. Its optimal solution is $W = b$ if and only if a 3-Partition exists. To see this, first note that each design requires exactly one operation and that therefore no precedence relations exist. Hence Constraint (5) is always satisfied. Moreover, the total processing time required for all products is tb, independent of the designs chosen. Since t machines are available, b establishes a lower bound for the maximum processing time W. Now suppose a 3-Partition exists and let $y_{p,k} = 1$ if $a_p \in T_k$ for all p and k. This assigns exactly 3 operations of the same type to each machine, yielding a perfectly balanced system where each machine has the same workload b. Conversely, suppose the answer to DSP is $W = b$. This implies that exactly 3 operations are assigned to each machine and that the processing times on each machine add up to b. Letting $a_p \in T_k$ if $y_{p,k} = 1$ for all p and k yields a 3-Partition. ■

Corollary 2 *$RDSP$ is unary NP-hard.*

Proof. The corollary follows trivially from the preceding proof. ■

3.2 Solution of the Design Selection Problem

Now we develop a procedure to solve the Design Selection Problem. The algorithm searches for an optimal solution in the space given by the choices among the alternative designs of each product. We shall denote the elements of this space σ as $\delta = (d_1, ..., d_{|P|})$ where d_p $(d_p = 1, ..., |D_p|)$ indicates the index of the design that is chosen for product p. The choices δ are relaxed solutions to DSP, since they do not necessarily satisfy the compatibility and other constraints. Let $W^i(\delta) = \sum_{p=1}^{|P|} s^i_{p,d_p}$ denote the total processing time of operation i associated with selection δ. The algorithm generates a series of relaxed solutions that is nondecreasing in W^r, where r is a reference operation. The *Design Selection Algorithm* (DSA) follows.

Design Selection Algorithm (DSA)

Step 0 Let $W^* = \infty$, $W^r = 0$, and $L = \sigma$.

While $L \neq \{\}$ and $W^r < W^*$ **do:**

Step 1 Let $W^r = \min_{\delta \in L} \{W^r(\delta)\}$, $\delta^0 = \arg\min_{\delta \in L} \{W^r(\delta)\}$, $W = \max_{i=1,\ldots,|N|} \{W^i(\delta^0)\}$, and $L \rightarrow L \backslash \{\delta^0\}$.

Step 2 If $W < W^*$ and if δ^0 satisfies constraints (3)and (5), then $W^* = W$ and $\delta^* = \delta^0$.

The algorithm works as follows: Initially, the processing time W^r for the reference solution is set to 0, the incumbent value W^* is set to infinity, and all relaxed solutions are placed in a list L. Step 1 generates the elements of the sequence of relaxed solutions that is increasing in W^r. At each iteration of the algorithm, the design δ^0 with the lowest processing time for the reference operation is removed from the list L and its largest processing time W is computed. Step 2 updates incumbent value W^* if δ^0 is feasible and an improvement over the incumbent solution. Since the sequence of relaxed solutions is increasing in the reference processing time W^r, the algorithm terminates once the reference processing time of the newly generated element is greater than the incumbent value W^*. Alternatively, the algorithm stops after emptying the list L.

Notice that the feasibility check in Step 2 requires to verify if constraints (3) and (5), regarding the staging capacity and compatibility are satisfied. DSA treats these constraints implicitly. Constraint (3) is violated if and only if the number of nodes in the superimposed graph exceeds the available staging capacity and Constraint (5) is violated if and only if the superimposed graph contains circuits. The latter can be checked efficiently by topologically ordering the nodes in the superimposed graph. The graph is acyclic if and only if a topological order exists (Deo, 1974). This approach of treating the compatibility constraints implicitly is similar to the approach of using independence tests to determine maximum-weight independent sets in matroids (c.f., Nemhauser and Wolsey 1988).

In the worst case, the DSA will stop after enumerating all the $\prod_p D_p$ relaxed solutions. However, the choice of the reference operation may significantly affect the computational requirement of the algorithm. A good choice for the reference operation will cause the $W^r \geq W^*$ condition to occur as early as possible. It is likely that this will occur if the operation with the largest average $s^i_{p,d}$ is selected as the reference operation. In practice, this rule works quite well with homogenous product designs, where processing times are similar across the designs. For non homogenous designs, the variance of the processing times should also be considered in the choice of the reference operation. Possibilities include using weighted averages of the mean and coefficient of variation, choosing the reference operation with the highest average of some number of the smallest processing times and others.

Solving $RDSP$ necessitates α-compatibility tests which require two steps. First, α needs to be computed which is simply the difference between the staging capacity and the number of operations (or nodes in the superimposed graph) required by the given design selection. The second step must determine if all cycles can be removed from the superimposed graph by duplicating at most α operations which is equivalent to

ascertain that a graph can become acyclic by removing at most α arcs. This is the well known, unary NP-hard (Karp, 1972) *minimum arc feedback* problem, which together with the closely related *linear ordering problem* occur in numerous applications such as matrix triangulization (Lawler, 1964), decision making (Korte and Oberhofer,1969), comparison ranking (Flueck and Korsh, 1974), preference ranking (Hochbaum and Levin, 2006), tournaments (Charbit et al. 2007) and others. As a consequence many solution procedures have been suggested during the last three decades (e.g. Kaas 1981, Grötschel et al., 1984, Barthelemy et al. 1989, Seymour 1995, Even et al. 1998, Aneja and Sokkalingam 2004, Hudry and Charon, 2006 or Wang 2008), which can be employed to determine α-compatibility. In many instances, to affirm α-compatibility, it may suffice to establish an upper bound for the minimum arc set feedback problem. One such upper bound is the number of cycles in the superimposed graph which is easily obtained by generating the reachability matrix (Deo, 1974). Nevertheless, $RDSP$ still may require an α-compatibility test at each iteration, so that running times can become much larger than for DSP. We therefore recommend $RDSP$ only if attempts to solve DSP have been futile.

3.3 Generation of Designs

A critical step in the algorithm is the search for the design with the smallest processing time of the reference operation in Step 1. In this section we discuss how to find this design efficiently and how this obliterates the need to store the entire list L. Notice that the k^{th} iteration of the DSA requires the design with the k^{th} smallest processing time for the reference operation, henceforth referred to as the k^{th} *minimum design* denoted by T^k. Consider the graph that combines the PTs of all products under one common AND root. This graph can be thought of as a *Total Process Tree* (TPT), whose components are the individual products. Moreover there is a one-to-one correspondence between design selections and solutions to the TPT and the search for the k^{th} minimum design can be performed on the TPT. To find T^k we observe that there must be at least one $h < k$ such that T^k and T^h have exactly one OR-node in common, for which they have distinct children.

For example, consider the graph in Figure 4. The dummy root node 0 combines products 1 and 2 (with root nodes 1 and 7 respectively) into one TPT. Processing times for reference operation 3 are given in parentheses in the respective nodes. Since there are only two design choices for each product, exactly four design choices exist as listed and ranked in the figure. Notice that the T^2 design differs from the T^1 design by choosing node 5 instead of node 2 as the son of node 1, while all other choices remain the same. Similarly, T^3 (indicated in the figure by the dark shaded nodes) differs from T^1 by choosing node 9 as the son of node 7, while otherwise making identical choices to T^1. Finally, T^4 differs from T^3 in choosing node 5 as the son

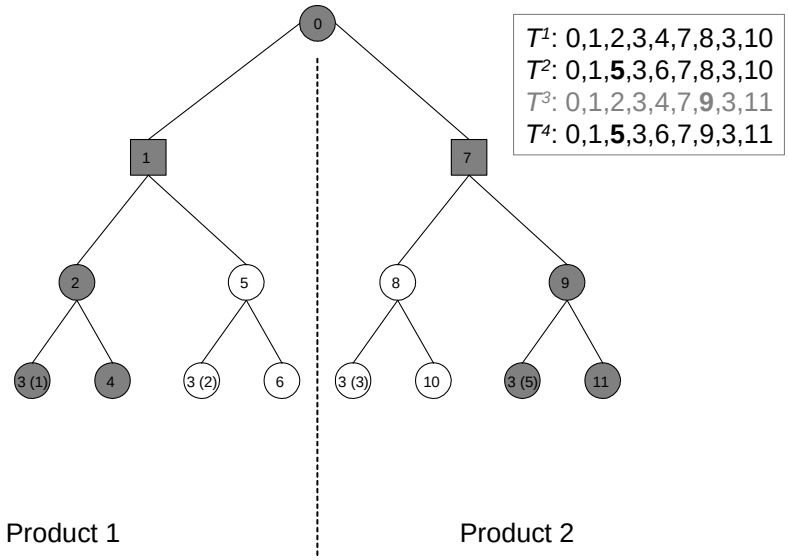


Figure 4: Example of Total Process Tree

of node 1 instead of node 2. This property is formally expressed in the following proposition.

Proposition 3 *For any $k > 1$, there exists a T^k such that for some $h < k$, T^k and T^h have exactly one OR-node in common, for which T^k and T^h have distinct sons.*

Proof. Clearly, if all OR-nodes of the two solutions have identical sons, then $T^k = T^h$, contradicting $h < k$. Consequently, at least one such OR-node must exist. Now suppose two of such OR-nodes exist, say node 1 and node 2 and denote their descendants by sets H_1, H_2, K_1, and K_2 respectively. Since T^h and T^k are identical except that $H_1 \neq K_1$ and $H_2 \neq K_2$, any difference in the total duration of the reference operation must stem from different contributions $C(\xi)$ ($\xi \in \{H_1, H_2, K_1, K_2\}$) by these four sets. Clearly $C(H_1) > C(K_1)$ and $C(H_2) > C(K_2)$ cannot hold since $h < k$. W.l.o.g. suppose that $C(H_1) \leq C(K_1)$ and consider tree T^g which is identical to tree T^k except that it has descendents H_1 for node 1 instead of K_1. If $C(H_2) \leq C(K_2)$ then the processing time for the reference operation for T^g cannot exceed that of T^k and either T^g is one of the best $k-1$ solutions or T^g is also a k-th minimum design. In the latter case, the proposition is satisfied by trees T^g and T^h in the former case by trees T^k and T^g. Conversely, if $C(H_2) > C(K_2)$ then the processing time for the reference operation of tree T^g must be less than that for tree T^h. Thus T^g must be among the $k-1$ best designs and the proposition holds for trees T^k and T^g. ■

Proposition 3 allows us to recursively generate T^k once T^1 is obtained. To obtain the latter, define V as the vertex set of TPT, $H(i)$ as the set of children of vertex i, and p_i as the processing time required for the reference operation at vertex i $(i = 1, ...|V|)$. The height h of a tree is defined as the maximum level among the vertices in TPT, where the level of vertex i is the length of the path from the root to vertex i (c.f. Manber, 1989). The following algorithm computes T^1 in $O(|V|)$ time:

Minimum Design Algorithm (MDA)

Step 0 Determine the level of all vertices and label the vertices accordingly.

Let $v_i = p_i$ for $i = 1, ...|V|$ and $t = h - 1$, $L = V$.

While $t > 0$ **do:**

Step 1 For each node at level t let $v_i \rightarrow \begin{cases} v_i + \min_{q \in H(i)} v_q & \text{if } i \text{ is an OR-node} \\ v_i + \sum_{j \in H(i)} v_j & \text{if } i \text{ is an AND-node} \end{cases}$

IF i is an OR-node, then let $v_q \leq v_j \ \ \forall j \in H(i)$. Set $L = L - \bigcup_{j \in H(i) \setminus q} H_j$.

Step 2 Let $t \rightarrow t - 1$.

The level of each vertex in Step 0 can be identified by any tree traversal algorithm such as preorder or postorder transversal (c.f. Sedgewick 1988). The solution is identified by the remaining vertices in List L.

To obtain T^k we define R^k as the set of OR nodes corresponding to T^k and T_n^{k-1} to be a *minimum deviation* from the design T^{k-1} at OR-node n in the following manner: T_n^{k-1} consists of all the nodes of T^{k-1} except for the descendants of node n. If these descendants constitute the tree t_i in T^{k-1} with root node $i \in H(n)$ and required processing time v_i then T_n^{k-1} is obtained by replacing t_i in T^{k-1} with the subtree t_j of TPT, such that $v_j = \min_{j \in H(n)} [v_j | v_j \geq v_i]$. The algorithm to find T^k follows. L_o is a list containing the solutions.

k^{th}-Minimum Design Algorithm $(k - MDA)$

Step 0 Find T^1 by applying MDA. Set $\kappa = 2$. Place T^1 in list L_o.

While $\kappa \leq k$ **do:**

Step 1 For $n = 1, ..., |R^{\kappa-1}|$ determine T_n^{k-1} and place it into L_1.

Step 2 Find the minimum solution in list L_1. Denote this solution by T^κ and move it from L_1 to L_o. Let $\kappa \rightarrow \kappa + 1$.

Note that at the κ^{th} iteration all minimum deviations from the T^j, $(j = 1, ..., \kappa-2)$, already exist in L_1 so that only the deviations from $T^{\kappa-1}$ are needed in order to complete the list. The minimum design deviations in Step 1 can efficiently be computed using the information in the v_is from Step 0. The computational complexity of the algorithm is $O(k \cdot |R| \cdot H)$ where R is the set of OR-nodes in TPT and H is the maximum number of children of any OR-node in TPT.

With the Design Selection Algorithm we have provided a tool that generates process compatible designs from process tree representations under particular consideration of total system throughput. Each iteration of the DSA evaluates different design selections for the products. To generate these efficiently, we developed the k^{th}-Minimum Design Algorithm. The DSA considers all products simultaneously and ultimately forces them to follow the same linear flow through the entire manufacturing system. In the next section, we address how to partition the products into process compatible subsets, that are processed on separate, and thus shorter, synchronized sublines.

4 Product Set Partitioning for Synchronized Flow

Often times, WIP and lead times of synchronized solutions can further be improved by segmenting the production line into synchronized sublines. To see this, consider the four product example in Figure 5 and suppose that five machines are available, each with staging capacity $c = 2$. The shaded nodes in Figure 5 indicate the only solution satisfying these constraints. Figure 6a shows a corresponding assignment of the operations to machines $M1$ to $M5$. Notice that only the first two operations assigned to machine 1 are performed on all four products. Moreover, once an MPS exits the first machine, products A and B and products C and D respectively, share two of the remaining three operations. However, note that no two operations are common between these two groups of products. This suggests to partition the products into two distinct subsets $\{A, B\}$ and $\{C, D\}$ that are processed together only in an initial segment, while being processed in parallel in a second segment as in Figure 6b.

The subject of this section is to formally address the segmentation of the product set into process compatible subsets. One difficulty arising in this context is that this problem involves the decision at what point to segment the line into sublines. Lee and Tang (1997) provide a modeling framework that characterizes the optimal point of product differentiation after which the line segmentation should proceed. In our model we will assume that a suitable *Point of Segmentation* (PoS) is given. Complete product segmentation without a common initial line can be obtained when the PoS is placed at the beginning of the production line. Frequently, product variety is based on a common platform shared by a family of products or on cost-reduced versions, add-ons or enhancements to a core product (Shirley, 1990). In this case the product

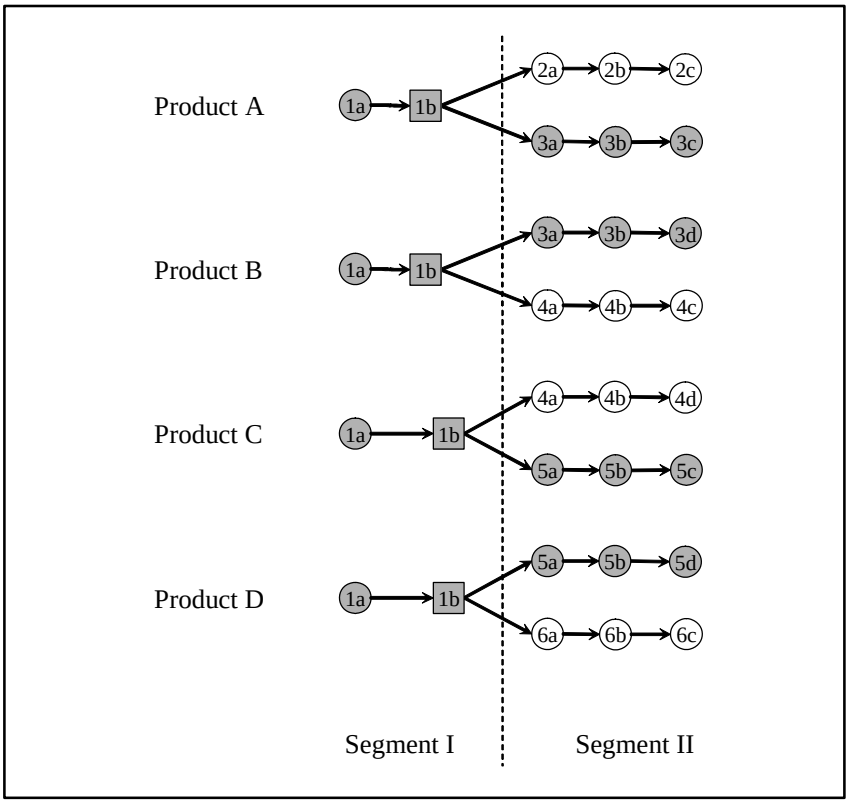


Figure 5: Suitable Designs for Line Segmenting

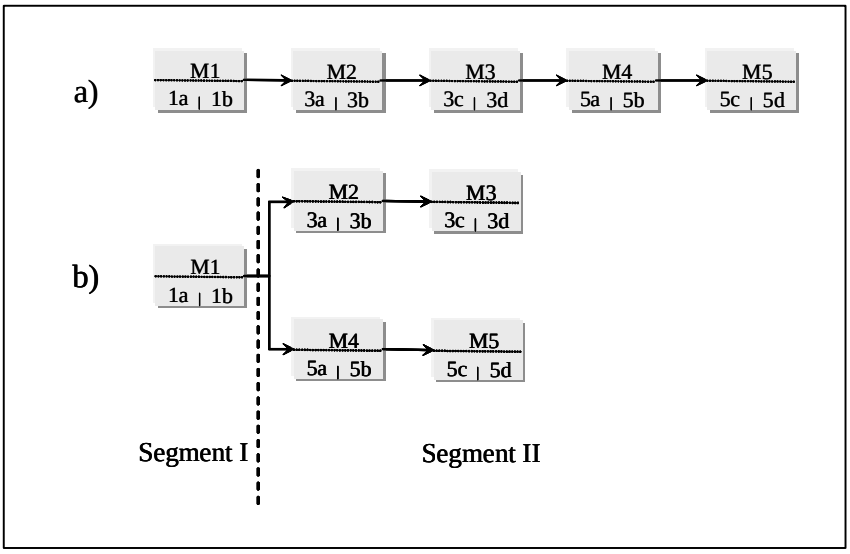


Figure 6: Line Segmenting

platform or the core product can be produced on a single line followed by individual sublines.

4.1 The Product Partitioning Problem

To model this problem, we will minimize the number of sublines or groups to be formed subject to machine availability constraints for each group and to the condition that all designs within a group are process compatible. The objective function serves as a good surrogate for minimizing capacity requirements, since additional groups require more operation duplications and thus more capacity. At the same time, the constraints on the number of machines for each group control total WIP and the lead time of the entire system. To minimize the number of groups, we will again exploit the availability of alternative designs. These alternative designs are not the designs for the entire product, but only for the part of the product to be produced after the PoS and correspond to the alternative branches in the TPT that follow the operations in the designs chosen up to the PoS. The model requires the following additional notation.

Parameters

$\left\|D'_p\right\|$	Number of designs of product p after the PoS.
M'	Maximum number of machines per group.
W'	Upper bound on the maximum processing time.

Indices

d	Design indices, $d = 1...\left\|D'_p\right\|$.
g	Group indices, $g = 1, ..., \|P\|$.

Variables

G	Number of Groups.
$y_{p,d,g}$	$\left\{ \begin{array}{ll} 1, & \text{if design } d \text{ for product } p \text{ is chosen and assigned to group } g. \\ 0, & \text{otherwise.} \end{array} \right\}$
$x_{i,g}$	$\left\{ \begin{array}{ll} 1, & \text{if operation } i \text{ is performed in group } g. \\ 0, & \text{otherwise.} \end{array} \right\}$
z_g	$\left\{ \begin{array}{ll} 1, & \text{if group } g \text{ is nonempty.} \\ 0, & \text{otherwise.} \end{array} \right\}$

The Product Partitioning Problem (PPP)

Minimize G s.t.

$$\sum_{g=1}^{|P|} z_g \leq G \tag{10}$$

$$z_g \geq y_{p,d,g} \quad \forall p, d, g \tag{11}$$

$$\sum_{g=1}^{|P|} \sum_{p \in P} \sum_{d \in D_p} s^i_{p,d} \cdot y_{p,d,g} \leq W' \quad \forall i \tag{12}$$

$$x_{i,g} \geq a^i_{p,d} \cdot y_{p,d,g} \quad \forall i, p, d, g \tag{13}$$

$$\sum_{i \in N} x_{i,g} \leq M' \cdot c \quad \forall g \tag{14}$$

$$\sum_{g=1}^{|P|} \sum_{d \in D_p} y_{p,d} \leq 1 \quad \forall p \tag{15}$$

The selected designs in each group must be compatible (16)

$$y_{p,d,g} \in \{0, 1\} \quad \forall p, d, g \tag{17}$$

$$x_i \in \{0, 1\} \quad \forall i \tag{18}$$

$$z_g \in \{0, 1\} \quad \forall g \tag{19}$$

Constraint 10 defines the total number of groups and constraints 11 are logical constraints that force z_g to unity once a product has been assigned to Group g. Constraints 12 limit the maximum processing time

for any operation to allow for a good line balancing solution. If the first segment, that is the line before the PoS, has been determined by the DSA then the objective value from this solution can be used. Otherwise, W' can be set as a percentage of the throughput rate (T) of the first segment. If it is imperative that the throughput rate of the second segment does not exceed T, then $\frac{T}{c}$ can be chosen for W'. However, we found this condition to be too restrictive in most applications and only justified if the variation between processing times was very small. Constraints 14 limit the number of machines available to each group and thus control WIP and, in connection with constraints 12, the lead time as well. The remaining constraints are generalizations of constraints 2 and constraints 4 to 7 in DSP.

In complete analogy to $RDSP$, duplication of operations within groups can be incorporated by the following problem relaxation ($RPPP$), where constraints 14 and 16 in PPP are modified as follows:

$$\sum_{i \in N} x_{i,g} + \alpha_g \leq M' \cdot c \quad \forall g \tag{20}$$

$$\text{The selected designs in group } g \text{ must be } \alpha_g\text{-compatible for all } g. \tag{21}$$

Notice that a solution to PPP with $G = 1$ exists if and only if a solution to DSP exists with $W \leq W'$. It follows therefore directly from Proposition 1 that PPP and $RPPP$ are unary NP-hard.

4.2 Heuristic solution for the Product Partitioning Problem

Since PPP is a generalization of DSP and unary NP-hard we will in this section only present a simple greedy heuristic to generate solutions to PPP. Let $\Theta(d_p)$ be the set of operations required by design d for product p, $\Theta(\Gamma) = \bigcup_{d_p \in \Gamma} \Theta(d_p)$ be the set of operations required by a collection Γ of designs, $f(\delta_p)$ be a *preference score* for design d_p and $R(p)$ be a *regret value* associated with product p. These two scores mainly serve to select designs for new groups. A good design choice should allow the accommodation of many additional products within the same group, but should also consider workload balancing criteria. A good preference score should therefore evaluate a design based on

- its number of operations,
- the degree to which its operations are common among all designs,
- its total processing time, and
- its maximum processing time.

We have obtained good results by equating the preference score to the number of operations required by a design, while successively applying the remaining three factors as tie breakers. The regret value evaluates the ratio between the preference score of the two most preferable designs for each product. If this ratio is large, then not choosing the most preferable design is likely to create conflicts later on. Consequently, products with high regret values should be considered first when creating new groups. In the *Product Set Partition Heuristic* below, lower preference scores indicate more preferable designs:

Product Set Partitioning Heuristic $(PSPH)$

Step 0 Let $g = 0$, $\Pi = P$.

While $\Pi \neq \{\}$ **do:**

Step 1 Let $g \rightarrow g+1$, $\widehat{D}_p = D_p$ for all $p \in \Pi$,
$\overline{p} = \arg\max_{p\in\Pi} \{R(p)\}$, $\Pi \rightarrow \Pi \backslash \{\overline{p}\}$ and $\Gamma_g = \left\{\delta_{\overline{p}} | f(\delta_{\overline{p}}) = \min_{d_{\overline{p}}\in\widehat{D}_p} \{f(d_{\overline{p}})\}\right\}$

Step 2 Determine design δ_π such that $|\Theta(\delta_\pi) \bigcup \Theta(\Gamma_g)| = \min_{p\in\Pi, d_{\overline{p}}\in\widehat{D}_p} \{\Theta(d_p) \bigcup \Theta(\Gamma_g)\}$.
If $|\Theta(\delta_\pi) \bigcup \Theta(\Gamma_g)| > M' \cdot c$ then goto Step 1; else let $\Gamma'_g = \Gamma_g \cup (\delta_g)$.

Step 3 If Γ'_g satisfies the compatibility constraints, then let $\Pi \rightarrow \Pi \backslash \{\pi\}$ and $\Gamma_g = \Gamma'_g$. Else let $D_\pi \rightarrow D_\pi \backslash \{\delta_\pi\}$. Goto Step 2.

Step 1 utilizes the regret value and the preference score to create a new group. For unassigned products, Step 2 determines the design that adds the least number of additional operations to an existing group. If adding this design to the group violates the capacity constraints 12, then no design can be added and Step 1 starts over. Otherwise, if Step 3 ascertains that the selected design satisfies the compatibility constraints, then it is added to the current group and the search for another compatible design from the remaining products continues. Checking for compatibility (or α-compatibility) can be done in the same manner as in the Design Selection Algorithm. If the design is not compatible, then it is excluded from further consideration for this group and the search for another compatible design starts over. Notice that excluding the first $k-1$ best designs amounts to finding the k^{th} best design, which can efficiently be computed by appropriately modifying the $k - MDA$ presented above. $PSPH$ terminates after at most $P\sum_p D_p$ iterations, each of which may require a compatibility test and invoke $k - MDA$.

In this section we have presented a line segmentation approach for synchronized flow manufacturing. The approach combines partitioning of the product set with design selection to satisfy the conditions for synchronized flow. Since this approach reduces unnecessary waiting times by grouping products that have similar processing requirements, WIP and lead times will be smaller than in the unpartitioned approach.

On the other hand, capacity requirements may increase and machine utilization decrease if operations have to be duplicated for different groups. Furthermore, since the existence of product groups limits the choices during the line balancing phase, throughput rates may also decrease unless counteracted by the additional capacity. In practical applications, we therefore recommend to evaluate different scenarios as illustrated in the example in the next section.

5 Numerical results

5.1 Illustrative Numerical Example

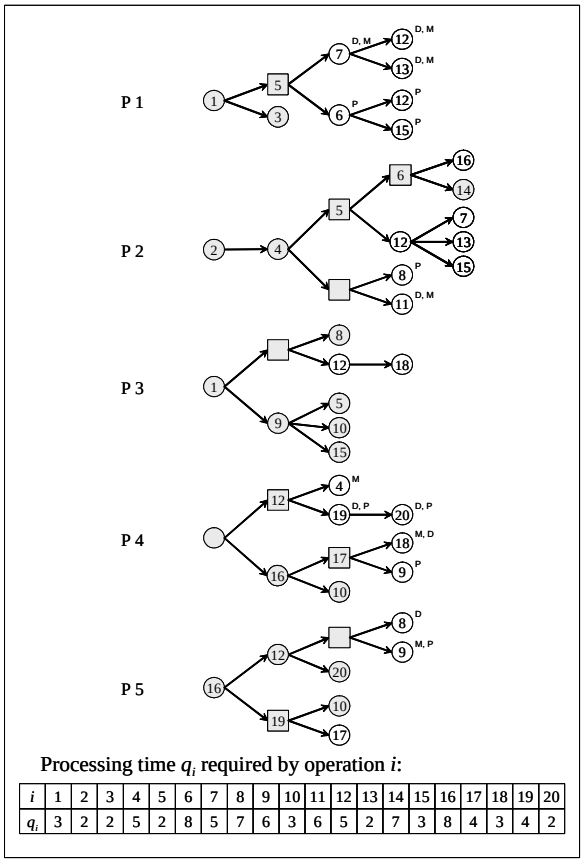
The following example illustrates the concepts for synchronized flow manufacturing: There are $|P| = 5$ products that must be produced at the same production rate on machines with staging capacity $c = 4$. Figure 7 shows the process trees for the products and the unit processing times for the $|N| = 20$ different operation types. The products in this example are substantially differ in terms of operation sequences as well as general tree structure. In particular, no common product platform can be distinguished. Thus the example will illustrate our methodology on rather adverse circumstances.

To evaluate the performance of the suggested approaches, we will establish two benchmark cases. Both cases consider manufacturing issues during design selection, but do not the incorporate the concept of synchronized flows. The first approach, the *minimum processing time* (MPT) solution, selects the design with the least total processing time for each product. Since the MPT solution also minimizes total processing time, it is expected to result in particularly short lead and cycle times. The second benchmark case is the design selection that requires the *minimum number of operations* ($MinOp$) for each design. While this selection does not necessarily yield the solution with the minimum number of distinct operations, it is nonetheless expected to yield good solutions without excessive machine requirements. Clearly, both solutions are easily obtained and should thus be of interest to practitioners.

The MPT solution requires 20 and the $MinOp$ solution 15 distinct operations[3], so that respectively five and four machines are necessary for their implementation[4]. To maximize throughput for each solution, the machine workloads were balanced by appropriate operation assignments. Determining such an assignment is a unary NP-hard problem for staging capacities larger than 2, but most Bin-Packing heuristics (Garey and Johnson, 1979) yield suitable assignments. To obtain a routing for the MPSs, we applied the cyclic

[3] The $MinOp$ solution is not unique in this example. In the spirit of testing our methodolgy under adverse conditions, we computed that solution that minimizes the total number of distinct operations as well.

[4] The dark shaded nodes in Figure 7 are common to all solutions. The remaining nodes are marked with a D, M, or, P if they belong to the DSP, MPT, or $MinOp$, solutions respectively.



i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
q_i	3	2	2	5	2	8	5	7	6	3	6	5	2	7	3	8	4	3	4	2

Figure 7: 5 Product Example

scheduling heuristic proposed by Graves et al. (1983), which is known to generate good results.

Next, we solved DSP and $RDSP$ for both, $M = 4$ and $M = 5$ henceforth referred to as $(R)DSP$ 4 and $(R)DSP$ 5 respectively. For each of these solutions, the machines were balanced under conservation of the precedence constraints. Finally, we solved the following instances of PPP: Up to three machines before the PoS were considered[5] where the line up to the PoS was identical to the DSP 5 solution. For each of these choices, we solved for all feasible and non-trivial values of machines per group M'. In all instances the upper bound W' on the maximum processing time, that is the RHS of Constraint 12, was set to $W' = 16$, the objective function value of the solution to DSP 5[6]. As in the non-partitioned synchronized solutions, the

[5]Note that this covers all necessary choices for the POS since DSP 5 is an optimal solution for the case where the PoS occurs after the fourth machine.

[6]It is interesting to observe that whenever we relaxed this constraint, we obtained solutions that were strictly dominated by others on all four performance criteria.

Approach \ Performance Measure	Machines	Cycle Time	WIP/ Machine	Lead Time	Distance	Total Proc. Time	Groups
Benchmarking Cases							
MPT	5	26	1.25	162	13.0	129	(1)
MinOp	4	35	1.31	184	9.0	138	(1)
Unpartitioned Solutions							
DSP 4	4	38	1	152	4.0	139	(1)
DSP 5	5	27	1	135	5.0	131	(1)
RDSP 5	5	27	1	135	5.0	130	(1)
Partitioned Solutions							
PPP 0\2	8	30	1	60	2.0	136	4
PPP 0\3	5	30	1	90	2.6	138	2
PPP 1\2	7	26	1	78	3.0	141	3
PPP 1\3	6	26	1	104	3.6	138	2
PPP 2\2	6	26	1	104	4.0	138	2
PPP 3\1	6	26	1	104	4.0	129	3

Figure 8: Overview of Solutions

machines were then balanced under conservation of the precedence constraints.

To compare the solutions from the different approaches along different dimensions, we computed the number of machines required, the cycle time, WIP and the lead times. In addition, we also computed a measure called *distance* - an approximate measure for how far an MPS has to travel through a set of machines. It is computed by adding up the absolute differences of the machine indices of two consecutively visited machines. For example, as depicted in Figure 9, the MPT-Solution routes the MPSs in order $M1, M2, M3, M4, M5, M1, M2, M4$, before exiting at "virtual machine" $M6$. For the distance d this yields $d = |2-1| + |3-2| + |4-3| + |5-4| + |1-5| + |2-1| + |4-2| + |6-4| = 13$. Notice that by this convention, for the synchronized solutions, distance is equal to the number of machines. Moreover, if the machines are arranged equidistantly along a line, our measure becomes directly proportional to the distance traveled through the machines. With increasing distance (relative to the number of machines), the actual management of the system becomes more challenging. As such, the distance may serve as a proxy for managerial complexity, as well as for system setup costs, since longer workpiece routes require higher investments in

hardware.

Figure 8 provides an overview of all results, where the notation $PPP\ m\backslash x$ indicates the solution to PPP with m machines up to the PoS and $M' = x$. Notice that no design selection dominates any other selection on all performance measures. Therefore all selections, including the benchmarking cases, are efficient, and the choice of a particular selection depends mainly on the relative importance of the measures in a given environment. For example, if short lead times and small WIP are of primary importance, then the selection generated by PPP $0\backslash 2$ should be considered. However, this selection is costly since it requires twice as many machines as the selections with the fewest machines.

Even at the same level of capacity, the synchronized solutions considerably outperform the benchmark cases in terms of WIP, lead times, and distance. In $MinOp$, the lead time is more than 13% longer than in DSP 4, the distance 125% greater, and it operates with up to 50% more WIP. Similarly, the lead time in the MPT solution is 20% (80%) longer than the lead time for DSP 5 (PPP $0\backslash 3$), the distance is 160% (400%) greater, and it operates with up to 25% (25%) more WIP than DSP 5 (PPP $0\backslash 3$) . Figure 9 indicates the reasons for these differences: In the MPT solution, each MPS initially traverses all five machines but then re-enters three machines causing an increase in WIP, distance and lead time. In the synchronized solutions, waiting occurs only to the extent that the machines are not perfectly balanced. Additional savings in the lead time for the PPP $0\backslash 3$ solution arise because the products visit fewer operations not required by them. Notably, in the PPP solution, each operation is on average performed on 80% of the products in a pallet, versus 32% in DSP 5 and 24% in MPT. Since the benchmark solutions disregard precedence constraints, they have more flexibility in balancing the lines. As a consequence, the benchmarking solutions feature shorter cycle times. In particular, the cycle time for $MinOp$ is $\approx 8\%$ shorter than the cycle time for DSP 4, and the cycle time for MPT is $\approx 4\%$ ($\approx 13\%$) shorter than the cycle time for DSP 5 (PPP $0\backslash 3$).

The performance of the synchronized flows increases further, when relaxing some of the assumptions: Suppose that setup times are not negligible. In that case, it is no longer efficient to use MPSs and the products should migrate through the system on pallets that contain larger batch sizes of the product mix. Consequently, WIP and lead time increase proportionally for all solutions, yielding still higher savings with synchronized flows. Moreover, if the setup times are operation- rather than product-specific, the performance of non-synchronized flows may deteriorate rapidly due to repeated setup of operations. In the MPT solution, operation 12 is performed during both visits at machine 2, and thus has to be setup twice. If the setup time is one unit or more, throughput of the MPT solution drops to, or below, the level of the DSP 5 solution.

Next, consider unscheduled downtimes of the machines. Since waiting lines are normal in the MPT-solution, no clear signal is sent upstream to cease production, causing further inventory pile ups as well as

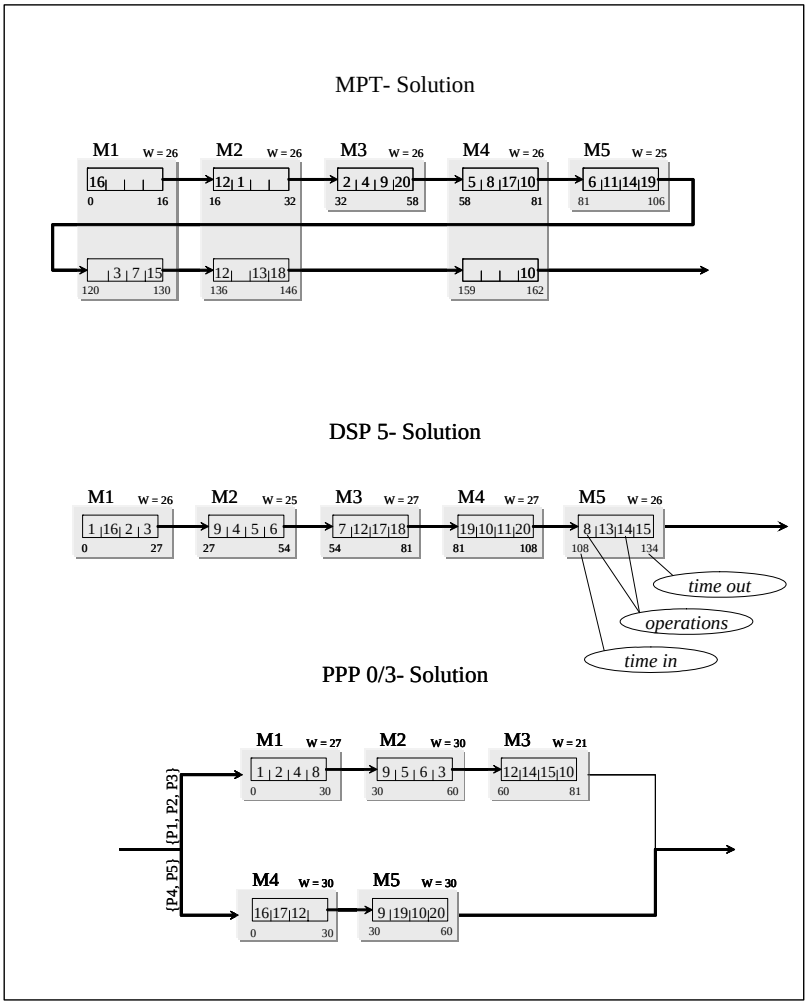


Figure 9: Process flows for several 5-machine solutions

longer lead times. Unless countered by appropriate measures, a single downtime at machine 4 of 15 units permanently increases the lead time by this amount and forces WIP up to 8 MPSs. Stochastic processing times or varying product mixes have a similar impact on the manufacturing system, and jeopardize the planned production schedule.

Finally, consider permanent changes in demand and thus in the mix ratios in the MPSs. This will not impact process compatibility, but the machines may have to be re-balanced. In the example, if the demand for product 3 doubles while the other demands remain constant, then both DSP 5 and MPT can be re-balanced to yield a maximum workload of 31 units. While this does not change the flow between the machines in DSP 5, the MPSs in MPT, after consecutively visiting each machine once, have to revisit machines 3,1, 2, and 4 in that order. If the line is not rebalanced, then maximum workload increases to 38. Otherwise, depending on the material handling system, potentially costly reconfigurations of the shop floor may be required.

5.2 Comparison of Approaches

In order to evaluate the performance of our proposed solutions for larger scale problems, we created 33 different scenarios. In the base scenario, we compared 20 different products ($|P| = 20$). For each product, a process tree is generated by the following procedure:

Algorithm 1 (Tree Generation) *INPUT:* $MinNodes$, $MaxNodes$, $P01$, $P2$, $P3$, $P4$.

Step 0: $L = \{1\}$, $m = n = 1$,

Step 1: Determine number of nodes ν *from uniform distribution between* $MinNodes$ *and* $MaxNodes$.

Do While $n < \nu$

Step 2: Determine number of sons s *for node* m *to be* $1, 2, 3,$ *or* 4 *with probabilities* $P01, P2, P3,$ *and* $P4$ *respectively.*

If $s = 1$ *and* $|L| > 1$, *then* $s = 0$ *with probability* $\frac{n}{\nu}$.

If $n + s > \nu$ *then* $s = \nu - n$.

Assign nodes $n + 1, ..., n + s$ *as sons to node* F.

Step 3: $m = m + 1$, $n = n + s$, $L = \{m, ..., n\}$.

Loop

The input determines the size and structure of the tree. In our base case we chose $MinNodes = 15$ and $MaxNodes = 25$, allowing the number of operations (or nodes) for each product to be uniformly distributed between 15 and 25. Values of $P01$ to $P4$ roughly correspond to the probability of a given node in the tree

having 0 to 4 sons. Initially, only node 1 (the root node) is in list L. Node m and n respectively denote the nodes in list L with the lowest and highest indices. Step 1 determines the number of nodes in the tree. In Step 2, node m is randomly assigned up to 4 sons, while ascertaining that the total number of nodes will not exceed ν. Step 3 updates the list by removing node m and adding its sons to the list. In our base case, we chose $P01 = 0.2$, $P2 = 0.7$, $P3 = 0.09$, and $P4 = 0.01$. The tree structure was completed by designating each node an OR node with probability $P\_OR = 0.5$ and an AND node otherwise.

This random generation of trees may give rise to the somewhat unrealistic scenario that one solution to a tree could consist of 2 operations only, while others could require more than 10 operations. To exclude such scenarios, we pruned the trees to restrict solutions to designs with $Spread = \pm 2$ operations around the median number of operations. Next, each node was randomly assigned an integer between 1 and $|N| = 50$, without assigning the same integer twice. Each integer thus represents a different type of operation. The duration of each operation is generated from a Normal Distribution with coefficient of variation $CV = 0.2$. Finally, the machine staging capacity for the base case is $c = 4$.

For this base case, we then randomly generated 100 problem instances. In each problem instance, we first computed the MPT and $MinOp$ solutions and determined the machine requirements for each. Using a binary search over the cycle time (i.e. bin size), we employed the *First Fit Decreasing* (FFD) procedure (Garey and Johnson, 1979) to assign operations to machines. To find a routing through the shop and determine the lead time, WIP and distance, we applied Graves' heuristic. We next generated the DSP and $RDSP$ solutions with the machine requirements M given by the machines required by the MPT and $MinOp$ solutions. Thus we ascertained equal machine requirements for the different solutions to enable a direct comparison of the four performance criteria (cycle time, lead time, WIP, and distance). Performing a binary search over the cycle time, we used a modified version of the FFD that accommodates precedence constraints, and assigned operations to machines. Given the cycle time and the machine requirements, the remaining performance criteria followed immediately for the simple linear flow of the $(R)\,DSP$ solutions. The relative performance of the heuristic was averaged over all 100 instances and the results are reported in Figure 10. Similar to the illustrative example above, the DSA drastically decreases lead time, WIP, and distance, at the expense of some increase in cycle time when compared to the $MinOp$ and MPT solutions. Moreover, part of this increase in cycle time is erased by $RDSA$ without much or any sacrifice along the other dimensions.

Next, we created 32 additional scenarios by varying the parameters in our experimental setup as depicted in the left hand side of Figure 11. Experiments 2 to 4 test our methodology on varying problem sizes that is on the number of products $|P|$. Experiments 5 to 10 vary the machine staging capacity c, and experiments 11

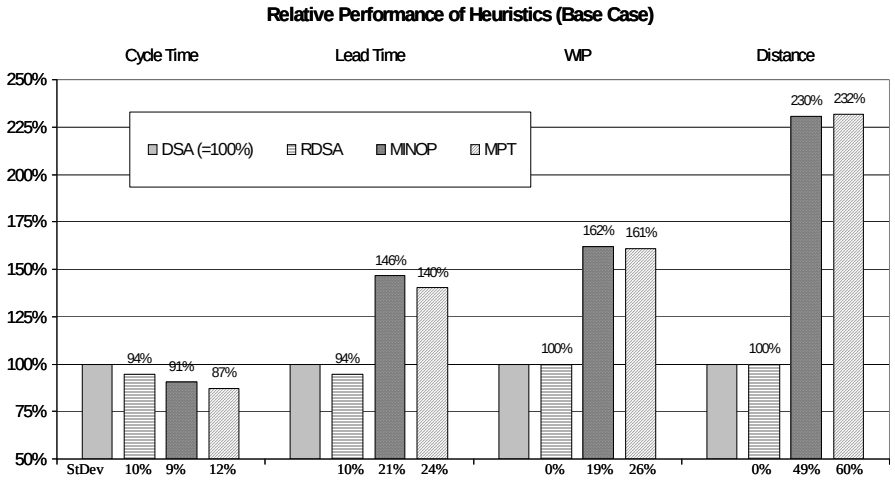


Figure 10: Average Relative Performance of Approaches for Base Case

to 16 address the size of the trees (Min_Nodes and Max_Nodes and proportionally the number of distinct operations $|N|$). By exclusively changing the number of distinct operations $|N|$ in experiments 17 to 19, we examine the impact of design similarities between the products, as a larger range of possible operations allows more distinct product designs. Experiments 20 to 23 examine the impact of more or less variable processing times, while the remaining experiments generate different tree structures. For each of these experiment settings we again ran 100 trials and computed the 4 performance criteria in the same manner as for the base case. To demonstrate implementability, we coded all procedures in rudimentary VBA within an Excel spreadsheet. Most sets of 100 experiments took less than 20 minutes to run, but none exceeded 3 hours. To compactly report the results of our trials, the right hand side of Figure 11 compares the best of the $MinOP$ and MPT solution to that of the DSA solution. Bold faced numbers indicate that the result differs by more than $\pm 10\%$ from the base case.

Over all generated instances, the DSA, on average, reduced lead times by 28%, WIP by 36% and our measure for managerial complexity, distance, by 54%, while adding 14% to the cycle time of the best benchmark solution. The results further show that the performance of the DSA solutions are very robust in respect to the problem instance. Common to all solutions are substantial improvements in lead time, WIP, and distance at the penalty of some drop in cycle time. Moreover, the magnitude of the latter seems fairly independent of the problem instance, hovering at a fairly constant 6% to 22% increase over the best of the benchmark heuristics. In contrast, distance appears most sensitive to the problem parameters, with

		Experimental Setting												Relative Performance of *DSA* to best of *MinOp* and *MPT (=100%)*			
	Experiment	|P|	c	Max_Nodes	Min_Nodes	|N|	CV	Spread	P01	P2	P3	P4	P_OR	Cycle Time	Lead Time	WIP	Distance
	1	20	4	25	15	50	0.2	2	0.2	0.7	0.09	0.01	0.5	115%	71%	62%	43%
Number of Trees	2	**10**	4	25	15	50	0.2	2	0.2	0.7	0.09	0.01	0.5	108%	69%	64%	**50%**
	3	**30**	4	25	15	50	0.2	2	0.2	0.7	0.09	0.01	0.5	121%	76%	63%	42%
	4	**40**	4	25	15	50	0.2	2	0.2	0.7	0.09	0.01	0.5	125%	75%	61%	40%
Machine Cpacity	5	**10**	**2**	25	15	50	0.2	2	0.2	0.7	0.09	0.01	0.5	113%	**82%**	**73%**	46%
	6	**10**	**6**	25	15	50	0.2	2	0.2	0.7	0.09	0.01	0.5	108%	66%	62%	**53%**
	7	**10**	**8**	25	15	50	0.2	2	0.2	0.7	0.09	0.01	0.5	108%	65%	60%	**55%**
	8	20	**2**	25	15	50	0.2	2	0.2	0.7	0.09	0.01	0.5	122%	**88%**	**71%**	40%
	9	20	**6**	25	15	50	0.2	2	0.2	0.7	0.09	0.01	0.5	111%	65%	59%	**48%**
	10	20	**8**	25	15	50	0.2	2	0.2	0.7	0.09	0.01	0.5	111%	**63%**	57%	**49%**
Size of Trees	11	**10**	**8**	**40**	**30**	**80**	0.2	2	0.2	0.7	0.09	0.01	0.5	106%	**60%**	57%	**48%**
	12	20	**8**	**40**	**30**	**80**	0.2	2	0.2	0.7	0.09	0.01	0.5	107%	**60%**	**55%**	40%
	16	20	4	**10**	**1**	**20**	0.2	2	0.2	0.7	0.09	0.01	0.5	110%	70%	66%	**61%**
	15	20	4	**15**	**5**	**30**	0.2	2	0.2	0.7	0.09	0.01	0.5	113%	72%	64%	**55%**
	13	20	4	**40**	**30**	**80**	0.2	2	0.2	0.7	0.09	0.01	0.5	112%	72%	66%	**39%**
	14	20	4	**55**	**45**	**110**	0.2	2	0.2	0.7	0.09	0.01	0.5	109%	71%	66%	**34%**
Similarity of Products	17	20	4	25	15	**65**	0.2	2	0.2	0.7	0.09	0.01	0.5	113%	74%	67%	**50%**
	18	20	4	25	15	**80**	0.2	2	0.2	0.7	0.09	0.01	0.5	110%	72%	67%	**53%**
	19	20	4	25	15	**95**	0.2	2	0.2	0.7	0.09	0.01	0.5	105%	67%	65%	46%
Processing Time	20	**10**	4	25	15	50	**0.4**	2	0.2	0.7	0.09	0.01	0.5	117%	67%	63%	47%
	21	**10**	4	25	15	50	**0.08**	2	0.2	0.7	0.09	0.01	0.5	108%	69%	66%	**51%**
	22	20	4	25	15	50	**0.4**	2	0.2	0.7	0.09	0.01	0.5	121%	74%	63%	43%
	23	20	4	25	15	50	**0.08**	2	0.2	0.7	0.09	0.01	0.5	112%	70%	63%	44%
Tree Structure	24	20	4	25	15	50	0.2	**4**	0.2	0.7	0.09	0.01	0.5	117%	73%	63%	46%
	25	20	4	25	15	50	0.2	**6**	0.2	0.7	0.09	0.01	0.5	119%	73%	62%	44%
	26	20	4	25	15	50	0.2	**20**	0.2	0.7	0.09	0.01	0.5	117%	72%	63%	44%
	27	20	4	25	15	50	0.2	2	0.2	**0.8**	**0**	0.01	0.5	116%	71%	62%	43%
	28	20	4	25	15	50	0.2	2	**0**	**1**	**0**	0.01	0.5	111%	68%	64%	45%
	29	20	4	25	15	50	0.2	2	**0**	**0**	**1**	**0**	0.5	111%	72%	65%	46%
	30	20	4	25	15	50	0.2	2	0.2	0.7	0.09	0.01	**0.25**	121%	78%	64%	45%
	31	20	4	25	15	50	0.2	2	0.2	0.7	0.09	0.01	**0.75**	121%	**79%**	65%	45%
	32	20	4	25	15	50	0.2	2	0.2	0.7	0.09	0.01	**0.9**	122%	78%	64%	44%
	33	20	4	25	15	50	0.2	2	0.2	0.7	0.09	0.01	**0.1**	122%	**80%**	65%	45%
Average														**114%**	**72%**	**64%**	**46%**

Figure 11: Overview of Experiments

reductions of the distance ranging from 39% for very small trees to 66% for very large ones. Indeed, the advantage of the DSA with respect to distance monotonically increases with an increase in tree size, machine capacity, or the number of products. With respect to lead time and WIP, the performance of the DSA also increases monotonically with the machine capacity, but we were unable to detect any further trends. In particular, the structure of the trees, that is the allowable spread, the number of sons of each node, or the prevalence of AND- or OR nodes, show no sign of impacting the quality of the $R(DSA)$ solution. The set of experiments thus illustrates that the $(R)\,DSA$ heuristics are easily implementable and applicable to industrial size problems. Moreover, the considerable advantages (and slight disadvantages) of the heuristics when compared to our benchmark solutions, fall within predictable levels. Ultimately, the results suggest that DSA solutions are particularly promising for large problem instances or high machine staging capacities. Moreover, as outlined in the preceding subsection, nonquantifiable benefits of the synchronized solutions significantly add further lure to their implementation, particularly in the presence of stochasticity.

6 Summary and Conclusion

We have provided two methodologies for addressing product and process design concurrently. The primary objective of both approaches is to select designs which allow a simple synchronized flow of products through the manufacturing system. The advantages of such systems include simple management control, high responsiveness, economical implementation, low WIP, short lead times, low managerial complexity, and high machine utilization (c.f. Ahmadi and Wurgaft, 1994a, Umble and Srikanth 1990). By considering the entire product line simultaneously, our research addresses the challenges of product variety explicitly, and contributes to the emerging literature (c.f. Ho and Tang, 1998, Pine 1999) that focuses on the complexities arising with increasing product proliferation.

In the first approach, we suggest the exact Design Selection Algorithm that encompasses all products simultaneously, enabling the same linear flow through the entire manufacturing system. The second approach partitions the product set into subsets that allow linear flows on separate sublines. Since this problem is a generalization of the first one, we present a heuristic to solve it. The resulting structure features lower WIP and shorter lead times than the unpartitioned case, but is often harder to balance. The trade-off between the two solutions should be evaluated in the context of the given environment. The computational results presented in the preceding section illustrate the implementability and computational efficiency of the DSA, and demonstrate the benefits of synchronized flow solutions as substantial and fairly constant over a large variety of different scenarios. In contrast, the downside, that is reduced cycle times, are modest throughout.

Moreover, most assumptions in this paper were made for the sake of clarity, and are not necessary for

applying either of the methods. Our methodologies can be applied as soon as process trees of the designs have been generated and are thus suitable for a wide variety of different scenarios. Furthermore, since synchronized flows are easily managed, their application is even more promising under less ideal assumptions, particularly in environments characterized by higher degrees of uncertainty.

7 References

References

[1] Ahmadi, R.H. and H. Wurgaft, "Design for Set Manufacturability" in: S. Dasu and C. Eastman (eds.), Management of Design - Engineering and Management Perspectives , Boston, Kluwer, 1994a.

[2] Ahmadi, R.H. and H. Wurgaft, "Design for Synchronized Flow Manufacturing," Management Science 40 (1994b), 1469-1483.

[3] Aneja, Y.P. and R.T. Sokkalingam, "The minimal feedback arc set problems," INFOR 42, 2, (2004), 107-112.

[4] Barthelemy, J.P., A. Guenoche and O. Hudry, "Median Linear Orders: Heuristics and a Branch and Bound Algorithm," European Journal of Operational Research 42 (1989), 313-325.

[5] Baybars, I., "A Survey of Exact Algorithms for the Simple Line Balancing Problem," Management Science 32 (1986), 909-931.

[6] Becker and Scholl "A survey on problems and methods in generalized assembly line balancing," European Journal of Operational Research 168, (2006), 694-715.

[7] Berger, I., Bourjoully, J.-M. and Laporte, G., "Branch-and Bound Algorithms for the Multi-Product Assembly Line Balancing Product," European Journal of Operations Research 58 (1992), 215-222.

[8] Blackburn, J., "Time-Based Competition," R.D. Irwin, Homewood, Illinois, 1992.

[9] Bock, S. "Using distributed search methods for balancing mixed-model assembly lines in the automotive industry," OR Spectrum 30, (2008), 551-578.

[10] Boctor, F.F., "A Multiple-rule Heuristic for Assembly Line Balancing," Journal of the Operational Research Society 46 (1995), 62-69.

[11] Charbit, P., S. Thomasse, and A. Yeo "The Mimimum Feedback Arc set Problem is NP-Hard for Tournaments," Combinatorics, Probability and Computing 16, 1, (2007), 1-4.

[12] Even, G., S. Naor, B. Schieber, and M. Sudan "Approximating minimum feedback sets and multicuts in directed graphs," Algorithmica 20 (1998), 151-174.

[13] Dar-El, E.M. and R.F. Cother, (1975). "Assembly line sequencing for model mix," International Journal of Production Research 13 (5) 463-477.

[14] Deo, N., "Graph Theory with Applications to Engineering and Computer Science," Englewood Cliffs, Prentice-Hall, 1974.

[15] Dolgui, A., B. Finel, N.N. Guschinsky, G.M. Levin, and F.B. Vernadat "MIP approach to balancing transfer lines with blocks of parallel operations," IIE Transactions 38 (2006), 869-882.

[16] Erel, E. and S.C. Sarin "A survey of the assembly line balancing procedures," Production Planning and Control 9, (1998), 414-434.

[17] Flueck, J.A. and J.F. Korsh "A Branch Search Algorithm for Maximum Likelihood Paired Comparison Ranking," Biometrika 61, (1974), 621-626.

[18] Eastman, C., "Automatic Composition in Design", in: S. Newsome et al. (eds.) Design Theory, New York, Springer, 1988.

[19] Garey, M.R. and D.S. Johnson, "Computer and Intractability - A Guide to the Theory of NP-Completeness," New York, Freeman, 1979.

[20] Graves S.C., H.C. Meal, D. Stefek and A.H. Zeghmi, "Scheduling the re-entrant flow shops", Journal of Operations Management 3 (1983), 197-207.

[21] Groetschel, M., Juenger, M. and Reinelt, G., "A Cutting Plane Algorithm for the Linear Ordering Problem," Operations Research 32 (1984) 6, 1195-1220.

[22] Hudry, O. and I. Charon, "A branch-and-bound algorithm to solve the linear ordering problem for weighted tournaments," Discrete Applied Mathematics 154, 15, (2006), 2097-2116.

[23] Hackman, S.T., M.J. Magazine and T.S. Wee, "Fast, Effective Algorithms for Simple Assembly Line Balancing Problems," Operations Research 37 (1989) 6, 916-924.

[24] Ho, T.-H. and C. Tang "Product Variety Management – Research Advances," Boston, Kluwer 1998.

[25] Hochbaum, D. S. and A. Levin "Methodologies and algorithms for group-rankings decision," Management Science 52, 9, (2006), 1394-1408.

[26] Hofmann, T.R., "Eureka: A Hybrid System for Assembly Line Balancing," Management Science 38 (1992) 1, 39 47.

[27] Hubka, V. and W.E. Eder "Theory of Technical Systems," Berlin, Springer, 1988.

[28] Ignall, E.J. "A review of Assembly Line Balancing," Journal of Industrial Engineering, 16 (1965), 244-254.

[29] Karmakar, U.S., S. Kekre and S. Kekre "Lotsizing in multi-item multi-machine job shops," IIE Transactions, 17 (1985) 3, 290 -298.

[30] Kaas, R., "A Branch and Bound Algorithm for the Acyclic Subgraph Problem," European Journal of Operational Research 8 (1981), 355-362.

[31] Karp, R.M., "Reducibility among Combinatorial Problems," in Miller and Tatcher (eds), Complexity of Computer Computations, New York, Plenum, 1972.

[32] Korte, B. and W. Oberhofer, "Zur Triangulation von Input-Output Matrizen," Jahrbuch für Nationalökonomie und Statistik 182 (1969), 398-433.

[33] Lawler. E.L. "A comment on minimum feedback arc sets," IEEE Transactions on Circuit Theory 11, (1964), 296-297.

[34] Lee, H.F. and R.V. Johnson, "A Line-Balancing Strategy for Designing Flexible Assembly Systems," International Journal of Flexible Manufacturing Systems 3 (1991), 91-120.

[35] Lee, H.L. and Tang, C.S., "Modeling the Costs and Benefits of Delayed Product Differentiation," Management Science 43 (1997) 1, 40-53.

[36] Lenstra, J.K., A.H.G. Rinnooy Kan and P. Brucker, "Complexity of Machine Scheduling Problems," Annals of Discrete Mathematics 1 (1977), 343-362.

[37] Lin, G. Y-J., and J.J. Solberg "Effectiveness of Flexible Routing," The International Journal of Flexible Manufacturing Systems 3 (1991), 189-212.

[38] Manber, U. "Introduction to Algorithms," Reading, Addsion-Wesley, 1989

[39] Marple, D., "The Decisions of Engineering Design," IEEE Transactions on Engineering Management, 1961, 55-71.

[40] Mastor, A.A. "An Experimental Investigation and Comparative Evaluation of Production Line Balancing Techniques," Management Science 16 (11) (1970), 728-746.

[41] McMullen and Tarasewich "Using ant techniques to solve the assembly line balancing problem," IIE Transactions 35, (2003), 605-617.

[42] Nemhauser, G. and L. Wolsey, "Integer and Combinatorial Optimization," New York, Wiley, 1988.

[43] Pine II, B.J., "Mass Customization: The New Frontier in Business Competition" Boston, Harvard Business School Press, 1999.

[44] Rembold, U., C. Blume and R. Dillman, "Computer Integrated Manufacturing," New York, Dekker, 1985.

[45] Salveson, M.E. "The Assembly Line Balancing Problem," Journal of Inductrial Engineering, 6, (1955), 18-25.

[46] Sedgewick, R. "Algorithms," Reading, Addison-Wesley, 1988.

[47] Seymour P.D. "Packing directed circuits fractionally," Combinatorica 15 (1995), 281-288.

[48] Shirley, G.V., "Models for Managing the Redesign and Manufacture of Product Sets," Journal of Manufacturing and Operations Management , 3 (1990), 85-104.

[49] Stecke, K.E., "Design, planning, scheduling, and control problems of flexible manufacturing systems," Annals of Operations Research , 3 (1985), 51-60.

[50] Stoll, H.W., "Design for Life-Cycle Manufacturing," in: J.E. Ettlie and H.W. Stoll, (eds.) Managing the design-manufacturing processes, New York, McGraw-Hill, 1990.

[51] Ulrich, K.T. and S.D. Eppinger, "Product Design and Development," New York, McGraw-Hill, 1995.

[52] Umble, M.M. and M.L. Srikanth, "Synchronous Manufacturing - Principles for World Class Excellence," Cincinnati, South-Western, 1990.

[53] Victor, H., "Trennen" in: Grote, K.-H. and Feldhusen, J. (eds.), Dubbel - Taschenbuch für den Maschinenbau, Berlin, 21. Aufl., Springer, 2005.

[54] Wang, Z.-S., "Improved Algorithm for minimal feedback arc set based on stochastic evolution," Computer Enginering and Applications, 44, 17, (2008), 45-48.

[55] Whitney D.E., "Manufacturing by Design," Harvard Business Review, July-August (1988).