The International Journal of Flexible Manufacturing Systems, 4 (1992): 183–209

An Iterative Approach to System Setup Problems in Flexible Manufacturing Systems

YEONG-DAE KIM
Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, Cheongryang P.O. Box 150, Seoul, Korea

CANDACE ARAI YANO
Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan 48109-2117

Abstract. System setup problems in flexible manufacturing systems deal with short-term planning problems such as part type selection, machine grouping, operation assignment, tooling, fixture and pallet allocation, and routing. In this article, we consider three of the subproblems: part type selection, machine grouping, and loading. We suggest a heuristic approach to solve the subproblems consistently with the objective of maximizing the expected production rate. The proposed procedure includes routines to generate all possible machine grouping alternatives for a given set of machines, to obtain optimal target workloads for each grouping alternative, and to allocate operations and tools to machine groups. These routines are executed iteratively until a good solution to the system setup problem is obtained. Computational experience is reported.

Key Words: system setup problems, part type selection, machine grouping, iterative solutions

1. Introduction

A flexible manufacturing system (FMS) is an automated manufacturing system consisting of numerically controlled machines capable of performing multiple functions, linked together by a material handling system, all controlled by a computer system. The aim of an FMS is to achieve the efficiency of automated high-volume mass production while retaining the flexibility of low-volume job shop production.

In order to achieve this efficiency, a variety of decisions must be made. Such decisions include designing the system, planning production for the upcoming production period, logically and physically configuring the system to support the production plan, scheduling operations in detail, and controlling and monitoring the system to ensure that the system is running as planned. (See Stecke 1985 and Kiran and Tansel 1986 for classification and description of the various decisions.) Once the system is designed and the production plan for the upcoming period is established, the remaining decision problems can be treated as a large scheduling problem with various resource constraints. However, such a problem is extremely difficult to solve. As an alternative, the large scheduling problem can be divided into two parts: one that uses aggregate information and deals with decisions that are difficult to change in real time, and another that uses detailed information to make real-time decisions. The former, which is called the system setup problem, has the goal of logically and physically configuring the system so the real-time scheduling problem is easier to solve, by ensuring that resources and requirements are consistent with one another.

The system setup problem consists of several subproblems, which Stecke (1983) defines as follows:

1. *Part type selection problem:* From a set of part types for which production requirements (or production orders) are specified, choose a subset of part types for immediate and simultaneous manufacture. Due dates may be considered. (This problem must be solved when all the parts cannot be manufactured with a single tool setup because of limited tool magazine capacities.)
2. *Machine grouping problem:* Partition the machines of each type into groups, where the machines in each group are identically tooled. (Pooling machines improves most system performance measures.)
3. *Loading problem:* Allocate the operations and cutting tools with the selected part types to the machines or machine groups.
4. *Production ratio problem:* Determine the relative production ratios in which the set of selected part types should be processed.
5. *Resource allocation problem:* Allocate the limited number of pallets and fixtures among the selected part types.

There is strong interdependence among these problems. The second through fifth problems cannot be solved until a set of part types has been selected. On the other hand, the solution to the part type selection problem may make the other problems infeasible. Therefore, the part type selection problem should be solved in such a way that the other problems are feasible to solve.

A considerable amount of research has been done on the system setup problem. Some approaches try to solve the subproblems simultaneously to find a global optimum, while others try to solve them separately, sometimes in a sequential fashion, to reduce computational complexity. First, we will review papers that use the former approach.

Chakravarty and Shtub (1987) present a nonlinear integer programming formulation for tool grouping and loading. In this formulation, they consider various constraints and parameters, such as machining time capacity, pallet availability, tool magazine capacity, inventory costs, and tool magazine setup costs. To avoid excessive computation, they develop a heuristic procedure based on the formulation and give an example. Rajagopalan (1986) formulates the part type selection, loading, and production ratio problems as a mixed-integer linear program under the assumption that each part type either has only one operation on a machine type or has all its operations on a machine type assigned to the same machine. Linearity is achieved by redefining certain variables. He also presents heuristic algorithms and computational results. O'Grady and Menon (1984, 1985, 1987) present goal-programming approaches to the system setup problem. In their mixed-integer programming formulation, they consider various goals including some related to tool magazine capacity, machine time capacity, due dates, alternative process routes, and expediting of certain orders. They give an example problem and solve it without pursuing a global optimum to avoid computational limitations of the formulation. The program was stopped when an "acceptable" integer solution was obtained.

As noted in the above papers, treating the system setup problem as a whole is computationally intractable. As an alternative, sequential decision methods, such as the one proposed in concept (but not implemented) by Stecke (1983), can be used. In the following, we will review the literature on the individual subproblems: the part type selection problem, the grouping problem, and the loading problem.

There are several approaches to the part type selection problem: the group technology approach, the sequential decision approach, and the constraint-directed approach. The *group technology approach* uses the concept of group technology in grouping parts; that is, parts with similar characteristics are grouped together. Kusiak (1984), Kumar et al. (1986), Chakravarty and Shtub (1984) use this type of approach. In the *sequential decision approach*, parts are included sequentially to maximize the probability of a desirable outcome or to maximize dollar savings. This approach is discussed in Whitney and Gaul (1984) and Whitney and Suri (1984). The *constraint-directed approach*, which is the most recent, considers due dates, processing capacities of the machine tools, and tool magazine capacities (Hwang 1986). The approaches of Kiran and Tansel (1986), Rajagopalan (1986), O'Grady and Menon (1984, 1985, 1987), and Chakravarty and Shtub (1987) can also be classified as constraint-directed, even though the part type selection problem is considered along with other subproblems in a single formulation or solution procedure. Using the last type of approach, Stecke and Kim (1986) suggest a method called a *flexible approach*, which considers dynamic events such as changes in the part mix and machine breakdowns. This approach is compared with the approaches of Whitney and Gaul (1984), Hwang (1986), and Rajagopalan (1986) in Stecke and Kim (1988).

Grouping machines increases system performance by decreasing the probability that a part will be blocked (i.e., that no machine will be available for the next operation). Having more than one machine in a group is one way to allow multiple routes for some parts. Using a closed queueing network model, Stecke and Solberg (1985) prove that fewer groups are better when the goal is to maximize the expected production rate. An approach to grouping is discussed in Stecke (1983).

The loading problem is studied initially by Stecke and Solberg (1981). They explore loading and scheduling strategies for a particular FMS. Later, Stecke (1983) formulates the FMS loading problem as a nonlinear mixed-integer program and solves it through linearization techniques. A branch-and-bound algorithm is developed by Berrada and Stecke (1986) for this formulation with the objective of balancing the workloads. This algorithm is modified to accommodate the objective of finding the best (unbalanced) workloads for machines groups of unequal sizes in Kim and Yano (1987). A branch-and-bound algorithm is presented by Kim and Yano (1989) for both of the objectives. Several heuristic algorithms are described by Stecke and Talbot (1985), but are not tested. Kim and Yano (1990) present a heuristic approach using multipass algorithms for multidimensional bin-packing problems and report computational results. Different formulations and heuristic procedures are suggested by Ammons et al. (1985), Lashkari et al. (1987), and Shanker and Tzen (1985). Also, the loading problem is studied in conjunction with other FMS-related problems in other research (Greene and Sadowski 1986; Rajagopalan 1986). A hierarchical approach for the grouping and loading problem is suggested by Stecke (1986).

The main contribution of this article is the *systematic integration* of a set of efficient procedures to solve the order (or part type) selection, grouping, and loading problems simultaneously. Because of space considerations, the details of the procedures appear elsewhere (Kim 1988; Kim and Yano 1987, 1989, 1990). These procedures were designed with the intent of integrating them, and the purpose of this article is to explain how the integration was accomplished. Thus, it is not necessary for the reader to understand the technical details of each procedure. It is only necessary to understand what each procedure does, and its inputs and outputs. To this end, we briefly describe each of the procedures as the article progresses.

Since the procedures were designed with integration in mind, they can be linked and used in a variety of different ways, depending upon the circumstances and goals of the particular manufacturing system. We concentrate on a particular implementation to illustrate the main features of the procedures and the links among them, but later describe what other possible implementations would entail.

Our approach to this problem differs from earlier approaches in that the procedures themselves are modular. Thus, large monolithic formulations are not essential. For the same reason, the approach is flexible, since different objectives can be considered. The modularity also contributes to the speed of computation and the quality of the solutions, because each module deals with a computationally tractable piece of the overall problem, and as a result, it is possible to solve each piece extremely well.

Our approach also differs from proposed solution approaches in which the individual subproblems are solved in sequence, with earlier subproblems providing constraints to later subproblems. We often go beyond finding the "optimal" solution to a subproblem; we typically generate a list of the best, and in some cases, all feasible alternatives. In this way, we try to avoid imposing constraints on the solution until the "downstream" impact can be evaluated. Most existing solution approaches involve solving optimization subproblems that provide only one solution. Thus, if a downstream subproblem is found to be infeasible, an upstream optimization problem must be modified, but the appropriate types or extent of changes are not always clear. We discuss this point in more detail later. In our approach, the upstream feedback is very specific.

Thus, our contribution is not the introduction of new problems, nor are we the first to suggest that the various subproblems should be solved in an integrated fashion, or that iterating among the subproblems to achieve a consistent solution might be worthwhile. Rather, our contributions are 1) a strategy for integrated solution in which iteration is performed in a well-defined and systematic manner, and 2) a demonstration that this strategy is computationally feasible and performs well. It is important to point out that, to our knowledge, this is the first completely automated implementation of a procedure to simultaneously solve the part type selection, machine grouping, and loading parobIems.

Despite the generality of the overall approach, there are some limitations of our procedure. First, it is designed principally for systems in which tool changes are done in a *tool setup period*, during which all of the equipment is shut down and relevant tools are changed. In the past several years, some companies have moved toward changing tools on a more dynamic basis. However, even recently, one of the authors has become aware of three highly automated "flexible" systems in Fortune 50 companies in which the entire system is shut down when tool changeovers are done. The companies cite various reasons, including the time required for calibrating equipment in very high precision applications, and increases in human error in setting up the tools and equipment when only a portion of the tools is changed. Thus, the use of tool setup periods appears to be a common mode of operation. We should note, however, that it should be possible to imlement our general approach even when tool changes are done dynamically, but it may be more difficult to do so.

Second, the procedures are designed with the assumption that tool magazine capacities represent constraints that must be considered. In each of the three cases cited above, tooling considerations are clearly a constraint; otherwise there would be no need for changeovers. Moreover, our procedures can be implemented even if tool magazine capacities are relatively large.

Third, we assume that there are multiple types of machines, and that each processing operation is preassigned to a machine type. In practice, most process plans specify a single preferred routing, and our assumption reflects this practice. However, alternate process plans, or mixes of various process plans for a particular part, can be accommodated by our procedure through modifications of the input data. The case of one machine type is a simplified version of the problem, and can be handled by our procedure.

Fourth, we assume that maximization of throughput (expected production rate) is one of the explicit goals of the system, or that maximizing throughput (for some selected mix of parts) over the short term can help to accomplish other goals, such as meeting due dates or reducing operating costs. A throughput maximization objective may be advantageous even in a system with low overall utilization. Indirect labor costs can be reduced by compressing the work into two shifts instead of three, or one shift instead of two. In a highly utilized system, the advantages of throughput maximization are self-evident.

Each order is defined by the part type, the order quantity, and a unique priority, which we explain in more detail later. Multiple orders for the same part type can be distinguished by their priorities. In our procedure, we do not select *part types* for processing; instead, we select *orders* for processing, and we may select multiple orders of the same part type simultaneously. Each order is treated as an indivisible entity in our approach. This means that once the processing of an order starts, that order must remain in the set of selected orders until the processing of that order is completed.

We consider a static problem with a given set of orders. For the time horizon required to complete these orders, the overall efficiency of the system is affected by the number of tool setups as well as the production rate during production periods. If no interruptions occur, we can partition the various orders into subsets, each with its own tool setup and production period, and operate the system as planned. However, in reality, machine breakdowns, arrivals of urgent orders, and other interruptions occur. Because of this, the focus of our research is on the development of a procedure to plan the immediate production period. The procedure would be repeated whenever the current mix of part types must be changed in response to completion of an order, the introduction of an urgent order, or a machine breakdown. In essence, we are solving an aggregate planning problem, but with more careful attention to resources, and with a shorter time frame than in the traditional production planning literature. Our procedure would generate updated solutions in much the same way that aggregate production plans are modified on a rolling horizon basis.

It is important to remember that we are not solving the detailed scheduling problem here. Moreover, with the exception of general studies of dispatching rules (e.g., Panwalkar and Iskander 1977; Blackstone et al. 1982; Elvers and Taube 1983; Russel et al. 1987; Vepsalainen and Morton 1987; Kim 1987), there is little research on the problem of scheduling multiple orders on multiple machines when each order has multiple parts (which can be processed independently of one another), but with a common due date for the parts within an order. Further research is needed on scheduling problems of this type. As a consequence of this lack of good scheduling procedures, it is difficult to determine exactly when each order will be completed for any given "aggregate" plan. Thus, for practical purposes, it is necessary to specify priorities of the various orders a priori. This might be done using an earliest due date (EDD) or slack-based rule when due dates are critical, or some other appropriate procedure. We might choose to deviate from this priority sequencing to consolidate orders for the same part type or to group orders with greater commonality of tool requirements (both of which would tend to reduce the number of tool changeovers), or to group orders with complementary machine usage (which would tend to decrease the makespan for a given set of orders). Our procedure is efficient enough so that many different modifications of an initial priority scheme can be evaluated. Throughout the remainder of this article, for ease of exposition, we will assume that the priority of each order is given and is inviolable. That is, a higher-priority order must be started before orders with lower priorities. In this situation, because each order has a unique priority, the problem reduces to one of determining how many orders to include in the first production period, and how to group the machines and assign operations to them.

In the next section, a formulation of the system setup problem is given. In section 3, an iterative approach is suggested, and a corresponding solution procedure is described. Results of computational experiments are reported in section 4. Section 5 concludes the article with a summary and comments on future research directions.

2. A formulation of the problem

Although we do not use the following formulation directly in solving the problem, it provides a formal statement of the problem addressed in this article. First, we present notation.

Indices

i: 1, 2, ..., $\|I\|$	Setups
j: 1, 2, ..., $\|J\|$	Orders (the index shows priority of an order, i.e., order j_1 should be included in tool setup no later than for order j_2, where $j_2 > j_1$)
v: 1, 2, ..., $\|V\|$	Machine types
w: 1, 2, ..., M_v	Machine groups (for machine type v)
q: 1, 2, ..., $\|Q\|$	Operations

Sets

Q_{jv}:	Set of operations for the jth order on machine type v
Q:	Set of all operations

Parameters

B:	A very large number
S:	Setup times

$c_{qt} \in \{0, 1\}$	= 1 if operation q requires tool t
d_v:	Tool magazine capacity (slots) of machine type v
s_t:	Number of slots occupied by tool t
p_q:	Processing time of operation q
M_v:	Number of machines of type v

Integer decision variables

k_{vwi}:	Number of machines in wth group of machine type v in ith setup

Binary (0–1) *decision variables*

Z_i	= 1, if there is an ith tool setup (and therefore an ith production period)
z_{ji}	= 1, if order j is included in the ith setup
x_{qvwi}	= 1, if operation q is assigned to machine group w of type v in the ith setup (production period)
y_{tvwi}	= 1, if tool t is assigned to machine group w of type v in the ith setup

Now, we present a nonlinear integer programming formulation of the problem. The objective is to minimize the time needed to complete all orders.

$$\text{Minimize} \sum_{i=1}^{|J|} \{SZ_i + L_i \,/\, (T_i \sum_v M_v)\}$$

subject to

$$\sum_{j=1}^{|J|} z_{ji} \leq BZ_i, \qquad \forall j \tag{1}$$

$$\sum_{i=1}^{|J|} z_{ji} = 1, \qquad \forall j \tag{2}$$

$$\sum_{i=1}^{n} z_{ji} \leq \sum_{i=1}^{n} z_{j-1,i}, \qquad n = 1, \ldots, |J|, \qquad \forall j \tag{3}$$

$$\sum_{w=1}^{M_v} k_{vwi} = M_v \qquad \forall v, i \tag{4}$$

$$\sum_{q \in Q} x_{qvwi} \leq B\, k_{vwi} \qquad \forall\ v, w, i \tag{5}$$

$$c_{qt}\, x_{qvwi} \leq y_{tvwi} \qquad \forall\ q, t, v, w, i \tag{6}$$

$$\sum_{t} s_t\, y_{tvwi} \leq d_v \qquad \forall\ v, w, i \tag{7}$$

$$X_{vwi} = \sum_{j=1}^{|J|} \sum_{q \in Q_{jv}} p_q\, x_{qvwi} \qquad \forall\ v, w, i \tag{8}$$

$$L_i = \sum_{v} \sum_{w} X_{vwi} \qquad \forall\ i \tag{9}$$

T_i is the throughput for the ith production period (i.e., ith setup) obtained from the CQN using X_{vwi}, k_{vwi} $\quad (T_i = 1 \text{ if } L_i = 0)$ (10)

$$Z_i,\ z_{ji},\ x_{qvwi},\ y_{tvwi} \in \{0, 1\} \qquad \forall\ q, t, v, w, j, i \tag{11}$$

$$k_{vwi} \geq 0, \quad \text{integers} \qquad \forall\ v, w, i \tag{12}$$

Constraint set (1) ensures that orders are included in the ith tool setup only when the ith tool setup exists. Constraints (2) and (3) relate to part type selection: each order can be included in one and only one tool setup, and orders with higher priorities must be selected no later than those with lower priorities. Constraints needed to generate all possible configurations for a given number of machines for each machine type can be written as equation (4) along with the integrality constraints (12).

The sets of constraints (5) through (7) pertain to the loading problems, where tool magazine capacities of machine tools must be considered. These constraints consider the impact of *tool commonality* (i.e., reduction of total tool slot requirements when operations requiring common tools are assigned to the same machine group). For any solution to a loading problem, we use equations (8) and (9) to calculate the system throughput given in definition (10). Since the throughput cannot be expressed in closed form, we have not stated definition (10) in mathematical form. The system throughput can be calculated by a numerical method given by Stecke and Solberg (1985) for a product-form closed queueing network (CQN) model.

3. The iterative procedure

Since the system setup problem in its general form is computationally intractable, one may choose to solve portions of the problem sequentially. In a *sequential* approach, the individual subproblems are solved one at a time, with one subproblem using the solution of the previous subproblems as inputs, usually as constraints. If, however, constraints generated at one step make it infeasible to solve any of the remaining subproblems, human intervention may be required to resolve the difficulty. For existing sequential approaches, a different solution procedure or a different set of parameters would need to be used to modify the problematic decisions. We are not aware of any attempts to formalize or automate this feedback process within these sequential approaches.

As an alternative, we suggest a *formally* iterative approach to the system setup problem. In this approach, the subproblems are solved in such a way that the final solution is feasible for all three subproblems. The nature of the iteration is such that if the solution to a loading problem is not feasible, the next machine grouping alternative in a well-defined list is considered. Similarly, if for any machine type, there is no machine grouping that produces a feasible solution for the associated loading problem, a smaller number of part types is considered. Consequently, the procedure solves a loading problem for different machine groupings and for different sets of part types until we obtain a solution that satisfies the tool magazine capacity constraints. Therefore, we can obtain solutions for the three subproblems at the same time.

While the iterative approach is better than sequential approaches in the sense that it will provide a feasible solution without human intervention, it requires more computation time. Because of this computational burden, in some parts of the procedure we will use heuristic methods rather than optimal solution methods. (There is no known polynomially bounded algorithm even for some individual subproblems.) This intractability stems from the fact that each machine is quite versatile and capable of performing many different operations, the system can machine several part types simultaneously, and each part may have alternate routes through the system.

Before giving a detailed description, we give an overview of the iterative procedure. The first portion of the procedure has the goal of preparing a ranked list of expected production rates (system throughput values) for the given set of machines. In order to determine the maximum system throughputs, it is necessary to find the optimal (continuous) workload allocation for each machine grouping alternative. The maximum possible throughputs and their associated workload allocations serve two roles. First, the maximum possible throughputs are used to rank the machine grouping alternatives. Second, the optimal continuous workload allocations serve as targets in the loading problem, as we explain later. It is important to point out that this list is generated only once for a given set of machines, and there is no need to create another one unless the set of machines changes, or if one or more is expected to be down for a long time. Indeed, if machine failures are an important factor, it would be possible to prepare ranked lists for commonly occurring states of the system, where a state is defined by the number of operating machines of each type.

The next major portion of the procedure is to determine priorities. As mentioned earlier, in this article, we assume that the priorities are given and that the priorities imply strict precedence in terms of the *starting times* of the orders. Although we assume that the priorities are given, we actually use a method developed by Kim (1988) in which a very aggregated version of the scheduling problem is used as the basis for setting priorities.

What remains to be decided is the number of orders in the first production period. This constitutes the third major portion of the procedure. We later present two possible search methods for doing this. For each candidate number of orders, we solve the loading problem for a number of machine grouping alternatives in order to identify the machine grouping (and corresponding loading plan) with the highest system throughput. Therefore, by the time the decision has been made regarding the number of orders, the related grouping and loading plans have already been developed.

We present two methods for deciding the number of orders in the imminent production period. One (method 1) has the goal of maximizing the number of orders in the first production period. This goal might be useful when tool changeovers are time-consuming, since this approach tends to increase the time between changeovers. Figure 1 depicts method 1. The other method (method 2) has the goal of maximizing system throughput during the first production period. In method 2, successively larger sets of orders are considered, and their system throughput values are compared. These comparisons are continued until orders no longer can be added without violating capacity constraints. This method would be more advantageous when tool changeovers are short in comparison to production periods. Method 2 is depicted in figure 2. Method 2 explicitly pursues the maximum throughput by comparing all alternative numbers of orders in the first production period. On the other hand, method 1 tries to achieve the goal with a surrogate objective, maximizing the number of orders included in a single production period. This surrogate objective is used in Hwang (1986). Note that a bisection search method can be used to save computation time in method 1, while we have to consider all possible alternatives in method 2.

Observe that this procedure differs from sequential approaches. In sequential approaches, when an upstream decision is made (for example, part-type selection is an upstream decision for the grouping and loading problems), it is fixed for downstream problems. If the decision leads to a constraint set that is infeasible or unsatisfactory for downstream problems, the initial decision must be modified. In a sequential procedure, there is no formal mechanism for providing this feedback, nor is there a systematic means of modifying the decision. In this iterative procedure, the feedback loop is always specific, as is the means of modifying the problematic decision(s). Note the feedback loops in step 6 of both methods 1 and 2. With this method, infeasibility can be resolved automatically without the intervention of the user if a feasible solution exists.

In the remainder of this section, each step of the procedure is discussed in more detail, although technical descriptions of the algorithms used in each step are not given. We will start with method 1. Recall that the goal is to maximize the number of orders selected.

Step 1 (Generating possible machine group configurations). If there are two or more machines of the same type in an FMS, they can be partitioned into groups. The following problem (*PI*) must be solved *for each machine type* to generate all possible configurations for a given number of machines, M, where $(k_1, k_2, \ldots, k_m)$ denotes a configuration with k_i machines in the ith group for $i = 1, \ldots, m$. The subscripts for machine types and setups used in section 2 are dropped for simplicity. Here, we must take into account that some partitions are effectively identical (for example, the configurations (1, 1, 2) and (1, 2, 1) are the same). This is accomplished by imposing the first constraint set in the formulation on page 195.

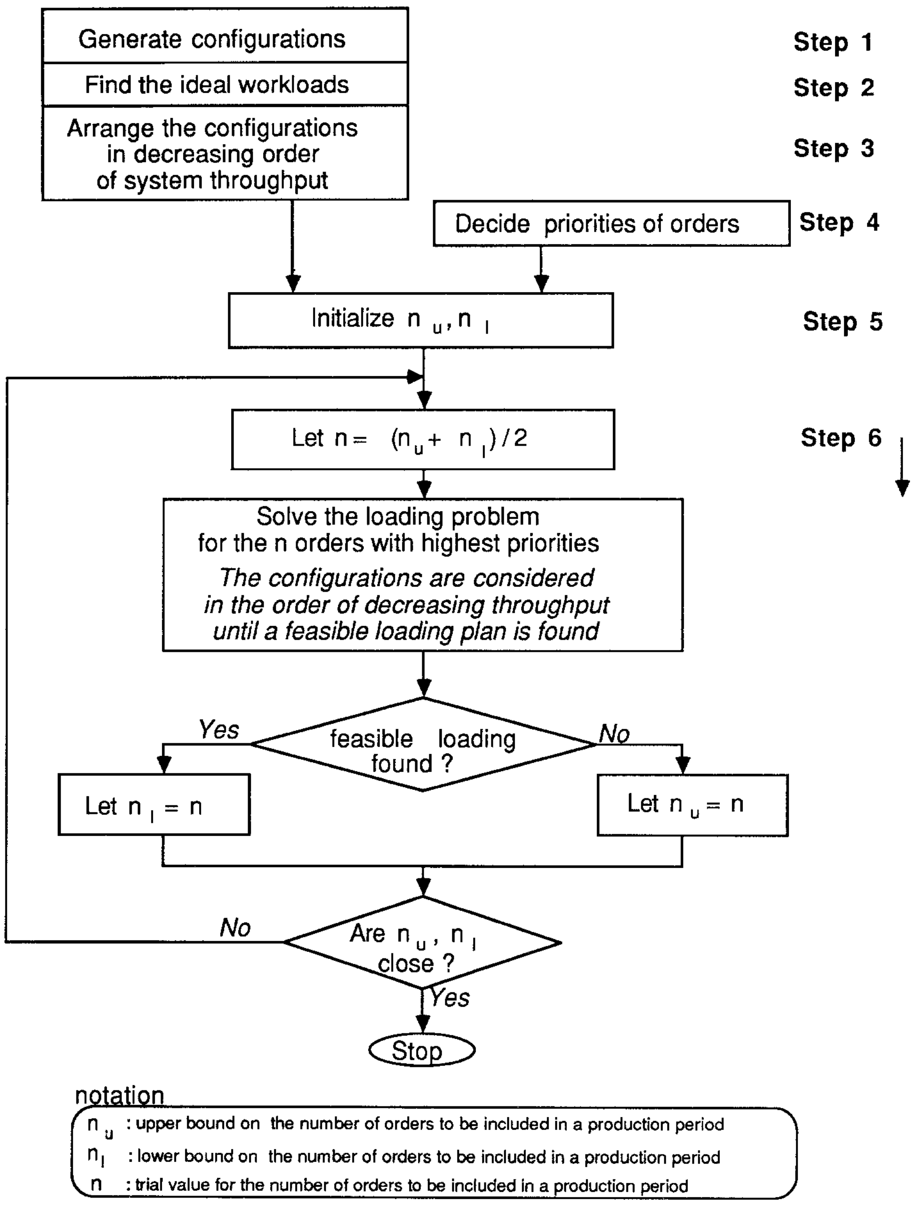
Generate configurations
Find the ideal workloads
Arrange the configurations in decreasing order of system throughput
Step 1
Step 2
Step 3
Decide priorities of orders
Step 4
Initialize n_u, n_l
Step 5
Let $n = (n_u + n_l)/2$
Step 6
Solve the loading problem for the n orders with highest priorities
The configurations are considered in the order of decreasing throughput until a feasible loading plan is found
feasible loading found ?
Yes
No
Let $n_l = n$
Let $n_u = n$
Are n_u, n_l close ?
No
Yes
Stop
notation
n_u : upper bound on the number of orders to be included in a production period
n_l : lower bound on the number of orders to be included in a production period
n : trial value for the number of orders to be included in a production period

Figure 1. The iterative procedure (method 1).

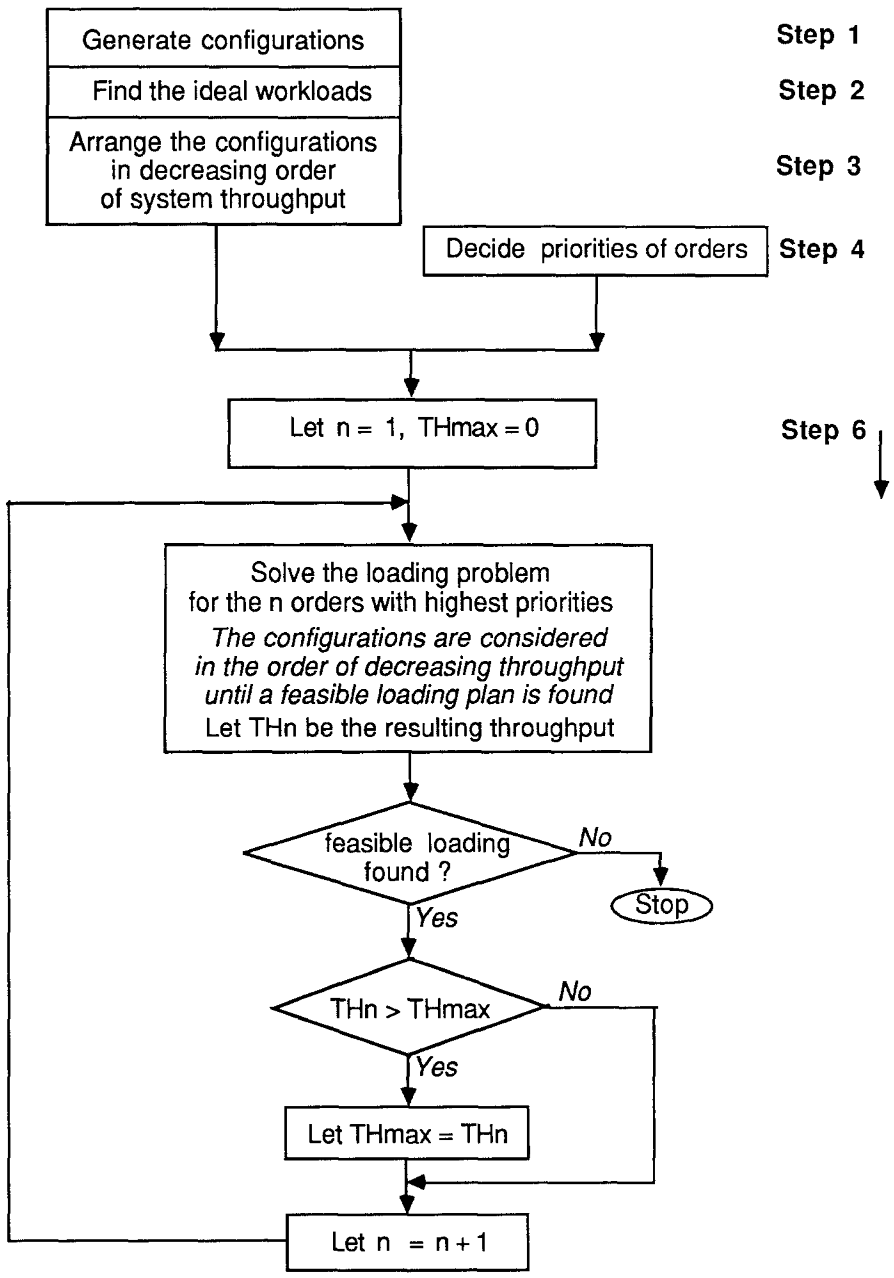
Generate configurations
Find the ideal workloads
Arrange the configurations in decreasing order of system throughput
Decide priorities of orders
Step 1
Step 2
Step 3
Step 4
Let n = 1, THmax = 0
Step 6
Solve the loading problem for the n orders with highest priorities
The configurations are considered in the order of decreasing throughput until a feasible loading plan is found
Let THn be the resulting throughput
feasible loading found ?
No
Stop
Yes
THn > THmax
No
Yes
Let THmax = THn
Let n = n + 1

Figure 2. The iterative procedure (method 2).

(P1) Find all configurations $(k_1, k_2, \ldots, k_m)$, for $m = 1, \ldots, M$

such that

$$k_1 \leq k_2 \leq \ldots \leq k_m,$$

$$\sum_{i=1}^{m} k_i = M,$$

$$k_i > 0, \qquad \text{for } i = 1, \ldots, m.$$

The number of possible configurations increases very quickly as the number of machines increases. For example, for 10, 20, 30, and 40 machines, the total numbers of possible configurations are 42, 627, 5604, and 37,338, respectively. However, most real FMSs have a small number of machines of the same type, so the number of configurations will be limited. See Kim (1988) for an efficient procedure to generate all configurations. The total number of *system* configurations is the product of the numbers of machine group configurations.

Step 2 (Finding the ideal workloads allocation). When a configuration is balanced, i.e., when there is an equal number of machines in each group, balancing workloads can minimize in-process inventories (Shanthikumar and Stecke 1986) as well as maximize the system throughput (Stecke and Morin 1985). For an unbalanced configuration, however, the system throughput can be maximized by a particular assignment in which the workload per machine is unbalanced across groups (Stecke and Solberg 1985; Stecke and Kim 1989). We refer to the continuous workload allocation that maximizes system throughput as the "ideal workloads."

In this step, we need to solve the following problem for all v and i, given k_{vwi}:

(P2) Maximize T_i

subject to

$$\sum_{w} IX_{vwi} = M_v$$

T_i is the throughput obtained from the CQN using XI_{vwi}, k_{vwi}

where IX_{vwi} denotes the ideal workload of machine group w. Here, the ideal workload is expressed as a relative value normalized in such a way that the sum of the values is equal to the number of machines.

In our approach, we use the throughput expression for a closed queueing network (CQN) given by Stecke and Solberg (1985). We use a *numerical steepest ascent (NSA) method* to find the workload allocation that maximizes system throughput. This method is similar to the steepest ascent method (Zangwill 1969; Avriel 1976), except that it uses discrete gradients (differences) and uses a numerical method to estimate these.

The NSA method is especially useful for our problem, since there are no closed-form expressions for the gradients, and even the system throughput for a given workload vector must be calculated by a numerical method such as that developed by Buzen (1973). Thus, the only feasible approach is to estimate gradients numerically. Detailed description of the method and the result of computational tests appear in Kim (1988).

Step 3 (Ranking system configurations). After the ideal workloads for each configuration and the corresponding system throughputs (considering all machine types) are calculated, the configuration alternatives are arranged in decreasing order of system throughput and are stored in a list in that order. The result is an ordered list of configurations, their ideal workloads, and corresponding maximum throughput values that can be used in every system setup phase. The lists can be stored in auxiliary computer memory space and retrieved whenever needed.

Step 4 (Setting order priorities). Since we assume that priorities are given, techniques to set priorities are beyond the scope of this article. However, it is useful to discuss why it is so difficult to *optimize* the priorities at this stage. Recall that the grouping and loading problems have not been solved at this point. Thus, the possible routings for each part cannot be determined. As a result, it is impossible even to *evaluate* a priority ranking with respect to any objective function, much less to determine an optimal priority ranking. Recall, however, that we use a procedure that uses due date information to specify a unique, inviolable priority for each customer order.

Step 5 (Initialization of lower and upper bounds on n). In this step, we give initial values for a lower bound (n_l) and an upper bound (n_u) on the number of orders that can be included in a single production period. These bounds can be specified easily by considering the space required by the tools, the number of tool slots needed for each order, and the tool magazine capacities of the machines.

To compute the initial lower bound, we assume there is only one group for each machine type. Therefore, the number of tool slots available for a machine type is the tool magazine capacity of a single machine of the type. The lower bound is obtained by including orders one by one until the number of tool slots required (the sum of the slots needed for these orders) first exceeds the number available in any type. Note that the tool savings that would result from orders sharing common tools are not considered here.

On the other hand, the upper bound can be calculated by assuming the maximum possible sharing of common tools. Here, the number of tool slots required for a set of orders is determined by finding the minimal set of tools needed for the orders and summing their space requirements. Additionally, the number of tool slots available for each machine type is calculated by assuming that there are as many groups as there are machines of that type (i.e., there is one machine in each group). Therefore, the number of tool slots is the product of the tool magazine capacity of the machine and the number of machines of that type. The upper bound is then determined by a method similar to that described above.

Step 6 (Solving loading problems). In this step, we need to solve the machine grouping and loading problem for a given setup i and selected orders j such that $z_{ji} = 1$:

(P3) Maximize T_i

subject to constraints (4), (5), (6), (7), (8), (10), (11), and (12).

Before explaining how we solve the problem for the system as a whole, we first briefly describe our approach to the loading problem for a given machine grouping. It is well known that the throughput function for a CQN is a unimodal function of the workload allocation vector. Thus, it is desirable to assign workloads to servers (or, in our problem, operations to machine groups) so as to match the ideal workloads as closely as possible. (Recall that the binary assignment of operations results in discrete workload allocations, but the ideal workloads are determined in the continuous domain.) We use a particular metric for "closeness" that performed the best among 12 metrics that we tested (see Kim and Yano 1987). With this metric, the loading problem can be viewed as a variation of a two-dimensional bin-packing problem in which the dimensions represent tool slots and processing times, respectively. In the two-dimensional bin-packing problem, bins (machines or machine groups) and items (operations) are represented as rectangles. The width of a rectangle corresponds to the number of tool slots, and the height corresponds to the processing time. The problem, then, is to allocate the items to the bins in such a way that the width capacities of the bins are not exceeded and the sums of the heights of the assigned items are as close as possible to the ideal heights (ideal workloads of groups).

Since operations may share the same tools, we need to consider tool savings that are achieved when these operations are assigned to the same group. In the algorithm, the tool savings are considered explicitly by a labeling method. In our method, each operation is labeled with the number of additional tool slots needed on each machine group. Also, each machine group is labeled with the set of tools and the set of operations assigned to it. These labels change as operations are assigned to machine groups.

We devised various heuristics in which longest processing time (LPT) and MULTIFIT procedures (commonly used for bin-packing) are adapted to our problem. The adaptations were needed for two reasons. First, we needed to deal with the sharing of tools by operations, as described above. Second, while the tool magazine capacity (one dimension of a bin) represents a hard constraint, the ideal workload (the other dimension of a bin) represents a target. Thus, in order to apply bin-packing approaches that require hard constraints in both dimensions, we had to construct a systematic search procedure to position the hard constraints for the workload dimension (one for each machine group). It is important to point out that since the ideal workloads differ across groups, the problem involves nonidentical bins.

From these individual heuristics, which are very fast computationally, we constructed a *composite algorithm* in which several algorithms are executed sequentially. Computational results reported in Kim and Yano (1990) indicate that this composite algorithm provides better solutions than the ϵ-optimal algorithm developed in Berrada and Stecke (1986), in some test problems in a fraction of the computation time (refer to these papers for additional details).

We now consider the grouping and loading problems for the system as a whole, given that n orders with the highest priority are tentatively selected. Recall that the system has multiple machine types. Thus, to find the best loading for the system as a whole, it is necessary to identify the machine grouping alternative and an associated feasible loading of operations *for each machine type* that collectively give the best overall throughput. We attack this problem using the heuristic bin-packing approaches mentioned earlier, one machine type at a time.

Recall that for each machine type, the machine group configurations are already ranked in decreasing order of their maximum possible throughput values. We start with the configuration at the top of the list and solve the loading problem. If the problem is feasible, the resulting throughput (for the actual loading) eliminates from consideration all configurations with a maximum throughput lower than this value. On the other hand, if the loading problem is infeasible, we proceed to the next configuration on this list. We implicitly enumerate all configurations, solving loading problems and eliminating dominated configurations. For the n selected orders, we can calculate a lower bound on the number of machine groups by considering the number of tool slots needed for the n orders and the tool magazine capacity. Therefore, we can eliminate configurations for which the number of machine groups is less than the lower bound even before applying the loading algorithm.

If there is a feasible loading for at least one machine configuration for the first machine type, we repeat the process for the next machine type. This procedure is repeated until a good feasible loading has been identified or no feasible loading is found.

The entire procedure is repeated for each value of n considered in the bisection search. This search is initiated using the upper and lower bounds on n identified in step 5.

When the procedure terminates, the current number n_l is the number of orders that will be selected for the first production period, and the n_l orders with the highest priorities will be selected. Therefore, the order-selection problem will have been solved. Also, we have a configuration for which there is a feasible loading plan. This configuration is the solution to the grouping problem, and the corresponding loading plan is the solution to the loading problem.

Method 2 is identical to method 1 except for some parts of steps 5 and 6. In method 2, the bisection search procedure of method 1 is not used to find the number n. Instead, we apply the grouping/loading routine (step 6 of method 1) for all values of n that give feasible solutions, and select the one that maximizes the system throughput. Since we consider only the immediate production period, this is a greedy method.

4. Computational experiments

4.1. Test problems

Since flexible manufacturing systems differ widely, a solution procedure that performs well on one system is not guaranteed to perform well on other systems. Moreover, even for a particular systems, the demand characteristics can affect the performance of a solution procedure. Since our approach to the system setup problem is designed to be general rather than to be system specific, it is desirable to test the procedure on a wide range of problems.

Unfortunately, we were unable to obtain actual data for a wide variety of systems, partly because of the proprietary nature of such data. However, the test problems were generated in such a way that the resulting data represent characteristics of real systems relatively well. (The parameters for the problems came from one author's experience at a manufacturing company. The resulting data are reflective of FMSs within that company.)

The 40 test problems are defined by the number of machine tools in the system, the number of machine types, and the number of orders to be processed in the upcoming planning horizon. Table 1 shows the selected combinations of these parameters. Here, the number of machines reflects the total for all machine types. The upper number in each cell is the number of test problems, and the number of orders considered in those test problems appear in parentheses.

The input data are generated independently for each problem. For each problem, the other required data are generated as follows:

1. For each machine type, the tool magazine capacity ranges from 40 to 100, which is a selected parameter for each problem.
2. The order quantity of each order is generated randomly using a discrete uniform distribution between 5 and 20.
3. The number of operations needed for each part type is generated by the same method as in point 2, with a minimum of 7 and a maximum of 15.
4. The processing time for each operation also is generated by the same method (minimum = 4, maximum = 30).
5. The index of the machine type that can process an operation is generated randomly using a geometric distribution that results in a value between 1 and the number of machine types.

Table 1. Data for test problems.

Number of machines	Number of machine types			
	3	4	5	6
10	3 (10, 10, 15)	2 (15, 20)		
15	2 (10, 15)	4 (10, 15, 20, 20)	4 (10, 15, 20, 25)	
20		4 (15, 15, 20, 25)	4 (15, 20, 20, 25)	2 (20, 25)
25		2 (20, 25)	3 (15, 20, 25)	5 (15, 20, 20, 25, 25)
30			2 (20, 25)	3 (20, 20, 25)

Notes: The numbers in each cell denote the number of test problems for the corresponding combination of number of machines and machine types. The numbers in parentheses denote the number of part types considered in the problems.

6. The total number of tool types considered in the test is generated by multiplying the total number of operations by a uniform random number between 2 and 3.
7. The number of tools needed for each operation is generated randomly, but it is generated in such a way that an operation with a longer processing time has a tendency to have more tools than an operation with shorter processing time. The number of tools needed for an operation is calculated by multiplying a random number (generated from a uniform distribution between 6 and 8) by the ratio of its processing time to the maximum processing time among all operations.
8. The number of tool slots needed for each tool ranges from 1 to 5. The probabilities for the values are 0.65, 0.05. 0.20, 0.05, and 0.05, respectively, and the number of tool slots is chosen randomly using this probability mass function.
9. Tools are assigned to operations as follows. The number of operations requiring each tool is generated from a discrete uniform distribution between 1 and 4. For each tool, the indices of the operations sharing it are randomly selected from among those operations for which the appropriate number of tools from step 7 has not yet been assigned. When there is a conflict between the results of step 7 and the number of operations requiring specific tools (which occurs because of interdependency of these two sets of data), the number of tools per operation has priority. That is, the number of operations requiring the tools is modified as long as it does not exceed a maximum of 4.
10. Tools that are used solely by a single operation are treated as a single tool for simplicity. The number of tools slots needed for the single artificial tool is the sum of the tool slots required by the individual tools.
11. Other tools are shared by 2 to 4 operations. The operations that share each of these tools are randomly selected.
12. The number of machines of a machine type is determined so that the number is proportional to the sum of processing times of the operations to be processed on the machine type. That is, the greater the processing time requirements on a machine type, the larger the number of machines is.
13. Tool setup time for a machine is generated by multiplying the number of tool slots in the tool magazine of the machine by a random number between 0.5 and 2.0. (This random number reflects that the time needed to change a tool is between 0.5 and 2.0 minutes in many real systems.)
14. Due dates are generated with the method presented in Potts and Van Wassenhove (1982) using the given tardiness factors (T) and relative ranges of due dates (R). In this method, due dates are generated from a uniform distribution $[P(1 - T - R/2), P(1 - T + R/2)]$, where P is the sum of processing times of all operations divided by the number of machines. The tardiness factors range from 0.3 to 0.6, and the relative ranges of due dates range from 0.4 to 1.4.

4.2. *Computational results*

In this section, we report results of the 40 test problems. The program was coded in FORTRAN and run with an optimizing compiler on an IBM 3090-600E system. Subroutines for priority determination based on due dates and for the loading problem are adapted from computer codes used in Kim (1988) and Kim and Yano (1990). For each problem, the list of machine configurations, ideal workloads, and corresponding system throughputs are obtained from the algorithm presented in Kim (1988). This list is used as an input to the iterative procedure.

Before presenting the results of the test problems, we give an example of outputs of the procedure for a small problem (see figure 3). In this problem, there are 3 machine types, a total of 10 machines, and 15 orders. The orders are indexed according to their priorities. The output shows (a) the selected orders for each production period, (b) the number of

Problem Data

number of orders : 15
number of machine types : 3
number of machines : 10
tool magazine capacities of machine types : 100, 90, 80

Method 1

For the 1st batch

a	b	c	d
1	3	0.692	0.591
1,2	3	0.748	0.685
1,2,3	4	0.725	0.681
1,2,3,4	3	0.726	0.692
1,2,3,4,5	3	0.712	0.683
1,2,3,4,5,6	4	0.723	0.698
1,2,3,4,5,6,7	4	0.706	0.685

selected orders : 1 – 7
overall throughput for 7 orders : 0.685

For the 2nd batch

a	b	c	d
8	5	0.653	0.555
8,9	4	0.853	0.785
8,9,10	7	0.789	0.744
8,9,10,11	5	0.798	0.764
8,9,10,11,12	5	0.804	0.775

selected orders : 8 – 12
overall throughput for 12 orders : 0.726

For the 3rd batch

a	b	c	d
13	6	0.861	0.720
13,14	4	0.873	0.802
13,14,15	6	0.776	0.725

selected orders : 13 – 15
overall throughput for 15 orders : 0.726

Method 2

For the 1st batch

a	b	c	d
1	3	0.692	0.591
1,2	3	0.748	0.685
1,2,3	4	0.725	0.681
1,2,3,4	3	0.726	0.692
1,2,3,4,5	3	0.712	0.683
1,2,3,4,5,6	4	0.723	0.698
1,2,3,4,5,6,7	4	0.706	0.685

selected orders : 1 – 6
overall throughput for 6 orders : 0.698

For the 2nd batch

a	b	c	d
7	5	0.726	0.592
7,8	4	0.751	0.683
7,8,9	6	0.815	0.768
7,8,9,10	5	0.774	0.739
7,8,9,10,11	5	0.800	0.771
7,8,9,10,11,12	5	0.794	0.770

selected orders : 7 – 11
overall throughput for 11 orders : 0.734

For the 3rd batch

a	b	c	d
12	5	0.825	0.676
12,13	4	0.843	0.764
12,13,14	5	0.845	0.795
12,13,14,15	3	0.775	0.735

selected orders : 12 – 14
overall throughput for 14 orders : 0.748

For the 4th batch

a	b	c	d
15	3	0.630	0.471

selected orders : 15
overall throughput for 15 orders : 0.734

a orders considered
b number of loading problems solved additionally when one more order is included
c throughput during the production period
d throughput recomputed after considering setup time

Figure 3. Results of an example problem.

loading problems solved, (c) throughputs during the production periods only, and (d) (overall) throughputs during production and setup periods. Tool setup times were considered when calculating the overall throughputs.

Since throughputs from the CQN model used in this research (that of Stecke and Solberg 1985) are relative values, we use relative values for the overall throughputs. These are also normalized so that the maximum possible throughput is 1. The time needed to complete the orders, also known as the makespan, is calculated for the orders in each production period by assuming that the steady-state throughput is achieved during each production period. The makespans for the production periods are summed, and setup times between production periods are added to give a makespan for the entire set of orders. The relative overall throughput can be computed as the reciprocal of the makespan because the total number of parts is given and constant for each problem. The maximum possible throughput of 1 can be achieved only when there is maximum pooling and no setups. The maximum pooling effect can be achieved when there is only one machine type and only one machine group in the system, and therefore, all machines can process all operations.

As can be seen in figure 3, including more orders in a single production period tends to increase the throughput. Although producing orders 1 and 2 gives the highest throughput (0.748) during the production period for the first batch, too-frequent tool setups reduce the overall throughput to 0.685, which is lower than those for other alternatives. However, maximizing the number of orders included in a single production period does not always give the best result. For example, including order 7 in the first set of orders gives a lower throughput than excluding it, even after considering tool setup time. This is because the set of operations needed for six orders led to a loading problem for which a higher throughput could be obtained than for the corresponding problem with seven orders.

The number of loading problems solved for each alternative grouping of orders ranges from 3 to 7. Since there are three machine types in the system, at least three loading problems must be solved for each alternative. The fact that no more than seven loading problems had to be solved shows that configurations near the top of the list were feasible for the loading problem in this example. However, it may be necessary to solve more loading problems in other cases.

The results of the 40 test problems are shown in table 2. The CPU times in the table indicate the time needed to determine priorities of parts, to decide the number of orders in the immediate production period, and to find a grouping/loading plan for the selected orders. Note that the CPU time increases only moderately with the number of machines, number of machine types, and number of part types. This occurs because good feasible solutions are found after enumerating only a few machine configurations, and consequently, only a few loading problems need to be solved. Beyond the anticipated CPU time growth due to the combinatorial aspects of the problem, we did not observe any particular sensitivity of the solution procedure to problem data.

The table also shows the number of orders selected in each production period, the system throughput from each method, and an upper bound on the system throughput. While the CPU time indicates the time needed for the immediate production period, the system throughput in the table indicates the overall system throughput for all the orders in the problem. The overall throughput is calculated by applying the iterative procedure repetitively until all the orders are completed over several production periods. The upper bounds on the overall system throughputs were obtained as follows:

Table 2. Computational results for the iterative procedure.

	Number of			Method 1			Method 2			
Prob. no.	M/C (total)	M/C types	Part types	CPU time (sec.)	NSO	System through-put	CPU time (sec.)	NSO	System through-put	Upper bound on through-put
1	10	3	10	.184	3,2,2,2,1	.689	.241	3,2,1,2,2	.696	.856
2	10	3	10	.190	3,3,3,1	.738	.243	2,2,3,1,2	.728	.865
3	10	3	15	.677	7,5,3	.704	.641	4,2,2,5,2	.725	.880
4	10	4	15	.203	3,3,1,3,3,2	.690	.250	2,3,1,1,3,3,2	.702	.824
5	10	4	20	.282	2,3,3,2,2,3,2,3	.636	.171	2,2,2,4,2,2,2,1,2,1	.654	.813
6	15	3	10	.618	6,4	.690	.877	2,3,2,3	.734	.864
7	15	3	15	.767	5,7,3	.658	.616	2,5,2,3,3	.699	.861
8	15	4	10	.348	4,4,2	.646	.496	4,2,2,2	.660	.813
9	15	4	15	.738	6,5,3,1	.656	.941	2,5,3,3,2	.666	.810
10	15	4	20	.677	9,6,5	.663	1.206	6,6,4,3,1	.657	.827
11	15	4	20	.549	10,6,4	.673	1.279	3,5,4,6,2	.690	.828
12	15	5	10	.174	4,4,2	.646	.391	4,4,2	.646	.790
13	15	5	15	.398	3,3,4,2,2,1	.578	.288	2,4,4,2,2,1	.582	.768
14	15	5	20	.306	8,7,5	.664	.797	6,5,3,5,1	.678	.799
15	15	5	25	.257	7,7,6,5	.631	.652	6,7,6,6	.662	.793
16	20	4	15	.794	8,7	.624	1.218	5,4,4,2	.664	.804
17	20	4	15	.791	8,5,2	.609	1.251	5,3,4,3	.656	.792
18	20	4	20	.607	9,9,2	.606	1.347	3,4,4,5,3,1	.658	.798
19	20	4	25	.892	11,12,2	.599	1.792	6,5,5,3,4,2	.683	.805
20	20	5	15	.429	6,7,2	.597	.927	2,3,7,2,1	.616	.765
21	20	5	20	.327	7,6,5,2	.592	.509	3,4,3,3,2,3,2	.617	.764
22	20	5	20	.815	6,6,6,2	.601	1.017	5,3,2,3,5,2	.615	.758
23	20	5	25	.843	11,3,10,1	.578	1.466	6,3,4,1,3,3,5	.585	.767
24	20	6	20	.546	7,7,6	.595	.937	5,5,4,4,2	.604	.749
25	20	6	25	.706	9,5,6,4,1	.530	1.260	4,8,5,5,3	.564	.732
26	25	4	20	2.208	14,6	.556	5.514	5,5,4,4,2	.641	.779
27	25	4	25	1.772	13,9,3	.559	3.818	6,4,2,5,3,3,2	.631	.769
28	25	4	15	.710	5,6,4	.571	.800	5,5,4,1	.569	.726
29	25	5	20	.981	10,5,5	.565	2.003	7,3,4,4,1,1	.582	.736
30	25	5	25	.836	10,9,6	.533	2.253	4,4,4,4,3,5,1	.585	.742
31	25	6	15	.719	7,7,1	.550	1.161	4,3,7,1	.560	.706
32	25	6	20	.623	7,9,4	.525	.774	7,4,7,2	.542	.715
33	25	6	20	.852	9,9,2	.550	1.742	4,5,3,4,4	.591	.711
34	25	6	25	1.079	10,9,6	.544	1.642	10,8,7	.549	.715
35	25	6	25	1.065	10,9,6	.530	2.213	7,5,5,8	.556	.714
36	30	5	20	1.951	14,6	.531	4.595	5,6,5,4	.576	.701
37	30	5	25	1.985	9,10,6	.537	2.241	2,4,6,4,5,3,1	.561	.695
38	30	6	20	1.363	12,8	.528	2.673	4,6,5,4,1	.552	.680
39	30	6	20	1.371	10,9,1	.489	2.144	4,5,4,4,3	.541	.671
40	30	6	25	.736	9,3,9,4	.509	1.669	9,3,5,4,3,1	.517	.663

Notes: NSO denotes the numbers of selected orders for a series of production periods.
CPU times are for an IBM 3090-600E computer.

1. When computing the throughput during production periods, all machines of the same type are assumed to be pooled completely. This gives the maximum pooling, which is known to maximize the throughput. In addition, the ideal workloads (those that maximize throughput for the given configuration) are allocated to the machine types.

2. The number of tool setups is computed as if there were one machine in each group. Thus the number of tool slots available for a machine type is equal to the maximum possible value.

A proof of the validity of this upper bound is given in the appendix. Although the bounds are valid, they are very loose, since it is unlikely that a feasible solution to the original problem satisfies both relaxations. Note that we have relaxed the integrality constraints on operation allocation as well as the technical constraints that limit the machines that can process each operation. The above relaxations are used, however, since it is very difficult to obtain a sharper bound because of the complexity of the problem.

The CPU time for method 1 is shorter than that for method 2 because method 1 uses a bisection search procedure while the other cannot use it. Nevertheless, even the CPU times for method 2 are short enough to be implemented in practice. This indicates that most system setup problems can be solved with our iterative approach. The CPU time decreases as the number of machine types increases, with the number of machines in the system and the number of orders held constant. This is because the number of machines of the same type decreases, which results in fewer configuration alternatives for a given machine type, and therefore fewer loading problems need to be solved.

The speed of the procedure was achieved by using fast heuristic algorithms for the loading problems. To select a set of orders for the first tool setup, more than 10 loading problems had to be solved within the iterative procedure in most cases, and some of the test problems required the solution of almost 140 loading problems. Therefore, if an existing optimal algorithm for loading problems were used in our iterative approach, the resulting computation time would be prohibitive even on fast mainframe computers. This is what necessitates the use of heuristics for this problem.

Overall, method 2 is better in maximizing system throughput than method 1. In some of the test problems, method 2 gave a system throughput almost 10% higher than method 1. Method 2 gave better solutions than method 1 in all the test problems except three (problems 2, 10, and 28). In these three problems, the results appear to be the consequence of end-of-horizon effects. In method 2, there were only a few orders in the last production period, and therefore only a few operations to be allocated. Thus, the resulting operation assignments gave actual workloads that were far from the ideal workloads. In real systems, we would be able to avoid these end-of-horizon effects by combining these few orders with the orders for the upcoming planning horizon.

We use the loading algorithm of Kim and Yano (1990) in our procedure. They report computational results that indicate that the difference between the expected system throughput for the solutions from their procedure (as computed by the CQN model of Stecke and Solberg 1985) and the expected system throughput for the optimal *continuous* workload allocation is less than 1.0% in most cases, and the maximum difference among 60 test problems is 1.8%. Since the loading algorithm affects the performance of our procedure most significantly, it is unlikely that significantly higher throughputs could be achieved, even with complete enumeration.

Although method 2 appears to be more attractive than method 1 for the objective of maximizing system throughput, it may not always be the best for other objectives. Observe that method 2 chooses a smaller number of orders than method 1. Therefore, the nature of the resulting scheduling problems differs. We explain this with an example. Suppose there are six orders to be scheduled and all can be included in one production period. Also suppose that method 2 chooses to have two setups, each with three orders. Under method 1, all six orders are scheduled simultaneously. All other things being equal, there would be many more alternative schedules than under method 2. However, because of tool magazine capacity constraints, each machine type is likely to be partitioned into more groups, and consequently, the number of machines in each group will be smaller than under method 2. Thus, the number of alternative routings for each part will be more limited as well. It is not obvious how these factors interact to influence the quality of the schedule, and the differences between the methods may be situation specific.

In practice, other methods to select orders may be needed, depending upon the long- and short-term goals of the manufacturing operation. The grouping and loading procedures presented here can be used with any part type selection procedure, both in real-time operation and as a decision support tool to analyze various part type selection procedures.

In conclusion, our iterative approach provides solutions for the part type selection problem, the grouping problem, and the loading problem altogether in a reasonable amount of computing time, and provides very good solutions as well.

5. Summary and concluding remarks

In this article, we have described a *formally* iterative (closed-loop) procedure for simultaneously solving the problems of selecting orders for immediate processing, partitioning machines of each type into groups (where machines in a group are tooled identically), and assigning operations required by the selected orders to these machine groups. The procedure consists of an integrated collection of enumeration, optimization, and heuristic algorithms that were designed to facilitate feedback. Because of the combinatorial nature of the order selection problem and its dependence on the particular application, we assume that strict priorities are already given, but that the number of orders can be decided.

This iterative approach was designed to cope with problems inherent in sequential approaches. In a sequential approach, each subproblem is solved in sequence using the solutions of upstream subproblems, but without specific feedback on how to modify upstream decisions when downstream subproblems are infeasible. In our iterative approach, on the other hand, the subproblems may be solved many times with specific upstream feedback until a good solution for the system setup problem is found. Since even the individual subproblems cannot be solved optimally in a reasonable amount of time, heuristic approaches were employed for some portions of the problem.

To demonstrate the viability of such a procedure, we describe and test two variations on a particular implementation in which the objective is to maximize the throughput of the system. The overall iterative procedure performed well in 40 test problems. It not only was very fast computationally but also provided solutions with relatively high system throughputs. It also provides a feasible solution (if one exists), while sequential approaches may fail to do so.

More broadly, the results show that part type selection, machine grouping, and machine loading can be accomplished simultaneously with good results if fast, near-optimal heuristics can be developed for the subproblems. Earlier in the article (step 1), we briefly described an efficient procedure to generate all machine grouping alternatives. This algorithm could be used in the context of solving any problem involving machine grouping. To deal with the part type selection and machine loading aspects of the problem for objectives other than throughput maximization will require the development of optimal or good heuristic procedures that address those objectives. As mentioned earlier, it is impossible to anticipate the impact of a part type selection procedure on any performance measure without having solved the loading and grouping problems. Thus, this problem will remain a difficult one. On the other hand, it is possible that good heuristics could be developed for the loading problem with a variety of objective functions. Almost all of the work to date has focused on throughput maximization and related measures. Further research is needed to develop a better understanding of how to model the various subproblems when there are different objectives.

Our approach to the system setup problem can be extended in various ways. For each step of the iterative procedure, more efficient algorithms can be developed. Much of the computational burden in our procedure is attributable to determining the workload allocation that maximizes throughput for a given configuration. Faster algorithms for this problem will permit problems to be solved on mini- and microcomputers. Also, the concept of the flexible approach of Stecke and Kim (1986) can be implemented in the context of our iterative approach to cope with dynamic events, such as part mix changes and machine breakdowns. The iterative procedure, then, will be reexecuted whenever a part type is completed, urgent orders or part types arrive, or machines break down. Finally, off-line scheduling of an FMS can be done in conjunction with the system setup. Coordination of these two is important to our research as well, since it is useful to assess how system setup decisions affect the quality of detailed schedules.

Since the system setup problem can be solved quickly, we can use the procedure when testing various product mix alternatives. In other words, we can evaluate different production plans for the upcoming planning horizon and choose the best one. This will help to improve the quality of production plans as well. Thus, our iterative approach can be used not only as a decision-making tool but also as a decision support tool in real systems.

Appendix. Proof of the validity of the upper bounds in table 2.

Let TH_o and TH_u be the overall system throughput of an optimal solution and the upper bound obtained with the above relaxations, respectively. Then,

$$TH_o = \frac{G}{\sum_{i=1}^{r} L_i^o \,/\, T_i^o + r \cdot S},$$

where L_i^o is the total workload of the part types included in the ith tool setup, T_i^o is the system throughput for the ith production period, and r is the number of tool setups, in the optimal solution. Additionally, S is tool setup time for each tool setup and G is the normalization constant that makes the maximum possible throughput equal to 1.

From the method by which the upper bound is obtained, TH_u can be expressed as

$$TH_u = \frac{G}{L_T / T^u + R \cdot S},$$

where T^u is the system throughput obtained with the first relaxation described above, L_T is the total workload of all part types, i.e.,

$$L_T = \sum_{i=1}^{r} L_i^o,$$

and R is the number of tool setups obtained with the second relaxation.

We first show that $T_i^o \leq T^u$ for any i. Since operations to be processed on machines of one type cannot be processed on machines of other types, we cannot group machines of different types together. Under this restriction on machine grouping, the maximum possible throughput is the one obtained from the relaxation, since the grouping in the relaxation is the best under this restriction. Moreover, the ideal workloads are used to calculate the throughput for the grouping. (The ideal workloads are defined as the workloads of machines that result in the maximum throughput.)

Next, we show that $R \leq r$. Since for each machine type, the machines within the same group are identically tooled, the total number of tool slots available is equal to the tool-magazine capacity (tool slots) multiplied by the number of groups. By assuming there is one machine in each group, we maximize the number of groups. This gives the maximum number of tool slots for each machine type, and therefore permits the largest number of different tools to be installed in a single tool setup. This minimizes the number of tool setups needed for all part types ($R \leq r$).

Since

$$\sum_{i=1}^{r} L_i^o / T_i^o \geq \sum_{i=1}^{r} L_i^o / T^u = L_T / T^u,$$

we have

$$L_T / T^u + R \cdot S \leq \sum_{i=1}^{r} L_i^o / T_i^o + r \cdot S,$$

which results in $TH_o \leq TH_u$.

References

Ammons, J.C., Lofgren, C.B., and McGinnis, L.F., "A Large Scale Machine Loading Problem in Flexible Assembly," *Annals of Operations Research*, Vol. 3, pp. 319–322 (1985).

Avriel, M., *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, Englewood Cliffs, NJ (1976).

Baker, K.R. and Kanet, J.J., "Job Shop Scheduling with Modified Due Dates," *Journal of Operations Management*, Vol. 4, No. 1, pp. 11–22 (1983).

Berrada, M. and Stecke, K.E., "A Branch and Bound Approach for Machine Load Balancing in Flexible Manufacturing Systems," *Management Science*, Vol. 32, No. 10, pp. 1316–1335 (1986).

Blackstone, J.H. Jr., Phillips, D.T., and Hogg, G.L., "A State-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations," *International Journal of Production Research*, Vol. 20, No. 1, pp. 27–45 (1982).

Buzen, J.P., "Computational Algorithms for Closed Queueing Networks with Exponential Servers," *Communications of the ACM*, Vol. 16, No. 9, pp. 527–531 (1973).

Chakravarty, A.K. and Shtub, S., "Selecting Parts and Loading Flexible Manufacturing Systems," *Proc. of the 1st ORSA/TIMS Special Interest Conf. on Flexible Manufacturing Systems*, Ann Arbor, MI, K.E. Stecke and R. Suri (Eds.), pp. 284–289 (1984).

Chakravarty, A.K. and Shtub, S., "Capacity, Cost, and Scheduling Analysis for a Multiproduct Flexible Manufacturing Cell," *International Journal of Production Research*, Vol. 25, No. 8, pp. 1143–1156 (1987).

Elvers, D.A. and Taube, L.R., "Time Completion for Various Dispatching Rules in Job Shops," *OMEGA*, Vol. 11, No. 1, pp. 81–89 (1983).

Greene, T.J. and Sadowski, R.P., "A Mixed Integer Program for Loading and Scheduling Multiple Flexible Manufacturing Cells," *European Journal of Operational Research*, Vol. 24, No. 3, pp. 379–386 (1986).

Hwang, S., "A Constraint-Directed Method to Solve the Part Selection Problem in Flexible Manufacturing Systems Planning Stage," *Proc. of the 2nd ORSA/TIMS Conf. on Flexible Manufacturing Systems*, Ann Arbor, MI, K.E. Stecke and R. Suri (Eds.), Elsevier Science Publishers B.V., Amsterdam, pp. 297–309 (1986).

Lashkari, R.S., Dutta, S.P., and Padhye, A.M., "A New Formulation of Operation Allocation Problem in Flexible Manufacturing Systems: Mathematical Modelling and Computational Experience," *International Journal of Production Research*, Vol. 25, No. 9, pp. 1267–1283 (1987).

Kim, Y-D., "On the Superiority of a Backward Approach in List Scheduling Algorithms for Multi-Machine Makespan Problems," *International Journal of Production Research*, Vol. 25, No. 12, pp. 1751–1759 (1987).

Kim, Y-D., "An Interative Approach for System Setup Problems of Flexible Manufacturing Systems," Ph.D. dissertation, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI (1988).

Kim, Y-D. and Yano, C.A., "A Branch and Bound Approach for the Loading Problem in Flexible Manufacturing Systems: An Unbalancing Case," Technical Report 87-18, Dept. of I&OE, The University of Michigan, Ann Arbor, MI (1987).

Kim, Y-D. and Yano, C.A., "A New Branch and Bound Approach for Loading Problems in Flexible Manufacturing Systems," Technical Report 89-5, Dept. of I&OE, The University of Michigan (1989).

Kim, Y-D. and Yano, C.A., "Heuristic Approaches for Loading Problems in Flexible Manufacturing Systems," to appear in *IIE Transactions* (1990).

Kiran, A.S. and Tansel, B.C., "The System Setup in FMS: Concepts and Formulation," *Proc. of the 2nd ORSA/TIMS Conf. on Flexible Manufacturing Systems*, Ann Arbor, MI, K.E. Stecke and R. Suri (Eds.), Elsevier Science Publishers B.V., Amsterdam, pp. 321–332 (1986).

Kumar, K.R., Kusiak, A., and Vannelli, A., "Grouping of Parts and Components in Flexible Manufacturing Systems," *European Journal of Operational Research*, Vol. 24, No. 3, pp. 387–397 (1986).

Kusiak, A., "The Part Families Problem in Flexible Manufacturing Systems," *Proc. of the 1st ORSA/TIMS Special Interest Conf. on Flexible Manufacturing Systems*, Ann Arbor, MI, K.E. Stecke and R. Suri (Eds.), pp. 237–242 (1984).

O'Grady, P.J. and Menon, U., "A Flexible Multiobjective Production Planning Framework for Automated Manufacturing Systems," *Engineering Costs and Production Economics*, Vol. 8, pp. 189–198 (1984).

O'Grady, P.J. and Menon, U., "A Multiple Criteria Approach for Production Planning of Automated Manufacturing," *Engineering Optimization*, Vol. 8, No. 2, pp. 161–175 (1985).

O'Grady, P.J. and Menon, U., "Loading a Flexible Manufacturing System," *International Journal of Production Research*, Vol. 25, No. 7, pp. 1053–1068 (1987).

Panwalker, S.S. and Iskander, W., "A Survey of Scheduling Rules," *Operations Research*, Vol. 25, No. 1, pp. 45–61 (1977).

Potts, C.N. and Van Wassenhove, L.N., "A Decomposition Algorithm for the Single Machine Total Tardiness Problems," *Operations Research Letters*, Vol. 1, No. 5, pp. 177–181 (1982).

Rajagopalan, S. "Formulation and Heuristic Solutions for Parts Grouping and Tool Loading in Flexible Manufacturing Systems," *Proc. of the 2nd ORSA/TIMS Conf. on Flexible Manufacturing Systems,* Ann Arbor, MI, K.E. Stecke and R. Suri (Eds.), Elsevier Science Publishers B.V., Amsterdam, pp. 312-320 (1986).

Russel, R.S., Dar-El, E.M., and Taylor, B.S. III, "A Comparative Analysis of the COVERT Job Sequencing Rule Using Various Shop Performance Measures," *International Journal of Production Research,* Vol. 25, No. 10, pp. 1523-1540 (1987).

Shanker, K. and Tzen, Y-J.J., "A Loading and Dispatching Problem in a Random Flexible Manufacturing System," *International Journal of Production Research,* Vol. 23, No. 3, pp. 579-595 (1985).

Shanthikumar, J.G. and Stecke, K.E., "Reducing Work-in-Process Inventory in Certain Classes of Flexible Manufacturing Systems," *European Journal of Operational Research,* Vol. 26, No. 2, pp. 266-271 (1986).

Stecke, K.E., "Formulation and Solution of Nonlinear Integer Production Planning Problems for Flexible Manufacturing Systems," *Management Science,* Vol. 29, No. 3, pp. 273-288 (1983).

Stecke, K.E., "Design, Planning, Scheduling, and Control Problems of Flexible Manufacturing Systems," *Annals of Operations Research,* Vol. 3, pp. 3-12 (1985).

Stecke, K.E., "A Hierarchical Approach to Solving Machine Grouping and Loading Problems of Flexible Manufacturing Systems," *European Journal of Operational Research,* Vol. 24, No. 3, pp. 369-378 (1986).

Stecke, K.E. and Kim, I., "A Flexible Approach to Implementing the Short-Term FMS Planning Function," *Proc. of the 2nd ORSA/TIMS Conf. on Flexible Manufacturing Systems,* Ann Arbor, MI, K.E. Stecke and R. Suri (Eds.), Elsevier Science Publishers B.V., Amsterdam, pp. 283-295 (1986).

Stecke, K.E. and Kim, I., "Performance Evaluation for Systems of Pooled Machines of Unequal Sizes: Unbalancing versus Balancing," *European Journal of Operational Research,* Vol. 42, No. 1, pp. 22-38 (1989).

Stecke, K.E. and Kim, I., "A Study of FMS Part Type Selection Approaches for Short-Term Production Planning," *International Journal of Flexible Manufacturing Systems,* Vol. 1, No. 1, pp. 7-29 (1988).

Stecke, K.E. and Morin, T.L., "The Optimality of Balancing Workloads in Certain Types of Flexible Manufacturing Systems," *European Journal of Operational Research,* Vol. 20, pp. 68-82 (1985).

Stecke, K.E. and Solberg, J.J., "Loading and Control Policies for a Flexible Manufacturing System," *International Journal of Production Research,* Vol. 19, No. 5, pp. 481-490 (1981).

Stecke, K.E. and Solberg, J.J., "The Optimality of Unbalancing Both Workloads and Machine Group Sizes in Closed Queueing Networks of Multi-Server Queues," *Operations Research,* Vol. 33, No. 4, pp. 882-910 (1985).

Stecke, K.E. and Talbot, F.B., "Heuristics for Loading Flexible Manufacturing Systems," in *Flexible Manufacturing: Recent Developments in FMS, Robotics, CAD/CAM, CIM,* A. Raouf and S.I. Ahmad (Eds.), Elsevier Science Publishers B.V., Amsterdam, pp. 73-85 (1985).

Vepsalainen, A.P.J. and Morton, T.E., "Priority Rules for Job Shops with Weighted Tardiness Costs," *Management Science,* Vol. 33, No. 8, pp. 1035-1047 (1987).

Whitney, C.K. and Gaul, T.S., "Sequential Decision Procedures for Batching and Balancing in Flexible Manufacturing Systems," *Proc. of the 1st ORSA/TIMS Special Interest Conf. on Flexible Manufacturing Systems,* Ann Arbor, MI, K.E. Stecke and R. Suri (Eds.), pp. 243-248 (1984).

Whitney, C.K. and Suri, R., "Decision Aids for FMS Part and Machine Selection," *Proc. of the 1st ORSA/TIMS Special Interest Conf. on Flexible Manufacturing Systems,* Ann Arbor, MI, I.E. Stecke and R. Suri (Eds.), pp. 205-210 (1984).

Zangwill, W.I., *Nonlinear Programming: A Unified Approach,* Prentice-Hall, Englewood Cliffs, NJ (1969).