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

Technical Report 89-7

March 1989

Revised December 1989, January 1991
and May 1991

AN ITERATIVE APPROACH TO SYSTEM SETUP PROBLEMS IN FLEXIBLE MANUFACTURING SYSTEMS

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 paper, 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.

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 all its operations on a machine type are 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.

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 grouping and loading problems is suggested by Stecke (1986).

The main contribution of this paper 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 in this paper 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 paper 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, as 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 which 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.

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 distiguished 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 as 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 (Panwalkar and Iskander 1977, Blackstone *et al.* 1982, Elvers and Taube 1983, Russel *et al.* 1987, Vepsalainen and Morton 1987, and Kim 1987, for example), there is little research on the problem of scheduling multiple orders on multiple machines when each order has multiple units (which can be processed independently of one another), but with a common due date for the units 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

Sets

Q_{jv}:	set of operations for the j-th 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 w-th group of machine type v in i-th setup

Binary (0-1) *Decision Variables*

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

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

$$\text{Minimize} \quad \sum_{i=1}^{|J|} \{ S Z_i + L_i / (T_i \sum_{v} M_v) \}$$

subject to

$$\sum_{j=1}^{|J|} z_{ji} \le B Z_i \tag{1}$$

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

defined in (10). Since the throughput cannot be expressed in closed form, we have not stated (10) in mathematical form. The system throughput can be calculated by a numerical method given by Stecke and Solberg (1985) for a product-form 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

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.

Insert Figures 1 and 2 here

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 which 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 (*P1*) must be solved *for each machine type* to generate all possible configurations for a given number of machines, M, where $(k_1,k_2,\cdots,k_m)$ denotes a configuration with k_i machines in the i-th 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 below.

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 results 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 lists of configurations, their ideal workloads, and corresponding maximum throughput values which can be used in every system setup phase. The lists can be stored in auxiliary computer memory space and retrieved whenever they are needed.

Step 4 ***(Setting order priorities)***

Since we assume that priorities are given, techniques to set priorities are beyond the scope of this paper. 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 which 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

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 which 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 non-identical 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 many 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 .

4. Computational Experiments

4.1. Test problems

Since flexible manufacturing systems differ widely, a solution procedure which performs well on one system is not guaranteed to perform well on other systems. Moreover, even for a particular system, 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 forty 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.

Insert Table 1 here

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 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).

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 three machine types, a total of ten machines, and fifteen orders. The orders are indexed according to their priorities. The output shows (a) the selected orders for each production period, (b) the number of 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.

Insert Figure 3 here

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

repetitively until all the orders are completed over several production periods. The upper bounds on the overall system throughputs were obtained as follows.

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 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 machine types 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 decide a set of orders selected 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

5. Summary and Concluding Remarks

In this paper, 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 which 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 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 paper (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

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, New Jersey (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, Jr., J. H., 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 Network 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* , (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 (editors), 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 Iterative Approach for System Setup Problems of Flexible Manufacturing Systems," Ph.D. Dissertation, Department of of Industrial and Operations Engineering (I&OE), The University of Michigan, Ann Arbor, MI (1988).

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 (editors), 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," pp.73-85 in *Flexible Manufacturing: Recent Developments in FMS, Robotics, CAD/CAM, CIM, Raouf, A. and Ahmad, S.I.* (eds.), Elsevier Science Publishers B.V., Amsterdam (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* (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* (1984).

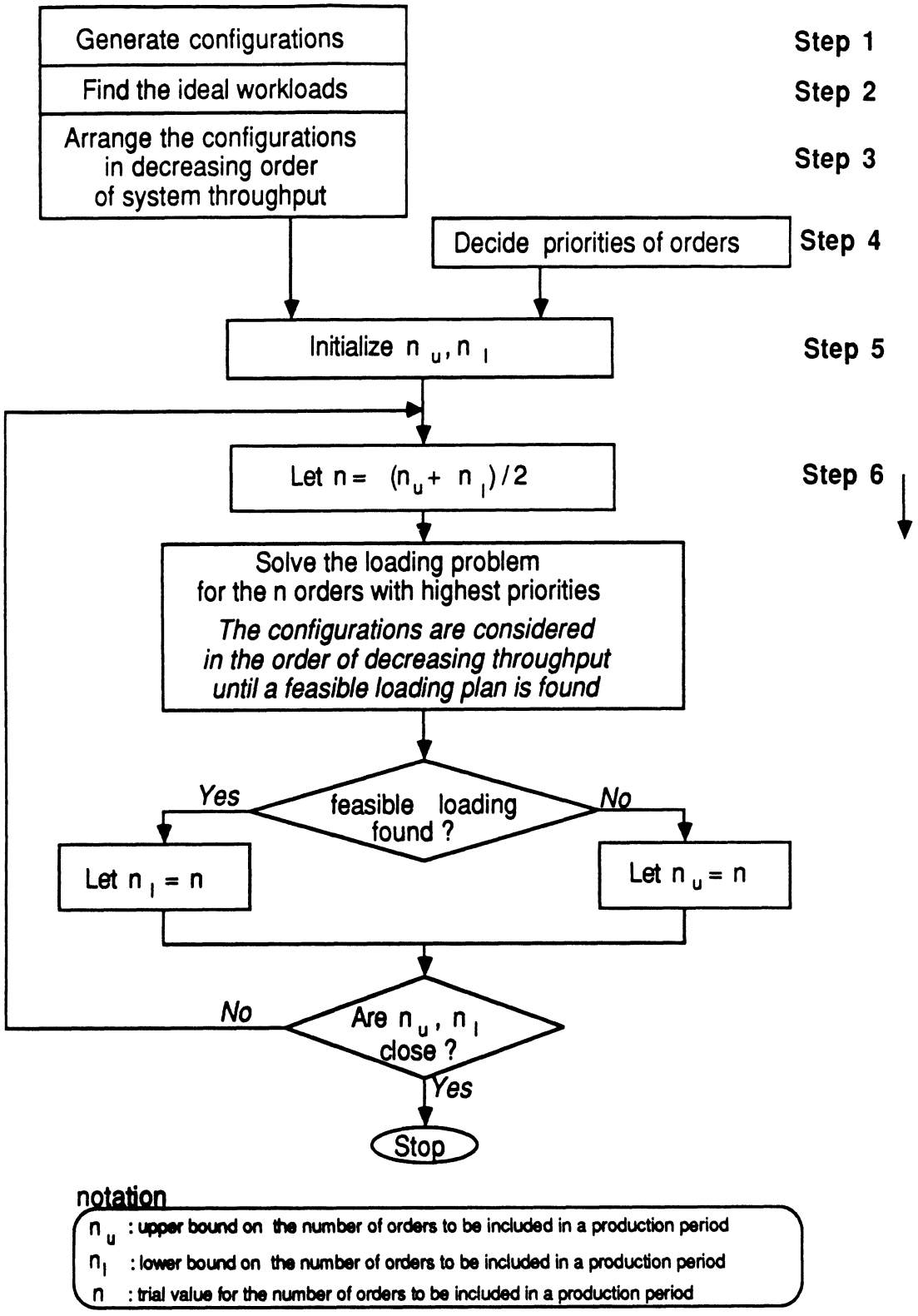

Figure 1. The iterative procedure (Method 1)

Table 1. Data for the 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)

† The numbers in each cell denotes the number of test problems for the corresponding combination of number of machines and machine types. The numbers in the parenthesis denote the number part types considered in the problems.

Table 2. Computational results for the iterative procedure

Prob. No.	Number of M/C (total)	M/C types	Part types	Method 1 CPU time (sec.)	Method 1 NSO †	Method 1 System throughput	Method 2 CPU time (sec.)	Method 2 NSO †	Method 2 System throughput	Upper bound on throughput
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

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

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 $TH_o \leq TH_u$.