HEURISTIC APPROACHES FOR LOADING PROBLEMS IN FLEXIBLE MANUFACTURING SYSTEMS

Yeong-Dae Kim
Department of Industrial Engineering
Korea Advanced Institue 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 87-21

September 1987

Revised
March 1989, March 1990, and November 1990

HEURISTIC APPROACHES FOR LOADING PROBLEMS IN FLEXIBLE MANUFACTURING SYSTEMS

Abstract

Flexible manufacturing systems (FMSs) are able to process a wide variety of operations, but the specific mix of operations that can be performed at any point in time depends upon the combination of tools loaded onto the machines. The machines have tool magazines with finite capacities. We consider the problem of assigning operations and their associated tools to machines (or groups of machines) to maximize the throughput for a specified steady-state mix of orders. Since this objective is difficult to deal with directly, we use an intermediate objective of meeting workload targets for each machine group as closely as possible. A certain form of this intermediate objective has been shown to correlate highly with the original objective.

Since it is computationally intractable to find optimal solutions for problems with more than 20 operations, fast heuristic algorithms are developed. These algorithms are adapted from multi-dimensional bin-packing algorithms. Computational results are reported.

1. Introduction

The production capacity of Flexible Manufacturing Systems (FMSs) is commonly analyzed by closed queueing network (CQN) models. Several CQN-based models are commercially available and are being used at major manufacturing companies. To estimate production capacity using a CQN model, one needs to specify a routing matrix and the average processing time of the operations assigned to each machine group, where machines in a group have identical capabilities. Normally, the routing matrix is easy to determine once it has been decided what operations will be performed by each machine group, and these operation assignments determine the workload allocation among the groups.

Some research has been done on the allocation of workload among machine groups when the total workload can be divided continuously (e.g., Yao 1985, Stecke and Morin 1985, Shanthikumar and Stecke 1986, Yao and Kim 1987a, and Yao and Kim 1987b). Stecke and Solberg (1985) have shown that the throughput (or expected production rate) of a system is maximized by a specific continuous workload allocation, also referred to as *ideal workloads*. When two or more machine groups are identical, it is optimal to divide the work among them equally, which they call *balancing*. On the other hand, when machine groups are not identical, it is better to assign larger workloads per machine to larger groups to take advantage of the effects of pooling. They refer to this optimal allocation as *unbalancing*.

Operation assignments result in discrete allocations of workloads to machine groups, and the number of operations that can be assigned to each machine group is limited by the tool magazine capacities. We develop a new procedure for assigning a given set of operations to machine groups so as to maximize the throughput of the system while ensuring that tool limitations are satisfied. We assume there is a specified steady-state mix of orders, and the throughput reflects this mix. Although this problem has been studied before, existing optimization procedures are computationally impractical when there are more than 20 operations.

The new procedure can be used in several different ways. First, it can be used as a capacity planning tool to provide more accurate estimates of throughput, taking into account practical constraints. Since a typical FMS can manufacture a wide variety of parts, it is usually not possible to process all of them simultaneously because of the tool limitations. Consequently, to obtain robust estimates, one actually needs to evaluate the capacity of a system under many different assumptions about the mix of parts that might be processed simultaneously. Secondly, our procedure can be used to aid in the decision of

which parts should be produced on the FMS, and if there is more than one FMS, how to partition parts among them. The throughput estimates derived from the approach can provide information about the compatibility of parts with regard to utilization of equipment and tools. Finally, the procedure can be used to solve the short-term problem of allocating operations and tools to machine groups for the imminent production run. This is known in the literature as the *FMS loading problem*. Since the CQN models used in the analysis are steady-state models, they are applicable only when a constant mix of parts is produced for a sufficiently long enough duration. There are, however, systems with common due dates for parts (e.g., once a week shipments) and relatively short operation processing times (e.g., minutes), where steady-state approximations are realistic.

Most of the literature on this topic has focused on the short-term loading aspect of this problem. Stecke (1983) formulates the loading problem with the workload balancing objective as a nonlinear mixed integer program and solves it through linearization techniques. She uses a surrogate objective of minimizing weighted under and overloads. A branch and bound algorithm is developed by Berrada and Stecke (1986) for this formulation. Their computational study suggests that some problems with up to 40 operations can be solved, but only with a considerable amount of CPU time. Stecke and Talbot (1985) describe (but do not test) several heuristics, two of which include workload (un)balancing considerations and are based upon bin-packing concepts. Ammons *et al.* (1985) formulate the problem with the objective of minimizing the weighted sum of workload imbalance and station-to-station material movements. When all of the weight is on the former objective, they attempt to minimize the maximum workload across machines, and suggest a modification of the best-fit-decreasing procedure for bin packing. For one case study, they develop the efficient frontier for their bicriterion objective.

Shanker and Tzen (1985) use the objective of minimizing weighted under- and overloads as a surrogate for balancing workloads. They also consider the bicriterion objective of balancing the workloads and minimizing the number of late jobs. A complicated heuristic is proposed for the balancing objective, and job tardiness is considered heuristically by attempting to schedule urgent jobs in the current production setup. Lashkari *et al.* (1987) formulate the problem as a nonlinear mixed integer program for two different objectives, minimizing transport load and minimizing refixturing activities. Kusiak (1985) presents several models based on the generalized assignment problem and the generalized transportation problem, considering the cost of processing an operation at a station. Carrie and Perera (1986) emphasize the life of cutting tools and Sarin and Chen (1987) focus on the machining costs corresponding to tool-machine

combinations. The loading problem is studied together with other FMS-related problems in other research (Greene and Sadowski 1986, Rajagopalan 1986, and Stecke 1986).

Our objective for the loading problem is to maximize system throughput. However, because the throughput function for a CQN is so complicated, one cannot maximize it directly. However, it is thought to be quasiconcave, and therefore, a common surrogate objective is to be as close as possible to the optimal continuous workload allocation. Kim and Yano (1989) show that certain forms of this surrogate objective are highly correlated with the original objective.

Our problem, then, is *to allocate operations and tools to machines or machine groups in such a way that workloads of the machine groups are as close as possible to the ideal workloads subject to tool magazine capacity constraints and machine capacity constraints*. We explicitly consider the fact that operations may share common tools, which we call *tool commonality*.

The reason for solving the short-term loading problem first in an approximate, aggregate way is to reduce the difficulty of the scheduling problem. In essence, the solution to the loading problem provides a physical and logical configuration of the system that should make scheduling easier by ensuring an appropriate balance in the use of resources. There is, however, no guarantee that the throughput (or any other performance measure) suggested by the solution to the loading problem can actually be achieved. Thus, unless one has a system that satisfies the CQN assumptions (e.g., exponential service, infinite buffers), optimal solutions to the loading problem may not be necessary. A heuristic with consistently good performance may be more than adequate. Also, branch and bound algorithms can be improved by using a good heuristic algorithm to provide an initial feasible solution. We use the solution from the heuristic as an initial solution in our branch-and-bound procedure (see Kim and Yano 1989).

We develop heuristic algorithms by viewing the FMS loading problem as a makespan scheduling problem with additional constraints to cope with the tool magazine capacity. The heuristic algorithms are modifications of various single-pass and multiple-pass algorithms for multi-dimensional bin-packing problems. Although some of the earlier research on the problem has used bin-packing approaches, the proposed heuristics differ from the earlier work in two ways. First, they use multi-pass rather than single-pass bin-packing approaches. Secondly, they use new assignment rules that explicitly consider tool magazine capacity constraints.

In the next section, basic ideas for the heuristic algorithms are discussed and related research results are reviewed. Section 3 presents several algorithms developed from these

ideas, and their computational results are given in Section 4. Finally, concluding remarks appear in Section 5.

2. Basic Idea

In most heuristic solution procedures for FMS loading problems, the problem is viewed as a bin-packing problem. The bin-packing problem involves assigning items of various sizes into bins (possibly of different sizes) to achieve some objective. The bins and the items may have more than one dimension. In one-dimensional bin-packing problems, the sizes of bins and items are specified by their lengths. On the other hand, in two-dimensional bin-packing problems, bins and items can be expressed as rectangles, and the sizes of bins and items can be specified by their heights and widths.

In this research, the FMS loading problem is viewed as a two-dimensional bin-packing problem in which the dimensions denote tool slots and processing times. Each machine group is represented by a bin, and operations are represented by items. Widths of rectangles associated with operations denote the number of tool slots needed, and heights denote workloads. Since the product mix is given, operation assignments actually involve the assignments of workloads associated with the respective operations. Thus, when an operation is assigned to a machine group, the workload induced by that assignment is the unit operation processing time multiplied by the total demand per unit time for that operation. The width of a bin associated with a machine group corresponds to its tool magazine capacity, which is equal for all machines of the same type. The heights of the bins may differ among machine groups of the same type, since the groups may differ in size.

As mentioned in the previous section, tool commonality must be considered in the loading problem. Figure 1 shows the effect of the tool commonality on our bin-packing problem approach. In the figure, each operation is denoted by one or more rectangles which are disconnected horizontally, while each tool is represented by a specific vertical column. Rectangles denoting different operations can occupy common horizontal space (i.e., there may be more than one rectangle intersecting a vertical column), if those operations share identical tools. However, the rectangles cannot share common vertical space since processing time capacity cannot be shared.

The problem is to allocate the items to the bins in such a way that the widths of the bins are not exceeded and the sums of the heights (processing times) of the assigned items are as close as possible to the ideal heights of the bins. The ideal heights can be determined for a given configuration, which is assumed to be pre-specified. There are several available

methods to obtain the ideal workloads for the given configuration. In this paper, the model of Stecke and Solberg is used and the ideal workloads for a specific problem instance are calculated by a numerical steepest ascent method devised in Kim (1988) for their model. This procedure is based on the assumption that the throughput function is sufficiently well-behaved (e.g., unimodal) so that an ascent method will find the optimal workload allocation. It is similar in spirit to standard steepest ascent methods except that the gradients are estimated numerically. This numerical estimation is necessary because the throughput function cannot be expressed in closed form.

Many heuristics have been developed for one-dimensional bin-packing problems, and for some of them worst case performance bounds have been derived. Friesen and Langston (1986) analyze algorithms for a problem with different bin sizes, which is similar to our problem with the objective of unbalancing. See Garey and Johnson (1981) for details. Several assignment rules for simple bin-packing algorithms are given in the appendix.

For multi-dimensional cases, worst case performance ratios are known for simple heuristics (Garey and Graham 1975, Garey *et al.* 1976, and Yao 1980). In these simple heuristics, an attribute of the items (for example, height or width in our loading problem) or is used to prioritize the items before assigning them to bins. Kou and Markowsky (1977) give some examples of preprocessing criteria for the first-fit-decreasing algorithm or the best-fit-decreasing algorithm. Maruyama *et al.* (1977) have found in a computational experiment that it is better to use the attribute for which the bin capacity is more likely to be exceeded.

The loading problem can be viewed as a scheduling problem as well. For the balancing case, the surrogate objective is to minimize the maximum workload among machine groups for balancing. For the unbalancing case, the surrogate objective is to minimize the maximum ratio (among machine groups) of assigned workload to the ideal workload. (See Kim and Yano 1989 for support of these surrogate objectives.) If we consider machine groups as processors and the ideal workload for a group as the speed of the processor, then the loading problem can be viewed as a uniform processor scheduling problem to minimize the makespan. Here, uniform processors can process the same operations but the speeds of the processors may differ. If the ideal workloads are all equal as in the balancing case, the problem is similar to an identical processor scheduling problem.

Various algorithms exist for the identical processor minimum-makespan scheduling problem, and for some of them the worst case performance ratios are known. The LPT (longest processing time) algorithm has a worst case performance ratio of $4/3-1/3m$, where m is the number of processors (machines), and the MULTIFIT algorithm has a worst case performance ratio of $1.2+(1/2)^k$, where k is the number of iterations in the algorithm. See Graham (1969), Coffman *et al.* (1978), and Friesen (1984) for details.

For uniform processor minimum-makespan scheduling problems, Gonzalez *et al.* (1977), Ibarra and Kim (1977), Dobson (1984), and Friesen (1987) give algorithms and analyze their worst case performance ratios. MULTIFIT algorithms can be applied to this problem as well. Friesen and Langston (1983), and Kunde and Steppat (1985) give worst case performance ratios for some algorithms of this type.

The approaches to these scheduling problems need to be modified for the loading problem, since they consider only processing times, but not the tool magazine capacities. However, the basic ideas of these approaches can be used to develop algorithms for the loading problem.

3. Heuristic Algorithms

As stated in Section 2, the FMS loading problem is viewed as a two-dimensional bin-packing problem, and as a uniform processor scheduling problem to minimize the makespan with additional (tool magazine capacity) constraints. We explicitly consider tool savings that are achieved by assigning operations that require the same tools to the same group.

In our approach, the tool savings are considered explicitly by a labeling method. Initially for each tool, a set of operations requiring that tool is constructed from the problem data. Then we label each operation with the number of additional tool slots needed on each machine group. We also label each machine group with the set of tools and the set of operations assigned to it. These labels are changed as operations are assigned to machine groups, thereby incorporating tool savings explicitly.

Several types of algorithms have been developed in this research: LPT-type algorithms; simple MULTIFIT-type algorithms; combinations of these two; and MULTIFIT-type algorithms with additional considerations. The LPT-type algorithms are executed in one iteration while the others may need several iterations.

We first give an overview of the algorithms and then explain the various operation selection and assignment rules adapted from LPT and MULTIFIT algorithms.

Procedure 1. (for the algorithms of the LPT type)

Step 1. Initialize the (maximum) capacities of the machine groups and the labels of relevant machine groups and operations.

Step 2. Assign an operation using a selection rule considering the remaining capacities of the machine groups. If no operation can be assigned feasibly, stop. (The algorithm cannot generate a feasible loading plan.)

Step 3. Update the labels.

Step 4. If all the operations are assigned, stop. Otherwise, go to Step 2.

Procedure 2. (for the algorithms of the other three types)

Step 1. Initialize the bin sizes with the (maximum) time capacities and the tool magazine capacities, and the labels for machine groups and operations.

Step 2. Assign operations to machine groups using a selection rule considering the remaining capacities of the machine groups. Update the labels whenever an operation is assigned to a machine group. If all operations are assigned, set the upper bound of each bin sizes equal the current bin size and go to Step 3. If any remaining operations cannot be assigned, stop. (The algorithm cannot generate a feasible loading plan.)

Step 3. Re-initialize the bin sizes with the time capacities corresponding to the ideal workloads and the tool magazine capacities. Also re-initialize the labels.

Step 4. Assign the operations as in Step 2. If all the operations can be assigned, stop. (The resulting loading plan is optimal.) Otherwise, set the lower bound of each bin size equal to the current bin size, and go to Step 5.

Step 5. If the corresponding lower and upper bounds are close enough, stop. Otherwise, re-initialize the bin sizes with the midpoints of the two bounds. Also, re-initialize the labels.

Step 6. Assign operations as in Step 2. If all the operations can be assigned, reset the upper bound with the current bin sizes. Otherwise, reset the lower bounds with the current bin sizes. Go to Step 5.

Before stating the algorithms, we define some additional terms.

1) Time Slot Ratio (T/S) for a machine group:

the ratio of the remaining processing-time capacity of the machine group to the remaining tool magazine capacity.

2) Time Slot Ratio (T/S) for an operation:

the ratio of the processing time to the number of tool slots needed for the operation. Each operation has a T/S ratio for each machine group. These ratios may change when

an operation is assigned to a machine group, and may differ among machine groups even for the same operation, because of tool commonality .

3) Prospective Tightness of tool magazine capacity constraints (*PT*) :

$$PT = (TN_1 / TA_1) * (TA_2 / TN_2),$$

where TN_1 is the number of tool slots needed for unassigned operations when tool commonality is not considered, TA_1 is the number of tool slots available at the moment, TN_2 is the number of tool slots needed for assigned operations without considering tool savings, and TA_2 is the number of tool slots already used at the moment. *PT* is the expected utilization of the remaining space in the tool magazine under the assumption that the trend of tool savings will continue. Note that TA_2 / TN_2 represents the ratio of slots actually used to the sum of slots needed for the assigned operations, and thereby reflects the extent of tool savings thus far. *PT* changes when an operation is assigned to a machine group.

4) Machine Preference Ratio (*PR*):

$$PR = (NS_1 - NS_2) / N_n ,$$

where NS_1 is the number of tool slots saved when an operation is assigned to the most *preferred* machine group (the group for which the fewest additional tool slots are required), NS_2 is the number of tools slots saved when the operation is assigned to the second most preferred machine group, and N_n is the sum of tool slots needed for all tools required for the operation. *PR* reflects the percentage savings of tool slots achieved by assigning an operation to the most preferred machine group rather than the second most preferred group. We are assuming here that the operation can be feasibly assigned to its most preferred and second most preferred machine groups.

Now we state the selection rules for operation assignments. Note that *machine* and *machine group* are used synonymously in these descriptions.

A) LPT type algorithms

LPTO : The operation with the longest processing time is assigned to the machine with the largest remaining time capacity.

LPTL : The operation with the longest processing time is assigned to the machine which has the minimum load after the operation is assigned to it.

B) Simple MULTIFIT type algorithms

MTDI : First, the operations are sorted in a descending order of processing times, and the machines are sorted in an ascending order of processing-time capacities. Then, the first-fit rule is used for assigning in the MULTIFIT algorithm.

MTDD : This is the same as MTDI except that the machines are sorted in descending order of processing-time capacities.

C) Combinations of the two

CPT : The MULTIFIT algorithm is used to find the minimum necessary initial processing-time capacities of the various machine groups to assign all the operations. The assignment rule is the same as that of LPTO for each initial capacity.

CPL : This is the same as CPT except that this uses LPTL instead of LPTO.

D) MULTIFIT type algorithms with additional considerations

APS : The operation that requires the largest number of tool slots on the most preferred machine is assigned to that machine.

APS2 : The operation that needs the smallest number of tool slots on the most preferred machine is assigned to that machine.

ARM : After the machine with the largest T/S value is found, the unassigned operation with the largest T/S value on that machine is assigned to that machine.

APM : If prospective tightness (PT) is greater than some value (for example, 1), the operation with the highest machine preference ratio (PR) is assigned to its most preferred machine. Otherwise, ARM is used.

It is possible to construct several variations of the rules. For example, we can replace the largest T/S value with the smallest T/S value in the description for ARM and APM. This may be meaningful since ordinary dominance of the first-fit-decreasing rule over the first-fit-increasing rule has not been shown for our problem. Also, instead of the ratio T/S, the product $T \cdot S$ can be used because the area of a bin or an item is another simple criterion for assignment in the two-dimensional bin-packing problem. Several variations, which are included in the test, are listed in the appendix.

Although these algorithms differ from existing heuristics in that they consider tool magazine capacity and tool commonality, those in the last category are more novel than the others. In these algorithms, tool magazine capacity and tool commonality effects have more influence on the assignments. The LPTL criterion is the same as that of Dobson (1984). The basic idea of the ARM and APM selection criteria is that larger items are packed in larger bins to achieve a better loading. Since we must consider tool magazine

capacity as well as processing time capacity, the idea of the T/S ratio is used in ARM and APM. Prospective tightness (PT) was suggested by Maruyama *et al.* (1977) as a means of emphasizing feasibility.

Since these algorithms are expected to be very fast, they can be executed sequentially within one procedure. An algorithm of this type is called a *composite algorithm*. For example, in the MULTIFIT type algorithms, for a given bin size a composite algorithm tries to obtain a feasible packing using the algorithms one by one, and terminates when a feasible solution has been found. If we order the algorithms from the "best" to the "worst," the CPU time is not expected to increase very much. The composite algorithms described in this paper will be called ALLBAL and ALLUNB for the balancing and unbalancing objectives, respectively.

Figure 2 illustrates the composite algorithms devised for the loading problem. In the figure, R is the number of individual heuristic algorithms included in the composite algorithm, and r is the index of each individual algorithm. As can be seen from the figure, this procedure is identical to the procedure for the MULTIFIT type algorithms (Procedure 2) except that this procedure tries up to R assignment rules for each bin size until a feasible solution is found.

4. Computational Results

Existing optimal algorithms may not be able to find solutions for realistic problems in a reasonable amount of time. Therefore, we first selected the best-performing algorithm among the heuristics on a broad range of loading problems, then compared it with optimal algorithms. The problems were necessarily limited in size because of the computational requirements of the optimal procedures.

In this section, we first compare the heuristic algorithms with each other and then with existing optimal algorithms. Since our heuristic approach to the loading problem is designed to be general rather than to be system-specific, it is desirable to test the algorithms on a wide range of problems. Unfortunately, because of the proprietary nature of the data, it is extremely difficult to obtain actual data for a wide variety of systems. Instead, the test problems were generated randomly so that the resulting data are representative of real systems relatively well. In fact, 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.

Comparison of Heuristics

To compare the heuristic algorithms, four sets of loading problems were generated, one for each combination of the objectives (balancing or unbalancing) and level of tightness of the tool magazine capacity constraints (tight or not tight). Tightness was considered since the capability to find a feasible solution is also an important criterion in the evaluation of heuristic loading algorithms. Each set has 30 problems. The number of operations considered in the test problems ranges from 15 to 40 (15, 20, 25, 30, 35, and 40) and the number of machine groups ranges from 3 to 6. The same set of 30 problems was used for the two levels of tightness of the tool magazine capacity constraints. Tightness of the constraints was controlled by modifying the number of tool slots in the magazine. For the not-tight cases, the number of tool slots in the magazine was selected to be the multiple of 10 nearest to the sum of tool slots needed for all operations (without any tool commonality) divided by the number of machine groups and then multiplied by 0.9. Thus, on the average, 90% of all tools could be loaded onto the machines if no tools were shared by operations. For the tight cases, these numbers were reduced by 15 to 30 percent from those of the not-tight cases. Also the resulting numbers were rounded to be a multiple of 10. The resulting tool magazine capacities range from 55 to 150. For each problem, the other required data were generated as described below:

1) The processing time for each operation is generated randomly using a discrete uniform distribution. The minimum value ranges from 10 to 40, and the maximum value ranges from 35 to 450.
2) The total number of tools ranges from 40 to 80.
3) The number of tool slots needed for each tool ranges from 1 to 5. The probabilities for these values are 0.65, 0.05, 0.20, 0.05, and 0.05, respectively.
4) The number of tools needed for each operation is generated randomly from a discrete uniform distribution between 4 and 15.
5) Tools are assigned to operations as follows. The number of operations requiring each tool is generated from a discrete uniform distribution between 1 and 5. For each tool, the indices of operations sharing it are randomly selected from among those operations for which the number of tools from Step 5 has not yet been assigned. When there is a conflict between the results of Step 5 and the number of operations requiring specific tools (which occurs because of the interdependence of these two sets of data), the latter is modified as long as it does not exceed a maximum of 5.

6) Tools that are used solely by a single operation are treated as a single tool for simplicity. The number of tool slots needed for the single artificial tool is the sum of the tool slots required by the individual tools.
7) The number of machines in each group is generated from a discrete uniform distribution between 1 and 3.
8) The number of jobs (also pallets) circulating in the system was set equal to 10, 15, or 20, with the value selected to be approximately proportional to the total number of machines.

All the algorithms were coded in FORTRAN and run on an AMDAHL 5860 with an optimizing compiler. The results are given in Tables 1 through 4. Two performance measures are considered: *maximum deviation ratio* and *relative performance ratio*. The former is the maximum ratio of actual workload to the ideal workload among machine groups. The latter is the ratio of the maximum deviation ratio of an algorithm to that of the best algorithm. The best algorithm is defined as that with the smallest maximum deviation ratio for the problem. The results of the algorithms listed in the appendix were included in the analysis but are not reported here since they were inferior to the others. The mean values and the maximum values of the ratios in the tables were calculated using only the values from feasible solutions. Therefore, the algorithms with the smallest ratios are not necessarily the best.

As can be seen in Tables 1 and 2, when the objective is balancing the workloads, there is no exceptionally dominating algorithm, but MTDI, MTDD, LPTO, LPTL, CPT, and CPL are slightly better than the others. Note that MTDI and MTDD are identical for the objective of balancing and so are LPTO and LPTL, since the processing-time capacities of the machine groups are equal. When the tool magazine capacity constraints are tight, ARM and APM are good in finding feasible solutions, since they consider both workloads and tool magazine capacities at the same time.

Table 1. Performance of the heuristic algorithms (balancing, not tight)

Algorithm	Number of Solutions †		Maximum Deviation Ratio		Relative Performance Ratio ‡		CPU Time * (milliseconds)		
	feas.	best ‡	Mean	Max.	Mean	Max.	20	30	40
LPTO	30	8 (8)	1.017	1.055	1.010 (1.010)	1.051 (1.051)	3	3	4
LPTL	30	8 (8)	1.017	1.055	1.010 (1.010)	1.051 (1.051)	3	3	4
MTDI	30	14 (14)	1.012	1.046	1.005 (1.005)	1.046 (1.046)	21	28	36
MTDD	30	14 (14)	1.012	1.046	1.005 (1.005)	1.046 (1.046)	21	29	35
CPT	30	8 (8)	1.017	1.055	1.010 (1.010)	1.051 (1.051)	23	32	41
CPL	30	8 (8)	1.017	1.055	1.010 (1.010)	1.051 (1.051)	21	28	35
APS	30	0 (0)	1.016	1.306	1.099 (1.099)	1.296 (1.296)	8	14	20
APS2	30	0 (0)	1.144	1.270	1.137 (1.137)	1.261 (1.261)	8	13	20
ARM	29	1 (1)	1.043	1.154	1.036 (1.036)	1.132 (1.132)	22	30	37
APM	29	1 (1)	1.041	1.154	1.034 (1.034)	1.132 (1.132)	23	31	38
ALLBAL	30	(30)	1.006	1.019	(1.000)	(1.000)	103	119	160

† The figures in these two columns denote the number of problems for which the algorithm found a feasible solution and the best solution among those from the various algorithms tested in this research.

‡ The figures in parentheses were obtained when the algirhtm ALLBAL is included in the analysis.

* The CPU times are the means for 5 twenty-, thirty-, and forty-operation problems which were run on an AMDAHL 5860.

When the objective is unbalancing the workloads (Tables 3 and 4), ARM and APM dominate the other algorithms. While LPT-type algorithms and simple MULTIFIT-type algorithms (those in the first three categories) worked better for the balancing objective, the more complicated MULTIFIT-type algorithms (those in the last category) were better for the unbalancing objective. This can be explained as follows. The algorithms in the earlier categories consider the current workloads of the machine groups when assigning operations, and therefore may not effectively consider the differences in the processing time capacities among the groups. On the other hand, the other algorithms (e.g., ARM and APM) consider the current remaining processing time capacities of the groups when assigning operations. Hence, it is likely that a larger number of operations is assigned to a group with a large capacity, making the workloads closer to the ideal workloads.

Table 2. Performance of the heuristic algorithms (balancing, tight)

Algorithm	Number of Solutions †		Maximum Deviation Ratio		Relative Performance Ratio ‡		CPU Time * (milliseconds)		
	feas.	best ‡	Mean	Max.	Mean	Max.	20	30	40
LPTO	25	6 (6)	1.021	1.059	1.013 (1.013)	1.055 (1.055)	3	3	4
LPTL	25	6 (6)	1.021	1.059	1.013 (1.013)	1.055 (1.055)	3	4	4
MTDI	23	6 (6)	1.041	1.126	1.031 (1.032)	1.122 (1.122)	20	28	35
MTDD	23	6 (6)	1.041	1.126	1.031 (1.032)	1.122 (1.122)	21	28	34
CPT	28	11 (9)	1.024	1.103	1.013 (1.015)	1.099 (1.099)	23	32	40
CPL	22	6 (6)	1.021	1.056	1.012 (1.012)	1.053 (1.053)	20	27	35
APS	25	1 (1)	1.094	1.254	1.084 (1.084)	1.246 (1.248)	14	23	36
APS2	27	0 (0)	1.125	1.270	1.113 (1.115)	1.261 (1.261)	13	14	22
ARM	29	4 (2)	1.042	1.171	1.029 (1.031)	1.123 (1.153)	22	30	37
APM	29	3 (3)	1.045	1.171	1.032 (1.034)	1.123 (1.153)	24	31	39
ALLBAL	30	(27)	1.014	1.064	(1.001)	(1.017)	109	131	171

†, ‡, * See the footnotes of Table 1.

However, many algorithms in the last category (including those in the Appendix) did not perform well. This may be due to the characteristics of the assignment rules. Tool magazine capacity and tool slots needed for an operation were overemphasized and therefore processing time, the criterion directly related to the objectives, was not adequately considered. Moreover, APS and APS2, which select operations and machines considering tool slots, and so presumably save them, failed to find a feasible solution many times. In these rules, however, the machine preference ratio (PR) is not used. Note that APM, which uses PR, is good in finding a feasible solution. This indicates the advantage of using the PR values.

Table 3. Performance of the heuristic algorithms (unbalancing, not tight)

Algorithm	Number of Solutions †		Maximum Deviation Ratio		Relative Performance Ratio ‡		CPU Time * (milliseconds)		
	feas.	best ‡	Mean	Max.	Mean	Max.	20	30	40
LPTO	30	1 (0)	1.348	2.099	1.174 (1.183)	1.443 (1.443)	3	3	4
LPTL	30	0 (0)	1.495	2.065	1.309 (1.318)	1.569 (1.569)	3	4	4
MTDI	27	0 (0)	1.491	2.341	1.285 (1.291)	1.820 (1.820)	21	28	35
MTDD	30	4 (1)	1.351	2.162	1.172 (1.180)	1.553 (1.553)	21	28	34
CPT	30	1 (0)	1.338	2.099	1.163 (1.171)	1.405 (1.405)	24	32	39
CPL	30	0 (0)	1.403	2.397	1.217 (1.226)	1.589 (1.589)	20	28	34
APS	30	0 (0)	1.701	2.641	1.490 (1.502)	2.440 (2.505)	18	41	63
APS2	30	0 (0)	1.503	2.542	1.305 (1.314)	1.791 (1.791)	23	41	92
ARM	30	22 (16)	1.152	1.508	1.007 (1.015)	1.081 (1.119)	23	31	40
APM	30	22 (16)	1.152	1.508	1.007 (1.015)	1.081 (1.119)	24	33	41
ALLUNB	30	(27)	1.136	1.508	(1.000)	(1.004)	78	96	151

†, * See the footnotes of Table 1.

‡ The figures in parentheses in these columns were obtained after ALLUNB was included in the analysis.

The CPU times for these algorithms are short enough to be run several times as a subroutine in the context of solving more global problems. LPTO and LPTL needed no more than 5 milliseconds for the forty-operation problems. In MULTIFIT algorithms, the CPU time depends on the number of iterations. For the number of iterations used in our algorithms, 8, the CPU time did not exceed 150 milliseconds for the forty-operation problems. Even more favorably, the algorithms which perform well did not need more than 50 milliseconds.

The figures within parentheses in Tables 1 through 4 show the performance of the algorithms when the two composite algorithms, ALLBAL and ALLUNB, were included in the analysis. These procedures include the individual algorithms presented in the previous section. As stated in Langston (1987), composite algorithms can find not only the best solution among those from individual algorithms but may also find even better solutions in many cases. This can be seen by comparing the number of best solutions found with or without including the composite algorithms. For example, in Table 4, the number of the best solutions found by APM dropped from 26 to 16, which means that ALLUNB found at least 10 new best solutions.

Table 4. Performance of the heuristic algorithms (unbalancing, tight)

Algorithm	Number of Solutions †		Maximum Deviation Ratio		Relative Performance Ratio ‡		CPU Time * (milliseconds)		
	feas.	best ‡	Mean	Max.	Mean	Max.	20	30	40
LPTO	16	3 (2)	1.618	2.178	1.164 (1.187)	1.693 (1.693)	3	3	4
LPTL	15	0 (0)	1.829	2.446	1.298 (1.323)	1.573 (1.573)	3	3	4
MTDI	6	1 (1)	1.701	1.969	1.211 (1.236)	1.394 (1.394)	20	29	34
MTDD	13	0 (0)	1.826	2.294	1.227 (1.268)	1.349 (1.425)	20	27	33
CPT	16	1 (0)	1.593	2.198	1.131 (1.154)	1.414 (1.414)	23	31	40
CPL	8	0 (0)	1.793	2.446	1.257 (1.300)	1.573 (1.573)	21	26	33
APS	15	1 (1)	2.113	3.057	1.405 (1.425)	1.877 (1.877)	35	36	94
APS2	16	0 (0)	1.954	2.644	1.383 (1.429)	1.701 (1.762)	35	59	95
ARM	29	24 (15)	1.528	2.203	1.008 (1.033)	1.139 (1.180)	22	30	39
APM	29	26 (16)	1.525	2.203	1.006 (1.031)	1.139 (1.180)	23	31	40
ALLUNB	29	(28)	1.482	2.203	(1.000)	(1.001)	92	112	135

†, ‡, * See the footnotes of Table 3.

Also, it is apparent that the composite algorithms did not find the best solution in all cases, since for some problems the best solutions were found by algorithms that were not included in the composite algorithms. Some individual algorithms were omitted from the composite algorithm because their contribution was expected to be minimal.

Evaluation of Heuristics

We compare the heuristics with branch and bound algorithms developed for the same objectives because existing heuristic procedures do not explicitly consider tool magazine capacity constraints, and therefore cannot be guaranteed to find a feasible solution. Moreover, some of the existing heuristics were designed for entirely different objectives. Among the existing heuristics, only those described by Stecke and Talbot (1985), Ammons *et al.* (1985), and Shanker Tzen (1985) were designed for similar objectives. The latter two papers use surrogate objectives for throughput maximization which have been shown elsewhere (Kim and Yano 1989) not to perform quite as well as the objective that we use here. The two heuristics briefly described by Stecke and Talbot are similar in spirit to a few of the individual algorithms tested here.

We use the same data sets as those which were used in Kim and Yano (1989). In that paper, two sets of loading problems are generated randomly, one for balancing and one for unbalancing. Each set has 30 problems with 10 to 20 operations and 2 to 5 machine groups. For each of the 30 problems, the other required data were generated with a method similar to the one used in this paper.

Three different solution procedures were tested for each set: the composite heuristic algorithm; the branch and bound algorithm of Berrada and Stecke (1986); and the branch and bound algorithm suggested by Kim and Yano (1989). These are called ALLBAL or ALLUNB, B&B1, and B&B2, respectively. Solutions from ALLBAL or ALLUNB are used in B&B2 as initial incumbent solutions. We also performed some preliminary tests on a few of the simple individual heuristics (e.g., LPTO), and decided not to test them further because of their poor performance in finding feasible solutions, as described earlier.

These procedures were coded in FORTRAN and run with an optimizing compiler for up to 30 seconds of CPU time on an IBM 3090-400, which is approximately twice as fast as the AMDAHL 5860. The computer code for the original branch and bound algorithm of Berrada and Stecke was provided by them. Note that this branch and bound algorithm is an ε-optimal algorithm. The value ε was set in the same way as in the original code, where ε is the minimum value of the processing times of the operations divided by the number of machine groups.

The results are shown in Tables 5 and 6. Two performance measures appear in the tables. The maximum deviation ratio is defined as in Tables 1–4, and the throughputs, or *expected production rates* (EPR), in the tables were calculated using the closed queueing network model of Stecke and Solberg (1985). The latter measure was included to compare performance of the algorithms for the final objective, maximizing the throughput. This is a relative value which is normalized so that the maximum possible value is 1.0, which can be achieved when there is only one group, i.e., all machines can process all operations.

When the objective is balancing workloads, the composite heuristic algorithm (ALLBAL) worked well. It was not only faster but also gave better solutions (lower maximum deviation ratios) than the ε-optimal algorithm, B&B1. It also gave solutions with higher throughputs than B&B1 except in three problems (problems 4, 9, and 11), in which the differences were less than 0.5%.

The heuristic algorithm can also enhance the performance of branch and bound algorithms, when it is used as an initial feasible solution. This can be seen from a comparison of B&B1 and B&B2, which uses the heuristic solution as the initial incumbent. In the computational comparison, B&B2 outperformed B&B2 in both computation time and solution quality.

Table 5. Computational results for the balancing objective

Problem†	ALLBAL			B&B1			B&B2		
	Max. Dev. Ratio	EPR	CPU Time (sec.)	Max. Dev. Ratio	EPR	CPU Time (sec.)	Max. Dev. Ratio	EPR	CPU Time (sec.)
1 (10,2)	‡ 1.002	.937	.022	1.027	.934	.050	1.002	.937	.026
2 (10,2)	1.006	.952	.023	1.010	.952	.028	1.002	.952	.026
3 (10,3)	1.016	.882	.031	1.049	.872	.109	1.006	.882	.091
4 (10,3)	1.047	.895	.037	1.053	.898	.077	1.006	.909	.105
5 (10,3)	1.037	.899	.059	1.054	.898	.756	1.004	.909	.127
6 (12,2)	‡ 1.001	.952	.025	1.026	.948	1.119	1.001	.952	.028
7 (12,3)	‡ 1.008	.909	.027	1.019	.908	2.858	1.008	.909	.030
8 (12,3)	‡ 1.000	.909	.003	1.032	.905	4.295	1.000	.909	.003
9 (12,4)	1.057	.859	.035	1.072	.859	5.210	1.042	.861	2.021
10 (12,4)	1.009	.869	.035	1.117	.841	5.908	1.003	.869	.271
11 (14,3)	1.030	.902	.037	1.030	.906	9.025	1.000	.909	.083
12 (14,3)	1.035	.905	.057	1.028	.903	3.680	1.000	.909	.353
13 (14,4)	1.012	.869	.040	1.012	.869	5.766	1.006	.870	2.507
14 (14,4)	1.037	.866	.055	1.046	.863	13.978	1.004	.869	.410
15 (14,5)	1.043	.828	.039	1.104	.815	11.794	1.001	.833	.806
16 (16,3)	1.005	.909	.042	1.028	.907	14.714	1.002	.909	.076
17 (16,3)	1.005	.926	.052	* 1.963	.510	30.	1.002	.926	.056
18 (16,4)	1.012	.892	.049	* 2.617	.382	30.	1.001	.893	.092
19 (16,4)	1.025	.891	.039	1.066	.878	17.968	1.000	.893	.070
20 (16,5)	1.050	.853	.051	1.117	.833	28.792	1.003	.862	15.574
21 (18,3)	1.008	.909	.043	* 2.101	.476	30.	1.001	.909	.119
22 (18,4)	1.009	.893	.046	* 1.358	.715	30.	1.000	.893	9.904
23 (18,4)	1.014	.892	.067	* 1.602	.614	30.	1.001	.893	.090
24 (18.5)	1.019	.860	.056	* 1.450	.654	30.	1.004	.862	25.675
25 (18,5)	1.049	.868	.074	* 1.301	.722	30.	* 1.025	.876	30.
26 (20,3)	1.009	.909	.058	* 1.041	.901	30.	1.002	.909	.062
27 (20,4)	1.007	.893	.048	* 2.442	.410	30.	1.001	.893	13.807
28 (20,4)	1.006	.893	.059	* 2.060	.478	30.	1.003	.893	.063
29 (20,5)	1.027	.860	.073	* 1.919	.505	30.	* 1.027	.861	30.
30 (20,5)	1.022	.877	.061	* 2.257	.442	30.	* 1.022	.877	30.

† The numbers in parentheses denote the number of operations and number of machine groups.
‡ Heuristic solutions which were proven to be optimal.
* Incumbent solutions after 30 seconds of CPU time on an IBM3090-400.

Although B&B2 gave slightly better solutions, ALLBAL performed very well. For most of the test problems, the maximum deviation ratios from ALLBAL are very close to those from B&B2. The largest difference between these ratios was 4.7%. However, the difference of the expected production rates was smaller, with the worst difference being 1.5%. This suggests that the heuristic algorithms perform well for the final objective of maximizing throughput.

Table 6. Computational results for the unbalancing objective

Problem†	ALLUNB			B&B1			B&B2		
	Max. Dev. Ratio	EPR	CPU Time (sec.)	Max. Dev. Ratio	EPR	CPU Time (sec.)	Max. Dev. Ratio	EPR	CPU Time (sec.)
1 (10,2)	1.002	.941	.017	1.026	.940	.045	1.000	.941	.023
2 (10,2)	1.010	.955	.017	1.016	.952	.017	1.000	.955	.020
3 (10,3)	‡ 1.008	.886	.018	1.019	.885	.047	1.008	.886	.051
4 (10,3)	1.029	.917	.019	1.058	.897	.023	1.003	.919	.030
5 (10,3)	‡ 1.015	.916	.023	1.057	.897	.650	1.015	.916	.033
6 (12,2)	‡ 1.001	.956	.020	1.036	.942	2.792	1.001	.956	.040
7 (12,3)	‡ 1.033	.905	.024	1.067	.904	6.218	1.004	.912	.236
8 (12,3)	‡ 1.018	.915	.024	1.064	.893	3.487	1.018	.915	.106
9 (12,4)	1.057	.859	.026	1.083	.838	6.267	1.039	.870	.516
10 (12,4)	1.020	.878	.024	1.088	.867	3.823	1.011	.880	.123
11 (14,3)	1.029	.905	.024	1.030	.906	22.551	1.001	.913	1.311
12 (14,3)	‡ 1.008	.919	.027	1.008	.919	11.728	1.008	.919	.193
13 (14,4)	1.012	.872	.032	1.045	.864	7.048	1.004	.873	2.157
14 (14,4)	1.031	.872	.025	1.031	.877	5.738	1.009	.880	.793
15 (14,5)	1.077	.824	.029	1.042	.834	7.904	1.009	.836	5.970
16 (16,3)	1.023	.909	.033	**	**	30.	1.001	.913	10.125
17 (16,3)	‡ 1.052	.912	.026	**	**	30.	1.052	.912	3.670
18 (16,4)	1.013	.895	.032	**	**	30.	* 1.002	.896	30.
19 (16,4)	1.028	.899	.034	**	**	30.	1.003	.902	7.404
20 (16,5)	1.061	.849	.035	**	**	30.	* 1.009	.865	30.
21 (18,3)	1.009	.912	.032	**	**	30.	1.000	.913	•30.039
22 (18,4)	1.021	.895	.037	**	**	30.	* 1.020	.892	30.
23 (18,4)	1.023	.897	.034	**	**	30.	* 1.023	.897	30.
24 (18.5)	1.030	.860	.040	**	**	30.	1.003	.865	•30.037
25 (18,5)	1.016	.888	.029	**	**	30.	* 1.013	.889	30.
26 (20,3)	1.011	.912	.030	**	**	30.	* 1.011	.912	30.
27 (20,4)	1.007	.895	.036	**	**	30.	* 1.003	.896	30.
28 (20,4)	1.029	.892	.036	**	**	30.	* 1.029	.892	30.
29 (20,5)	1.017	.863	.040	**	**	30.	* 1.005	.865	30.
30 (20,5)	1.028	.884	.040	**	**	30.	* 1.022	.886	30.

†, ‡, * See the footnotes of Table 5.

** No feasible solution was found in 30 seconds of CPU time on an IBM3090-400.

• More than 30 seconds was spent to obtain an optimal solution, since the solution obatined after 30 seconds was close to the lower bound.

Similar results can be seen for the case of unbalancing. ALLUNB performed better than B&B1 in both time and solution quality except in Problem 15, for which the difference between the production rates was less than 1.2%. The worst differences between the maximum deviation ratios and the production rates of ALLUNB and B&B2 are 6.7% and 1.8%, respectively. In this case too, B&B2, which uses ALLUNB for an initial solution by far outperformed B&B1.

Since all of the machines are of the same type, the processing time of each operation is the same on all machine groups. This may be a reason that B&B2 was very slow for relatively small problems. Berrada and Stecke (1986) observed that if the processing times are the same, branch and bound algorithms are not be able to eliminate dominated alternatives as efficiently as otherwise. Consequently, for *easier* problems, the CPU time may be much smaller.

5. Summary and Concluding Remarks

A heuristic approach is suggested for the FMS loading problem with the objective of maximizing the expected production rate. Since it is difficult to achieve this objective directly, workload balancing or unbalancing was used as an intermediate objective. With these objectives, the FMS loading problem was viewed as a two dimensional bin packing problem and/or as a uniform processor scheduling problem to minimize the makespan.

Various heuristics were devised using basic concepts of the LPT and MULTIFIT algorithms. To the best of our knowledge, the MULTIFIT algorithm has not been yet applied to the FMS loading problem in published research. Moreover, our algorithms explicitly consider the savings due to tool commonality using a labeling method. The MULTIFIT algorithm with consideration of the tool magazine capacity performed better than the others for the objective of unbalancing, while there was little difference in performance of the better algorithms for the balancing objective.

Also devised and tested was a composite algorithm in which several heuristics are executed sequentially. The composite algorithm not only was fast but also provided very good solutions. For most problems tested in this paper, it gave a better solution than an existing ε-optimal branch and bound algorithm with less CPU time and performed nearly as well as optimal procedures in terms of solution quality.

The results in this paper suggest that the FMS loading problems can be solved near-optimally in a short time. Our approach can be used as a subroutine in larger decision problems (see Kim 1988 for an example of its use). Also it will be useful when loading problems must be solved frequently or on mini- or micro-computers. In this paper we used the objective of throughput maximization. Further research is needed to incorporate other objectives such as due dates.

Appendix

A. *Assignment rules of simple bin-packing algorithms (one-dimensional case)*

Next Fit: With arbitrarily given indexes of items and bins, pack the lowest-indexed item among items not yet assigned into the highest indexed non-empty bin.

First Fit: Pack the lowest-indexed item into the lowest indexed bin into which the item will fit.

Best Fit: Pack the lowest-indexed item into the bin which will have the smallest remaining capacity after the item is assigned

First Fit Decreasing (Increasing): Sort items in decreasing (increasing) order of size and then apply the first fit rule.

Best Fit Decreasing (Increasing): Sort items in decreasing (increasing) order of size and then apply the best fit rule.

B. *Additional selection rules for MULTIFIT type algorithms*

ARO : The operation and machine pair which have the largest T/S value are selected, and the operation is assigned to that machine.

APO : This is the same as APM except that ARO is used when PT is less than a threshold value.

ARMI : After the machine with the smallest T/S value is found, the unassigned operation with the smallest T/S value on that machine is assigned to that machine.

APMI : This is the same as APM except that ARMI is used when PT is less than a specified value.

AMM : This is the same as ARM except that the product of time and tool slots ($T \cdot S$) is used instead of their ratio.

AMO : This is the same as ARO except that the product of time and tool slots ($T \cdot S$) is used instead of their ratio.

APMM : This is the same as APM except that the product of time and tool slots ($T \cdot S$) is used instead of their ratio.

APMO : This is the same as APO except that the product of time and tool slots ($T \cdot S$) is used instead of their ratio.

References

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

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

Carrie, A. S. and D. T. S. Perera, 1986. Work Scheduling in FMS under Tool Availability Constraints. *Int. J. Prod. Res.*, Vol.24, No.6, pp.1299-1308.

Coffman, Jr., E. G., M. R. Garey, and D. S. Johnson, 1978. An Application of Bin-Packing to Multiprocessor scheduling. *SIAM Journal on Computing*, Vol.7, No.1, pp.1-17.

Dobson, G., 1984. Scheduling Independent Tasks on Uniform Processors. *SIAM Journal on Computing*, Vol.13, No.4, pp.705-716.

Friesen, D. K., 1984. Tighter Bounds for the MULTIFIT Processor Scheduling Algorithm. *SIAM Journal on Computing*, Vol.13, No.1, pp.170-181.

Friesen, D. K., 1987. Tighter Bounds for LPT Scheduling on Uniform Processors. *SIAM Journal on Computing*, Vol.16, No.3, pp.554-560.

Friesen, D. K. and M. A. Langston, 1983. Bounds for MULTIFIT Scheduling on Uniform Processors. *SIAM Journal on Computing*, Vol.12, No.1, pp.60-70.

Friesen, D. K. and M. A. Langston, 1986. Variable Sized Bin Packing. *SIAM Journal on Computing*, Vol.15, No.1, pp.222-230.

Garey, M. R. and R. L. Graham, 1975. Bounds for Multiprocessor Scheduling with Resource Constraints. *SIAM Journal on Computing*, Vol.4, No.2, pp.187-200.

Garey, M. R., R. L. Graham, D. S. Johnson, and A. C. Yao, 1976. Resource Constrained Scheduling as Generalized Bin Packing. *Journal of Combinatorial Theory(A)*, Vol.21, pp.257-298.

Garey, M. R. and D. S. Johnson, 1981. Approximation Algorithms for Bin Packing Problems: A Survey. in *Analysis and Design of Algorithms in Combinatorial Optimization*, G. Ausiello and M. Lucertini, Eds., Springer-Verlag, New York.

Gonzalez, T., O. H. Ibarra, and S. Sahni, 1977. Bounds for LPT Schedules on Uniform Processors. *SIAM Journal on Computing*, Vol.6, No.1, pp.155-166.

Graham, R. L., 1969. Bounds on Multiprocessing Timing Anomalies. *SIAM Journal on Applied Mathematics*, Vol.17, pp.416-429.

Greene, T. J. and R. P. Sadowski, 1986. A Mixed Integer Program for Loading and Scheduling Multiple Flexible Manufacturing Cells. *European Journal of Operational Research*, Vol.24, pp.379-386.

Ibarra, O. H. and C. E. Kim, 1977. Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors. *Journal of the Association for Computing Machinery*, Vol.24, No.2, pp.280-289.

Kim, Y-D., 1988. An Iterative Approach for System Setup Problems of Flexible Manufacturing Systems. Ph.D. Dissertation, The University of Michigan.

Kim, Y-D. and C. A. Yano, 1989. A New Branch and Bound Approach for Loading Problems in Flexible Manufacturing Systems. Technical Report 89-5, Dept. of Industrial and Operations Engineering, The University of Michigan.

Kou, L. T. and G. Markowsky, 1977. Multidimensional Bin Packing Algorithms. IBM *Journal of Research and Development*, Vol.21, pp.443-448.

Kunde, M. and H. Steppat, 1985. First Fit Decreasing Scheduling on Uniform Processors. *Discrete Applied Mathematics*, Vol.10, pp.165-177.

Kusiak, A., 1985. Loading Models in Flexible Manufacturing Systems. *Flexible Manufacturing: Recent Development on FMS, Robotics, CAD/CAM, CIM*, Raouf, A. and Ahmad, S. I. eds., Elsevier Science Publishers B.V., Amsterdam, pp.119-132.

Langston, M. A., 1987. A Study of Composite Heuristic Algorithms. *Journal of Operational Research Society*, Vol.38, No.6, pp.539-544.

Maruyama, K., S. K. Chang, and D. T. Tang, 1977. A General Packing Algorithm for Multidimensional Resource Requirement. *International Journal of Computer and Information Sciences*, Vol.6, No.2, pp.131-149.

Lashkari, R. S., S. P. Dutta, and A. M. Padhye, 1987. A New Formulation of Operation Allocation Problem in Flexible Manufacturing Systems: Mathematical Modelling and Computational Experience. *Int. J. Prod. Res.*, Vol.25, No.9, pp.1267-1283.

Rajagopalan, S., 1986. 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*, pp.312-314.

Sarin, S. C. and C. S. Chen, 1987. The Machine Loading and Tool Allocation Problem in a Flexible Manufacturing System. *Int. J. Prod. Res.*, Vol.25, No.7, pp.1081-1094.

Shanker, K. and Y-J. J. Tzen, 1985. A Loading and Dispatching Problem in a Random Flexible Manufacturing System. *Int. J. Prod. Res.*, Vol.23, pp.579-595.

Shanthikumar, J. G. and K. E. Stecke, 1986. Reducing Work-in-Process Inventory in Certain Classes of Flexible Manufacturing Systems. *European Journal of Operational Research*, Vol.26, pp.266-271.

Stecke, K. E., 1983. Formulation and Solution of Nonlinear Integer Production Planning Problems for Flexible Manufacturing Systems. *Management Science*, Vol.29, No.3, pp.273-288.3

Stecke, K. E., 1986. A Hierarchical Approach to Solving Machine Grouping and Loading Problems of Flexible Manufacturing Systems. *European Journal of Operational Research*, Vol.24, pp.369-378.

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

Stecke, K. E. and J. J. Solberg, 1985. 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.

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

Yao, A. C., 1980. New Algorithms for Bin Packing. *Journal of the Association for Computing Machinery*, Vol.27, No.2, pp.207-227.

Yao, D. D., 1985. Some Properties of the Throughput Function of Closed Networks of Queues. *Operations Research Letters*, Vol.3, No.6, pp.313-317.

Yao, D. D. and S. C. Kim, 1987a. Some Order Relations in Closed Networks of Queues with Multiserver Stations. *Naval Research Logistics*, Vol.34, No.1, pp.53-66.

Yao, D. D. and S. C. Kim, 1987b. Reducing the Congestion in a Class of Job Shops. *Management Science*, Vol.33, No.9, pp.1165-1172.

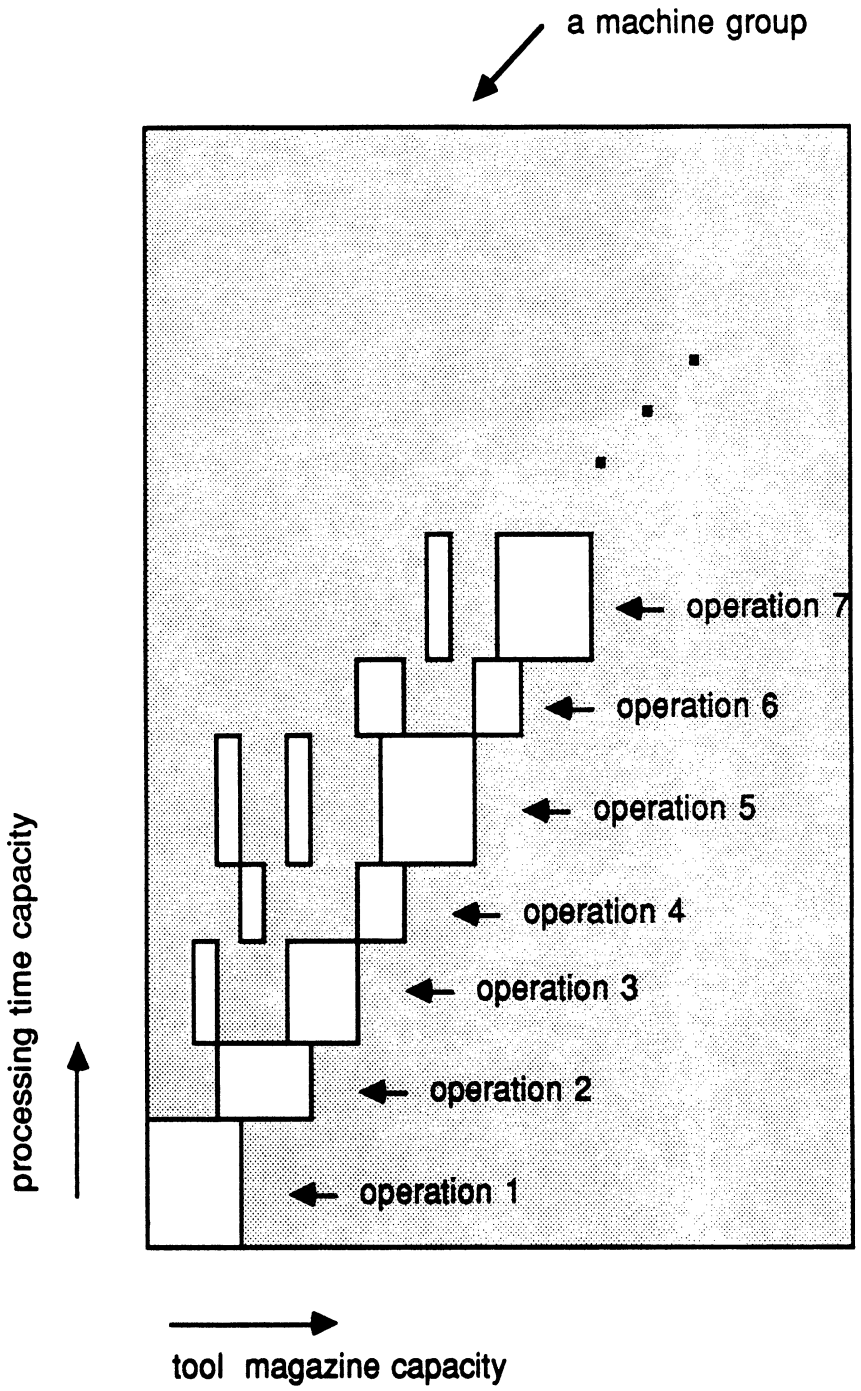

Figure 1. The effect of tool commonality in the bin-packing problem

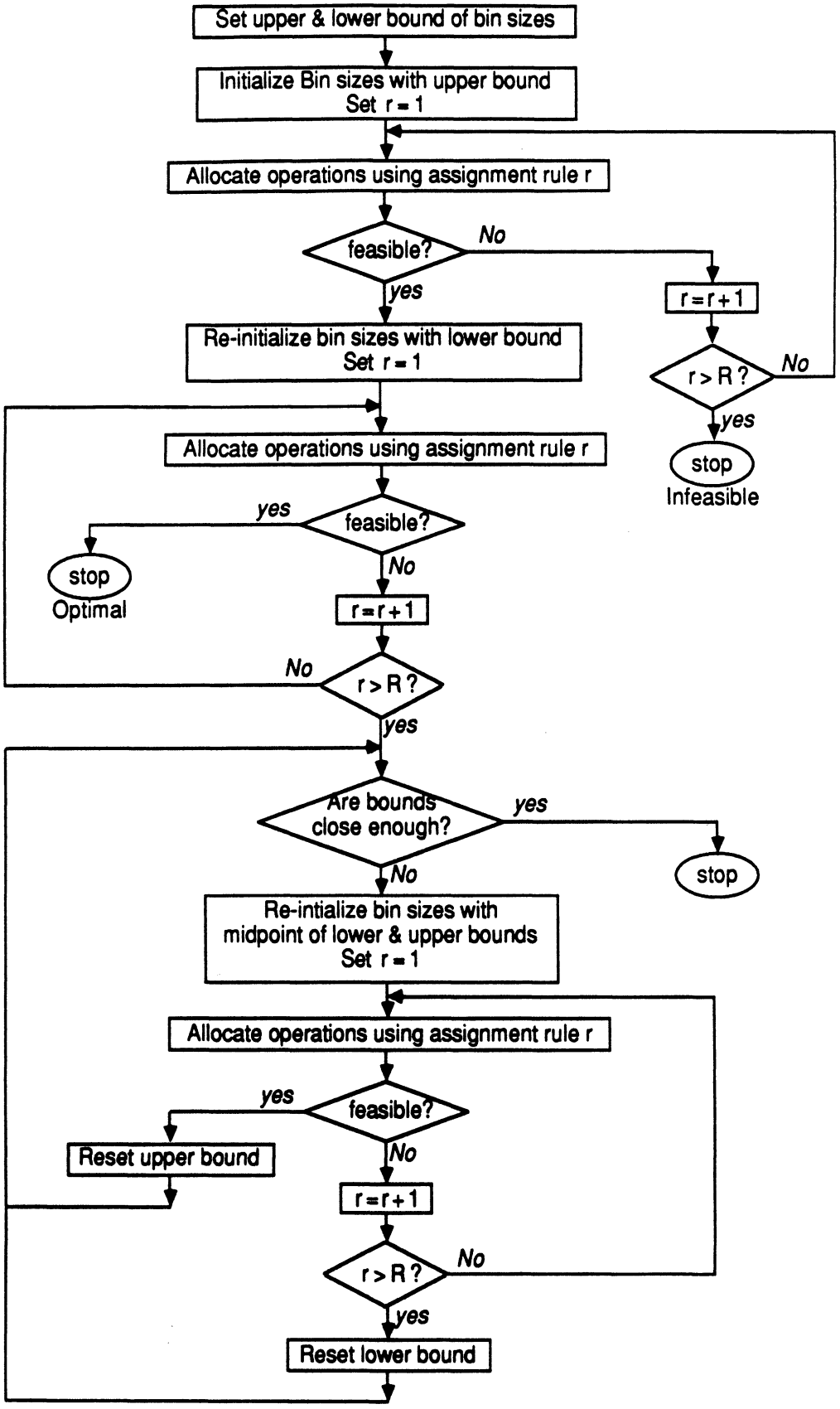

Figure 2. A flow chart for the composite algorithms