Division of Research
Graduate School of Business Administration
The University of Michigan

April 1983
Revised Dec. 1984
Revised Oct. 1985

A BRANCH AND BOUND APPROACH FOR MACHINE LOAD BALANCING IN FLEXIBLE MANUFACTURING SYSTEMS

Working Paper No. 329-C

Mohammed Berrada
Centre d'Études et de Recherches de Toulouse
Toulouse, France

Kathryn E. Stecke
The University of Michigan
Ann Arbor, Michigan

Management Science, forthcoming, 1986

ABSTRACT

A flexible manufacturing system (FMS) is an integrated system of computer numerically controlled machine tools connected with automated material handling. A set of production planning problems for FMSs has been defined (Stecke [1983]), and this paper considers one called the loading problem. This problem involves assigning to the machine tools, operations and associated cutting tools required for part types that have been selected to be produced simultaneously. The part types will be machined during the upcoming production period (say, of one to three weeks duration on average) and according to a prespecified part mix. This assignment is constrained by the capacity of each machine's tool magazine as well as by the production capacities of both the system, and each machine type. There are several loading objectives that are applicable in a flexible manufacturing situation. This paper considers the most commonly applied one, that of balancing the workload on all machines.

This paper first discusses a nonlinear integer mathematical programming formulation of the loading problem. The problem is formulated in all detail. Then an efficient solution procedure is proposed and illustrated with an example. Computational results are provided to demonstrate the efficiency of the suggested special-purpose procedures.

A flexible manufacturing system (FMS) can be thought of as an automated job shop consisting of computer numerically controlled machine tools, each with automatic tool-changing capabilities. A computer also controls the movements of parts between machines tools via some material handling system such as wire-guided carts or conveyors. Descriptions of particular FMSs can be found in Stecke (1977), Cavaillé, Forestier, and Bel (1981), Stecke and Solberg (1981a), Barash (1982), and Stecke and Browne (1985).

The cutting tools required for all operations that might be performed by a particular machine tool are stored in that machine's limited-capacity tool magazine. The technological sophistication of the automatic tool-changing devices virtually eliminates set-up time between consecutive operations for each machine tool. However, a careful system set-up is required before production begins, to best utilize potential production capacity. This planning phase involves several production planning problems (see Stecke (1983)). In brief, these problems are to: (1) select compatible part types for simultaneous machining for the upcoming time period; (2) partition machines into machine groups, each of which can perform the same operations; (3) determine production ratios at which the part types should follow; (4) determine minimum inventory requirements (pallets and fixtures) to maintain the production ratios; (5) allocate operations and cutting tools to the limited capacity tool magazines.

This paper focuses on the fifth problem, called the FMS loading problem. To expound a bit, this problem is to assign to the machines, the operations of the selected part types and the tools necessary to perform these operations, subject to the FMS technological and capacity constraints and according to some loading objective, in a way that will best utilize the machines, or maximize production, when the system is running. Once this problem is solved and the cutting tools are loaded into their assigned tool magazine(s), then the system is ready to begin production.

Such loading problems associated with conventional manufacturing systems require consideration of lot-sizing (because of long set-up times) or assembly line balancing (during the design phase of a production line). The latter sorts of loading problems may require solution only once a year.

However, because of FMS capabilities and flexibility, the <u>FMS loading problem</u> need not be restricted by these considerations. Also, this problem requires frequent solution (which can range from about a week to say three weeks). In particular, a new <u>FMS loading problem</u> has to be solved whenever: production orders change; or a part type finishes its requirements (space in the tool magazine is now free for other part types/cutting tools); or a new part type is to begin production (its cutting tools now have to fit in the tool magazines somehow); or when a machine tool goes down, to name a few situations. Because the technology associated with FMSs is still relatively new, various aspects of the FMS loading problem have only recently been studied. See, for example, Stecke (1977, 1983, 1985b), Stecke and Solberg (1981b, 1985), Stecke and Schmeiser (1982), Kusiak (1983), and Stecke and Morin (1985).

There are different loading objectives that can be followed, each applicable in different situations. This paper focuses on the loading objective of balancing the workload on all machines, while assigning each operation to only one machine. While somewhat restrictive for an FMS, it nevertheless is the most studied and applied objective, with conventional manufacturing methods as well as flexible manufacturing. There is a very large literature on assembly line balancing and balancing workloads in job shops and FMSs. Balancing is appropriate for a flexible assembly system and an automated transfer line. In these systems, parts flow unidirectionally from machine to machine, sometimes skipping machines. Stecke and Morin (1985) have shown that if each operation is assigned to only one machine, balancing workload per machine maximizes expected production. Stecke and Solberg (1985) show that if functionally similar machines are pooled into machine groups** of equal size, then balancing workloads again maximizes expected production. Shanthikumar and Stecke (1986) show that balancing workloads also minimizes work-in-process inventory in FMSs that contain only one machine in a group. This is observed in Stecke (1985a). We note that if an FMS is very reliable, sometimes it can be treated deterministically and operated as a transfer line between periods of change (such as when a new part type is to be introduced for machining or an old part type has completed its production requirements or a breakdown has occurred). Minimal redundancy in machine assignments might be allowed. In some systems, there is no capacity available to allow pooling. A balancing objective is relevant in such situations.

However, Stecke and Solberg (1985) show that for a more flexibly-operated system having real-time control, in addition to advantages gained by pooling machines into groups, performance is theoretically further improved by unbalanced rather than balanced partitions of machines into groups. Moreover, for these better unbalanced partitions, expected production is maximized by a particular unbalanced assignment of workload per machine. We shall discuss pooling further in §5.

Other loading objectives have also proved to perform better than balancing with respect to achieving maximum production for certain types of FMSs (see Stecke and Solberg (1981b)). For example, if travel time from machine to machine is long relative to processing time, or if the transporter mechanism is a bottleneck, then minimizing the number of movements can be a better objective than balancing. In fact, the objective of minimizing movements has

**Machines in a machine group are said to be pooled, are identically tooled, and are assigned the same operations.

proven better than balancing for a particular FMS, even when travel time was not long and carts were not bottlenecks (see Stecke and Solberg (1981b)).

In Stecke (1983), both the grouping problem (how to best partition m machines into g groups) and the loading problem are defined and mathematically formulated as nonlinear integer programming problems. The nonlinear terms result, in part, from considering the possiblility of assigning several operations, having some common tools, to the same machine. Several loading objectives, each applicable to different types of FMSs, are also defined and formulated. The approach taken to solve these problems was to linearize the nonlinear terms. Several linearization methods were applied (those of Balas (1964), Glover (1975), and Glover and Woolsey (1973, 1974)). The linearizations resulted in much larger constraint formulations. Then the linearized formulations were applied to data from an existing FMS and run to optimality, taking minutes on a CDC 6600 using a standard MIP code. The nonlinear integer formulations can handle any reasonably-sized FMS loading problem.

However, in addition to the computer time, the manual linearization process is too time-consuming and unwieldy to warrant frequent application. The procedure could not be easily implemented by management. In addition, the linearized integer problems get large quickly. Either a nonlinear integer code or an integer code (if linearization is performed) is needed. What is required is an efficient and easy to use optimum-producing code. The advent of these new technologies provides opportunities for operations researchers to develop efficient tools that an FMS manager can use to solve his/her everyday production planning and operating problems. It is this goal--to develop a usable, efficient, specialized and self-contained procedure to solve a particular version of the FMS loading problem--that spurred the research presented here.

The approach taken in this paper is to bypass the manual linearization step, and to solve the nonlinear problems directly and automatically. A sequence of subproblems is defined. In brief, each subproblem is solved by an efficient branch and bound procedure that first solves a simple, relaxed assignment problem, then checks for feasibility, and finally modifies the assignment to correct violated constraints. The modification of the assignment is obtained by solving very small integer problems via branch and backtrack for each machine where a constraint is violated. The optimum solution to the relaxed problem provides a lower bound to the original problem that is used both to fathom nodes in the binary enumeration tree and to select a node for further branching. An intelligent selection criterion is used to choose the most suitable branching variable. Only one of the loading objectives of Stecke (1983) is used, that of balancing the assigned workload on each machine tool. Each operation is assigned to only one machine tool. This is by far the most widely applied loading objective.

The paper is organized as follows. In §1, the nonlinear integer formulation of the particular FMS loading problem, which is to balance the assigned workload on each machine tool, is reviewed. The proposed solution method is described in detail in §2. An example that illustrates the procedure is provided in §3, while computational results are given in §4. §5 summarizes the results of the paper and discusses the extension of the procedures to another loading objective.

1. MATHEMATICAL FORMULATION

After the variables and notation are defined, the constraints and objective function of the considered FMS loading problem will be reviewed.

1.1 Variables and Parameters

The subscripts, some preliminary input parameters, and the decision variables of the FMS loading problem are provided in Table I. Additional notation will be provided as required, when the solution procedure is described.

TABLE I

Notation

Parameters and Subscripts:

I	$= \{i \mid i=1,\ldots,b\}$ = set of operations
J	$= \{j \mid j=1,\ldots,m\}$ = set of machine tools
p_{ij}	= processing time of operation i on machine tool j
d_i	= number of slots required in a tool magazine for holding the cutting tools of operation i
t_j	= capacity of the tool magazine of machine tool j
w_{ik}	= number of slots saved as a result of having common tools when operations i and k are assigned to the same machine tool = count of the number of spaces (slots) occupied by the tools contained in the intersection of the sets of tools required by operations i and k
w_B	= number of slots saved when the operations in subset B are assigned to the same machine tool
P	= index set of compatible part types that are to be produced simultaneously on the system of machine tools in the upcoming production period
a_{ki}	= production ratio (relative to the remaining part types in $P-\{k\}$) at which operation i of part type k will be produced
r_j	= relative workload, or utilization, of machine tool j

Decision Variables:

$$x_{ij} = \begin{cases} 1, & \text{if operation } i \text{ is assigned to machine tool } j; \\ 0, & \text{otherwise.} \end{cases}$$

1.2 Constraint Formulations

The constraints of the loading problem are as follows.

First, each operation must be assigned to at least one machine tool of the machine type required by the operation. Throughout, each operation is assigned to only one machine tool. Then

$$\sum_{j=1}^{m} x_{ij} = 1, \quad i=1,\dots,b. \tag{1}$$

It follows that $x_{ij} = 0$ if operation i cannot be performed by the machine type specified by machine tool j. In this case, $p_{ij} = \infty$.

Second, the tool magazine capacity constraints, which relate the number of tool slots required by the operations assigned to each machine tool to the total number of slots in each machine's tool magazine, in their simplest form, are

$$\sum_{i=1}^{b} d_i x_{ij} \leq t_j, \quad j=1,\dots,m.$$

However, this capacity constraint is only sufficient to insure that the tools required by the operations that are assigned to machine tool j can all be contained in its tool magazine. If the <u>tools</u> that are common to several operations and the <u>slots</u> that are saved by a suitable positioning of the tools are taken into consideration (see Figure 1 to demonstrate the latter), the capacity constraints are then more accurately expressed by:

$$\sum_{i=1}^{b} d_i x_{ij} + TC_j(x) \leq t_j, \quad j=1,\dots,m, \tag{2}$$

where

$$TC_j(x) = \sum_{p=2}^{b} (-1)^{p+1} \sum_{\substack{\forall B \subseteq I \\ \ni |B|=p}} w_B \prod_{i \in B} x_{ij},$$

or

$$= \sum_{\substack{\forall B \subseteq I \\ \ni |B| \geq 2}} (-1)^{|B|+1} w_B \min_{i \in B}\{x_{ij}\}, \quad j=1,\ldots,m. \tag{3}$$

Note that $TC_j(x) \leq 0$, for all j. The nonlinear terms of equation (3) arise from the tool slot overlap. For example, if you've taken out common tools from all pairs of 3 operations, some tools may have been taken out too much and so have to be put back in.

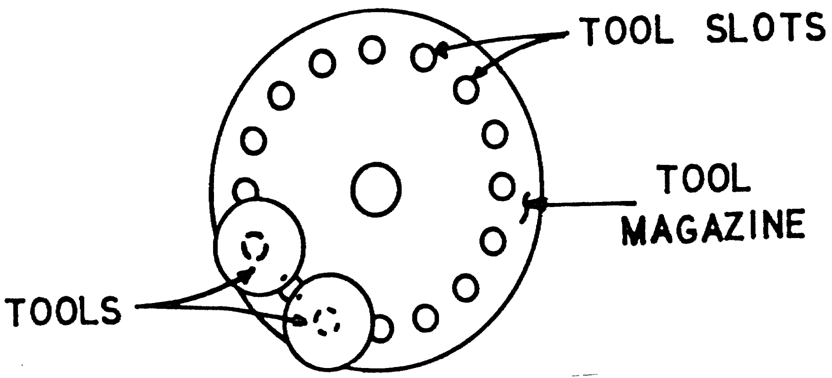

FIGURE 1. Tool Magazine.

A third constraint type is required to specify the relative workload assigned to machine tool j:

$$r_j = \sum_{i=1}^{b} a_i p_{ij} x_{ij}, \quad j=1,\ldots,m.$$

Throughout, the subscript k is dropped from the production ratios a_{ki}. These are relative ratios at which the selected part types are to be maintained during production. Methods to help determine these, for situations of both dependent and independent demand, are provided in Stecke (1985a). When all r_j are equal, the system is perfectly balanced. However, perfect balance is usually impossible to attain because of the discreteness of the processing times.

Finally, there are the integrality constraints:

$$x_{ij} = 0 \text{ or } 1, \text{ for all } i, j. \tag{4}$$

1.3 Problem Formulation

The production rate of a particular type of FMS, a deterministic transfer line, is limited by the throughput of the bottleneck machine, which is specified by the largest r_j. This is the intuition behind the common belief that balancing workloads corresponds to maximizing the production rate.

The approach taken here to balance the FMS is to minimize the workload of the bottleneck machine. Let δ be an upper bound on the machine workloads. Then the problem is to find $\{x_{ij} | i=1,\dots,b,\ j=1,\dots,m\}$ to:

Problem (P)

minimize δ

subject to (1), (2), (3), (4), and

$$\sum_{i=1}^{b} a_i p_{ij} x_{ij} \leq \delta, \quad j=1,\dots,m. \tag{5}$$

2. SOLUTION METHOD

The solution procedure can be summarized as follows. In §2.1, a sequence of subproblems is defined by fixing the maximum workload, δ, (in equation (5)) to be δ_ℓ for each subproblem ℓ. The sequence of solutions converges to an an ε-optimal assignment. If δ^* is the optimal solution of problem (P), then an ε-optimal solution, δ^ε, is such that $|\delta^* - \delta^\varepsilon| \leq \varepsilon$. To decrease the size of the subsequent problems, we set δ_ℓ to be the midpoint of an interval that contains the optimal δ of Problem (P). Then a check is made to see if δ is greater (or not) then δ_ℓ. The solution of each subproblem is used to decrease the interval size: depending on this solution, either the lower bound increases or both the lower bound increases and the upper bound decreases. An efficient branch and bound procedure is developed in §2.2 to solve each subproblem optimally. First, a relaxed integer problem is solved (in §2.2.1) and feasibility is checked. Then the assignment is modified in the least-cost way for violated constraints by using a branch and backtrack procedure on very small single-machine integer problems. The optimal solution of this relaxed problem provides a lower bound on the solution of the original problem to allow node selection and fathoming. Finally, the criterion for selecting the branching variable is described in §2.2.2.

2.1 Subproblem Generation

Problems (P_{δ_ℓ}) are a sequence of subproblems, each finding a <u>feasible solution</u> of (P), that converges to an ε-optimal solution to Problem (P). Rather than settling for an arbitrary feasible solution, the procedure uses an auxiliary objective function for each subproblem, to find a solution that allows a maximum reduction of the interval containing δ_ℓ in the subsequent subproblem. Also, each subproblem, (P_{δ_ℓ}), is defined by a <u>fixed</u> workload upper bound, δ_ℓ.

Problem (P_{δ_ℓ}) is reformulated from Problem (P) as follows. For a fixed δ,

maximizing $(\delta - r_j)$ for all j can be achieved by

maximizing $(m\delta - \sum_{j=1}^{m} r_j) \Longleftrightarrow$ minimizing $\sum_{j=1}^{m} \sum_{i=1}^{b} a_i p_{ij} x_{ij}$

Thus, the problem is to find x_{ij} to:

<u>Problem (P_{δ_ℓ})</u>

$$\text{minimize } Z = \sum_{j=1}^{m} \sum_{i=1}^{b} a_i p_{ij} x_{ij}$$

subject to (1), (2), (3), (4), and

$$\sum_{i=1}^{b} a_i p_{ij} x_{ij} \leq \delta_\ell, \quad j=1, \ldots, m. \tag{5'}$$

The subproblems are generated as follows. Let LV and UV be a lower bound and upper bound, respectively, of δ_ℓ. (Initially, LV = 0 and UV = ∞.)

Fix $\delta_\ell = (UV + LV)/2$. (Initially, $\delta_0 = \infty$.) An optimal solution to Problem (P_{δ_ℓ}) is found. If a solution exists, then UV is updated to equal the largest machine workload associated with this solution. The auxiliary objective helps to reduce this largest machine workload. Also, LV is set equal to $\sum_{j=1}^{m} r_j/m$, if $\sum_{j=1}^{m} r_j/m > LV$. The reasoning behind this updating of LV is provided in Appendix C. If a feasible solution does not exist, then LV is updated to equal δ_ℓ.

The procedure is terminated when the difference between UV and LV is smaller than ε. An ε-optimal solution is sufficient because:

1. processing times are allocated among machines in discrete quantities; and
2. processing times may fluctuate slightly because of behavioral anomalies and because of adaptive processing capabilities of machine tools.

ε is set equal to $\min_{i,j} \{a_i p_{ij}/m\}$. The choice of ε is motivated as follows.

Let x and x' be two feasible solutions of Problem (P). The distance between x and x', as measured by the objective function of Problem (P_{δ_ℓ}), can be evaluated as:

$$d(x,x') = \left|\max_j r_j(x) - \max_j r_j(x')\right|.$$

Define ε^* as:

$$\varepsilon^* = \min \{d(x,x') \mid x,x' \text{ are such that } d(x,x') > 0\},$$

when (P) is feasible.

Since the set of feasible solutions is finite, ε^* is greater than zero (unless all feasible solutions are optimal!). Clearly, for all $\varepsilon \leq \varepsilon^*$, an ε-optimal solution is optimal if one exists. Increasing the number of machines tends to increase the number of feasible solutions, and hence decrease ε^*. To account for this,

$$\varepsilon = \min_{i,j} \{a_i p_{ij}/m\}$$

provides a reasonable heuristic approximation of ε^*. An accurate value of ε^* is very difficult to calculate, and unnecessary in any case.

The generation of subproblems required to solve the FMS loading problem is precisely described by the following algorithm.

The Solution Algorithm

```
STEP 1.  INITIALIZATION:
             Set $LV = 0$, $UV = \infty$, $\ell = 0$, $\delta_0 = \infty$, and
                 $\varepsilon = \min_{i,j} \{\frac{a_i p_{ij}}{m}\}$.
STEP 2.  SOLUTION:
             Find an optimal solution, x, of Problem ($P_{\delta_\ell}$)
                 (to be described in §2.2).
STEP 3.  UPDATE BOUNDS:
             If a solution exists, set $UV = \max_j \{r_j\}$ and
                 $LV = \text{Max}\{LV, (\sum_{j=1}^{m} r_j)/m\}$ ;
                 otherwise, set $LV = \delta_\ell$.
STEP 4.  CHECK STOPPING THRESHOLD:
             If $UV - LV > \varepsilon$, set $\delta_{\ell+1} = (UV + LV)/2$ and go to STEP 2;
                 otherwise, go to STEP 5.
STEP 5.  STOP:
             If $\delta_\ell = \infty$, then no solution exists; otherwise, the
                 solution found last is $\varepsilon$-optimal.
```

The initialization of δ_0 to be infinity is equivalent to dropping the utilization constraint (5) of Problem (P_{δ_ℓ}), which allows an initial test for problem feasibility.

2.2 Solving Subproblem (P_{δ_ℓ})

Finding a feasible solution to an integer problem is as computationally complex as finding an optimal solution (Karp [1975]). However, stopping an optimization problem early with a feasible solution is actually much quicker in practice. Hence, it might seem that finding a feasible solution of (P_{δ_ℓ}), without updating LV, could suffice in STEP 2. However, computational experience has indicated that, in this case, the number of generated subproblems (P_{δ_ℓ}) is very large and most of them are infeasible, i.e., time-consuming to deal with. It would be much more efficient to find optimal solutions. Decreasing the size of interval [LV, UV] is then important. Consequently, many subproblems (P_{δ_ℓ}) (mostly the infeasible ones) are no longer considered (because of the improved LV). This is proven in Appendix C.

Problems somewhat similar to (P_{δ_ℓ}) have been considered by Sandi (1975), Ross and Soland (1975), Caie, Linden, and Maxwell (1980), and Graves and Lamar (1981). The main difference here is the inclusion of the highly nonlinear terms in $TC_j(x)$ of constraint (3).

(P_{δ_ℓ}) is solved via branch and bound (see Garfinkel and Nemhauser (1972) and Salkin (1975)). The binary enumeration tree for the branch and bound procedure consists of nodes which are partial assignments. At each node, a lower bound of the objective function is obtained by solving very small zero-one problems. At every step, the node with the least lower bound is selected from the list of dangling nodes for further branching.

2.2.1 Calculation of the Lower Bound at Node K during Solution of a Relaxation of Problem (P_{δ_ℓ})

Let F^K be the set of operations that have been assigned at node K:

$$F^K = \{i \varepsilon I \mid \text{there exists } j_i \varepsilon J \text{ such that } x^K_{ij_i} = 1\},$$

where x^K is the partial solution at node K. $Z(x^K)$ is the value of the partial solution.

We consider the nonlinear tool magazine capacity constraints, (2) and (3), and the workload constraints, (5), to be the "complicating" constraints of Problem (P_{δ_ℓ}), and the remaining constraints, (1) and (4), to be "easy" constraints. Our <u>relaxation</u> of Problem (P_{δ_ℓ}) consists of dropping the complicatting constraints. Thus at node K the relaxed problem is the following simple integer program:

<u>Problem $(P_{\delta_\ell} R)^K$</u>

$$\text{minimize } Z = \sum_{j=1}^{m} \sum_{i\varepsilon(I-F^K)} a_i p_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j=1}^{m} x_{ij} = 1, \qquad i\varepsilon(I-F^K) \tag{1}$$

$$x_{ij} = 0 \text{ or } 1, \qquad j\varepsilon J \text{ and } i\varepsilon(I-F^K). \tag{4}$$

The solution to problem $(P_{\delta_\ell} R)^K$ is that each unassigned operation is assigned to the machine tool which results in the least load.

A solution (x^{*K}) of $(P_{\delta_\ell} R)^K$ is easily found for each unassigned operation $i\varepsilon(I-F^K)$ from the index $j_i \varepsilon J$ such that:

$$a_i p_{ij_i} = \min_{j\varepsilon J} \{a_i p_{ij} \mid (i,j) \notin G^K\},$$

where

$$G^K = \{(i,j) \mid x_{ij}^K = 0\}.$$

Then the solution (x^{*K}) is defined by $x_{ij_i}^{*K} = 1$, and $x_{ij}^{*K} = 0$ for $j \neq j_i$ and $i \notin F^K$. In the case of a tie, the algorithm selects the first minimum $\{a_i p_{ij}\}$.

A first estimate of the lower bound of the objective function is thus:

$$LB^K = Z(x^K) + Z(x^{*K}) = Z(x^K \cup x^{*K})$$
$$= \sum_{i \in F^K} a_i p_{ij_i} + \sum_{i \notin F^k} a_i p_{ij_i} .$$

If (x^K) completed by (x^{*K}) satisfies the constraints (2), (3), and (5), then it is an optimal solution of (P_{δ_ℓ}). Otherwise, for each machine whose corresponding constraints are violated, we must modify the assignment with the least possible increase of the objective function in order to find a feasible solution to Problem (P_{δ_ℓ}). In addition, the lower bound is improved.

Let

I_j^K be the set of operations assigned to machine j in the solution (x^{*K}); $I_j^K \cap F^K = \emptyset$ (see Figure 2); and

J'^K be the set of machines for which either the capacity constraint or the workload constraint or both are violated.

We define for all $i \in I_j^k$:

c_i = the minimal incremental cost that would result from a reassignment of operation i

$$= \min_k \{a_i p_{ik} - a_i p_{ij_i} \mid k \neq j_i \text{ and } (i,k) \in G^K\};$$

b_j = the overflow of the violated capacity constraint, i.e., the additional number of slots that are required in the magazine of machine tool j

$$= \sum_{i \in F^K} d_i x_{ij}^K + \sum_{i \notin F^K} d_i x_{ij}^{*K} + TC_j(x^K \cup x^{*K}) - t_j;$$

α_j = the workload overflow of machine j

$$= \sum_{i \in F^K} a_i p_{ij} x_{ij}^K + \sum_{i \notin F^K} a_i p_{ij} x_{ij}^{*K} - \delta_\ell .$$

For <u>each</u> $j \in J'^K$ we must reassign one or more operations to other machines. Let y_{ij} be the reassignment variable to be used in the <u>reassignment problem</u>, $(PR_j)^K$, for each machine $j \in J'^K$ at node K. The problem is:

Problem $(PR_j)^K$

$$\text{minimize } Z_j = \sum_{i \epsilon I_j^K} c_i y_{ij}$$

subject to

$$\sum_{i \epsilon I_j^K} d_i y_{ij} + TC_j'(y) \geq b_j$$

$$TC_j'(y) = \sum_{\substack{\forall B \subseteq I_j^K \\ \ni |B| \geq 2}} (-1)^{|B|+1} \; w_B \max_{i \epsilon B} \{y_{ij}\}^{(*)}$$

$$\sum_{i \epsilon I_j^K} a_i p_{ij} y_{ij} \geq \alpha_j$$

$$y_{ij} = 0 \text{ or } 1, \qquad i \epsilon I_j^K,$$

where the decision variables, y_{ij}, are:

$$y_{ij} = \begin{cases} 1, & \text{if operation } i \epsilon I_j^K \text{ must be reassigned to another machine;} \\ 0, & \text{otherwise.} \end{cases}$$

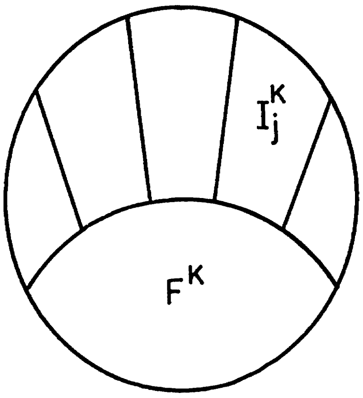

FIGURE 2. Subsets of Assigned Operations at Node K.

(*) The proof that the maximum operator is correct here is provided through the derivation of the $TC_j'(y)$ term in Appendix A.

The lower bound is improved by adding to $Z(x^K \cup x^{*K})$, the sum of the objective function values $Z_j(y^{*K})$ obtained by solving problems $(PR_j)^K$ for all $j \in J'^K$:

$$LB^K = Z(x^K) + Z(x^{*K}) + \sum_{j\in J'^K} Z_j(y^{*K}).$$

The reassignment problems, $(PR_j)^K$, for each machine $j\in J'^K$ are solved by a branch and backtrack procedure (Balas [1965]). This procedure is very quick because of the small size of the single-machine problems, $(PR_j)^K$.

2.2.2 Selection of the Branching Variable

The selection of which variable among the $x^{*K}_{ij_i}$ to branch on is one such that $y^{*K}_{ij_i} = 0$, since these variables correspond to operations which will not be reassigned. Intuitively, it is better to choose i^* and j^*_i so that the objective function of (P_{δ_ℓ}) would increase more if i^* were assigned to a machine other than j^*_i. Hence, the branching variable, $x_{i^*j^*_i}$, can be found by using the following priority indices:

$$i^* \in \operatorname{Arg}\max_i \{c_i \mid y^{*K}_{ij_i} = 0\}.$$

However, some preliminary computational experience has indicated that it is important to consider also both the remaining number of slots and the amount of work capacity available at each machine tool at node K to get a better choice of the branching variable. Tests with various sets of examples have shown that convergence towards the ε-optimal solution is, in most cases, faster by using the following rule: select $x_{i^*j^*_i}$ such that:

$$i^* \in \operatorname{Arg}\max_i \{c_i \,/\, [d_i/(t_{j_i} - \sum_{k\in F^K_{j_i}} d_k) + a_i p_{ij_i}/(\delta_\ell - \sum_{k\in F^K_{j_i}} a_k p_{kj_i})]\},$$

such that $y^{*K}_{ij_i} = 0$ and where $F^K_{j_i}$ is the set of operations already assigned to machine j_i at node K.

This selection rule introduces an intelligent weighting of the reassignment penalties, c_i. The c_i's are divided by a sum of:

1. a tool-storage-saving factor, which is the number of tool slots required by operation i per number of slots remaining in machine tool j_i's tool magazine;
2. a utilization-saving factor, which is the workload required for operation i on machine tool j_i per unit of work capacity remaining.

This branching rule gives a lower reassignment priority to those machine tools that are already significantly loaded or tooled, among those machine tools that are candidates for reassignment of the operations.

Once the branching variable is chosen, two nodes (two new candidate problems) are generated and the parameters associated with problem (P_{δ_ℓ}) are updated to account for the new partial solutions at both nodes K + 1 and K + 2.

Once problem (P_{δ_ℓ}) is solved, the bounds of the maximum workload δ_ℓ are updated. Then a check is made to see if the interval size is less than ε. If not, the next subproblem $(P_{\delta_{\ell+1}})$ is solved. Otherwise, the ε-optimal solution has been found (if a solution exists).

3. EXAMPLE

To illustrate the branch and bound procedure, consider the following example consisting of three machine tools and eight operations. The data, including the weighted operation times, are provided in Table II below:

TABLE II

Three-Machine Tool, Eight-Operation Data

j \ i	1	2	3	4	5	6	7	8	t_j
1	3.0	4.0	2.5	6.0	4.0	2.0	2.4	5.0	20
2	3.4	3.5	3.0	5.5	4.1	3.0	2.0	4.7	20
3	2.8	4.2	2.7	6.0	5.0	2.5	2.6	5.2	20
d_i	6	7	6	10	8	5	4	9	*

An entry (i,j) in the table represents the term $a_i p_{ij}$. Moreover, assigning operations 1 and 2 together can save two slots in a tool magazine, assigning operations 1 and 4 to the same machine can save three slots, operations 3 and 5 can save two slots, operations 5 and 7 can save one slot, and operations 7 and 8 can save two slots.

The Solution Algorithm

STEP 1. INITIALIZATION:

Set $LV = 0$, $UV = \infty$, $\delta_0 = \infty$, and

$\varepsilon = 2/3 = 0.67$.

STEP 2. SOLUTION OF (P_{δ_0}):

Node K = 1:

$$F^1 = \phi;\ G^1 = \phi;\ x^1 = \phi.$$

By inspection of Table II, the solution of $(P_{\delta_0}R)^1$ is

$$x^{*1}_{13} = x^{*1}_{22} = x^{*1}_{31} = x^{*1}_{42} = x^{*1}_{51} = x^{*1}_{61} = x^{*1}_{72} = x^{*1}_{82} = 1,$$

and all other $x^{*1}_{ij} = 0$.

The lower bound is

$$LB^1 = Z(x^{*1}) = 27.$$

However, this solution is not feasible for (P_{δ_0}) because the capacity constraint on machine 2 is violated $(d_2 + d_4 + d_7 + d_8 - w_{78} > t_2)$. To solve the problem, we solve a reassignment problem for machine 2:

$(PR_2)^1$

$$\text{minimize } Z_2 = .5y_{22} + .5y_{42} + .4y_{72} + .3y_{82}$$

subject to

$$7y_{22} + 10y_{42} + 4y_{72} + 9y_{82} - 2\max\{y_{72}, y_{82}\} \geq 8$$

$$y_{ij} = 0 \text{ or } 1, \text{ for all } i,j.$$

Problem $(PR_2)^1$ is solved by a branch and backtrack procedure (see Figure 3). The optimal solution is $y_{42}^{*1} = 1$ and all other y_{ij} are 0. Then $Z_2^* = .5$, and the tighter, improved lower bound is now 27.5.

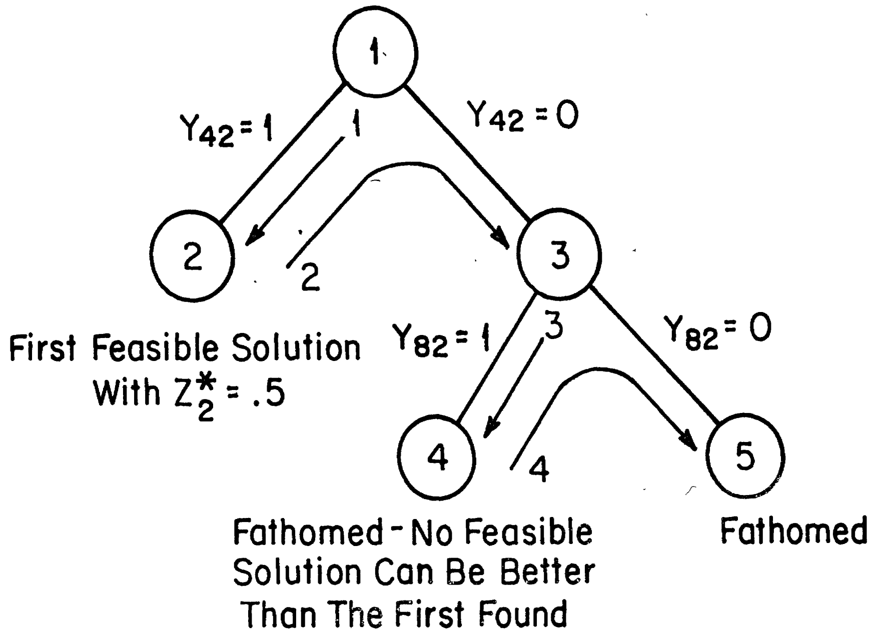
1
2
3
4
5
Y42 = 1
Y42 = 0
Y82 = 1
Y82 = 0
1
2
3
4
First Feasible Solution With Z*2 = .5
Fathomed - No Feasible Solution Can Be Better Than The First Found
Fathomed

FIGURE 3. Branch and Backtrack Tree for Problem $(PR_2)^1$.

The branching variable is x_{72} because:

$7 = \text{Arg} \max_i \{c_i/[d_i/t_{2i}]$ such that $y_{i2}^{*1} = 0\}$,

and the process is repeated (see Figure 4).

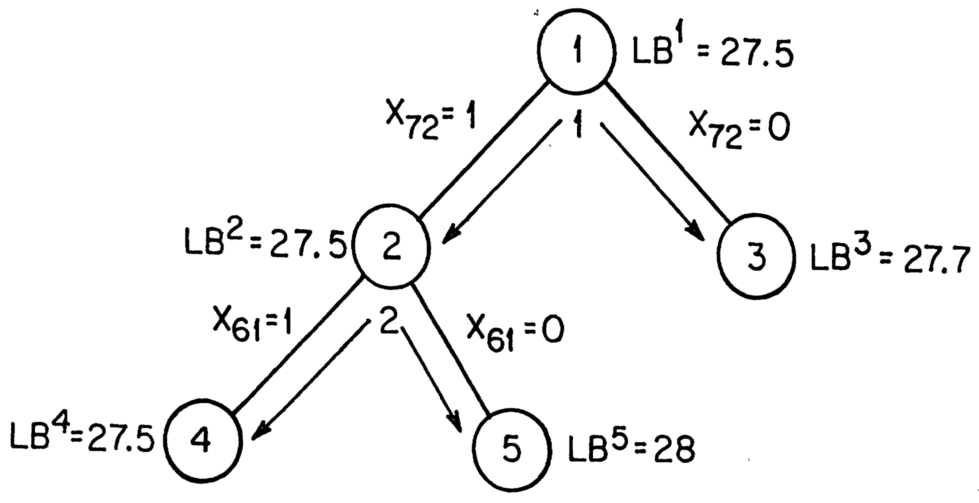

FIGURE 4. Branch and Bound Tree for Problem (P_{δ_0}).

Node K = 2:

$F^2 = \{7\}$; $G^2 = \{(4,2)\}$, because $y^{*1}_{42} = 1$; $x^2 = \{x_{72} = 1\}$.

The solution of $(P_{\delta_0}R)^2$ is

$$x^{*2}_{13} = x^{*2}_{22} = x^{*2}_{31} = x^{*2}_{41} = x^{*2}_{51} = x^{*2}_{61} = x^{*2}_{82} = 1,$$

and all other $x^{*2}_{ij} = 0$.

Note that setting x^{*2}_{43} to 1 rather that x^{*2}_{41} provides the same minimum $a_i p_{ij} = 6$. The algorithm arbitrarily selects the first minimum ($x^{*2}_{41}=1$).

The lower bound is

$$LB^2 = Z(x^2) + Z(x^{*2}) = 27.5.$$

This solution is not feasible for (P_{δ_0}) because the capacity constraint on machine 1 is violated ($d_3 + d_4 + d_5 + d_6 - w_{36} > t_1$). We solve a reassignment problem for machine 1:

$(PR_1)^2$

$$\text{minimize } Z_1 = .2y_{31} + 0y_{41} + .1y_{51} + .5y_{61}$$

subject to

$$6y_{31} + 10y_{41} + 8y_{51} + 5y_{61} - 2 \max \{y_{31}, y_{51}\} \geq 7$$

$$y_{ij} = 0 \text{ or } 1, \text{ for all } i,j.$$

The optimal solution, found by branch and backtrack, is $y^*_{41} = 1$ and and all other y_{i1} are zero. Then $Z^*_1 = 0$. The branching variable is x_{61}—see Figure 4.

Node K = 3:

$F^3 = \phi$; $G^3 = \{(7,2)\}$; $x^3 = \{x_{72} = 0\}$.

The solution of $(P_{\delta_0}R)^3$ is

$$x^{*3}_{13} = x^{*3}_{22} = x^{*3}_{31} = x^{*3}_{42} = x^{*3}_{51} = x^{*3}_{61} = x^{*3}_{71} = x^{*3}_{82} = 1,$$

and all other $x_{ij}^{*3} = 0$.

The lower bound is

$$LB^3 = Z(x^{*3}) = 27.4.$$

This solution is not feasible for (P_{δ_0}) because the capacity constraint on machine 2 is violated $(d_2 + d_4 + d_8 > t_2)$. We solve a reassignment problem for machine 2:

$(PR_2)^3$

$$\text{minimize } .5y_{22} + .5y_{42} + .3y_{82}$$

subject to

$$7y_{22} + 10y_{42} + 9y_{82} \geq 6$$

$$y_{i2} = 0 \text{ or } 1, \text{ for all i, j.}$$

The optimal solution found by branch and backtrack is $y_{82}^{*3} = 1$ and $y_{22}^{*3} = y_{42}^{*3} = 0$. Then $Z_2^* = .3$ and the improved lower bound is now

$$LB^3 = 27.7.$$

The branching variable (from node 3) would be x_{22}.

However, the selected dangling node to branch on is node 2 (because $LB^2 < LB^3$--see Figure 4).

<u>Node K = 4</u>:

$$F^4 = \{7,6\};\ G^4 = \{(4,2),\ (4,1)\};\ x^4 = \{x_{72} = 1,\ x_{61} = 1\}.$$

The solution of $(P_{\delta_0}R)^4$ is

$$x_{13}^{*4} = x_{22}^{*4} = x_{31}^{*4} = x_{43}^{*4} = x_{51}^{*4} = x_{82}^{*4} = 1,$$

and all other $x_{ij}^{*4} = 0$.

The lower bound is

$$LB^4 = Z(x^{*4}) + Z(x^4) = 27.5.$$

This solution is feasible for (P_{δ_0}).

Node K = 5:

$$F^5 = \{7\};\ G^5 = \{(4,2),(6,1)\};\ x^5 = \{x_{72} = 1,\ x_{61} = 0\}.$$

The solution of $(P_{\delta_0}R)^5$ is

$$x^{*5}_{13} = x^{*5}_{22} = x^{*5}_{31} = x^{*5}_{41} = x^{*5}_{51} = x^{*5}_{63} = x^{*5}_{82} = 1,$$

and all other $x^{*5}_{ij} = 0$.

The lower bound is

$$LB^5 = Z(x^{*5}) + Z(x^5) = 28.$$

This solution is not feasible for (P_{δ_0}) because the capacity constraint on machine 1 is violated $(d_3 + d_4 + d_5 - w_{35} > t_1)$. We solve a reassignment problem for machine 1:

$(PR_1)^5$:

minimize $.2y_{31} + 0y_{41} + .1y_{51}$

subject to

$$6y_{31} + 10y_{41} + 8y_{51} - 2\max\{y_{31},y_{51}\} \geq 2$$

$$y_{ij} = 0 \text{ or } 1, \text{ for all } i,j.$$

The optimal solution is $y^{*5}_{41} = 1$ and $y^{*5}_{31} = y^{*5}_{51} = 0$. Then $Z^*_1 = 0$. The branching variable from node 5 would be x_{31}.

However, the selected dangling node to branch on is node 4 (because $LB^4 < LB^5$—see Figure 4). Since the solution $(x^{*4} \cup x^4)$ is feasible, we stop.

STEP 3. UPDATE BOUNDS:

$$UV = \max_j \{r_j\} = 10.2;\ LV = (\sum_{j=1}^{m} r_j)/m = 9.16.$$

(Notice the drastic improvement in the size of the interval [LV, UV].)

STEP 4. CHECK STOPPING THRESHOLD:

$UV - LV = 1.04 > \varepsilon = .67$; then $\delta_1 = (UV + LV)/2 = 9.68$ and go to STEP 2.

STEP 2. Apply the same procedures used for (P_{δ_0}) to obtain, again at the fifth node, the following feasible solution of (P_{δ_1}):

$$x_{13}^{*5} = x_{22}^{*5} = x_{31}^{*5} = x_{43}^{*5} = x_{52}^{*5} = x_{61}^{*5} = x_{72}^{*5} = x_{81}^{*5} = 1,$$

and all other $x_{ij} = 0$.

STEP 3. UPDATE BOUNDS:

$UV = 9.6$; $LV = 9.3$.

STEP 4. CHECK STOPPING THRESHOLD:

$UV - LV = .3 < \varepsilon = .67$.

STEP 5. STOP:

$\delta_1 < \infty$. The ε-optimal solution is given in Table III.

TABLE III

The ε-Optimal Solution

Machine Tools	Operations	Workloads
1	3, 6, 8	9.50
2	2, 5, 7	9.60
3	1, 4	8.80

4. COMPUTATIONAL RESULTS

The algorithm has been implemented in FORTRAN IV. The ten problems of Table IV were run on a PERKIN ELMER computer. The 45 problems of Table V were run on an AMDAHL 5860. The problems have been solved to optimality in very short CPU times.

Our experimental plan is now described. The size of the problems of Table IV vary from 3 to 9 machine tools, which is the range of most FMSs. The number of operations to be assigned range from 7 to 48. For the problems of Table V, the number of machines range from 4 to 15 and the number of operations range from 10 to 40. In most of the problems, the operation times are different on different machine tools, similar to the processing times described in Table II of §3. However, two of the problems of Table IV contain machine tools for which the operation times on those machines are identical. In particular, Problem Number 10 of Table IV consists of 3 machine types: 4 machine tools of 1 type, 3 of another type, and 2 of the third type. Also, Problem Number 26 consists of 3 machine types: 4 of one type, 2 of another, and 1 of a third type. These problems, although not all real examples of FMSs (some are), provide a representative range of the sizes of the problems that would have to be run frequently to retool and reconfigure an FMS.

TABLE IV

Computational Results on a PERKIN ELMER

Problem Number	Number of Machines	Number of Operations	Number of Nodes	Number of Subproblems	CPU Time (seconds)
1	3	8	10	2	0.484
2*	3	8	79	3	3.839
3	3	12	19	3	0.548
4	4	7	65	3	1.109
5	4	12	23	5	3.095
6	4	24	47	5	3.592
7	5	20	182	6	10.103
8	6	24	157	5	10.306
9	6	48	237	6	38.531
10*	9	13	61	3	2.203

* In these problems, operation processing times are identical on all machine tools of a particular machine type.

TABLE V

Computational Results on an AMDAHL 5860

Problem Number	Number of Machines	Number of Operations	Number of Nodes	Number of Subproblems	CPU Time (seconds)
11(1)*	4	12	23	5	0.145
12(1)*	4	12	91	5	0.456
13(2)*	4	12	27	5	0.065
14(2)*	4	12	26	4	0.114
15	4	15	252	6	1.745
16(3)*	5	10	83	3	0.114
17(3)*	5	10	64	4	0.114
18(4)*	5	12	47	5	0.221
19(4)*	5	12	333	7	0.465
20(5)*	5	14	253	5	1.010
21(5)*	5	14	291	5	1.507
22(6)*	6	12	70	4	0.113
23(6)*	6	12	39	5	0.118
24(7)*	6	14	41	5	0.121
25(7)*	6	14	53	5	0.209
26	7	10	59	5	0.237
27(8)*	7	12	58	6	0.121
28(8)*	7	12	67	5	0.204
29(9)*	8	12	52	4	0.107
30(9)*	8	12	1,424	16	2.871
31(10)*	8	14	159	5	0.256
32(10)*	8	14	145	5	0.354
33(11)*	8	15	213	6	0.486
34(11)*	8	15	1,092	12	3.673

TABLE V (Continued)

Problem Number	Number of Machines	Number of Operations	Number of Nodes	Number of Subproblems	CPU Time (seconds)
35(12)*	9	15	28	6	0.116
36(12)*	9	15	26	6	0.127
37(13)*	9	20	53	5	0.232
38(13)*	9	20	47	5	0.251
39(14)*	10	20	338	7	1.046
40(14)*	10	20	338	7	1.188
41	10	25	904	12	5.278
42	11	20	369	6	1.250
43	11	20	369	6	1.371
44(15)*	11	24	542	8	3.154
45(15)*	11	30	443	9	6.370
46	11	30	223	7	5.529
47(16)*	12	20	754	12	2.269
48(16)*	12	20	854	12	2.376
49(17)*	12	35	88	6	1.476
50(17)*	12	35	618	9	12.922
51(18)*	13	36	771	10	13.674
52(18)*	13	36	771	10	15.111
53	14	35	656	10	7.655
54	15	30	732	10	5.309
55	15	40	849	11	16.163

*In these problems, operation processing times and the number of slots for holding the tools of each operation are identical on all machines of a particular machine type. The larger problem number of each pair (marked in parentheses) has a relatively smaller capacity of the tool magazine.

The computational results of these 55 problems are provided in Tables IV and V. In general, the algorithm performs better when the processing time of each operation is different on each machine tool or when the machine tools are of different types. This is because when there are many identical machine tools, there are many similar, equivalent solutions and the algorithm will not be able to prune the branch and bound trees as efficiently as otherwise. Such differences in performance can be seen by comparing Problem Numbers 1 and 2 of Table IV, both having 3 machines and 8 operations. For Problem Number 2, which has identical machines, the CPU time of 3.84 seconds compares unfavorably with .48 seconds. However, Problem Numbers 10 and 26 also consider identical machine tools and the CPU time is reasonable. A heuristic method to handle these particular types of systems having identical or pooled machines is currently being developed.

Of the 45 problems that are reported in Table V, there are 18 pairs, which are noted in the parentheses. The larger problem number of each pair has a relatively smaller tool magazine capacity. These problems all take longer to solve because the smaller magazine capacity is harder to satisfy. In these cases, the run times appear to be somewhat dependent on the ratio of tool magazine capacity to the number of slots required. In addition, increasing problem size increases both the number of subproblems as well as the number of nodes considered, as might be expected. However, all of these 45 problems were solved in less than 16 seconds of CPU time. It appears that run time would not limit the solution of practical problems.

These computational results are encouraging. The program is easy to apply, specialized, and self-contained. (This contrasts with the original solution procedure (Stecke (1983)), which is unwieldy and time consuming when performing the linearizations. Also it requires the availability of a mixed integer code). The CPU times are low enough to allow the frequent (one to three weeks, say, on average) re-solving of the FMS loading problem.

5. SUMMARY AND FUTURE RESEARCH NEEDS

This paper provides a self-contained and easy to use branch and bound approach for allocating operations and cutting tools to limited-capacity machine tools. This special purpose algorithm is tailored for FMS applications. Significant computational results demonstrate the efficiency of the algorithm. The focus here is on only one of several loading objectives, that of balancing the workload per machine while assigning each operation to only one machine.

However, the procedures described in §2 can be generalized to one of the pooling objectives of Stecke (1983). Prior to describing that extension, we recall some advantages of pooling. A group of pooled machine tools are similar and identically tooled. "Similar" means that all machines in a group can perform the same operations with identical processing times. Some advantages from the increased flexibility resulting from machine pooling include:

- an improvement in the utilization of machine tools;
- an increase in productivity;
- a decrease in the mean flow time of parts in the system;
- a decrease in waiting times;
- a reduction in in-process inventory.

Redundancy in the case of machine breakdowns is automatically provided with pooling. The processing sequence has several possible routes and the system can automatically cope with machine breakdowns and congestion. Machine pooling enhances the flexibility of real-time control and hence the production capacity of the FMS.

The efficiency of grouping several single-servers into one multi-server queue is established (Kleinrock (1976)). However, because of tool magazine capacity constraints, maximum grouping (i.e., pooling all machines into a single group) is generally infeasible because no machine's tool magazine can store all tools necessary to perform all required operations. In addition, machines of different types usually cannot belong to the same group.

The algorithm presented in §2 can be adapted to certain FMS situations with pooled machines. However, the tool magazine capacity constraint is harder to satisfy because each machine tool in a group must be able to perform any operation assigned to the group. The corresponding cutting tools are duplicated at each machine. Hence, the tool magazine capacity constraint for a group of machines is identical to the capacity constraint of any single machine in the group.

To use the procedures described in §2, the objective function of Problem (P_{δ_ℓ}) is modified to fit a pooled machine situation. If the objective is to balance the workload per machine, this quantity is obtained by dividing the workload assigned to a group by the number of machines in the group. This version of the FMS loading problem is:

<u>Problem (PG)</u>

$$\text{minimize } \left[\underset{k}{\text{maximum}} \left(\sum_{i=1}^{b} \frac{a_i p_{ik} x_{ik}}{s_k} \right)\right]$$

subject to:

$$\sum_{k=1}^{g} x_{ik} = 1, \qquad i \varepsilon I$$

$$\sum_{i=1}^{b} d_i x_{ik} + TC_k(x) \leq t_k, \qquad k=1,\ldots,g$$

$$x_{ik} = 0 \text{ or } 1, \qquad i \varepsilon I \text{ and } k=1,\ldots,g,$$

where

g is the number of machine groups;

s_k is the number of machine tools in group k;

t_k is the capacity of the tool magazine for any machine of group k;*

$TC_k(x)$ is given in equation (3).

The algorithm presented in §2 can solve problem (PG) provided that the number of machine tools in each group (s_k, $k=1,\ldots,g$) is known.

*This assumes that the tool magazine capacities of all machines in group k are identical. If not, t_k is the smallest of these capacities.

The objective function of problem (PG) may not be appropriate when the s_ℓ are not all equal. Stecke and Solberg (1985) show that when group sizes are unbalanced, workload per machine should also be unbalanced--in particular, each machine tool in a large group should be loaded more heavily than each machine in a small group, to maximize expected FMS production. This is shown using a closed network of multiserver queues (see Solberg (1977)). This model, although qualitatively robust (see Dukhovny and Koenigsberg (1981) and Suri (1983)), does not always provide reliable quantitative information. The optimal, unbalanced workload per machine of each group can be determined by using this model. There is not enough evidence to date that these workloads are also optimal for real FMSs having deterministic processing times. More research validating this point is needed before an unbalanced loading objective can be implemented on a shop floor. However, the unbalancing phenomenon _is_ qualitatively and theoretically true. A similar efficient branch and bound algorithm, or perhaps another procedure, could be developed for this situation as well as for other loading objectives.

APPENDIX A

DERIVATION OF THE $TC_j'(y)$ TERM IN PROBLEM (PR_j^K)

Suppose that at node K, for a particular assignment of operations, $\bar{x}$, the tool magazine capacity constraint is violated for machine j; then we have:

$$\sum_{i=1}^{b} d_i \bar{x}_{ij} + TC_j(\bar{x}) - t_j \equiv b_j > 0,$$

where

$$TC_j(\bar{x}) = \sum_{\substack{\forall B \subseteq I \\ \ni |B| \geq 2}} (-1)^{|B|+1} w_B \min_{i \in B} \{\bar{x}_{ij}\}$$

$$= \sum_{\forall B \subseteq I_J^K} (-1)^{|B|+1} w_B,$$

where

$$I_j^K = \{i \mid \bar{x}_{ij} = 1\}.$$

I_j^K contains the indices of the operations assigned to machine j (see Figure 1).

To obtain a feasible solution that satisfies this tool magazine constraint, one or more operations in I_j^K must be reassigned to other machines. If the set of reassignment variables is $\{y_{ij} \mid i \varepsilon I_j^K\}$, then the number of slots required by the operations remaining on machine j is:

$$\sum_{i=1}^{b} d_i \bar{x}_{ij} - \sum_{i \varepsilon I_j^K} d_i y_{ij} + \sum_{\substack{\forall B \subseteq I_j^K \\ \ni |B| \geq 2}} (-1)^{|B|+1} w_B \min_{i \varepsilon I_j^K} \{1 - y_{ij}\}. \qquad \text{(A1)}$$

The last term dictates that the amount of tool slot capacity previously taken (or returned), as measured by w_B, is now returned (or taken), if there is at least one operation i in B that is reassigned ($y_{ij} = 1$). The last term in equation (A1) can then be expressed as:

$$TC_j(\bar{x}) - \sum_{\substack{\forall B \subseteq I_j^K \\ \ni |B| \geq 2}} (-1)^{|B|+1} w_B \max_{i \varepsilon I_j^K} \{y_{ij}\} \equiv TC_j(\bar{x}) - TC_j'(y),$$

since

$$\min_i \{1 - y_{ij}\} = 1 - \max_i \{y_{ij}\}.$$

To insure a feasible solution that satisfies the tool magazine constraint of machine j, we have:

$$\sum_{i=1}^{b} d_i \bar{x}_{ij} - \sum_{i \varepsilon I_j^K} d_i y_{ij} + TC_j(\bar{x}) - TC_j'(y) \leq t_j,$$

which directly implies that:

$$\sum_{i \varepsilon I_j^K} d_i y_{ij} + TC_j'(y) \geq b_j. \quad ||$$

APPENDIX B

EFFICIENT STORAGE OF DATA ASSOCIATED WITH COMMON TOOLING CONSIDERATIONS

Computational time is decreased by providing a suitable means of storing the data pertaining to slots in a tool magazine that are saved when:

1. several cutting tools are common to two or more operations; and
2. the larger tools are positioned correctly in a tool magazine.

Let

p = number of subsets, B, of operations for which space can be saved in a tool magazine

b = total number of operations

A = $p \times 3$ array containing the relevant cutting tool overlap information

$B = \{i_1, i_2, \ldots, i_{|B|}\}$.

Consider row L corresponding to some subset, B, of operations. Then A consists of the following entries:

$$A(L,1) = |B|$$

$$\begin{aligned} A(L,2) &= \text{index that characterizes subset } B \\ &= 1 + \sum_{k=1}^{|B|} (i_k - 1) \cdot b^{|B|-k} \end{aligned}$$

$$A(L,3) = w_B.$$

$A(L,2)$ provides a unique ordered numeration of the elements of a multi-dimensional matrix in order to get a one-dimensional vector.

This information, which is stored compactly in A, can be used efficiently during enumeration as follows:

Suppose that a set, B_j, of operations has been assigned to machine tool j. Then

$$TC_j(B_j) = \sum_{\substack{\forall B \subseteq B_j \\ \ni |B| \geq 2}} (-1)^{|B|+1} w_B.$$

If another operation k ($\notin B_j$) is now also assigned to machine tool j, then

$$TC_j(B_j \cup \{k\}) = TC_j(B_j) + \sum_{\substack{\forall B \subseteq B_j \\ \ni |B| \geq 2}} (-1)^{|B|-1} w_{B\cup\{k\}},$$

where $w_{B\cup\{k\}}$ is read from array A, if this entry exists.

This approach to the calculation of the TC_j's can be somewhat burdensome computationally if many subsets of operations are listed in A. The maximum number of rows in A (which is p) could be up to $2^b - b - 1$. In practice, the number of rows of A is not very large because only tools common to two or three operations need be considered.

APPENDIX C

UPDATING LV WHEN $(P\delta_\ell)$ IS FEASIBLE

Problem $(P\delta_\ell)$ is to find an optimal solution, $x^*(\delta_\ell)$, minimizing

$$Z = \sum_{j=1}^{m} \sum_{i=1}^{m} a_i p_{ij} x_{ij} = \sum_{j=1}^{m} r_j(x)$$

under the condition: $x \epsilon D(\delta_\ell)$,

where $D(\delta_\ell)$ is the set of feasible solutions satisfying constraints (1), (2), (3), (4), and (5').

Assume that (P_{δ_ℓ}) has an optimal solution, $x^*(\delta_\ell)$. Updating LV, by setting $LV = \frac{Z^*(\delta_\ell)}{m} = \frac{1}{m} \sum_{j=1}^{m} r_j(x^*(\delta_\ell))$, is valid if we prove the following.

Proposition: Let $\delta_\ell' < \delta_\ell$. If $x^*(\delta_\ell')$ exists, then $\delta_\ell' \geq \frac{Z^*(\delta_\ell)}{m}$.

Proof: Note that $\delta_\ell' < \delta_\ell \implies D(\delta_\ell') \leq D(\delta_\ell)$, because of constraint (5').

Hence, if $x^*(\delta_\ell')$ exists, then

$$Z^*(\delta_\ell') \geq Z^*(\delta_\ell).$$

But $\forall$ j = 1,...,m, $r_j(x^*(\delta_\ell')) \leq \delta_\ell'$. (from (5'))

This implies that

$$Z^*(\delta_\ell') \leq m\,\delta_\ell' \,.$$

Hence, $\delta_\ell' \geq \frac{Z^*(\delta_\ell')}{m} \geq \frac{Z^*(\delta_\ell)}{m}$. Q.E.D

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Didier Dubois of Paul Sabatier University in Toulouse for helpful discussions during the preparation of this manuscript as well as for useful suggestions concerning the derivation in Appendix A. We would also like to thank Mr. Ilyong Kim of The University of Michigan for providing many of the computational results and the Referees and the Associate Editor for unusually thorough reading and helpful suggestions on some details and organization. This research was supported in part by grants from Ford Motor Company and from the Graduate School of Business Administration of The University of Michigan.

REFERENCES

BALAS, EGON, "Extension de l'algorithme additif à la programmation en nombres entiers et à la programmation non linéaire", C.R. Acad. Sc. Paris (May 1964).

BALAS, EGON, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables", Operations Research, Vol. 13, No. 4, pp. 517-546 (1965).

BARASH, MOSHE M., "Computerized Manufacturing Systems for Discrete Products", Ch. VII-9 in The Handbook of Industrial Engineering, edited by Gavriel Salvendy, John Wiley & Sons, Inc., New York NY (1982).

CAIE, JAMES, LINDEN, JANET and MAXWELL, WILLIAM, L., "Solution of a Single Stage Machine Load Planning Problem", OMEGA, Vol. 8, No. 3, pp. 355-360 (1980).

CAVAILLÉ, JEAN-BERNARD, FORESTIER, J. P. and BEL, GERARD, "A Simulation Program for Analysis and Design of a Flexible Manufacturing System", in Proceedings, IEEE Conference on Cybernetics and Society, Atlanta GA, pp. 257-259 (October 1981).

DUKHOVNY, ISAAC M. and KOENIGSBERG, ERNEST, "Invariance Properties of Queueing Networks and Their Application to Computer/Communications Systems", Information Systems and Operations Research, Vol. 19, No. 3, pp. 185-204, (August 1981).

GARFINKEL, ROBERT S. and NEMHAUSER, GEORGE L., Integer Programming, John Wiley & Sons, Inc., New York NY (1972).

GLOVER, FRED, "Improved Linear Integer Programming Formulations of Non-Linear Integer Problems", Management Science, Vol. 22, pp. 455-460 (1975).

GLOVER, FRED and WOOLSEY, EUGENE, "Further Reduction of Zero-One Polynomial Programming Problems to Zero-One Linear Programming Problems", Operations Research, Vol. 21, pp. 156-161 (1973).

GLOVER, FRED and WOOLSEY, R. E., "Converting the 0-1 Polynomial Programming Problem to a 0-1 Linear Program", Operations Research, Vol. 22, pp. 180-182 (1974).

GRAVES, STEPHEN C. and LAMAR, BRUCE W., "An Integer Programming Procedure for Assembly System Design Problems", presented at the Joint National CORS/ORSA/TIMS Meeting, Toronto, Ontario, Canada (May 1981).

KARP, RICHARD M., "On the Computational Complexity of Combinatorial Problems", Networks, Vol. 5, pp. 45-68 (1975).

KLEINROCK, LEONARD, Queueing Systems, Volume 2: Computer Applications, John Wiley & Sons, Inc., New York NY (1976).

KUSIAK, ANDREW, "Loading Models in Flexible Manufacturing Systems", WP#05/83, Technical University of Nova Scotia, Department of Industrial Engineering, Nova Scotia, Canada (March 1983).

ROSS, G. TERRY and SOLAND, RICHARD M., "A Branch and Bound Algorithm for the Generalized Assignment Problem", Mathematical Programming, Vol. 8, pp. 91-103 (1975).

SALKIN, HARVEY M., Integer Programming, Addison-Wesley, Reading MA (1975).

SANDI, CLAUDIO, "Solution of the Machine Loading Problem with Binary Variables", in Combinatorial Programming: Methods and Applications, edited by B. Roy, D. Reidel, Dordrecht-Holland, pp. 371-378 (1975).

SHANTHIKUMAR, J. GEORGE and STECKE, KATHRYN E., "Reducing Work-In-Process Inventory in Certain Types of Flexible Manufacturing Systems", European Journal of Operational Research, 1986, forthcoming.

SOLBERG, JAMES J., "A Mathematical Model of Computerized Manufacturing Systems", in Proceedings, 4th International Conference on Production Research, Tokyo, Japan (August 1977).

STECKE, KATHRYN E., "Experimental Investigation of a Computerized Manufacturing System", Master's Thesis, Purdue University, W. Lafayette IN (December 1977).

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

STECKE, KATHRYN E., "Procedures to Determine Both Appropriate Production Ratios and Minimum Inventory Requirements to Maintain These Ratios in Flexible Manufacturing Systems", Working Paper No. 448, Division of Research, Graduate School of Business Administration, The University of Michigan, Ann Arbor MI (October 1985a).

STECKE, KATHRYN E., "A Hierarchical Approach to Solving Machine Grouping and Loading Problems of Flexible Manufacturing Systems", European Journal of Operational Research (1985b), forthcoming.

STECKE, KATHRYN E. and BROWNE, JIM, "Variations in Flexible Manufacturing Systems According to the Relevant Types of Automated Materials Handling", Material Flow, Vol. 2, No. 2, pp. 179-185 (July 1985).

STECKE, KATHRYN E. and MORIN, THOMAS L., "The Optimality of Balancing Workloads in Certain Types of Flexible Manufacturing Systems", European Journal of Operational Research, Vol. 20, No. 1, pp. 68-82 (April 1985).

STECKE, KATHRYN E. and SCHMEISER, BRUCE W., "Equivalent Representations of System Throughput in Closed Queueing Network Models of Multiserver Queues", Working Paper No. 324, Division of Research, Graduate School of Business Administration, The University of Michigan, Ann Arbor MI (December 1982).

STECKE, KATHRYN E. and SOLBERG, JAMES J., "The CMS Loading Problem", Report No. 20, NSF Grant No. APR 74 15256, School of Industrial Engineering, Purdue University, W. Lafayette IN (February 1981a).

STECKE, KATHRYN E. and SOLBERG, JAMES J., "Loading and Control Policies for a Flexible Manufacturing System", International Journal of Production Research, Vol. 19, No. 5, pp. 481-490 (September-October 1981b).

STECKE, KATHRYN E. and SOLBERG, JAMES J., "The Optimality of Unbalancing Both Workloads and Machine Group Sizes in Closed Queueing Networks of Multiserver Queues", Operations Research, Vol. 33, No. 4, pp. 882-910 (July-August 1985).

SURI, RAJAN, "Robustness of Queueing Network Formulae", Journal of the Association for Computing Machinery, Vol. 30, No. 3, pp. 564-594 (July 1983).