SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853-3801

TECHNICAL REPORT NO. 887

January 1990
Revised June 1992

AN APPROACH TO PRODUCTION PLANNING, SCHEDULING, AND DUE-DATE QUOTATION IN CYCLICALLY SCHEDULED MANUFACTURING SYSTEMS

by

Andrew G. Loerch and John A. Muckstadt

This work was supported in part by National Science Foundation Grant DDM-8819542.

This research was conducted using the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which is funded in part by the National Science Foundation, New York State, the IBM Corporation and members of the Center's Corporate Research Institute.

Abstract

The use of cyclic schedules for manufacturing systems, particularly ones with fairly stable demand, has been found to have several advantages over other methods of scheduling. These advantages include ease of communication to the shop floor, consistency with the just-in-time philosophy, and less myopic nature. When a cyclic schedule is put in place, additional structure is imposed on the system, and precedence relations are established between operations.

We exploit the structure of the cyclically scheduled manufacturing facility by modelling the system as an acyclic directed network, whose arc lengths correspond to the durations of the various production operations. By computing the longest path through the network, we obtain the start times of the operations, as well as a good estimate of the time needed to meet a specified demand. This time can then be compared with a desired due-date to determine if the due-date can be met.

For the situation where the due-date cannot be met, we formulate the problem of expending additional resources, such as overtime and subcontracting of parts, to meet the due-date, as a linear program. We describe an efficient algorithm based on Dantzig-Wolfe decomposition, to solve the potentially large problem of finding the least cost combination of additional resources needed to meet the demand by the due-date.

Finally, we give the results of a computational test of the algorithm using two sets of complex and realistic factory data.

1. Introduction

The purpose of this paper is twofold. First a methodology to solve the problem of production planning and scheduling in a manufacturing facility in which a cyclic schedule is in use is developed. Second, we show the methodology to be a potentially useful tool through the use of computational experiments. Where the need exists due to uncertainty in demand, to apply additional resources to the system to maintain the cycle length or to meet due-dates for orders, the algorithm presented here determines the least cost combination of resources needed to accomplish this task. These resources might be overtime, subcontracting of parts, or expediting operations.

As mentioned, we focus our attention on cyclic schedules, where a set of components is produced on a set of workcenters in a prescribed sequence, with the sequence repeated at regular time intervals. The use of these schedules has become increasingly prevalent in recent years in various manufacturing environments because they are significantly simpler to describe, understand, and implement than many other scheduling schemes. For instance, their use improves communication with the shop floor since the sequence of operations is fixed. Thus, the need for complicated dispatching procedures is reduced. Cyclic scheduling is consistent with the just-in-time philosophy. Efforts to reduce setup times can be concentrated on only the combinations of operations that occur in sequence. Also the task of coordinating other schedules such as raw material delivery, preventive maintenance, and work force becomes much simpler.

Because of these and other potential advantages, cyclic schedules are sometimes used in manufacturing environments in which they are not completely appropriate. An example of such an environment would be one with highly variable demand. In fact methods used to develop cyclic schedules typically assume constant and continuous demand. See, for example , Graves, et al. [1983] and Del Pilar and Jackson [1989]. Constant and continuous demand patterns are seldom seen in reality. Also, cyclic schedules would not be appropriate in environments where frequent disruptions caused by machine breakdowns or "rush" orders occur. Reestablishing the sequences in such facilities would be difficult. If these problems are not too severe the benefits of cyclic scheduling might be seen to outweigh its shortcomings, and they could be implemented and used.

The algorithm presented here can be used in situations where cyclic scheduling is not necessarily appropriate with respect to demand variability, and efforts must be made to increase capacity in order to maintain cycle lengths and to meet established due-dates. We show how it can be useful in establishing order due-dates as well.

The remainder of this paper is composed of three main parts. Following a brief literature review, a network model of the manufacturing system is given. The linear programming formulation of the problem and the presentation of its solution by Dantzig-Wolfe decomposition is described. The results of a computational demonstration of the methodology on realistic data from the microchip and automotive industries is then discussed. We conclude with a brief summary.

2. Literature Review

To our knowledge, no one has specifically addressed the problem of optimal allocation of additional resources to meet a due-date in a cyclically scheduled manufacturing facility. Much work has been done, however, in the areas of cyclic scheduling, due-date quotation, and optimization in production planning that provides a basis for the research presented here.

The preponderance of work done in the area of cyclic scheduling revolves around computing cyclic schedules for various manufacturing environments. Graves, et al. [1983] develop a heuristic algorithm to generate a cyclic schedule for re-entrant flow shops, in which jobs can be considered identical from a processing standpoint, and may return to any workcenter for further processing. McCormick, et al.[1988] formulate the problem of finding a cyclic schedule to maximize the throughput in a buffered serial line, solving it as an equivalent maximum flow problem. Matsuo [1986] considers cyclic sequencing of jobs in a two machine flow shop, both with and without intermediate buffer space. This work is the outgrowth of Hitz [1980] and Wittrock [1985] who studied cyclic scheduling in a flexible flow shop context. Roundy [1988] studies cyclic schedules for job shops with identical jobs.

For our computational work, we use the method of Del Pilar and Jackson[1989] who formulate the general production scheduling problem in the constant and continuous demand environment as a mixed integer program with a nonlinear objective function. Their heuristic solution to this problem builds on the work of Roundy [1986] and Jackson, Maxwell, and Muckstadt [1988] by using the power-of-two reorder intervals to simplify the objective function. They then employ a heuristic for determining sequences of operations on the various workcenters. They show these schedules to be near optimal.

Whereas our purpose is to discover a way to meet the various due-dates for orders given the status of the shop and the availability of additional resources that can be applied to the system, most of the research previously done in the area of due-date quotation has dealt with establishing rules for sequencing jobs in order to optimize some criteria of "goodness" with respect to service levels. These rules are typically functions of job class, urgency of the jobs, and due-dates of the orders.

The greatest part of the literature concerning due-dates falls into two categories. The first involves analytical, mathematical modelling of dynamic service systems with due-dates. See, for example, Jackson [1964], Kleinrock [1967], Holtzman [1971], and Goldberg [1977]. The other deals with the performance of various sequencing methods in simulated systems, consisting mainly of proposed dispatching rules, with occasional examinations of due-date quotations. The work of Conway, et al. [1967], Elvers [1973], Weeks [1979], Baker and Bertran [1981], Bertran [1983], Miuazaki [1981], and Baker [1984] fall into this category. A third category has emerged in which several authors have considered the problem of systematically quoting appropriate due-dates for arriving orders in a dynamic manufacturing system. See Seidmann and Smith [1981], Kanet and Christy [1984], Bookbinder and Noor [1985], Wein [1988], and Tate [1989].

Many attempts have been made over the years to apply linear programming techniques to the production planning and scheduling problem. Bowman [1956] first formulated the production planning problem as a transportation problem. Since then, others have suggested models for various special case production systems. However, real life production planning problems are typically so large that they are difficult to solve efficiently.

In order to overcome this problem, some researchers have suggested decomposition approaches. For instance, Dzielinski and Gomory [1965] solve approximately the problem formulated by Manne [1958] using Dantzig-Wolfe decomposition. Recently, McLain, Thomas, and Weiss. [1989] suggest a Dantzig-Wolfe decomposition approach to solve a linear programming formulation of the production scheduling problem which also considers capacity constraints. They improve efficiency by using the Leontief structure of the subproblem for its quick solution. They report, however, that large percentages of the CPU time needed to solve these decomposition problems is used to obtain the last few percent of improvement of the objective function to optimality. Thus it is often a good idea to employ heuristic stopping rules to reduce CPU time expenditure without too much deviation from optimality.

3. Precedence Structure and Network Formulation

Assumptions

Before we present our model and solution procedures, let us state some basic assumptions and introduce some basic nomenclature. The model and solution methodology are based on the following assumptions. We first assume that a good cyclic schedule is known. We will use the sequence of operations given in this schedule throughout the planning horizon being considered. Although the sequence will remain fixed, we show how the batch sizes and timing of each operation can be calculated for each cycle. this cyclic schedule can be obtained, say, using the method described by Del Pilar and Jackson [1989].

We note that the methodology presented here proceeds on the assumption that the sequences of operations on all machines are fixed. Obviously a cyclically scheduled factory satisfies this assumption. The methodology is also appropriate in any other scheduling situation where the need exists to keep the sequence of operations unchanged. We believe, however, that such a requirement would be rare for a non-cyclically scheduled factory, and we therefore concentrate on those that are cyclically scheduled.

We assume that a bill-of-materials for the production system has been specified and will remain fixed for the duration of the planning horizon as well. External demands for components produced by any operation and in any cycle throughout the planning horizon are assumed known and constant. The demands for parts which are produced in a certain operation need not be the same in each cycle, however.

The amounts of time needed to process each component in each operation and to set up the various machines to perform the operations are assumed to be known and constant. Both processing times and setup times may be given as a function of the operations involved. That is, if operation i precedes operations j and k, then processing and setup times between operation i and operation j, and operation i and operation k need not be the same.

Notation

The following notation will be used throughout this paper.

$a_{j,k}$ = setup (fixed) time for operation j when succeeded immediately by operation k,

$b_{j,k}$ = processing (variable) time of operation j when succeeded immediately by operation k,

d_j = external demand for parts produced by operation j within a particular cycle,

$\varphi_{j,k}$ = number of parts produced in operation j needed to produce one part in operation k (GOZINTO parameter),

S_j = start time of operation j,

S_0 = start time of first operation (earliest operation),

S_d = completion time of last operation,

S_d^* = due-date (given),

Q_j = batch size of operation j,

J = number of operations,

M = number of machines.

We note here that the above notation is nonstandard in the sense that the model we use departs from the "time bucket" scheduling of most MRP systems. Thus the usual time indices are missing in the above notation, and a continuous time representation is used. This approach will be explained fully as we proceed.

Precedence Structure

We will now associate a precedence structure with the given production system. This structure is a combinatorial representation of the way in which operations are sequenced. Sequencing is based on both the order in which jobs are performed on the workcenters, and on bill of material relations.

Consider a production system with M machines indexed by m. Each machine performs a set of operations repeatedly, with the sequence of the operations fixed.

The operations on each machine are viewed, for modelling purposes, as distinct operations. However, an operation that is performed might be the same as some previous operation on that machine.

A bill-of-materials network is given, representing the production system. Each node in this network corresponds to a production operation, and it is assumed that each node represents the production of a single item type. The arcs of the network show how the items flow to subsequent operations in the network. Associated with each arc is a parameter, $\varphi_{i,j}$, which is the number of item i required in operation j.

Combining the information in the cyclic schedule sequence and the bill-of-material network, we obtain a new network which we will call the precedence network. As in the bill-of-material network, each node corresponds to a single production operation. The arcs in the precedence network arise in two ways. First, there will be an arc (i,j) in the precedence network if operation i directly precedes operation j on some machine m. Second, there will be an arc (i,j) in the precedence network if there exists such an arc in the bill-of-material network, that is, if $\varphi_{i,j} > 0$.

To complete the precedence structure, we augment the precedence network by adding two nodes. The first node represents the start of production, which is denoted node 0. Arcs emanating from node 0 will terminate at a node representing the first operation on a particular machine, and there will be such an arc for each machine.

The second node added to the network is denoted node d, which represents the conclusion of all operations. There is an arc from each node that represents an operation for which there is external demand for its output to the terminating node d. Let $G = (V,E)$ be the augmented precedence network as described above, with node set V and arc set E.

Operations in subsequent cycles are renumbered sequentially starting with the next integer greater than the last operation number in the previous cycle. This re-numbering is performed because each operation is considered separately. However, multiple occurrences of the same operation in the network are not precluded. In fact the same operation might be performed several times within the same cycle. These operations will also have different node numbers and will be considered individually by the algorithm.

A simple example of the construction of the augmented precedence network is shown in figure 1. Figure 1(a) is a bill-of-materials for an assembly system, and figure 1(b) shows a Gantt chart for two cycles of the cyclic schedule. The bill-of-material relationships are depicted as arrows on the Gantt chart. Figure 1(c) is the resulting network for the system. Note the renumbering of the nodes in the second cycle, representing repeated performance of operations separately.

Decision Variables and Constraints

The production system will be operated such that at time S_j, operation j will begin and produce a batch of size Q_j. The batch sizes can be computed using the following relationships.

$$Q_j = \sum_{(j,k)\in E} \varphi_{j,k} Q_k + d_j, \quad \text{for all } j\in V. \tag{1}$$

These relations are the familiar bill-of-material equations that are commonly used in Material Resource Planning (MRP) systems. Products from any operation can be used to satisfy external demand, or may be consumed in subsequent operations. We assume that no product can be consumed making itself.

By construction of the augmented precedence network, we see that no operation can begin prior to the completion of all its predecessor operations. So we have the following constraints on the starting times.

$$S_j + (a_{j,k} + b_{j,k}Q_j) \le S_k, \quad \text{for all } (j,k) \in E, \tag{2}$$

where $(a_{j,k} + b_{j,k}Q_j)$ is the processing time, including time for setup, needed to make Q_j items in operation j. We note here that the total processing time associated with an arc in the network is also a function of the terminating node k. This complication is necessary to allow the consideration of, for instance, transportation time differences between different machines. Combining (1) and (2) and rearranging terms we get:

$$\begin{aligned} &S_0 && \ge 0, \\ &S_k - S_j - b_{j,k}Q_j && \ge a_{j,k}, \quad \forall\, (j,k) \in E, \qquad (3) \\ &Q_j - \sum_{(j,k)\in E} \varphi_{j,k} Q_k && = d_j, \quad \forall\, j \in V, \forall\, (j,k) \in E. \end{aligned}$$

Scheduling and Feasibility

We now compute values for the starting times and batch sizes. We note that the submatrix formed by the batch size constraints in (3) is triangular, and thus easily solved by back substitution, giving the values of all Q_j. We also note that in (2), the minimum duration of any operation in the network is given as

$$l_{j,k} = a_{j,k} + b_{j,k}Q_j. \qquad (4)$$

So, using the values of Q_j calculated previously, (4) gives the length of each arc in the network.

The problem of determining the starting times of each operation and whether or not the schedule is feasible with respect to a given due-date is reduced to finding the maximum distance from node 0 to node d. This problem is the familiar one that is solved in the Critical Path Method (CPM) or the related Program Evaluation and Review Technique (PERT), as mentioned in chapter 2. We can write this problem as a linear programming problem in the following manner.

$$\text{minimize } S_d - S_0$$
$$\text{subject to: constraints (3).}$$

This problem is easily solved using the longest path algorithm, which is described in Lawler [1976]. This algorithm computes the length of the longest path from node 0 to every other node in the network. Note that in order to use this algorithm, one must have an acyclic directed network, and the nodes must be numbered such that $i<j$ for any arc (i,j). An algorithm to re-number the nodes of the network to satisfy this requirement is given in Bradley, et al.[1977].

The distances computed from node 0 represent the earliest starting times for each node. The distance from node 0 to node d thus represents the shortest possible time needed to complete all the operations. As such, the distance from node 0 to node d can be compared with the due-date to determine whether the due-date can be achieved.

If the length of the longest path from node 0 to node d, namely S_d, is less than the due-date, we can infer that there is slack in the system. Often, no benefit is gained from finishing operations early because unnecessary inventory is generated. By repeating the longest path algorithm beginning at node d, and finding the lengths of the longest paths to all nodes back to node 0, we can compute the latest starting times for each operation. Let the length from node d to node j be denoted as l_j^d. Then we compute the latest feasible starting time for node j, S_j^d, as

$$S_j^d = S_d^* - l_j^d. \tag{5}$$

A decision must be made as to whether it is prudent to take up this slack and finish each operation "just-in-time". Under the assumption of constant and known processing and setup times, and known demand over the planning horizon, it would likely be a good idea.

If $S_d = S_d^*$, then the due-date is achieved exactly, and no adjustments to starting times of operations on the longest path can be made. However, the procedure can be used to adjust other starting times.

4. Linear Programming Model and Solution

Formulation and Decomposition

We define an infeasible schedule as one that cannot be completed by the due-date. That is, $S_d > S_d^*$. In this event, if we have additional resources that can be applied to the system to shorten the completion time, we want to determine how best can they be employed.

The extra resources that might be applied could be overtime, purchasing parts externally to reduce the number that would need to be produced, speeding the completion of some operations, and so forth. We assume that a known cost is associated with each additional resource and that the effect of adding a resource to an operation would be to shorten its duration.

The problem can be formulated as a linear program in a straight- forward manner. We first define the following additional notation. Let

$x_{i,j}$ = amount of time saved by additional resource applied to operation i that terminates with operation j (on arc (i,j)),

y_k = number of parts purchased externally to replace output of operation k (corresponding to node k),

$p_{i,j}$ = cost of one unit of time saving using $x_{i,j}$,

q_k = cost of one part purchased for operation k,

$u_{i,j}$ = upper bound on $x_{i,j}$,

u'_k = upper bound on y_k.

It should be noted at this point that the operations on which overtime, expediting, or external parts purchase are possible are usually limited in practical applications. We would not expect that a manager would want to examine every possible operation. Thus it is very likely that only a more manageable subset of operations would be considered. Many of the arcs and nodes would not be candidates for additional resources. So we define the following notation. Let X be the subset of E containing all arcs on which overtime or other time savings can occur. Then let Y be the subset of V which contains all nodes for which parts can be externally obtained.

In practice, the amount of available additional resources will be limited. These restrictions will occur in two ways. First, there may exist some limit on the amount of resource that can be applied to an operation. Perhaps an operator will only be available for no more than a specified number of overtime hours, or a process can be expedited only to a certain level.

Second, we would logically restrict the amount of overtime, say, on an operation to the duration of the operation. That is, if an operation required 4 hours to complete, and the length of the time required for the entire manufacturing process exceeded the due-date by 5 hours, i.e., $l - S_d = 5$, we would not want to apply all 5 hours of overtime, or another additional resource, to the 4 hour operation. Unless we consider bounds on the additional resources, a situation like this one could arise.

Our goal is to solve the problem of finding a schedule of start times and batch sizes, modified by the least cost combination of additional resources, that will increase shop capacity sufficiently to meet the desired due-date, S_d^*. We can simplify the above problem by observing that no additional resources need be added if the schedule is already feasible. Whether or not feasibility has been attained has already been determined by computing the longest path through the network, as described earlier. This problem would not be solved unless the schedule was infeasible, and in order to achieve feasibility at least cost we would want $S_d = S_d^*$. So we will fix $S_d = S_d^*$. We also assume without loss of generality that $S_o = 0$. We then obtain the following LP formulation.

(P) Minimize $\sum_{(i,j)\in X} p_{ij}x_{ij} + \sum_{k\in Y} q_k y_k$ (6)

subject to:

$$S_k - S_j - b_{jk}Q_j + x_{jk} \geq a_{jk}, \ \forall\, (j,k)\in E,\ k\neq d \quad (7)$$

$$-S_j - b_{jd}Q_j + x_{jd} \geq a_{jd} - S_d^*, \ \forall\, (j,d)\in E \quad (8)$$

$$Q_j - \sum_{(i,j)\in E} \varphi_{ij}Q_i + y_j = d_j, \ \forall\, j\in V \quad (9)$$

$$0 \leq x_{ij} \leq u_{i,j}, \ \forall\, (i,j)\in X \quad (10)$$

$$x_{ij} = 0, \ \forall\, (i,j)\in E\setminus X \quad (11)$$

$$0 \leq y_k \leq u'_k, \ \forall\, k\in Y \quad (12)$$

$$y_k = 0, \forall\, k\in V\setminus Y. \quad (13)$$

In matrix form, (P) can be expressed as follows.

(P) Minimize $\mathbf{C}\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$

subject to: $G\mathbf{s} + A\mathbf{Q} + H'\mathbf{x} \geq \mathbf{a}$ (14)

$D\mathbf{Q} + F'\mathbf{y} = \mathbf{d}$ (15)

$\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}'', \mathbf{0} \leq \mathbf{y} \leq \mathbf{u}'$,

where $\mathbf{C} = (\mathbf{p}^t, \mathbf{q}^t)$, the vector of costs of the resource variables,

$\mathbf{x}$ = vector of overtime variables, with bounds $\mathbf{u}''$,

$\mathbf{y}$ = vector of purchased parts variables, with bounds $\mathbf{u}'$,

G = matrix of coefficients of S_j forming the node-arc incidence matrix of the network,

A = matrix of processing times,

D = triangular BOM matrix,

H′ = matrix of coefficients of the x_{jk},

H = matrix of starting time resource variables,

F′ = matrix of coefficients of the y_k, F = matrix of material balance resource variables.

In addition to bounds on the additional resource variables, it also may be important to consider constraints on linear combinations of these variables. For example, we might want to constrain the total number of overtime hours expended to some specified amount, or it may be necessary to limit the total budgetary expenditure. In order to accommodate these additional constraints, we let B represent the matrix of constraint coefficients, including bounds and let $\mathbf{u}^t = (\mathbf{u}'^t, \mathbf{u}''^t, \mathbf{u}'''^t)$, where $\mathbf{u}'''$ = the right hand side vector of the constraints. So we have

$$-B\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} \geq -\mathbf{u}.$$

Then (P) is reformulated as

(P_c) Minimize $\mathbf{C}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}$

$$\begin{aligned}\text{subject to: } G\mathbf{s} + A\mathbf{Q} + H'\mathbf{x} &\geq \mathbf{a} && (14)\\ D\mathbf{Q} + F'\mathbf{y} &= \mathbf{d} && (15)\\ -B\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} &\geq -\mathbf{u} && (16)\\ \mathbf{x},\mathbf{y} &\geq \mathbf{0}.\end{aligned}$$

In order to solve this problem, we consider its dual which can be expressed as follows.

(D_c) Maximize $\mathbf{a}^t\alpha + \mathbf{d}^t\beta - \mathbf{u}^t\gamma$

$$\begin{aligned}\text{subject to: } (H')^t\alpha + (F')^t\beta - B^t\gamma &\leq \mathbf{C} && (17)\\ G^t\alpha &\leq \mathbf{0} && (18)\\ A^t\alpha + D^t\beta &= \mathbf{0} && (19)\\ \alpha \geq \mathbf{0}, \gamma &\geq \mathbf{0}.\end{aligned}$$

Examining the structure of program (D_c), we immediately see the following. First, if constraints (17) are ignored, the feasible region defined by the constraints that remain is the same as the dual of the longest path problem discussed previously. This structure suggests an easy method to solve this problem with constraints (17) relaxed. As such, Dantzig-Wolfe decomposition can be used to solve this potentially large problem. This approach is fully described in Dantzig and Wolfe [1960] or Bradley, et al. [1977].

Second, we note that the third terms of both the objective function and constraints (17) are independent of the variables involved in constraints (18) and (19). Thus, further decomposition of the problem is also suggested.

Finally, the relaxation of (D_c) with respect to constraints (17) leaves a feasible region that has the form of a polyhedral cone, meaning that the feasible set is unbounded. Therefore, only extreme rays, i.e. directions of unboundedness, are generated by the solution of the relaxed problem. As we will see, this effect greatly simplifies the solution of the master problem in the decomposition.

Relaxation of (17) and rearrangement of terms in the objective function gives the following reformulation of (D_c).

$$\begin{array}{lll} (S_D) \text{ Maximize} & \mathbf{a}^t\alpha + \mathbf{d}^t\beta - \rho^t[(H')^t\alpha + (F')^t\beta] - [\mathbf{u}^t\gamma - \rho^t B^t\gamma] & \\ \quad \text{subject to:} & G^t\alpha & \leq \mathbf{0} \\ & A^t\alpha + D^t\beta & = \mathbf{0} \\ & \alpha, \gamma \geq \mathbf{0}, \ \beta \text{ unrestricted.} & \end{array}$$

We then decompose the problem (S_D) further by forming the two subproblems. The first subproblem deals only with variables α and β, and the second deals only with the variable γ. Thus we obtain the following:

$$\begin{array}{llll} (S_{D1}) \text{ Maximize} & \mathbf{a}^t\alpha + \mathbf{d}^t\beta - \rho^t[(H')^t\alpha + (F')^t\beta] & & \\ \quad \text{subject to:} & G^t\alpha & \leq \mathbf{0} & \\ & A^t\alpha + D^t\beta & = \mathbf{0} & (20) \\ & \alpha \geq \mathbf{0}, \ \beta \ \text{ unrestricted,} & & \\ (S_{D2}) \text{ Maximize} & -\mathbf{u}^t\gamma - \rho^t(-B^t\gamma) = -(\mathbf{u}^t - \rho^t B^t)\gamma & & \\ \quad \text{subject to:} & \gamma \geq \mathbf{0}. & & \end{array}$$

Letting $\mathbf{w_{1i}} = (\alpha_i, \beta_i, \mathbf{0})$, $\mathbf{w_{2i}} = (\mathbf{0}, \mathbf{0}, \gamma_i)$, with $\mathbf{r_1} = (\mathbf{a}, \mathbf{d}, \mathbf{0})$, and $\mathbf{r_2} = (\mathbf{0}, \mathbf{0}, -\mathbf{u})$; and matrices $P_1 = (H'^t, F'^t, 0)$ and $P_2 = (0, 0, -B^t)$, we have the following formulation of the master problem incorporating the two separate subproblems.

$$(\mathrm{M_c})\quad \text{Maximize} \quad \sum_{i=1}^{2}\left(\sum_{j=1}^{N_i} \lambda_{ij}\mathbf{r_i}\mathbf{v_{ij}} + \sum_{j=1}^{M_i}\mu_{ij}\mathbf{r_i}\mathbf{w_{ij}}\right)$$

$$\text{subject to:} \quad \sum_{i=1}^{2}\left(\sum_{j=1}^{N_i} \lambda_{ij}P_i\mathbf{v_{ij}} + \sum_{j=1}^{M_i}\mu_{ij}P_i\mathbf{w_{ij}}\right) \leq \mathbf{C}$$

$$\sum_{j=1}^{N_1}\lambda_{1j} = 1 \qquad (21)$$

$$\sum_{j=1}^{N_2}\lambda_{2j} = 1 \qquad (22)$$

$$\lambda_{ij},\ \mu_{ij} \geq 0,$$

where

$\mathbf{w_{ij}}$ = the extreme ray of the jth iteration of the ith subproblem

$\mathbf{v_{ij}}$ = the extreme point of the jth iteration of the ith subproblem

N_i = the number of extreme points generated by the ith subproblem

M_i = the number of extreme rays generated by the ith subproblem.

Solution of (M) provides values for the dual variables, ρ, which will modify the objective functions of the subproblems. These variables are nothing more than the **x** and **y** variables from (P). So solving (M) gives the values of the additional resources needed to meet the due-date.

As previously mentioned, the feasible regions of the two subproblems are polyhedral cones. Therefore, they each have only one extreme point such that $\mathbf{v_{11}} = \mathbf{v_{21}} = \mathbf{0}$. For this reason we know by (21) and (22) that $N_1 = N_2 = 1$, and $\lambda_{11} = \lambda_{21} = 1$. If we let σ_1 and σ_2 be the dual variables associated with (21) and (22), respectively, we also know that $\sigma_1 = \sigma_2 = 0$ as in the unconstrained problem of the previous section. So the condition for termination for the subproblems will be such that the optimal objective function value must be less than or equal to zero for both subproblems. The meaning of this condition will become clear as we discuss the solution of the two subproblems.

Solution of the First Subproblem (S_{D1})

In order to solve (S_{D1}) we consider its dual, which is written as follows.

(D_{S1}) Minimize $\mathbf{0s} + \mathbf{0Q}$

$$\begin{aligned} \text{subject to: } G\mathbf{s} + A\mathbf{Q} &\geq \mathbf{a} \\ D\mathbf{Q} &\geq \mathbf{d} \\ \mathbf{s} \geq \mathbf{0}, \ \mathbf{Q} &\text{ unrestricted.} \end{aligned}$$

Note that here, **a** and **d** have been modified by the dual variables from the master problem.

We see immediately that (D_{S1}) has the same feasible region as the longest path problem described previously. We also see that the objective function of this problem is such that any feasible solution would give the same objective function value, namely zero. So if we find a feasible solution we have also found an optimal solution to (D_{S1}). We know that solving the longest path problem will give a feasible solution to this problem. Thus we have an easily performed algorithm for solving (D_{S1}).

After solving (D_{S1}), we need to compute the dual variables α and β. To do this we first note that α is the vector of dual variables that correspond to the timing constraints of (P). We saw before that for a specified value of **Q**, these constraints form the longest path problem. So the dual variables α can be calculated immediately as follows.

$$\alpha_{i,j} = \begin{cases} 1, & \text{if (i,j) is on the longest path,} \\ 0, & \text{otherwise.} \end{cases}$$

This result is well known and can be found in Hu [1979].

To calculate β, use (20) so that

$$A^t\alpha + D^t\beta = \mathbf{0}.$$

This implies that

$$\beta = -(D^t)^{-1}A^t\alpha, \text{ or } \beta^t = -\alpha^t A D^{-1} \qquad (23)$$

We noted previously that under the given assumptions, D is a triangular matrix with one as every diagonal element, and is thus always invertible. So (23) always gives a closed form expression for β. The calculation of α and β form the extreme ray used to generate a new column in the master problem.

We observed previously that a condition for optimality is that the optimal value of the objective function of this subproblem must be less than or equal to zero. the above condition will be met, i.e., the optimal objective function value of the current subproblem will be less than or equal to 0, if the length of the longest path is less than or equal to the due-date. This can be shown by examining the following. Let l be the length of the longest path through the augmented precedence network from node 0 to node d. The quantity l can be expressed as

$$l = \sum_{(i,j)\in E} \alpha_{ij} l_{ij},$$

where l_{ij} = length of arc (i,j), and

$$\alpha_{i,j} = \begin{cases} 1, & \text{if } (i,j) \text{ is on the longest path,} \\ 0, & \text{otherwise.} \end{cases}$$

But,

$$l_{ij} = b_{ij}(Q_i - y_j') + a_{ij} - x_{ij}$$

with x_{ij} measuring the amount of time saved by using additional resources on the operation associated with arc (i,j),

and y_j' representing number parts saved by external purchases for operation j.

Now, the value of the objective function of the current subproblem is

$$Z = [(\mathbf{a}^t, \mathbf{d}^t) - \rho^t P]\binom{\alpha}{\beta},$$

where α and β are the dual variables from (D_{S1}). But,

$$\beta^t = -\alpha^t AD^{-1} \text{ and } \rho = \binom{\mathbf{x}}{\mathbf{y}}, \text{ so that}$$

$$\begin{aligned} Z &= \alpha^t\mathbf{a} - \alpha^t AD^{-1}\mathbf{d} - \rho^t P\binom{\alpha}{\beta} \\ &= \alpha^t\mathbf{a} - \alpha^t AD^{-1}\mathbf{d} - \mathbf{x}^t(H')^t\alpha + \mathbf{y}^t(F')^t(\alpha^t AD^{-1})^t \\ &= \alpha^t[\mathbf{a} - A(\mathbf{Q} - D^{-1}F'\mathbf{y}) - H'\mathbf{x}]. \end{aligned}$$

We expand this expression to get

$$Z = \sum_{(j,k)\in E} \alpha_{jk}[\tilde{a}_{jk} + b_{jk}(Q_j - y_j') - x_{jk}]$$

Recall that

$\mathbf{a}$ = vector of $\tilde{a}_{jk}$ variables, where

$$\tilde{a}_{jk} = \begin{cases} a_{jk}, \text{ if } k \neq d, \\ a_{jk} - S_d^*, \text{ otherwise.} \end{cases}$$

By definition of longest path, we know that only one arc can have node d as its end node. So,

$$Z = \sum_{(j,k)\in E} \alpha_{jk}[a_{jk} + b_{jk}(Q_j - y_j') - x_{jk}] - S_d^*$$

$$= 1 - S_d^*.$$

Thus, consistent with intuition, the optimal solution to (M), and therefore (P) and (D), will be obtained when the longest path through the augmented precedence network is reduced to the due-date by the application of additional resources.

Solution of the Second Subproblem (S_{D2})

We now discuss finding the extreme rays corresponding to the second subproblem, (S_{D2}). We first examine the structure of its objective function. Recall that the values of the elements of the ρ vector, which are the dual variables that are generated by solving the restricted master problem, are the imputed values of the additional resource variables. Thus $B\rho$ is the vector of values of the left hand sides of the constraints on the additional resource variables, while $\mathbf{u}$ is the vector of values of the right hand side of these constraints. Therefore, the objective function coefficients of problem (S_{D2}), namely ($\mathbf{u}$ - $B\rho$), measure whether or not the various constraints are violated at any particular iteration.

By restricting $\gamma \geq \mathbf{0}$, we see that when an element of -($\mathbf{u}$ - $B\rho$) is negative, (S_{D2}) is unbounded in the direction corresponding to that element, thus providing an easy way to determine the extreme rays of (S_{D2}) that are needed to generate new columns of the master problem.

Compute the kth element of the extreme ray generated by the second subproblem in the jth iteration, $w_{2,j,k}$, as follows. Let

$$\mathbf{C} = (C_1, C_2, \ldots, C_{m1}) = -(\mathbf{u} - B\rho)$$ with m1 the number of additional variable constraints. Then

$$w_{2,j,k} = \begin{cases} 0, & \text{if } C_k \geq 0 \text{ in the jth iteration,} \\ 1, & \text{if } C_k < 0 \text{ in the jth iteration.} \end{cases}$$

This result is intuitively pleasing in that an extreme ray is generated for the master problem whenever at least one of the constraints on resource variables is violated. This extreme ray is used to compute a new column for the restricted master problem which, when solved, corrects the violation.

Solution of the Master Problem (M)

The algorithm for solving (P) by Dantzig-Wolfe decomposition proceeds in a very intuitive manner. The initial conditions are such that there are no additional resources applied in the system. The first subproblem (S_{D1}) is solved with $\rho = \mathbf{0}$ using the longest path procedure as before. When an infeasible schedule is detected, a master problem column is generated and the restricted master problem is solved, producing a set of dual variables. These dual variables are the extra resource variables in the original problem. The second subproblem is then used to check to see if any constraints on the extra resource variables are violated. If so, an extreme ray is computed as described above from the second subproblem, which is used to generate another column for the restricted master problem. The dual solution to the master problem will be a set of extra resources with the previously violated constraint satisfied if there is one.

The algorithm proceeds by finding longest paths, adding additional resources to correct infeasibility, and then correcting violations on constraints until it terminates with a feasible longest path and no constraint violations, or until the master problem is determined to be infeasible.

Using Available Overtime Periods

Throughout the previous discussion, we have implicitly assumed that the application of an additional resource could be identified with a particular node or arc in the network. In many instances this assumption would be very reasonable. Frequently, however, overtime will be available as a block of time in the schedule. Perhaps overtime may be performed during a 20 hour period over the weekend, or maybe four hours are available each weekday evening. We want our methodology to be able to handle this real life situation.

In order to do so, we must consider two complicating factors. First, we must decide which operations could be done during the various overtime periods. Because the overtime periods are stationary in time, and the starting and ending times of the operations may change as we work toward a feasible schedule, close attention must be paid to the potential intersection of the arcs of the network and the overtime periods. Second, as the algorithm progresses, the bounds on the overtime variables could change depending on the value of resource variables applied on preceding arcs.

Unfortunately, the obvious method of constraining the additional resource variables in such a way as to control their relationship to the start times would introduce constraints that would destroy structure of problem, and would render the algorithm presented previously useless. The method presented below will avoid that problem.

The general approach that is used is to identify all arcs, (i,j), that could have overtime applied to them during the various overtime periods, based on the values of the initial infeasible S_j's, and associate an additional resource variable, x_{ij}, with each. We then solve the linear programming problem as before, except that at every iteration, we adjust the bounds on these variables to take into consideration overtime applied to preceding operations.

We assume that the cost of overtime during any period is the same, regardless of the machine on which it is applied. We allow multiple overtime periods, but we are required to examine them in order from earliest to latest, because overtime early in the schedule affects the selection of arcs on which overtime is assigned later.

We now introduce some additional notation.

t_{ot_k} = schedule time of kth overtime period,

T_{ot_k} = duration of kth overtime period,

l_j^d = length of longest path from node d to node j,

l_i^o = length of longest path from node 0 to node i

$l_{ij} = S_j - S_i$, the length of arc (i,j),

$l_{ij}{}^* = l_{ij} + l_j{}^d + l_i{}^o$, the length of the longest path through arc (i,j).

The first task that must be performed is to identify arcs on which overtime may be needed. We eliminate from consideration any arc (i,j) if $l_{ij}{}^* \leq S_d{}^*$, since the schedule will be feasible when the longest path includes (i,j). This check is applied to all arcs being considered.

A depth first search is begun at any arc (i,j) for which the condition $S_i \leq t_{ot_k} < S_j$, holds, that is, on arcs during which the overtime is scheduled to start. Overtime variables, x_{ij}, are assigned to these arcs. Successors are examined in the following manner. We also observe that any overtime applied in previous overtime periods makes it possible for the starting times of subsequent operations to be earlier. Therefore we must consider this effect when determining the candidate arcs downstream. We do this by noting that the earliest an operation can begin is the total overtime applied to previous operations prior to the calculated start time. That is, if we let

$$I = \{k : t_{ot_k} < S_j\},$$

then we compute

$$S_j' = S_j - \sum_{k \in I} T_{ot_k},$$

where S_j' = the earliest possible start time of operation j. An overtime variable is associated with a successor arc (i,j) if the arc intersects the overtime period, that is, if it meets the condition that $S_i' < t_{ot_k} + T_{ot_k}$.

Then subsequent arcs are tested with the following condition.

$$S_d{}^* - l_j{}^d \geq t_{ot_k} + T_{ot_k}.$$

This condition says that if the overtime period concludes before the latest start time of the operation of the ending node of the arc, we need not consider successors of that arc, since no more overtime is needed to make the path feasible.

The search is terminated for a path whenever a condition above is not met. Backtracking must be used to examine other paths emanating from a common stem. This backtracking procedure is continued until all paths are examined, and additional resource variables are assigned as necessary.

The second task involves calculating bounds on the variables that have been assigned in the above manner. Recall that these bounds might need adjustment after every solution of the master problem.

We determine bounds for arcs in the network beginning with those affected by the latest overtime period under consideration. For each of these arcs, we test first whether the overtime period intersects the arc at the current iteration. There will be no intersection for arc (i,j) if either of the following conditions hold.

$$S_i > t_{ot_h} + T_{ot_h} \tag{24}$$

$$S_j < t_{ot_h} - \sum_{k=1}^{|K(i,j)|} x_{(i,j)}^{k}, \tag{25}$$

where $x_{(i,j)}^k$ = current value of the kth additional resource variable associated with arc (i,j), and K(i,j) = $\{k{:}x_{(i,j)}$ has been assigned$\}$.

Condition (24) says that the operation begins later than the overtime period ends. Condition (25) means that the operation ends before the overtime period begins. In either case we let the bound on the overtime variable under consideration be zero.

If neither of these conditions hold, further consideration must be given to the bound on the overtime variable associated with the arc and overtime period in question. We calculate a candidate bound, U_c, based on the relative positions, and therefore intersections, of the operation times with the overtime period. Conditions that could exist and the resulting candidate bounds are given as follows for arc (i,j).

Condition (1) $t_{ot} \le S_i < t_{ot} + T_{ot} < S_j$

$$U_c = t_{ot} + T_{ot} - S_i$$

Condition (2) $t_{ot} \le S_i < S_j \le t_{ot} + T_{ot}$

$$U_c = S_j - S_i = l_{ij}$$

Condition (3) $S_i \le t_{ot} < t_{ot} + T_{ot} \le S_j$

$$U_c = T_{ot}$$

Condition (4) $S_i \le t_{ot} \le S_j \le t_{ot} + T_{ot}$

$$U_c = S_j - t_{ot}$$

We must now compare the candidate bound, U_c, as calculated above, with the amount of overtime left in the overtime period, t_r. As stated, no overtime will be applied to an arc unless its predecessors are completed prior to or during the overtime period. Thus, unless the predecessors are at their bounds for overtime no overtime can be applied to the successor. We can easily compute the value of t_r as follows.

$$t_r = T_{ot} - \sum_{(i',j')\in N} U_{(i',j')},$$

where $N = \{(i'j') : (i'j')$ is a predecessor of (i,j) on the longest path from 0 to d, associated with an overtime variable in the same period$\}$, and

$U_{(i,j)}$ = bound on x_{ij}, the overtime variable on arc (i,j).

The value of $U_{(i,j)}$ can then be computed as follows.

$$U_{(i,j)} = \min\{U_c, t_r\}.$$

This procedure is repeated each time the values of the dual variables of the restricted master problem (M) change, i.e., after each iteration.

Individual Due-Dates for Separate Orders

We allow external demands for the output of any operation in the production process. Previously, however, we assumed a common due-date for the whole system. This assumption is clearly not appropriate in many cases, and can be easily relaxed. This relaxation is accomplished by associating an additional length of time with each arc that represents an external demand.

Recall that there exist arcs in the network terminating at node d, and originating at any node associated with production to meet external demand. We assign a due-date to each one. The demand that has the latest due-date will define the overall due-date for the system, denoted S_d^* as before. Then each demand arc will be lengthened by the difference between its due-date and S_d^*. So the additional length of the demand arcs, l_{ij}^*, can be written as

$$l_{ij}^* = S_d^* - S_{d_i}^*,$$

where $S_{d_i}^*$ is the due-date of the ith demand node.

While it may be possible to apply overtime to a demand arc, we must insure that we not shorten it so that its length becomes less than l_{ij}^*. Otherwise the due-date for the output of the operation in question would be extended. Thus we bound overtime variable x_{ij} on demand arc (i,j) as follows.

$$x_{ij} \leq a_{ij} + b_{ij}Q_i$$

While noting that the length of arc (i,j), l_{ij}, can be written as

$$l_{ij} = a_{ij} + b_{ij}Q_i + l_{ij}^* .$$

This restriction is consistent with previous bounds placed on overtime variables, and including the bound allows the methodology to proceed as before.

5. Computational Results

In order to evaluate the algorithm presented in the previous two chapters, we programmed it in Fortran 77 so that it could be run on a mainframe computer. The main program, together with the various subroutines we wrote, performs the tasks of reading the input data, establishing the data structures, solving both of the subproblems discussed in section 4, and providing output. The master problem is solved using the Experimental Mathematical Programming (XMP) package. For a detailed description of this package see Marsten [1981].

In testing the algorithm's performance, runs were made on an IBM 4341 mainframe. Other test runs, as well as development work were made on the IBM 3090 supercomputer. There is no reason why the algorithm could not be used on any computer that supports Fortran, including microcomputers, with very little modification.

Two different, but realistic, sets of data describing manufacturing systems were used in our test. The first set came from a facility that produced automobile transmissions. The data was also used by Jackson, Maxwell, and Muckstadt [1988]. In the production process used in this factory, 154 types of final products were produced by 271 different operations. The bill of materials structure was extremely complex, and the given bill of materials network was composed of 775 arcs along with the 271 operation nodes. There was a great deal of commonality of the components among the final products, and 27 workcenters were used in the production process. We felt that this example would provide a realistic test of our algorithm.

The bill of materials structure, the average demands over time, and the setup and processing times of the various operations were available in the data set. It remained to define a good cyclic sequence for the operations performed on each machine. To accomplish this task, the Del Pilar and Jackson method was used. The results obtained allowed us to form the augmented precedence network for use in our algorithm. This network had 1815 nodes, together with 8513 arcs, and it included

8 final product cycles for each of the 154 products in one overall factory cycle which was 4 days long. Although we envision our algorithm to be useful over a much longer time horizon, we note that these data describe an extremely high volume process. The number of operations performed in this short amount of time is compatible to a much longer time horizon in other factory settings.

We conducted the experiments on this data set in the following manner. We first calculated the length of the longest path through the augmented precedence network to determine the length of time needed to meet the given demand without the application of additional resources. The demand data used were equivalent to the constant and continuous demand distributed proportionally among the various final product nodes. We then made a series of runs with tighter and tighter due-dates. For each run we defined an overtime period whose length exceeded the difference between the longest path length and the due-date. Thus a feasible solution could be found in each case. To insure that all the features of the algorithm were exercised, we constrained the total overtime in the system to be less than necessary to meet the due-date, and allowed external purchase of parts on one node of the longest path through the network. The cost associated with the external parts variable was high enough to insure its use as a last resort. In each case, the data were arranged so that the constraint on the total overtime was violated, and the optimal solution has a positive value for the external parts additional resource variable.

Results of the experiment are shown in figure 2. There, the CPU time expended to solve the problem to optimality, along with the number of additional resource variables generated by the program, are plotted against the percent reduction from the length of the longest path through the network without additional resources applied. As suspected, the CPU time required increased with due-date tightness. The greatest reduction we considered was 37%,which resulted in a CPU time requirement of 120 seconds to solve the problem to optimality. This amount of reduction is very great and we believe that, in practice, such a large reduction would occur very infrequently. Even so, 120 CPU seconds is not so great as to be unreasonable, and the time required for smaller reductions was considerably less.

For the case of 37% reduction, 99 variables were generated by the program. So solving this particular case to optimality is equivalent to solving a linear program with over 10,000 variables, not including slack and artificial variables, and well over

10,000 constraints. Of course changing the bounds on the additional resource variables during the course of solving the problem would not be possible using standard linear programming codes. To solve our problem then, perhaps many additional constraints would have to be included in the formulation to avoid unreasonable or illogical values of the additional resource variables. Thus our method is useful in keeping the dimensionality of the problem as small as possible.

Whereas the first data set described a fairly typical factory setting, with few bottlenecks among a large number of machines, the second data set was designed to test the algorithm in a highly constrained and tightly scheduled manufacturing environment. The data were collected at a facility that produces microchips, and it was originally used to test dispatching and job release rules. The facility is very complex due to the need for frequent rework and multiple reentry to the various work centers. Yield is also an important consideration, and we assumed a constant and known yield for each operation. See Blum, et al. [1989] for a complete description of the data.

The facility in question has six workcenters, each performing six operations per cycle. In our experiment we used a planning horizon of 30 cycles, corresponding to 30 eight hour shifts. The resulting augmented precedence network was composed of 1082 nodes, of which 174 were external demand nodes, and more than 5000 arcs. Instead of one large overtime period as was the case for the first set of data, we allowed a small period during each cycle. As such, the variables that were generated were constrained to be of shorter duration. For each run, we reduced the due-date for each cycle while maintaining the same demand. Since all the machines were at or near capacity, we found that both CPU time and the number of variables generated grew very quickly with reduction in due-date. See figure 3. Nevertheless, the algorithm still solved the problems to optimality in a reasonable time.

6. Conclusions

We have presented an efficient solution technique for the problem of production planning and scheduling in a factory that has implemented a cyclic schedule. The algorithm makes use of the inherent structure provided by the cyclic schedule to achieve its high level of efficiency, evaluates of alternatives quickly to insure cycle times are maintained and established due-dates are met. During negotiation regarding the establishment of order due-dates, this method would be useful to evaluate the cost of meeting a proposed due-date when factory capacity is exceeded, thus allowing managers to make better decisions in this important, and often neglected area.

In summary, we believe that the methodology presented here will serve as a first step toward a real time scheduling and due-date quotation tool that can be used to improve the efficiency and service levels of a large class of manufacturing systems.

References

BAKER, K.R. and J.W.M. Bertrand [1981]. An Investigation of Due-Date Assignment Rules with Constrained Tightness. *J. Op. Management*, **1**, 109-120.

BAKER, K.R. [1984]. Sequencing Rules and Due-Date Assignments in a Job Shop. *Mgmt. Sci.*, **30**, 1093-1104.

BERTRAND, J.W.M. [1983]. The Effect of Workload Dependent Due-Dates on Job Shop Performance. *Mgmt. Sci.*, **29**, 799-816.

BLUM, J., K.E. McKONE, and M.L. ROCKEY [1989]. An Evaluation of Policies Which Implement Set Management at the International Business Machines Corportation East Fishkill, New York Facility, Master of Engineering Project, Cornell University, Ithaca, NY.

BOOKBINDER, J.H. and A. Noor [1985]. Setting Job-Shop Due-Dates with Service Level Constraints. *J. Opl. Res. Soc.*, **36**, 1017-1026.

BOWMAN, E.H. [1956]. Production Scheduling by the Transportation Method of Linear Programming. *Ops. Res.*, **3**, 1.

BRADLEY, S.P., A.C. Hax, and T.L. Magnanti [1977]. *Applied Mathematical Programming.* Addison-Wesley Publishing Co., Reading, Mass.

CHVATAL, V. [1983]. *Linear Programming.* W.H. Freeman & Co., New York.

CONWAY, R., W.L. Maxwell, and L.W. Miller [1967]. *The Theory of*

Scheduling. Addison-Wesley Publishing Co., Reading, Mass.

DEL PILAR, R. and P.L. Jackson [1989]. The Cyclic Scheduling Problem Formulation. Tech. Rep. 868, School of O.R.I.E., Cornell University.

DZIELINSKI, B.P. and R.E. Gomory [1965]. Optimal Programming of Lot Sizes, Inventory, and Labor Allocation. *Mgmt. Sci.*,**11**, 9, 874-890.

ELVERS, D.A. [1973]. Job Shop Dispatching Rules Using Various Delivery Date Setting Criteria. *Prod. and Inv. Mgmt.*, 4th Quarter 1973, 62-73.

GOLDBERG, H.M. [1977]. Analysis of the Earliest Due-Date Scheduling Rule in Queueing Systems. *Math. of O.R.*, **2**, 145-154.

GRAVES, S.C., H.C. Meal, D. Stefek, and A.H. Zeghmi [1983]. Scheduling of Re-entrant Flow Shops. *J. of Ops. Mgmt.*, **3**, 4, 197-203.

HITZ, K.L. [1980]. Scheduling of Flexible Flow Shops - II. Tech. Rep. LIDS-R-1049, Laboratory for Information and Decision Systems, Mass. Inst. of Tech.

HOLTZMAN, J.M. [1971]. Bounds for a Dynamic-Priority Queue. *Ops. Res.*, **19**, 461-468.

HU, T.C. [1982]. *Combinatorial Algorithms*. Addison-Wesley Publishing Co., Reading, Mass.

JACKSON, J.R. [1961]. Queues with Dynamic Priority Discipline. *Mgmt. Sci.*, **8**, 18-34.

JACKSON, J.R. [1964]. Waiting-Time Distribution for Queues with Dynamic Priorities. *Naval Res. Logist. Quart.*, **9**, 31-36.

JACKSON, P.L., W.L. Maxwell, and J.A. Muckstadt [1988]. Determining Optimal Reorder Intervals in Capacitated Production-Distribution Systems. *Mgmt. Sci.*, **34**, 8.

KANET, J.J. and D.P. Christy [1984]. Manufacturing Systems with Forbidden Early Order Departure. *Int. J. Prod. Res.*, **22**, 41-50.

KLEINROCK, L. [1964]. A Delay Dependent Queue Discipline. *Naval Res. Log. Quart.*, **11**, 329-341.

KLEINROCK, L. and R.P. Finkelstein [1967]. Time-Dependent Priority Queues. *Opns. Res.*, **15**, 104-117.

MANNE, A.S. [1958]. Programming of Economic Lot Sizes. *Mgmt. Sci.*, **4**, 2, 115-135.

MARSTEN, R.E. [1981]. XMP: A Structured Library of Subroutines for Experimental Mathematical Programming. *ACM Trans. on Mathematical Software.*, **7**, 481.

MAXWELL, W.L., J.A. Muckstadt, P.L. Jackson, and R.O. Roundy [1986]. The Interaction Between Design and Scheduling in Repetitive Manufacturing Environments. Tech. Rep. 692, School of O.R.I.E., Cornell University.

MATSUO, H. [1987]. Cyclic Scheduling Problems in the Two-Machine Flow Shop: Complexity, Worst-Case, and Average Case Analysis.

Working Paper #87-12-4, Grad. Sch. of Bus., University of Texas.

McCLAIN, J.O., L.J. Thomas, and E.N. Weiss [1989]. Efficient Solutions to a Linear Programming Model for Production Scheduling with Capacity Constraints and No Initial Stock. *IIE Transactions,* **21**,2, 144-152.

McCORMICK, S.T., M.L. Pinedo, S. Shenker, and B. Wolf [1988]. Sequencing in an Assembly Line with Blocking to Minimize Cycle Time. Tech. Rep., School of O.R.I.E, Columbia University.

MIYAZAKI, S. [1981]. Combined Scheduling System for Reducing Job Tardiness in a Job Shop. *Int. J. Prod. Res.*,**19**, 201-211.

MODER, J.J. and C.R. Phillips [1970]. *Production Management with CPM and PERT*, 2nd ed., VanNostrand, New York.

ROUNDY, R. [1986]. A 98%-Effective Lot-Sizing Rule for a Multi-Product, Multi-Stage Production/Inventory System. *Mathematics of Operations Research*,**11**,4,699-727.

ROUNDY, R. [1988]. Cyclic Schedules for Job Shops With Identical Jobs. Technical Report #766, School of O.R.I.E., Cornell University, Ithaca, New York.

SCHRIJVER, A. [1986]. *Theory of Linear and Integer Programming*, John Wiley and Sons, New York.

SEIDMANN, A. and M.L.Smith [1981]. Due-Date Assignment for Production Systems. *Mgmt. Sci.*, **27**, 571-581.

TATE, D.M. [1989a]. Due-Date Based Performance Measures for

Queueing Systems. (To appear, *Int. J. Comp. and Manuf. with Applications*).

TATE, D.M. [1989b]. Due-Date Based Performance Measures in Dynamic, Stochastic Manufacturing Systems. Ph.D. Dissertation, Cornell University, Ithaca, New York.

WEEKS, J.K. [1979]. A Simulation Study of Predictable Due-Dates. *Mgmt. Sci.*, **25**, 363-373.

WEIN, L.M. [1988]. Due-Date Setting and Priority Sequencing in a Multiclass M/G/1 Queue. (Submitted for Publication).

WITTROCK, R.J. [1985]. Scheduling Algorithm for Flexible Flow Lines. *IBM J. of Res. and Dev.*, **29**, 4, 401-412.

Figure 1. Construction of the Augmented Precedence Network

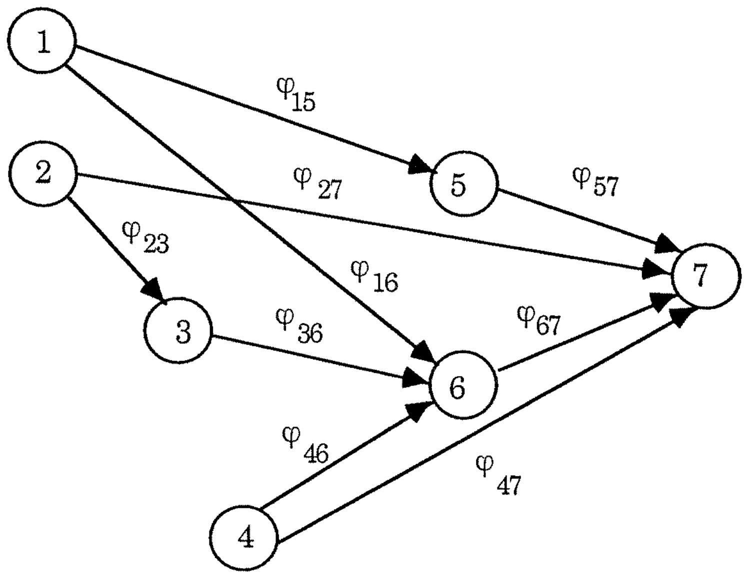

Figure 1(a) Bill-of-Materials Structure

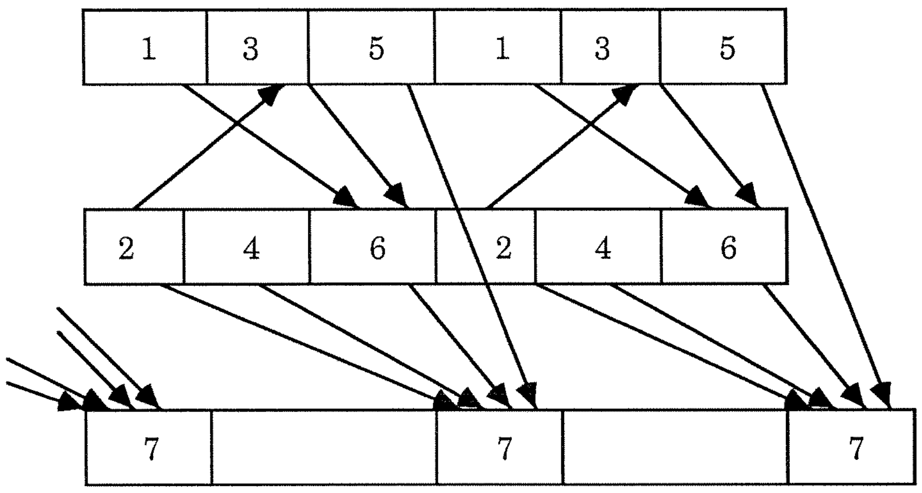

Figure 1(b) Cyclic Schedule Gantt Chart

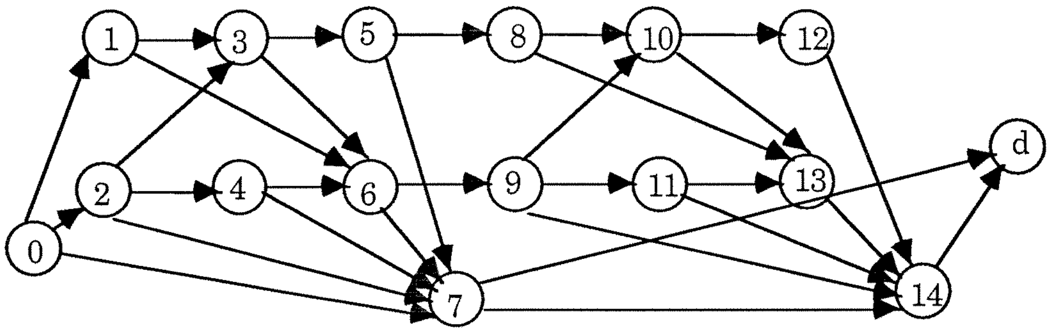

Figure 1(c). Augmented Precedence Network

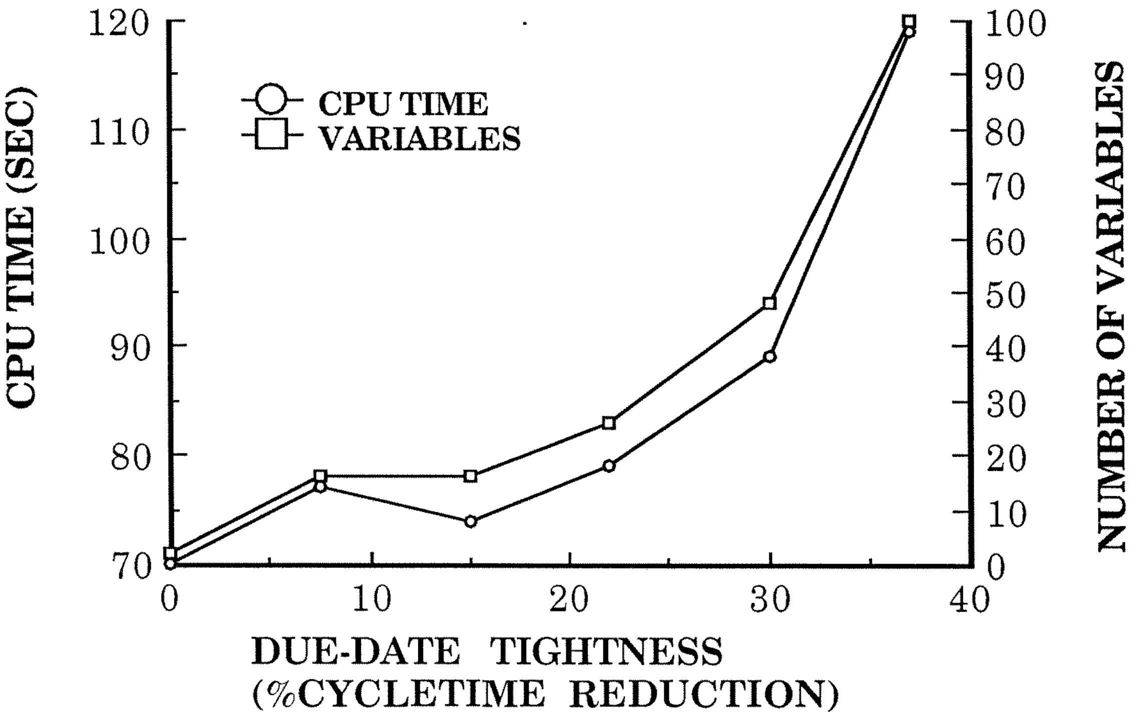

Figure 2. Experimental Results for Transmission Factory

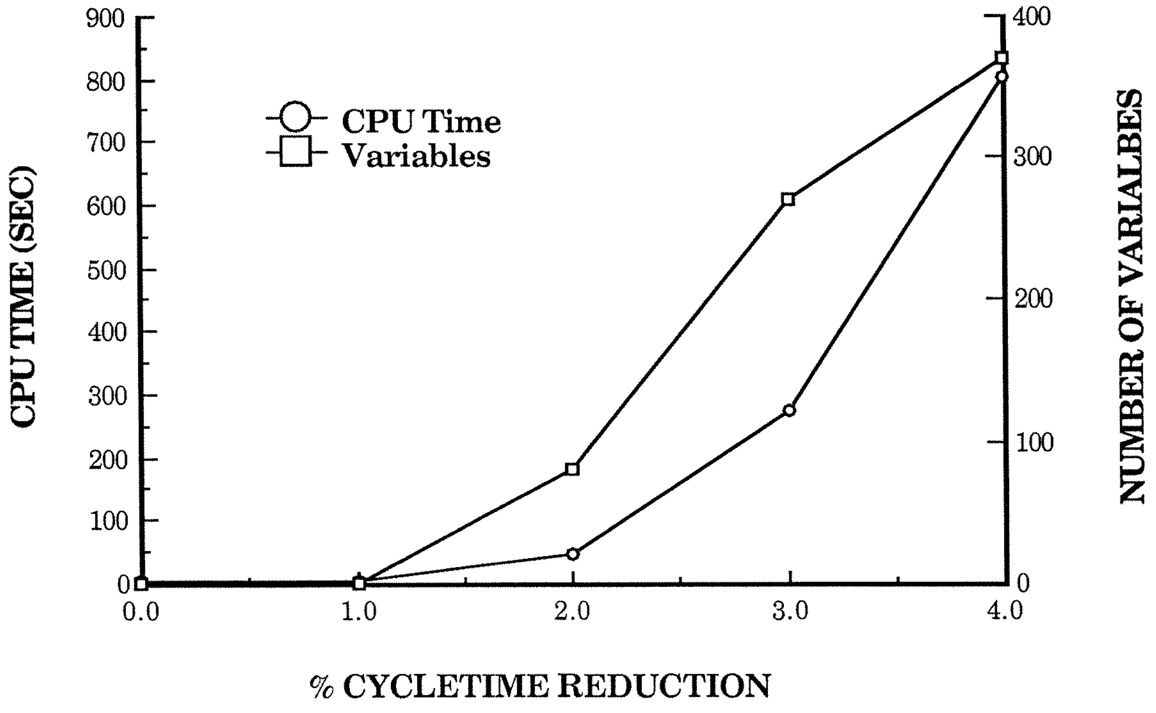

Figure 3. Experimental Results for Microchip Facility