MASSACHUSETTS INSTITUTE of TECHNOLOGY
LIBRARIES

M.I.T. LIBRARIES - DEWEY

Dewey

HD28
.M414
no.
3729-
94

17

Bottleneck Resource Allocation in Manufacturing

Anantaram Balakrishnan
Richard L. Francis
Stephen J. Grotzinger

#3729-94-MSA

October 1994

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

DEC 01 1994

LIBRARIES

Abstract

Many resource-allocation problems in manufacturing and service operations require selecting integer-valued levels for various activities that consume "nondecreasing amounts" of limited resources. System productivity, to be maximized, is limited by the least productive (bottleneck) activity. We first review a basic bisection method that can solve this discrete, monotonic resource-allocation problem even with nonlinear objective and constraints. We then generalize the basic algorithm to solve an enhanced version of the problem containing additional coupling constraints on the allocation decisions. This generalization applies to assembly-release planning (ARP) in a multiproduct assemble-to-forecast environment with part commonality. The ARP problem requires finding feasible amounts of each product to release for assembly in each period using the available parts. The objective is to maximize the minimum difference between the actual and desired service levels over all products and time periods. We also consider extensions of the ARP model incorporating precedence constraints and part substitutability, and show how to modify the bisection method to solve these problems.

1. Introduction

Many resource-allocation problems in manufacturing and service operations require choosing integer-valued levels for various activities that compete for limited, common resources in order to maximize the minimum return or productivity over all the activities. Applications of bottleneck (max-min or min-max) resource-allocation models include line balancing, raw material allocation, maintenance planning, and storage allocation.

These problems have the following general formulation. Let X be an n-vector of integer activity levels x_j. Each activity consumes varying amounts of m resources. For $i = 1, ..., m$, let S_i and $g_i(X)$ denote, respectively, the total availability of resource i and the amount of this resource consumed by the activity levels in X. If $f_j(x_j)$ denotes the "productivity" of activity j, the system's productivity f(X) is limited by the least productive or "bottleneck" activity, i.e., $f(X) = \min \{f_j(x_j): j = 1, ..., n\}$. We must select activity levels that maximize system productivity while satisfying resource constraints, and lower and upper bounds, $B = (B_j)$ and $C = (C_j)$. This *discrete, bottleneck resource-allocation problem* has the following nonlinear, integer programming formulation:

[P] $$\text{maximize } f(X) = \min_{j=1, ..., n} f_j(x_j) \tag{1}$$

subject to

$$g_i(X) \leq S_i \qquad \text{for all } i = 1, ..., m, \tag{2}$$

$$B \leq X \leq C\text{, and} \tag{3a}$$

$$x_j \text{ integer} \qquad \text{for all } j = 1, ..., n. \tag{3b}$$

We assume that as the activity levels increase the amount of each resource consumed does not decrease, i.e., for $i = 1, ..., m$, the resource-usage function $g_i(X)$ satisfies the following monotonicity (nondecreasing) property: given any two integral vectors X' and X" satisfying the upper and lower bounds (3a), $g_i(X') \leq g_i(X'')$ if $X' \leq X''$. Most practical resource-allocation applications satisfy this assumption. Nondecreasing functions are quite general; they can model economies of scale in the resource constraints, including fixed charges (e.g., fixed cost, setup time, and so on). The productivity functions $f_j(x_j)$ need not be monotonic. We only require $f_j(x_j)$ to be real-valued and defined for all integer activity levels in the closed interval $[B_j, C_j]$. We assume that the problem is feasible, i.e., $X = B$ satisfies constraints (2).

Past research on bottleneck resource allocation (see, for instance, Luss [1992] and Ibaraki and Katoh [1988] for reviews of this literature) has addressed continuous-variable versions of the problem and/or models with special structure (e.g., a single resource or linear functions). For the discrete allocation problem with one linear resource constraint (i.e., a knapsack constraint), Jacobsen [1971] proposed a marginal allocation approach. Porteus and Yormark [1972] developed an improved bisection search algorithm to solve this special case, and Brown [1979] described efficient solution procedures for knapsack-sharing problems with piecewise linear and nonlinear productivity functions.

For problems with multiple resources, Tang [1988] developed an $O(m\ n^2)$ algorithm for a linear model in which the resource-usage functions $g_i(X)$ are all linear (with nonnegative variable coefficients), and the productivity $f_j(x_j)$ of each activity is a strictly increasing linear function. The method finds an optimal integer solution by solving a sequence of relaxed problems without integer restrictions. Tang described various manufacturing applications of this model, including a storage space-allocation problem (also called the reel-allocation model, see Ahmadi, Grotzinger and Johnson [1988]). All of these applications satisfy the monotonicity (nondecreasing) assumption. Recently, Francis and Horak [1993] described a bisection search method to solve the storage space-allocation problem. The method is robust and easy to implement, and although its computational complexity is data-dependent, the method was quicker than Tang's algorithm in computational tests.

This paper, motivated by some resource-allocation decisions in electronics manufacturing and assembly, explores extensions of the bisection approach to address these decision problems. One of these applications is a cleanroom sampling problem described and formulated by Grotzinger and Cooper [1992] as a bottleneck resource-allocation model with a nonlinear objective function and a monotonic resource constraint. In this application, the activity levels are the sample sizes for n samples at n specified locations of a cleanroom. These samples are required for an inspection plan to certify the cleanroom. The cost of sampling varies by location, and increases with the number of samples. The cleanroom has a fixed budget for sampling. To meet the federal standard for certification as a particular class cleanroom, the sample average contaminant concentration at each location must be less than a prespecified class limit. Since the variance of the sample average at any location varies inversely with the size of the sample at that location, a larger sample size will increase the probability that the sample average at that location will be less than the class limit. However, the limited sampling budget introduces tradeoffs in allocating samples to various locations as we attempt to increase the overall likelihood that the cleanroom will be certified, i.e., the likelihood that the sample average will be below the class limit at all locations. To develop an effective inspection plan, Grotzinger and Cooper proposed a bottleneck model to select sample sizes that minimize the maximum variance of the sample averages over all the locations subject to a linear financial resource (budget) constraint. They developed a method to optimally solve the continuous relaxation of this problem (i.e., permitting fractional sample sizes), and proposed heuristic adjustments to the fractional solution to obtain an integer solution.

In Section 2 we review the basic bisection algorithm to solve discrete, monotonic (nonlinear) resource-allocation problems such as the cleanroom sampling model even with nonlinear (but nondecreasing) sampling cost functions that reflect economies of scale. In Section 3, we describe a multiperiod assembly-release planning (ARP) problem with serviceability constraints introduced by Grotzinger and O'Connor [1993] for an assemble-to-forecast environment with part commonality. This problem can be formulated as a monotonic resource-allocation model with additional coupling constraints on the allocation decisions. We discuss how to adapt the bisection approach to solve this model. In Sections 4 and 5 we consider more general ARP models with precedence constraints and parts substitutability. In Section 4, we first motivate ARP models with general precedence constraints governing the allocation of resources; these precedence constraints might stem from the interaction of production planning decisions across products and time periods. We then describe how to modify the bisection approach to solve this new class of problems. In Section 5, we consider assembly-release decisions when part substitutions are permitted, i.e., if we run out of a particular part, we can still produce products that normally use this part by using an alternate part from a specified set of substitutes. We further extend the bisection algorithm to incorporate substitutability of parts. To our knowledge, multiperiod ARP problems with precedence constraints and substitutable parts are unstudied. Our concluding discussion in Section 6 includes a broad interpretation of the various extensions to the monotonic resource-allocation problem [P] that this paper has addressed.

2. Bisection Method for Discrete, Bottleneck Resource Allocation

The bisection method for discrete, bottleneck resource allocation, although originally proposed for problems with linear productivity functions and resource constraints, also applies to the nonlinear, monotonic version [P] of the problem. However, before applying the method, we must replace any productivity functions that are not nondecreasing with equivalent nondecreasing functions. In this section, we first describe the transformation, and then present the bisection algorithm.

Suppose a given productivity function $f_j(x_j)$ is nonmonotonic or decreasing. We transform it into an "equivalent" nondecreasing function $f_j'(x_j)$ by setting

$$f_j'(x_j) \equiv \max \{f_j(y): y = B_j, B_j+1, \ldots, x\} \text{ for all } x = B_j, B_j+1, \ldots, C_j. \tag{4}$$

This transformation, proposed by Brown [1991] for bottleneck problems with linear knapsack constraints, is also valid for problems with nonlinear, nondecreasing resource-usage functions. To confirm its validity, first note that $f_j'(\cdot)$ and $f_j(\cdot)$ have the same maximum values, and the *smallest* value of x_j that maximizes $f_j(x_j)$ is also the smallest value maximizing $f_j'(x_j)$, and vice versa. Using this property, we can show that, if the resource-usage functions $g_i(X)$ are nondecreasing, then the optimal objective function value of the transformed problem [P], with $f_j'(\cdot)$ instead of $f_j(\cdot)$, equals the optimal value of the original problem. Moreover, the smallest optimal solution X* to the transformed problem is also optimal for the original problem. (Since the objective has the max-min form and the resource-usage functions are nondecreasing, problem [P] has an optimal solution X* that is component-wise less than or equal to all other alternate optimal solutions.) Our bisection method finds this smallest optimal solution. For convenience, we will assume in all of our subsequent discussions that the productivity functions are nondecreasing (i.e., the transformation (4) has been applied, if necessary).

We refer to any vector X satisfying constraints (2) and (3) as a feasible solution. We seek an *optimal feasible solution* (OFS) X*, i.e., a feasible solution that maximizes f(X). Let $z^* = f(X^*)$. Since the productivity functions are nondecreasing, $f(B) \leq f(X) \leq z^* \leq f(C)$ for any feasible solution X. Given any value $z \in [f(B), f(C)]$, define $y_j(z)$ by

$$y_j(z) \equiv \min \{x_j: z \leq f_j(x_j), x_j = B_j, B_j+1, \ldots, C_j\} \quad \text{for all } j = 1, \ldots, n. \tag{5}$$

Thus, $y_j(z)$ is the smallest value of the jth activity level x_j needed to ensure that the productivity of this activity equals or exceeds z. When the inverse of the function $f_j(\cdot)$ exists (for instance, if $f_j(\cdot)$ is linear), then

$$y_j(z) = \lceil f_j^{-1}(z) \rceil \text{ for all } f_j(B_j) \leq z \leq f_j(C_j),$$

where $\lceil u \rceil$ is the smallest integer no less than u. Otherwise, we can compute $y_j(z)$ using bisection search over the domain of $f_j(\cdot)$.

Given any positive number ε, we call a feasible solution X an ε-OFS if its objective value is within ε of the maximum value, i.e., if $z^* - \varepsilon \leq f(X) \leq z^*$. For the linear model, Francis and Horak [1993] showed that, given a value $z \in [f(B), f(C)]$, the following three conditions are equivalent: (i) $Y(z) = (y_j(z))$ is a feasible solution; (ii) $Y(z)$ satisfies constraints (2); and (iii) $z \leq z^*$. This result is the basis for their bisection algorithm. Both the result and the algorithm extend directly to discrete resource-allocation problems with nonlinear, nondecreasing resource-usage and productivity functions.

The underlying idea of the algorithm is to iteratively select a trial objective function value z at the midpoint of the current search interval, and check if the problem has a feasible solution X with $f(X) \geq z$. We check feasibility by first determining the smallest possible value $y_j(z)$ of x_j needed to ensure that the overall objective function value $f(X)$ equals or exceeds z. We then verify if the solution $Y(z) = (y_j(z))$ satisfies the resource constraints (2). If it does, then $Y(z)$ is a feasible solution to the problem, and so the trial value z is a lower bound on the optimal value z^* (actually, $f(Y(z)) \geq z$ is a better valid lower bound); otherwise, z overestimates z^*. We initialize the algorithm with lower and upper bounds, $LB = f(B)$ and $UB = f(C)$, on z^*. We iteratively select $z = (LB + UB)/2$, determine $Y(z)$, and update LB to $f(Y(z))$ or UB to z depending on whether or not $Y(z)$ satisfies (2). Iterations repeat until $UB - LB \leq \varepsilon$. At termination, the vector $Y(LB)$ is an ε-OFS.

One potential disadvantage of this approach is that its effort is data-dependent. The algorithm requires up to $\log_2((f(C) - f(B))/\varepsilon)$ iterations, each requiring n evaluations of the $y_j(z)$ values. We view this disadvantage as principally theoretical in nature; typically, the data is not so big as to make bisection impractical. The polynomial-order algorithms proposed in the literature assume special structure (e.g., linear resource constraints), whereas the bisection approach is quite general.

3. The Assembly-Release Planning Problem

In this section, we review a production planning problem in an assemble-to-forecast environment with part commonality. Grotzinger and O'Connor [1993] described and modeled this problem as a continuous, bottleneck resource-allocation problem. In this paper, we first consider the integer version of this model. The problem is formulated using variables for cumulative production quantities instead of monthly (or weekly) production quantities. The resource constraints, when expressed in terms of these cumulative production variables, satisfy the nondecreasing property. The model contains additional coupling constraints to ensure that the solution corresponds to a feasible production plan. We extend the basic bisection algorithm to incorporate these constraints. In Sections 4 and 5 we consider enhancements of the assembly-release planning model to account for precedence constraints and parts substitutability.

Assemble-to-forecast (ATF) or make-to-stock refers to systems that produce goods based on demand forecasts rather than actual orders (see, for instance, Baker [1993]) because the procurement and production (fabrication or assembly) lead times are longer than the customer lead time (i.e., the time between placing an order and shipping the product to the customer). Electronics assembly facilities producing personal computers provide one example of an ATF environment. In this context, the procurement and assembly lead times could range from several months (to procure certain specialized integrated circuits) to a few weeks whereas customer lead times might be only a few days. The personal computer assembly example also illustrates another feature, part commonality, found in many ATF environments. Commonality refers to using the same part (or component) for several different products. For instance, several personal computer models might require the same microprocessor. Increasing part commonality provides many benefits including decreasing ordering and inventory administration expenses, and reducing safety stocks of parts due to risk pooling (see, for instance, Bagchi and Gutierrez [1992], Baker, Magazine and Nuttle [1986], Collier [1982], Gerchak and Henig [1989], Gerchak, Magazine, and Gamble [1988], and Grotzinger et al. [1993]).

In ATF facilities that produce multiple products with part commonality, decisions regarding how many parts to procure in each period, how to allocate these parts to different products, and how many "kits" (component sets for each product) to release for assembly in each period must be based on estimates of probable future demands for each product, service level requirements, and inventory considerations. To address these decisions, Grotzinger and O'Connor [1993] developed a general nonlinear optimization model, and formulated a linear program for a "feasibility subproblem" incorporating the parts allocation and assembly-release decisions. We focus on the integer version of this feasibility problem, which we refer to as the *assembly-release planning (ARP)* problem. We next describe this problem.

Consider a facility that assembles to forecast n (finished) products using m part types (or parts). Suppose the procurement lead time is (p+1) time periods for all parts, and the assembly lead time is L periods for all products. For simplicity, the same procurement lead time for all parts, and the same assembly lead time for all products, is assumed. Item-dependent lead times are easy to incorporate. The parts ordered during the past p+1 periods will arrive at the beginning of the current period ($t = 0$) and subsequently for periods, $t = 1, \ldots, p$. We are concerned with how to allocate these parts (including those currently on-hand) to various products, and how many kits to release for assembly during each of the next p periods. Figure 1 shows a timeline depicting various events: procurement decisions, part arrivals, release of parts for assembly, and assembly completion.

We assume that the probability distribution of demand for every product during each of the next (L+p) periods is given. These distributions are revised each period based upon current market conditions and trends, and so the assembly-release plans must be updated each period. Since the assembly lead time is L time units, the number of kits for each product released for assembly at time t, for t = 0, ..., p, must be chosen to service the probable demand for that product in period (L+t). The planner's objective is to satisfy the demand for each product in every period L, ..., L+p with a given probability or *target service level*. If the available inventory of a finished product in any period is less than the demand during that period, we assume that the unsatisfied demand is backlogged.

We define the *cumulative availability* of a product at time t > 0 as its initial inventory at time t = 0 plus the total output (i.e., completed assemblies) from period 0 to period t. The *actual* service level for a product at time t is the probability that the cumulative availability of that product at time t equals or exceeds its cumulative demand up to and including time t. The *target* service level is the desired minimum value for this probability; this target might vary with product and time period. We refer to the difference between the actual and target service levels as the *service-level difference*. Grotzinger and O'Connor [1993] developed a linear program to determine if the target service level can be achieved for all products and time periods using the available parts, and if not to determine the minimum possible deviation from target. The model maximizes the minimum service-level difference over all products and periods. The integer version of this model, the *Assembly-Release Planning (ARP)* problem, has the following formal definition.

Given the current parts inventory, the quantities on order and to be received during the next p periods, the bill of materials for each product, and the demand distributions for the next (L+p) periods, the ARP problem seeks integer assembly-release quantities to *maximize the minimum service-level difference over all products and time periods* subject to parts availability.

If the optimal objective function value is nonnegative, then the currently available parts plus the anticipated receipts (based on prior procurement decisions) are adequate to provide the desired service levels for all products in periods L through (L+p); the ARP solution specifies a set of assembly-release quantities that meets the targets. Otherwise (if the optimal value is negative), the target service level cannot be attained for one or more products. In this case, the assembly-release quantities specified by the ARP solution ensure that the actual service level is as close to target as possible. Grotzinger and O'Connor [1993] discuss how to decide future parts procurement quantities based on the ARP solution.

Problem formulation:

For $j = 1, ..., n$, and $t = 0, ..., p$, we define nonnegative, integer variables q_{jt} representing the number of units of product j to be released for assembly in period t. For every part $i = 1, ..., m$, let a_{ij} denote the number of units of part i used per unit of product j. For $t = 0, ..., p$, let S_{it} denote the (known) cumulative supply of part i at time t, i.e., S_{it} is the sum of the initial on-hand inventory of part i at time 0 plus the total anticipated receipts of this part during periods 0 through t.

Let $A_{j,L-1}$ denote the (known) cumulative availability of (finished) product j in period (L–1), which is equal to product j's current inventory (at time $t = 0$, including assemblies just completed) plus the total quantity released for assembly during the past (L–1) periods (i.e., from period (–L+1) to period –1 inclusive). For $t = 0, ..., p$, the total amount of product j that we decide to release in periods 0 through t (and will complete in periods L through (L+t)) is $q_{j0}+ ... + q_{jt}$. Adding $A_{j,L-1}$ to this amount gives the cumulative availability of product j in period (L+t). Let $D_{j,L+t}$ be the cumulative demand (a random variable) for product j from period 0 to period (L+t). We define $f_{jt}(q_{j0}+ ...+q_{jt})$ as the service level difference for product j at time (L+t), i.e., for all $j = 1, ..., n$, and $t = 0, ..., p$,

$f_{jt}(q_{j0}+ ... +q_{jt}) = \text{Prob}\{ A_{j,L-1} + q_{j0} + ... + q_{jt} \geq D_{j,L+t}\}$ — Target service level for product j in period (L+t).

The minimum service-level difference f(Q) is:

$$f(Q) = \min\{f_{jt}(q_{j0}+ ...+q_{jt}): j = 1, ..., n, \text{ and } t = 0, ..., p\}. \quad (6)$$

Note that each $f_{jt}(\cdot)$, and hence f(Q), is a nondecreasing function of the release quantities, i.e., $Q \leq Q'$ implies $f(Q) \leq f(Q')$. Also, for a given vector Q, every service-level difference function $f_{jt}(q_{j0}+ ... +q_{jt})$ is nonnegative if and only if $f(Q) \geq 0$.

The ARP problem is formulated as follows:

[ARP] maximize f(Q)

subject to:

$$\sum_{j=1}^{n} a_{ij} (q_{j0} + \dots + q_{jt}) \leq S_{it} \quad \text{for all } i = 1, \dots, m, \text{ and } t = 0, \dots, p, \qquad (7)$$

$$q_{jt} \geq 0 \text{ and integer} \quad \text{for all } j = 1, \dots, n, \text{ and } t = 0, \dots, p. \qquad (8)$$

Constraints (7) ensure that the total amount of each part i used in all products during periods 0 through t does not exceed its cumulative supply S_{it}. We wish to find nonnegative, integer release quantities q_{jt} to maximize the minimum service-level difference over all products and all periods.

Although formulation [ARP] has linear parts-availability constraints (7), our bisection solution method (described in the next section) also applies to problems with nonlinear resource constraints of the form $G(Q) \leq S$, assuming that the vector function $G(\cdot)$ is *tier-nondecreasing* (TND), i.e., if $Q' = (q'_{jt})$ and $Q'' = (q''_{jt})$ are any two feasible solutions that satisfy

$$q'_{j0} + q'_{j1} + \dots q'_{jt} \leq q''_{j0} + q''_{j1} + \dots + q''_{jt} \qquad \text{for all } j = 1, \dots, n, \text{ and } t = 0, \dots, p,$$

then $G(Q')$ must be less than or equal to $G(Q'')$. Note that the objective function $f(\cdot)$ defined by (6) is also TND, but this property is not necessary. We can replace any given non-TND productivity function with an equivalent TND function, obtained by applying a transformation analogous to equation (4).

Formulation [ARP] uses the service level differences as the productivity functions. However, the enhanced bisection method that we describe in the next section applies to the broader class of problems in which each productivity function $f_{jt}(\cdot)$ is any real-valued and well-defined function (not necessarily TND) of the cumulative availability of product j at time t. As a special case, consider the ARP problem with known, deterministic demands. Suppose, for each product j, we assign a penalty π_j per unit shortfall of cumulative availability relative to cumulative demand; minimizing the maximum penalty over all products and time periods is the objective function. The bisection method can solve this problem, and can also accommodate additional TND resource constraints, such as labor and budget constraints with economies of scale and/or fixed charges. In Section 4, we consider further model extensions.

Solving the ARP problem:

Let Ω denote the set of all feasible solutions Q satisfying constraints (7) and (8), and let $z^* = \max\{f(Q): Q \in \Omega\}$. Set Ω is non-empty (since $Q = 0$ is a feasible ARP solution) and finite, and so z* exists. The bisection method requires an initial search interval for the optimal value; therefore, we first develop upper and lower bounds on z*. Since the function $f(\cdot)$ is monotonic, LB = f(0) is a valid lower bound on z*. To obtain an upper bound, note that for every $Q \in \Omega$,

$$a_{ij}\,(q_{j0} + \dots + q_{jt}) \le S_{it} \qquad \text{for all i, j, and t.}$$

Thus, if we define

$$u_{jt} = \min\{\lfloor S_{it}/a_{ij} \rfloor : \text{i such that } a_{ij} > 0\}, \tag{9}$$

where $\lfloor h \rfloor$ is the largest integer less than or equal to h, then, $q_{j0} + \dots + q_{jt} \le u_{jt}$ for all products j and time periods t. Let $U = (u_{jt})$. Since each function $f_{jt}(\cdot)$ is nondecreasing, UB = f(U) = min $\{f_{jt}(u_{jt})$: j = 1, ..., n and t = 0, ..., p$\}$ is a valid upper bound on z*. Furthermore, $q_{jt} \le u_{jt}$ for all j and t.

To better understand how to adapt the original bisection approach to the ARP model, we reformulate the problem by transforming the variables. Let

$$x_{jt} = q_{j0} + \dots + q_{jt} \qquad \text{for all } j = 1, \dots, n, \text{ and } t = 0, 1, \dots, p, \tag{10}$$

denote the *cumulative release quantity* of product j in periods 0 through t. Replacing the q-variables in formulation [ARP] with the x-variables, and defining

$$f''(X) = \min\{f_{jt}(x_{jt}) : j = 1, \dots, n, \text{ and } t = 0, \dots, p\}$$

we get the following equivalent model:

[ARP'] maximize $f''(X)$

subject to:

$$\sum_{j=1}^{n} a_{ij}\, x_{jt} \le S_{it} \qquad \text{for all } i = 1, \dots, m, \text{ and } t = 0, \dots, p, \tag{11}$$

$$x_{jt} \ge x_{j,t-1} \qquad \text{for all } j = 1, \dots, n, \text{ and } t = 1, \dots, p, \tag{12}$$

$$x_{jt} \le u_{jt} \qquad \text{for all } j = 1, \dots, n, \text{ and } t = 0, 1, \dots, p, \text{ and} \tag{13}$$

$$x_{jt} \ge 0 \text{ and integer} \qquad \text{for all } j = 1, \dots, n, \text{ and } t = 0, 1, \dots, p. \tag{14}$$

The *coupling constraints* (12), specifying that the cumulative quantity of product j released for assembly at time t must equal or exceed its cumulative release quantity at time (t–1), reflect the nonnegativity constraints on the variables q_{jt} in formulation [ARP]. Our solution method does not require the resource constraints (11) to be linear; it can also handle nonlinear resource constraints $G''(X) \le S$ as long as the function $G''(X)$ is nondecreasing for all vectors X that satisfy constraints (12), (13) and (14) (requiring $G''(\cdot)$ to be nondecreasing is equivalent to our previous TND condition on the resource-usage function G(Q)). Luss and Smith [1988] have considered a multiperiod model similar to [ARP'] but with continuous variables, linear resource constraints, and linear productivity functions $f_{jt}(x_{jt})$ that represent the relative deviation of cumulative production from cumulative demand.

Observation 1:

Every feasible solution X to [ARP'] has a corresponding feasible solution Q to [ARP] (with the same objective value) and vice versa.

Proof:

Given a feasible solution Q to [ARP], the solution X obtained using equation (10) is feasible in [ARP'] and has the same objective value. Conversely, if X satisfies the constraints of [ARP'], then we obtain a feasible solution $Q = (q_{jt})$ to [ARP] with the same objective function value by setting $q_{jt} = x_{jt} - x_{j,t-1}$ for all $t = 1, ..., p$, and $q_{j0} = x_{j0}$ for all $j = 1, ..., n$. ♦

The coupling constraints (12) differentiate model [ARP'] from the basic monotonic resource-allocation model [P]. Note that the function $h_{jt}(X) = x_{j,t-1} - x_{jt}$ is not monotonic in X, i.e., $X' \leq X$ does not necessarily imply that $h_{jt}(X') \leq h_{jt}(X)$. Therefore, we cannot include constraints (12) in the set of general resource constraints $G''(X) \leq S$, but need to treat them separately. Let us now explain intuitively how to modify the bisection method described in Section 2 to handle these additional constraints. Recall that, for a given trial value z of the objective function value, the value $y_j(z)$ defined using equation (5) is the smallest value of the resource-allocation variable x_j which ensures that the objective value equals or exceeds z. However, these values might not satisfy the coupling constraints (12) in formulation [ARP]. Therefore, we modify the definition of the y-values as follows. For all $j = 1, 2, ..., n$, and $t = 0, 1, ..., p$, let

$$w_{jt}(z) = \min \{x_{jt}:\ z \leq f_{jt}(x_{jt}),\ x_{jt} = 0, 1, ..., u_{jt}\}. \tag{15}$$

We then successively compute $y_{jt}(z)$, for $t = 1, ..., p$, as:

$$y_{j0}(z) = w_{j0}(z), \text{ and} \tag{16a}$$

$$y_{jt}(z) = \max \{y_{j,t-1}(z),\ w_{jt}(z)\} \qquad \text{if } t \geq 1. \tag{16b}$$

Note that each $y_{jt}(z)$ is a nondecreasing function of z.

Let Ω' denote the set of all vectors X satisfying (12), (13), and (14).

Observation 2:

For a given value $z \in [LB, UB]$, the vector $Y(z)$ obtained using equations (16a) and (16b) is the smallest element of Ω' with an objective value of at least z, i.e., $Y(z) \in \Omega'$, $f''(Y(z)) \geq z$, and $Y(z) \leq Y'$ for any other vector $Y' \in \Omega'$ with $f''(Y') \geq z$.

This observation stems from the monotonicity property of each function $f_{jt}(\cdot)$, and can be proven using a contradiction argument. Note that, instead of using equation (16b), we can equivalently define $y_{jt}(z)$ for $t = 1, \ldots, p$, as

$$y_{jt}(z) = \min \{x_{jt}: z \leq f_{jt}(x_{jt}), x_{jt} = y_{j,t-1}(z), y_{j,t-1}(z)+1, \ldots, u_{jt}\}. \tag{16c}$$

Observation 2 and the monotonicity of the resource-usage functions $G''(X)$ defined by (11) enable us to perform binary search to find an ε-optimal solution to problem [ARP']. We first state this algorithm before justifying its validity. The algorithm contains an embedded search procedure to evaluate $w_{jt}(z)$ (defined in equation (15)). We refer to this procedure as the W-evaluation subroutine.

<u>Bisection algorithm for ARP problem</u>:

Step 0: Initialization

```
β ← 0;          {initial vector of lower bounds on Y(z*)}
γ ← U;          {initial vector of upper bounds on Y(z*)}
LB ← f''(0);    {initial lower bound on z*}
UB ← f''(U);    {initial upper bound on z*}
```

Step 1: Search process

```
REPEAT
  $z \leftarrow (LB + UB)/2$;
  Compute $W(z)$ in $[\beta,\gamma]$;      {call W-evaluation subroutine}
  FOR $j = 1, \ldots, n$,
    Set $y_{j0}(z) \leftarrow w_{j0}(z)$;
    FOR $t = 1, \ldots, p$,
      Set $y_{jt}(z) \leftarrow \max \{y_{j,t-1}(z), w_{jt}(z)\}$;
  IF $Y(z)$ satisfies constraints (11)
    THEN set  $LB \leftarrow f''(Y(z))$ and $\beta \leftarrow Y(z)$;     {$z \leq z^*$ and $Y(z) \leq Y(z^*)$}
    ELSE set  $UB \leftarrow z$ and $\gamma \leftarrow Y(z)$;            {$z > z^*$ and $Y(z^*) \leq Y(z)$}
UNTIL $(UB - LB) \leq \varepsilon$;
```

Step 2: ε-optimal solution

```
FOR $j = 1, ..., n$,
  Set $q_{j0} = y_{j0}(LB)$;
  FOR $t = 1, ..., p$,
       Set $q_{jt} = y_{jt}(LB) - y_{j,t-1}(LB)$;

$Q = (q_{jt})$ is $\varepsilon$-optimal.
```

<u>W-evaluation subroutine</u>: To compute $w_{jt}(z)$ in $[\beta_{jt}, \gamma_{jt}]$ for all j and t.

```
FOR $j = 1, ..., n$,
 FOR $t = 0, ..., p$,
  Initialize $\lambda \leftarrow \beta_{jt}$ and $\upsilon \leftarrow \gamma_{jt}$;        $\{\beta_{jt} \leq w_{jt}(z) \leq \gamma_{jt}\}$
  IF $z \leq f_{jt}(\lambda)$
    THEN set $\upsilon \leftarrow \lambda$;
    ELSE
      REPEAT
        Set $u \leftarrow \lfloor(\lambda + \upsilon)/2\rfloor$;
        IF $z \leq f_{jt}(u)$
          THEN set $\upsilon \leftarrow u$;            $\{w_{jt}(z) \leq u\}$
          ELSE set $\lambda \leftarrow u+1$;          $\{u < w_{jt}(z)$ implies $u+1 \leq w_{jt}(z)\}$
      UNTIL $\lambda \geq \upsilon$;
  Set $w_{jt}(z) \leftarrow \upsilon$;
```

Starting with LB = $f^{*}(0)$ and UB = $f^{*}(U)$, the algorithm computes $y_{jt}(z)$, with $z = (LB+UB)/2$, for all j and t; let $Y(z) = (y_{jt}(z))$. If these values satisfy the resource capacity constraints (11), we update $LB \leftarrow f^{*}(Y(z))$; otherwise, we set $UB \leftarrow z$. We also update the upper and lower bounds on the optimal values of x_{jt} (thus narrowing the search interval for $w_{jt}(z)$). We repeat this procedure until $(UB - LB) \leq \varepsilon$. At termination, the solution Y(LB) is ε-optimal for problem [ARP']. We recover the ε-optimal solution to the original problem [ARP] by setting the release quantity in each period t equal to the difference between the cumulative release quantities (given by the solution Y(LB)) in periods t and (t–1).

To compute $y_{jt}(z)$, we first determine the smallest value $w_{jt}(z)$ of the cumulative release quantity for product j at time t to achieve the objective function value z, and then apply equations (16a) and (16b). The W-evaluation subroutine employs binary search in the interval $[\beta_{jt}, \gamma_{jt}]$ to determine $w_{jt}(z)$. This subroutine is not necessary if we can directly evaluate the inverse of the functions $f_{jt}(\cdot)$. The parameters β_{jt} and γ_{jt} respectively denote the current lower and upper bounds on the optimal value of the cumulative release quantity x_{jt}. Initially, $\beta_{jt} = 0$ and $\gamma_{jt} = u_{jt}$. If the solution Y(z) at a particular iteration satisfies the resource constraints (11), then Y(z) is a feasible solution to [ARP']. Therefore, $z^* \geq z$, and since the productivity functions $f_{jt}(\cdot)$ are nondecreasing, the problem must have an optimal solution X^* satisfying $X^* \geq Y(z)$. Hence, we increase the lower bound β to Y(z). Otherwise (if Y(z) does not satisfy (11)), $X^* \leq Y(z)$ and we reduce the upper bound γ to Y(z). Notice that, with this initialization and updating scheme, the lower and upper bounds satisfy the following condition at every iteration:

$$\beta_{j,t-1} \leq \beta_{jt} \text{ and } \gamma_{jt} \leq \gamma_{j,t-1} \qquad \text{for all } j = 1, ..., n, \text{ and } t = 1, ..., p.$$

We emphasize that, since the release quantities must be integral, the W-evaluation subroutine finds the exact value of $w_{jt}(z)$.

The validity of the bisection method rests upon the following result:

Proposition 3:

For the ARP problem with nondecreasing resource-usage functions, a trial objective function value z is a lower bound on the optimal value z^* if and only if Y(z), defined by equations (16a) and (16b), satisfies the resource constraints, i.e., iff $G''(Y(z)) \leq S$.

Proof:

Let X^* be any optimal solution to [ARP'] with optimal value $z^* = f''(X^*)$. The solution X^* is nonnegative, integral, and satisfies the resource capacity constraints (11) as well as the coupling constraints (12) and upper bounds (13). Suppose the trial value z is less than or equal to z^*. By observation 2, Y(z) is the smallest nonnegative, integral vector X satisfying constraints (12), (13), and $f''(X) \geq z$; therefore, $Y(z) \leq X^*$. Since the resource-usage function $G''(\cdot)$ is nondecreasing and X^* satisfies the resource constraints, $Y(z) \leq X^*$ implies that $G''(Y(z)) \leq G''(X^*) \leq S$, i.e., the solution Y(z) satisfies the resource constraints. Conversely, suppose for a given value of z, Y(z) satisfies the resource constraints (11). By definition, Y(z) also satisfies constraints (12), (13), and (14). Since $z \leq f''(Y(z))$ and Y(z) is a feasible solution to [ARP'], $z^* \geq f''(Y(z)) \geq z$. ♦

Computational effort:

To find an ε-optimal solution, the bisection algorithm requires at most $\log_2((f''(U)-f''(0))/\varepsilon)$ "major" iterations (i.e., number of repeats in Step 1). Since each service level difference is the difference of two probabilities, $-2 \leq f''(0) - f''(U) \leq 2$. Therefore, if $\varepsilon = 2^{-20} = 1/1{,}048{,}576$, no more than 21 iterations are required. As problem size increases, the method will likely be limited by memory rather than speed.

At each iteration, the method calls the W-evaluation subroutine n(p+1) times. Each evaluation requires at most $\log_2(\gamma_{jt}-\beta_{jt})$ iterations within the subroutine. We can reduce the search effort in the W-evaluation subroutine by modifying the overall algorithm as follows. At each iteration in Step 1, instead of first evaluating all the $w_{jt}(z)$ values before computing the $y_{jt}(z)$ values, we first evaluate $w_{j0}(z)$, and set $y_{j0}(z) = w_{j0}(z)$. Subsequently, for t = 1, ..., p in sequence, we specify the lower bound of the search interval for the W-evaluation subroutine as $\max\{y_{j,t-1}(z), \beta_{jt}\}$. Observe that the values returned by the W-evaluation subroutine now directly correspond to the $y_{jt}(z)$ values (i.e., they are not the $w_{jt}(z)$ values defined in equation (15)). Since the search intervals are shorter, this modified method has lower computational effort. Notice that, unlike some linear programming-based methods, the bisection method does not introduce cumulative floating point errors. Finally, if we are not interested in the exact value of z* but only want to know if $z^* \geq 0$ (i.e., whether we can achieve the target service levels for all products in all periods), then we can terminate the procedure as soon as LB becomes nonnegative.

The next two sections consider two types of generalizations of the ARP model, namely, precedence constraints and part substitutability, that the bisection method can solve.

4. Model Extensions I: Precedence-constrained resource allocation

The ARP problem is a constrained version of the basic resource-allocation model [P], containing the additional coupling constraints (12) to ensure that the cumulative release in a period equals or exceeds the cumulative release in the previous period. Let us now consider a broader class of "precedence" constraints that generalize the coupling constraints. Our initial development leads to our next generalization, (20) below, which is applicable to multistage production settings.

Suppose the production context requires that the cumulative release quantity of a product j at time t must equal or exceed the cumulative release quantity of product j' at time t', i.e., we must add constraints of the following type to formulation [ARP'] for a prespecified subset IP of ordered product-time *index pairs* <(j',t'),(j,t)>:

$$x_{jt} \geq x_{j't'} \qquad \text{for all } <(j',t'),(j,t)> \in \mathbb{P}. \qquad (17)$$

If $<(j',t'), (j,t)> \in$ IP, we say that (j',t') is a *predecessor* of (j,t). We will refer to discrete resource-allocation problems with precedence constraints (17) instead of the coupling constraints (12) as *precedence-constrained resource-allocation* problems.

Let us define a precedence graph PG to encode the required relationships specified in the set IP. This graph has one node for each product-time index (j,t), and contains a directed edge from node (j',t') to node (j,t) for every pair $<(j',t'), (j,t)> \in$ IP. Suppose PG contains a directed circuit DC. To satisfy all the precedence constraints (17) implied by the circuit we must necessarily set all the variables corresponding to the circuit's nodes to the same value. Thus, if N(DC) denotes the set of nodes in the circuit DC, we can reduce the size of the problem by replacing x_{jt} for every $(j,t) \in$ N(DC) by a single variable, say, x_{DC} in the resource constraints (11); in the objective function, we eliminate $f_{jt}(x_{jt})$ for all $(j,t) \in$ N(DC) and instead introduce $f_{DC}(x_{DC}) = \min \{f_{jt}(x_{DC})$: all $(j,t) \in$ N(DC)$\}$. Note that this substitution preserves the monotonicity of the resource-usage and productivity functions. In the precedence graph, the substitution corresponds to contracting all the nodes (j,t) in the circuit into a single node DC; all arcs originally incident from nodes not in N(DC) to any node in N(DC) are incident to node DC in the transformed graph. By successively eliminating all circuits in this manner, we obtain a circuit-free precedence graph PG. We do not require this graph to be connected.

Since PG contains no directed circuits, we can sequence its nodes such that for each node (j,t) all its predecessors occur before it in the sequence. (This node sequencing operation, analogous to the node indexing scheme for PERT/CPM networks, requires $O(|E|^2)$ computations, where $|E|$ is the number of edges in PG; see Lawler [1976].) We will refer to such a sequence $\mathbf{S}$ as a *y-evaluation sequence*. We can easily modify the bisection method to accommodate these precedence constraints as follows: in Step 1, after computing W(z) in $[\beta,\gamma]$ for all $j = 1, ..., n$, and $t = 0, ..., p$, we consider the indices (j,t) in the y-evaluation sequence $\mathbf{S}$, and set:

$$y_{jt}(z) \;=\; \max\left[w_{jt}(z), \max\{y_{j't'}(z)\text{: all } (j',t') \text{ such that } <(j',t'),(j,t)> \in \text{IP}\}\right]. \qquad (18)$$

The remaining steps of the algorithm are unaltered. Again, the validity of this method stems from the nondecreasing property of the resource-usage and productivity functions.

For our <u>original</u> ARP problem (described in Section 3), the set IP consists of index pairs $<(j,t-1), (j,t)>$ for all $j = 1, ..., n$, and $t = 1, ..., p$. Observe that the precedence graph PG for this special case of precedence-constrained resource allocation consists of n "line" subgraphs, one corresponding to each product; the adjacent nodes on each line j correspond to release quantities for product j in consecutive time periods. This precedence graph is circuit-free, and does not require any further reduction. The natural sequence ((j,t) for j = 1, ...,n, t = 0, ..., p) satisfies the required precedence property; the original and enhanced bisection methods are identical if we use this sequence as the y-evaluation sequence $\mathbf{S}$.

Let us now consider how to adapt the bisection method for discrete resource-allocation problems with precedence constraints that are more general than constraints (17). Suppose product j is an accessory board and product j' is the mother board for a personal computer. The production policy might specify that the cumulative release of product j at time t must equal or exceed a certain (nonnegative) proportion $b_{jt,j't'}$ of the cumulative release of product j' at time t' (t and t' might differ if the two board types have different assembly lead times). The parameter $b_{jt,j't'}$ might be based upon the relative yields of the two board types, spare parts requirements, and so on. Modeling this policy requires adding precedence constraints of the form:

$$x_{jt} \geq b_{jt,j't'}\, x_{j't'} \qquad \text{for all } <(j,t), (j',t')> \in \mathbb{P}. \tag{19}$$

A more general version of these forcing constraints arises in multi-stage assembly settings where product j is a subassembly that is used in final assemblies $j_1, j_2, ..., j_r$. If the assembly lead time for subassembly j is k time units, $b_{jj'}$ denotes the number of units of product j needed per unit of final assembly j', and I_{j0} represents the initial inventory of product j, then we require:

$$x_{jt} + I_{j0} \geq b_{jj_1}\, x_{j_1,t+k} + ... + b_{jj_r}\, x_{j_r,t+k}.$$

This constraint specifies that the cumulative release of subassembly j at time t plus the opening inventory must equal or exceed the cumulative requirements of this subassembly to meet the final assembly release quantities. In general, the precedence constraints might contain a (nonnegative) weighted combination of several variables of the form:

$$x_{jt} \geq e_{jt} + b_{jt,j_1t_1}\, x_{j_1t_1} + b_{jt,j_2t_2}\, x_{j_2t_2} \cdots + b_{jt,j_rt_r}\, x_{j_rt_r}, \tag{20}$$

where the parameter e_{jt} is unrestricted in sign, but the coefficients $b_{jt,j't'}$ are all nonnegative. The variable x_{jt} might be governed by several such precedence constraints, each with different right-hand sides. Let $\Pi(j,t)$ denote the subset of precedence constraints (20) that contain the variable x_{jt} in the left-hand side, and let P_{jt} be the set of all indices (j',t') such that $x_{j't'}$ occurs with positive coefficient $b_{jt,j't'}$ in the right-hand side of at least one constraint of $\Pi(j,t)$. We refer to (j',t') as a *predecessor* of (j,t). The precedence graph PG for this enhanced problem has one node corresponding to each product-time index (j,t); node (j,t) has incident arcs <(j',t'),(j,t)> for every $(j',t') \in P_{jt}$.

To apply the bisection method for resource-allocation problems with the general precedence constraints (19) or (20), we will assume that the corresponding precedence graph PG contains no directed cycles. (With constraints (19) or (20), we cannot always make the precedence graph acyclic using the successive substitution procedure we described for constraints (17).) The multi-stage assembly example that we described earlier satisfies this assumption. Since PG is assumed to be acyclic, we can sequence the product-time indices such that the index (j,t) occurs after the indices of all its predecessors. We then determine the values $y_{jt}(z)$ in this sequence $\mathbf{S}$ using a generalized version of equation (18). This version computes $y_{jt}(z)$ as the maximum of $w_{jt}(z)$ and the highest right-hand side value over all the precedence constraints in $\Pi(j,t)$. By considering the variables in the correct sequence, we ensure that the values of $y_{j't'}(z)$ for all $(j',t') \in P_{jt}$ are available before we compute the value of $y_{jt}(z)$. We can extend the same argument to handle even non-linear precedence constraints $x_{jt} \geq h_{jt}(Y_{jt})$, where $Y_{jt} = \{y_{j't'}(z)\text{: all } (j',t') \in P_{jt}\}$ as long as the functions $h_{jt}(\cdot)$ are non-decreasing and the corresponding precedence graph is acyclic.

5. Model Extensions II: Part substitutability

In Section 4, we showed how the bisection method extends to problems with general precedence constraints. We now discuss modifications to incorporate another complicating feature of some electronics assembly environments, namely, part substitutability. By parts substitutability we mean using a designated alternative part (or parts) if the preferred part for a product is not available. Allowing substitution yields risk-pooling benefits, and contributes to achieving desired service levels; however, it may also increase product costs and reduce product quality and reliability. Products with substitute parts, although functional, may not be tested as thoroughly as the nominal design. Thus, it is desirable to allow no more substitution than is needed to achieve service levels. Our approach incorporates this preference, i.e., it exploits the risk-pooling benefit of substitutability while maintaining cost, quality and reliability objectives. For any desired (and achievable) service level, we minimize the total amount (or cost) of substitution necessary by solving a minimum cost network flow problem.

Suppose we can replace part i with some other part without affecting the functionality of the products. Let R(i) be the index set of parts that can replace part i, and let RB(i) be the set of parts that can be replaced by part i, i.e., $RB(i) = \{i''\text{: } i \in R(i'')\}$. We assume one-for-one substitution, i.e., one unit of a part $i' \in R(i)$ replaces one unit of part i. Part substitutability, like commonality, provides risk pooling benefits. If part i is not available in stock, we can still assemble products that require this part by using an alternate part from R(i). Klein, Luss, and Rothblum [1993] developed an efficient specialized algorithm to solve linear, continuous resource-allocation problems with substitutions.

Let us first formulate the assembly-release planning problem with part substitutability, which we abbreviate as the APS problem. As before, let x_{jt} denote the cumulative release quantity of product j in periods 0 through t, for j = 1, ..., n, and t = 0, ..., p. For every part i, let b_{i0} be the initial availability of part i (i.e., opening inventory of part i in period 0 plus the quantity received in period 0), and for t = 1, ..., p, let b_{it} be the number of units of part i received in period t. In terms of our previous notation, $b_{i0} = S_{i0}$, and $b_{it} = S_{it} - S_{i,t-1}$ for t = 1, ..., p, where S_{it} is the cumulative availability of part i in period t. Note that $S_{it} = \sum_{t'=0}^{t} b_{it'}$ for all i and t. As before, $u_{jt} = \min\{\lfloor S_{it}/a_{ij} \rfloor$: i such that $a_{ij} > 0\}$ is a valid upper bound on x_{jt}. For simplicity, we will consider the simple precedence constraints (12) specifying that x_{jt} must equal or exceed $x_{j,t-1}$ for every product j and periods t = 1, ..., p. The model and the modified version of the bisection method that we describe later can also incorporate the more general precedence constraints (19) or (20).

To model part substitutability, we introduce new "substitution" variables $v_{ii't}$, for all i = 1, ..., m, i' ∈ RB(i), and t = 0, ..., p, denoting the **number of units of part i used to replace part i' during period t**. The APS problem then has the following nonlinear, integer programming formulation.

[APS] maximize $f''(X) = \min\{f_{jt}(x_{jt})\text{: } j = 1, \dots, n, \text{ and } t = 0, \dots, p\}$ (21)

subject to:

$$\sum_{j=1}^{n} a_{ij}\, x_{jt} + \sum_{t'=0}^{t} \sum_{i' \in RB(i)} v_{ii't'} \le \sum_{t'=0}^{t} b_{it'} + \sum_{t'=0}^{t} \sum_{i'' \in R(i)} v_{i''it'}$$

for all i = 1, ..., m, and t = 0,1, ...,p, (22)

$x_{jt} \ge x_{j,t-1}$ for all j = 1, ...,n, and t = 1, ...,p, (23)

$x_{jt} \le u_{jt}$ for all j = 1, ...,n, and t = 0,1, ...,p, (24)

$x_{jt} \ge 0$ and integer for all j = 1, ..., n, and t = 0,1, ...,p, and (25)

$v_{ii't} \ge 0$ and integer for all i = 1, ..., m, i' ∈ RB(i), t = 0,1, ...,p. (26)

Formulation [APS] is similar to our previous formulation [ARP'] except that constraints (22) now incorporate parts substitutability, and constraints (26) impose nonnegativity and integrality on the new substitution variables $v_{ii't}$. Constraints (22) have the following interpretation. For a particular part i and period t, the left-hand side of this constraint represents the cumulative "outflow" of part i in period t both to satisfy the cumulative demand of products j that use this part and the total quantity of part i used to replace other parts $i' \in RB(i)$ in periods 0, ..., t. The right-hand side of the constraint represents the cumulative "inflow," consisting of the opening inventory and cumulative receipts of part i plus the number of units of part i replaced by parts $i'' \in R(i)$ in periods 0, ..., t. Constraints (22) specify that the cumulative outflow must not exceed the cumulative inflow for each part and every period. Formulation [APS] is clearly feasible (set all variables to zero).

To solve problem [APS] we will follow the previous strategy of: (a) selecting, via bisection search, a trial value z of the objective function, (b) determining the minimum required cumulative release quantities $y_{jt}(z)$ for all products j and periods t satisfying constraints (23), (24), and (25) such that $f''(X)$ with $X = Y(z) = (y_{jt}(z))$ equals or exceeds z, and (c) verifying if the desired release quantities $y_{jt}(z)$ are feasible, i.e., if the available parts can meet these release quantities with substitutions permitted. We refer to Step (c) as *verifying the resource-feasibility of Y(z)*. This step is more complicated for model [APS] than for the basic model [ARP']. As we explain next, we must solve a network flow subproblem to verify resource-feasibility of Y(z) for the APS problem.

Given a vector $\hat{X} = (\hat{x}_{jt})$ (= $(y_{jt}(z))$ in our bisection method), Step (c) requires verifying if the partial solution $\hat{X}$ has a "feasible completion" in [APS], i.e., we must determine if there are nonnegative integer values for the variables $v_{ii't}$ that satisfy constraints (22) after we substitute $x_{jt} = \hat{x}_{jt}$ in these constraints. To motivate the procedure for checking resource-feasibility of $\hat{X}$ and if so determining feasible v-values, let us reformulate constraints (22) by introducing an additional set of variables $\mathbf{k_{it}}$ for all i = 1, ..., m, and t = 0, ..., p–1. We will define these variables via the following equations:

$$k_{i0} = b_{i0} + \sum_{i'' \in R(i)} v_{i''i0} - \sum_{i' \in RB(i)} v_{ii'0} \qquad \text{for all } i = 1, ..., m, \text{ and} \qquad (27)$$

$$k_{it} = b_{it} + k_{i,t-1} + \sum_{i'' \in R(i)} v_{i''it} - \sum_{i' \in RB(i)} v_{ii't} \qquad \text{for all } i = 1, ..., m,\ t = 1, ..., p-1. \qquad (28)$$

Notice that using these k-variables, we can simplify constraints (22) to:

$$\sum_{j=1}^{n} a_{ij}\, \hat{x}_{jt} \leq k_{it} \qquad \text{for all } i = 1, ..., m,\ t = 0, ..., p-1. \qquad (29a)$$

$$b_{ip} + k_{i,p-1} + \sum_{i'' \in R(i)} v_{i''ip} \geq \sum_{j=1}^{n} a_{ij} \hat{x}_{jp} \quad \text{for all } i = 1, \ldots, m. \tag{29b}$$

Therefore, **verifying the resource-feasibility of a given cumulative release vector $\hat{X}$ entails determining a solution $(V(\hat{X}),K(\hat{X}))$ satisfying constraints (26) through (29) or proving that no such solution exists.**

We now explain how we can interpret constraints (27) and (28) as the familiar flow conservation equations of a network flow problem, and constraints (29) as lower limits on the flows on certain arcs. For a given vector $\hat{X}$, consider a directed network $G'(\hat{X})$ containing nodes (i,t) corresponding to each product i = 1, ..., m, and every period t = 0, ..., p. The network contains two types of arcs: (i) "substitution" arcs from node (i,t) to node (i',t) for all i = 1, ..., m, $i' \in RB(i)$, and t = 0, ..., p; and, (ii) "k-arcs" from node (i,t) to node (i, t+1) for all i = 1, ..., m, and t = 0, ..., p–1. Figure 2 shows a portion of this network. In this figure, part i' can replace part i, while part i can replace part i". Each k-arc from (i,t) to (i,t+1) has a <u>lower limit</u> $LL_{it}(\hat{X}) = \sum_{j=1}^{n} a_{ij} \hat{x}_{jt}$ on its flow, i.e., the minimum required flow on this arc is $LL_{it}(\hat{X})$. Every node (i,t), for i = 1, ..., m and t = 0, ..., p, has a supply of b_{it} units. For i = 1, ..., m, node (i, p) has a demand (minimum required external outflow) of $\sum_{j=1}^{n} a_{ij}\hat{x}_{jp}$. (Since this node also has a supply of b_{ip} units, its net demand is $\sum_{j=1}^{n} a_{ij}\hat{x}_{jp} - b_{ip}$.)

Consider the following *flow feasibility* problem defined over the network $G'(\hat{X})$.

Find a feasible flow on $G'(\hat{X})$ satisfying the demand, supply, and minimum arc flow constraints, or prove that no feasible flow exists.

The mathematical formulation of this flow feasibility problem contains flow conservation equations at each node, constraints imposing the minimum flow requirements $LL_{it}(\hat{X})$ on the k-arcs, and nonnegativity constraints on the flow variables. If we interpret the flows on the substitution arcs as the v-variables, and the flows on the k-arcs as the k-variables, then we see that:

(i) the flow conservation equations at nodes (i,0) for all i = 1, ..., m are the same as constraints (27);

(ii) the flow conservation equations at nodes (i,t) for all i = 1, ..., m, and t = 1, ..., p-1, are the same as constraints (28);

(iii) the minimum flow requirements on the k-arcs correspond to constraints (29a); and,

(iv) for i = 1, ..., m, the demand constraints at nodes (i,p) are the same as constraints (29b).

Notice that, if the network $G'(\hat{X})$ has a feasible flow, then it must have an integral flow (assuming all the parameters a_{ij} and b_{it} are integral), i.e., constraints (26) are satisfied. We can easily incorporate upper bounds on the substitution variables and k-variables; these upper bounds become arc capacities for the substitution arcs and k-arcs, respectively, in $G'(\hat{X})$.

These observations establish the following lemma.

Lemma 4:

A given cumulative release vector $\hat{X}$ has a feasible completion if and only if the corresponding network flow problem defined over $G'(\hat{X})$ has a feasible solution.

Note that as $\hat{X}$ changes, only the minimum flow requirements $LL_{it}(\hat{X})$ change; the node supplies and demands do not change, and neither does the topology of network $G'(\hat{X})$. Any algorithm used for checking feasibility of a transshipment problem can be used to determine if $G'(\hat{X})$ has a feasible solution. Max-flow algorithms are commonly used for this purpose. The max-flow problem can be solved very efficiently in $O(|N|^2\sqrt{|E|})$ time, where $|N|$ and $|E|$ denote the number of nodes and edges in the network (Ahuja, Magnanti, and Orlin [1993]).

At each iteration, the bisection method applies the max-flow algorithm to verify the resource-feasibility of the cumulative release vector $Y(z)$ corresponding to the current target value z. We restate this iterative step (Step 1) of the modified bisection algorithm for solving the APS problem with general precedence constraints (20) (assuming these constraints satisfy the acyclic precedence graph property). The remaining steps and the W-evaluation subroutine are the same as before.

Modified bisection algorithm for the APS problem with general precedence constraints:

Step 1: Search process

```
Construct a y-evaluation sequence $\mathbf{S}$.
REPEAT
  $z \leftarrow (LB + UB)/2$;
  Compute $W(z)$ in $[\beta,\gamma]$;     {call W-evaluation subroutine}
  FOR successive $(j,t)$ in the sequence $\mathbf{S}$, set
    $y_{jt}(z) = \max\left[w_{jt}(z), \{\max \text{ RHS value over all precedence constraints (20) in } \Pi(j,t)\}\right]$;
  Construct the network $G'(Y(z))$;
```

```
    IF the network flow problem defined on $G'(Y(z))$ is feasible,
      THEN set  $LB \leftarrow f''(Y(z))$ and $\beta \leftarrow Y(z)$;
      ELSE set  $UB \leftarrow z$ and $\gamma \leftarrow Y(z)$;
  UNTIL $(UB - LB) \leq \varepsilon$;
```

Let $z' \in [LB, UB]$ be the final feasible target value when the bisection algorithm terminates, i.e., z' is the target value corresponding the last iteration in which the network flow problem defined on $G'(Y(z'))$ was feasible. As we noted earlier, we can determine the appropriate feasible v-values from the maximum flow solution corresponding to network $G'(Y(z'))$. One disadvantage of this solution is that it might entail "unnecessary" substitutions, i.e., the maximum flow algorithm might route flows on certain substitution arcs when in fact the problem has an alternate maximum flow solution that does not use these arcs. In practice, production managers might prefer to reduce substitutions, as far as possible, subject to the requirement that the objective value must be z'. We can incorporate this preference by solving at the end a minimum cost flow problem defined over the network $G'(Y(z'))$. The node supplies, demands, and arc flow lower bounds are as shown in Figure 2. We assign a penalty for flow along substitution arcs, i.e., each substitution arc has a cost $\delta > 0$ per unit of flow on that arc. We seek the min-cost network flow solution that meets (or exceeds) all the demands using the available supplies subject to the minimum flow requirements on the k-arcs. This min-cost flow problem is feasible since $Y(z')$ is resource-feasible. Since all substitutions are penalized equally, the optimal solution minimizes the total number of parts used as substitutes. The model can also incorporate part-dependent substitution penalties. Adding flow costs on the k-arcs allows modeling product-dependent finished goods inventory holding costs.

Note there is a tradeoff between the value of z and the (minimal) total amount of substitution needed. Let MCF(z) denote the min-cost flow substitution problem on the network $G'(Y(z))$, with minimal objective function value v(z). Since Y(z) is monotonic in z, as z increases the arc lower bounds $LL_{it}(Y(z))$ also increase. Thus if $z_1 < z_2$ then $MCF(z_2)$ is a restriction of $MCF(z_1)$, so that $v(z_1) \leq v(z_2)$. As we increase z we come closer to achieving our target service levels, but our substitution cost also increases. Clearly this approach could lead to a formal tradeoff of serviceability and substitutability.

The following proposition establishes the validity of the modified bisection algorithm for the APS problem.

Proposition 5:

The APS problem has a feasible solution with objective value z if and only if the network flow problem defined over G'(Y(z)) has a feasible solution.

Proof:

As we previously explained, for any value of z, the vector $\hat{X} = Y(z)$ computed in Step 1 is the smallest nonnegative integer vector with objective value less than or equal to z and satisfying the precedence constraints (20) (or (23)), and the upper bounds (24). If the network flow subproblem defined over G'(Y(z)) is feasible, then by Lemma 4 the solution (Y(z), V(Y(z))) is feasible in formulation [APS]. Hence, $f''(Y(z)) \geq z$ is a valid lower bound on the optimal value of [APS].

To complete the proof we need to show that if, for some value of z, the network flow subproblem defined over G'(Y(z)) is infeasible, then z is a valid upper bound, i.e., the model [APS] does not have any feasible solution with objective value greater than or equal to z. We establish this property via contradiction. Suppose, for a given value z, the network G'(Y(z)) does not have a feasible flow, but the APS problem has a feasible assembly-release solution, say, X' whose objective function value f"(X') equals or exceeds z. Since the productivity functions are nondecreasing and since Y(z) is the smallest vector of release quantities that achieves the objective function value of z while satisfying (20), (24), and (25), X' must be $\geq$ Y(z). Therefore, the arc flow lower limits $LL_{it}(X')$ in G'(X') must equal or exceed the lower limits $LL_{it}(Y(z))$ in G'(Y(z)). Since X' is feasible in [APS], the network G'(X') must have a feasible flow (by Lemma 4). The same flow must also be feasible in G'(Y(z)) (since $LL_{it}(Y(z)) \leq LL_{it}(X')$), contradicting the hypothesis. Therefore, if the network flow subproblem defined on G'(Y(z)) is infeasible, z is a valid upper bound on the optimal value of [APS]. ♦

Finally, we note that the modified bisection method can handle nonlinear, nondecreasing resource constraints as well.

6. Conclusions

This paper has discussed a set of increasingly complex planning problems in multi-product assemble-to-forecast environments with part commonality, and showed how to solve these problems using bisection methods. The bisection approach is a simple and versatile method to solve a wide class of discrete, monotonic bottleneck resource-allocation problems. Although its worst-case computational complexity depends on the data (e.g., the values of the upper bounds

u_{jt}), it can efficiently solve practical problems. The method offers the considerable advantage of finding integer solutions to problems with nonlinear productivity functions $f_{jl}(\cdot)$. This advantage stems from the fact that we only need to verify the feasibility of iterative solutions Y(z) with respect to the nonlinear, nondecreasing resource constraints, rather than incorporating these constraints directly in the optimization process. The nondecreasing requirement for the resource-usage functions $G(\cdot)$ is a relatively weak condition that holds for many practical applications.

The basic assembly-release planning model added coupling constraints (12) to the original monotonic bottleneck resource-allocation problem [P]. In Section 4, we generalized these constraints to include a large class of precedence constraints (e.g., (19) and (20)) that satisfy the acyclic precedence graph property. In Section 5, we introduced additional susbstitution variables in the problem formulation.

Let us now provide a broader interpretation of these developments as enhancements to the generic discrete, monotonic bottleneck resource-allocation model [P]. We will refer to the x-variables in this model as the *primary variables*; the bottleneck objective function depends only on these variables. The ARP problem's coupling and precedence constraints create dependencies between the primary variables. In general, we might represent these restrictions as the following set constraint:

$$X \in \Theta.$$

The condition that constraints (20) must satisfy the acyclic precedence graph property essentially translates into the following general requirement for the set Θ: for any target value z of the objective function, it must be "easy" to identify the smallest vector Y(z) in Θ that achieves the target value. For instance, the acyclic precedence graph property permits us to compute $y_j(z)$ easily after we determine the y-evaluation sequence $\mathbf{S}$.

The introduction of part substitutability generalized the basic model [P] along another dimension. In addition to the vector X of primary variables, we included a vector V of *auxiliary* variables that appear in the resource-usage constraints. Instead of constraints (2), the resource constraints for the ARP model with part substitutability have the following general form:

$$g_i(X) + h_i(V) \leq S_i \qquad \text{for all } i = 1, \ldots, m. \tag{30}$$

Furthermore, the auxililary variables might have additional restrictions:

$$V \in \Omega. \tag{31}$$

For the parts substitutability case, the set Ω was the set of all nonnegative integer vectors V. We defined the concept of resource-feasibility of any vector $\hat{X} \in \Theta$ as the requirement that constraints (30) must have a feasible completion $V(\hat{X}) \in \Omega$ when we substitute $X = \hat{X}$ in these constraints. With the general resource constraints (30) and auxiliary variables V, the original nondecreasing property of resource-usage functions for the bottleneck activity variables X is replaced by the following general *monotonic resource-feasibility* requirement:

> If a vector $X \in \Theta$ is not resource-feasible then neither is any other vector $X' \in \Theta$ with $X' \geq X$.

Finally, in order to apply the bisection method to this generalized bottleneck resource-allocation model, we also require a subroutine that can efficiently find a feasible completion $V(\hat{X}) \in \Omega$ for any bottleneck activity vector $\hat{X} \in \Theta$ that is resource-feasible. For the APS problem, the network flow subproblem served as this subroutine.

To summarize, the bisection method is effective for solving bottleneck resource-allocation problems with discrete variables, nonlinear productivity functions, and nonlinear resource constraints under the following three conditions: (i) determining the smallest bottleneck activity vector $Y(z) \in \Theta$ that achieves the target objective value z is relatively easy; (ii) the resource-usage functions satisfy the monotonic resource-feasibility requirement; and (iii) finding a feasible completion for any bottleneck activity vector $\hat{X} \in \Theta$ that is resource-feasible (or proving that this vector is not resource-feasible) is relatively easy. The assembly-release planning model with precedence constraints and part substitutability meets these conditions.

REFERENCES

Ahmadi, J., S. Grotzinger, and D. Johnson, "Component Allocation and Partitioning for a Dual Delivery Machine," *Operations Research*, **36** (1988) 176-191.

Ahmadi, R. H., and H. Matsuo, "The Line Segmentation Problem," *Operations Research*, **39** (1991) 42-55.

Ahuja, R. K., T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, New Jersey (1993).

Bagchi, U., and G. Gutierrez, "Effect of Increasing Component Commonality on Service Level and Holding Cost," *Naval Research Logistics*, **39** (1992) 815-832.

Baker, K. R., "Requirements Planning" in *Handbook of Operations Research and Management Science, Volume 4 (Logistics of Production and Inventory)*, S. C. Graves, A. H. G. Rinnoy Kan, and P. H. Zipkin (eds.), North-Holland, Amsterdam (1993).

Baker, K. R., M. J. Magazine, and H. L. W. Nuttle, "The Effect of Commonality on Safety Stock in a Simple Inventory Model," *Management Science*, **32** (1986) 982-988.

Brown, J. R., "The Knapsack Sharing Problem," *Operations Research* , **27** (1979) 341-355.

Brown, J. R., "Solving Knapsack Sharing Problems with General Tradeoff Functions," *Mathematical Programming* , **51** (1991) 55-73.

Collier, D. A., "Aggregate Safety Stock Levels and Component Part Commonality," *Management Science*, **28** (1982) 1296-1303.

Francis, R. L., and T. Horak, "An Efficient Algorithm for Solving the Reel Allocation Problem," to appear in *IIE Transactions* (1994)

Gerchak, Y., and M. Henig, "Component Commonality in Assemble-to-Order Systems: Models and Properties," *Naval Research Logistics*, **36** (1989) 61-68.

Gerchak, Y., M. J. Magazine, and B. Gamble, "Component Commonality with Service Level Requirements," *Management Science*, **34** (1988) 753-760.

Grotzinger, S. J., and D. W. Cooper, "Selecting a Cost-Effective Number of Samples to Use at Preselected Locations," *Journal of the IES*, (1992) 41-49.

Grotzinger, S. J., R. Srinivasan, R. Akella, and S. Bollapragada, "Component Procurement and Allocation for Products Assembled to Forecast: Risk Pooling Effects," *IBM Journal of Research and Development*, **37** (1993) 523-536.

Grotzinger, S., and M. O'Connor, "Procurement and Release with Uncertain Demands," Research Report RC 18890, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, May 1993.

Ibaraki, T., and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches*, The MIT Press, Cambridge, Massachusetts (1988)

Jacobsen, S., "On Marginal Allocation in Single Constraint Min-Max Problems," *Management Science*, **17** (1971) 780-783.

Klein, R. S., H. Luss, and U. G. Rothblum, "Minimax Resource Allocation Problems with Resource Substitutions Represented by Graphs," *Operations Research*, **41** (1993) 959-971.

Lawler, E. L., *Combinatorial Optimization: Networks and Matroids*, Holt, Reinhart and Winston, New York (1976).

Luss, H., "Minimax Resource Allocation Problems: Optimization and Parametric Analysis," *European Journal of Operational Research*, **60** (1992) 76-86.

Luss, H., and D. R. Smith, "Multiperiod Allocation of Limited Resources: A Minimax Approach," *Naval Research Logistics Quarterly*, **35** (1988) 493-501.

Porteus, E. L., and J. S. Yormark, "More on the Min-Max Allocation Problem," *Management Science*, **18** (1972) 502-507.

Tang, C. S., "A Max-Min Allocation Problem: Its Solutions and Applications," *Operations Research* , **36** (1988) 359-367.

Figure 1: Time line for procurement and production decisions and events

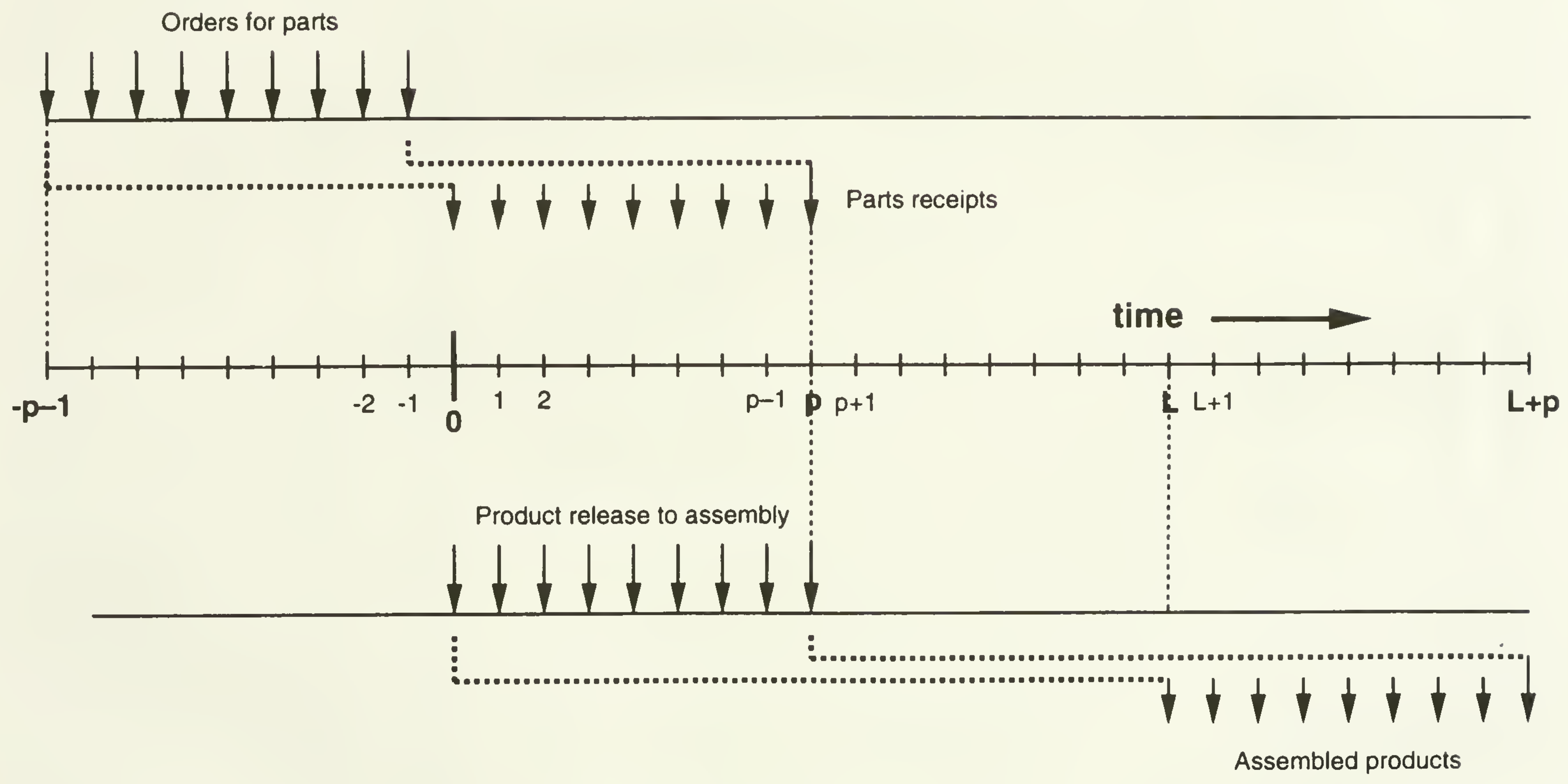
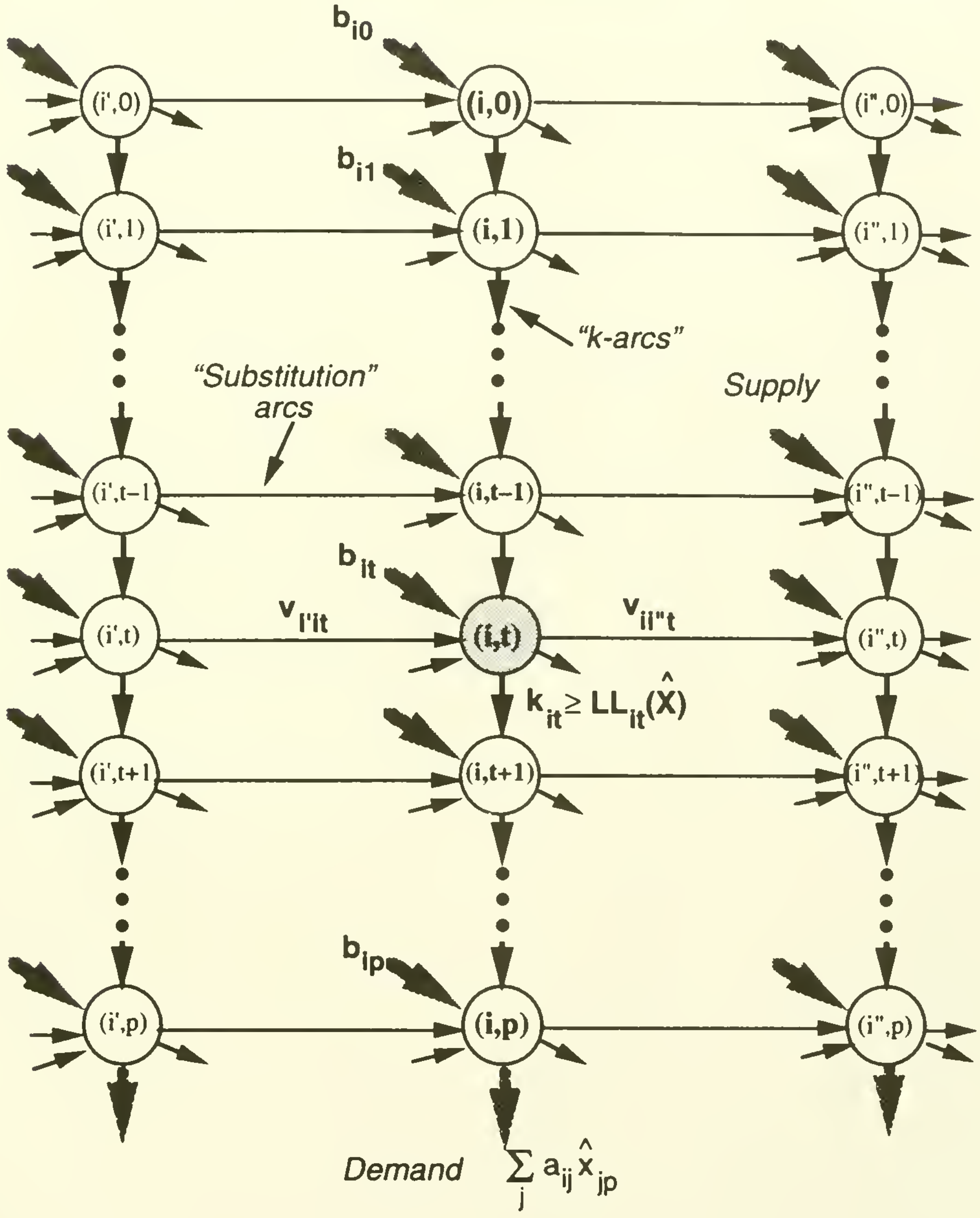


Figure 2: Network configuration to verify resource-feasibility for APS problem

MIT LIBRARIES
3 9080 00897 4864



2176 045

Date Due

SEP. 6 1995		
		Lib-26-67