ON EQUITABLE RESOURCE ALLOCATION PROBLEMS: A LEXICOGRAPHIC MINIMAX APPROACH

HANAN LUSS

Telcordia Technologies (formerly Bellcore), Piscataway, New Jersey

(Received February 1997; revision received April 1998; accepted May 1998)

In this expository paper, we review a variety of resource allocation problems in which it is desirable to allocate limited resources *equitably* among competing activities. Applications for such problems are found in diverse areas, including distribution planning, production planning and scheduling, and emergency services location. Each activity is associated with a performance function, representing, for example, the weighted shortfall of the selected activity level from a specified target. A resource allocation solution is called *equitable* if no performance function value can be improved without either violating a constraint or degrading an already equal or worse-off (i.e., larger) performance function value that is associated with a different activity. A *lexicographic minimax* solution determines this equitable solution; that is, it determines the lexicographically smallest vector whose elements, the performance function values, are sorted in nonincreasing order. The problems reviewed include large-scale allocation problems with multiple knapsack resource constraints, multiperiod allocation problems for storable resources, and problems with substitutable resources. The solution of large-scale problems necessitates the design of efficient algorithms that take advantage of special mathematical structures. Indeed, efficient algorithms for many models will be described. We expect that this paper will help practitioners to formulate and solve diverse resource allocation problems, and motivate researchers to explore new models and algorithmic approaches.

Resource allocation problems are concerned with the allocation of limited resources among competing activities so as to optimize some objective. Linear programming models with an objective function that minimizes, or maximizes, the sum of linear terms are widely used to formulate resource allocation problems. Advances in linear programming algorithms and computing power have facilitated the solution of large-scale problems. Integration of these algorithms with modeling languages and spreadsheets has encouraged the use of linear programming among practitioners and managers. However, such models suffer from certain drawbacks. Extreme point solutions often minimize a single overall objective, e.g., total costs, at the expense of disproportionately affecting some of the individual activities. Such solutions may be viewed by decision makers as undesirable and difficult to implement. Further, although small changes in input parameters typically lead to small changes in the optimal objective function value, they may lead to an extreme point with significantly different activity levels. Such "nervousness" in the optimal decision variables reduces a model's credibility and is difficult to comprehend.

Many papers describe resource allocation problems with special mathematical structures, solved by custom-made algorithms that take advantage of these structures. The books by Mjelde (1983b) and Ibaraki and Katoh (1988), and a recent paper by Katoh and Ibaraki (1997) describe a large collection of such resource allocation models and algorithms.

In this paper, we focus on an objective function that attempts to allocate resources *equitably* among the competing activities. We first consider the *minimax* objective function. Each activity is measured by a *performance function* that depends on the corresponding level assigned to that activity; a smaller objective function value is considered better. For instance, a performance function may represent the weighted shortfall of the level assigned to the activity from a specified target—the larger the assigned level (up to the target), the smaller is the shortfall. The weights may represent relative importance or cost parameters. A minimax objective function then minimizes the largest performance function among all activities: in our example, the largest weighted shortfall from the given target among all activities. Models with a minimax objective function attempt to allocate limited resources equitably among the worst-off activities, i.e., among activities whose performance function value is the largest. However, the minimax solution does not provide adequate guidance regarding the levels that should be assigned to all other activities. Thus, the question arises: Which solution from among those that yield the minimax objective value should be selected, while allocating resources equitably among all

Subject classifications: Programming, large-scale systems: resource allocation algorithms. Programming, multiple criteria: lexicographic minimax objective. Production/scheduling; applications: resource allocation models.
Area of review: OR CHRONICLE.

0030-364X/99/4703-0361 $05.00

activities? This is particularly important in large-scale problems, where, typically, each activity uses only a small subset of the resources. The *lexicographic minimax* objective function, explained below, addresses precisely this issue.

The lexicographic minimax solution determines *equitable* allocation of resources among all activities in the sense that no performance function value can be improved further without either violating a constraint or degrading an already equal or worse-off (i.e., larger) performance function value that is associated with a different activity. Let ν_1 and ν_2 be vectors with n elements each. Suppose that the first element in which vector $\nu_1 = (\nu_{11}, \nu_{12}, \ldots, \nu_{1n})$ differs from vector $\nu_2 = (\nu_{21}, \nu_{22}, \ldots, \nu_{2n})$ is element e. Then, ν_1 is lexicographically smaller than ν_2 if and only if $\nu_{1e} < \nu_{2e}$. Consider now the vector of performance function values. The lexicographic minimax solution is then defined as a solution that provides the lexicographically smallest vector whose elements, the performance function values, are sorted in nonincreasing order.

The various resource allocation problems addressed in this paper have mathematical structures that allow finding the lexicographic minimax solution by repeatedly solving problems with a minimax objective function. At each iteration, the optimal level for some activities is determined. A new minimax problem is then formulated without these activities and with updated parameters, such as available resource supplies.

We illustrate these ideas through a resource allocation problem, where the only constraints imposed are inequality resource constraints of the knapsack type (one unit of any activity consumes a specified nonnegative amount of each resource) and zero lower bounds on the activity levels. Also, suppose all performance functions are strictly decreasing functions. The minimax objective value is then obtained with one or more resource constraints, called *critical resources*, fully used. Since, in most large-scale problems, each activity uses only a small subset of the resources, many activities may not use the critical resources. Hence, the optimal levels assigned to all activities are not necessarily unique. Among all vectors of activity levels that provide the minimax objective value for this problem, an appealing one is called the *minimal solution*. Let the vector x^* be the vector of activity levels that constitutes the minimal solution. Then, $x^* \leqslant y^*$, where y^* is any alternate vector that provides the same objective function value. Note that for problems with knapsack constraints, there exists a minimal solution; furthermore, this solution provides the maximal amount of leftover resources.

The tableau in Figure 1 shows resource constraints for a problem with m resources (rows) and n activities (columns). Suppose the minimal solution has resource i_c as the single critical resource. Each P in row i_c implies that the corresponding activity uses a positive amount of resource i_c, and each 0 implies that it does not use resource i_c. The

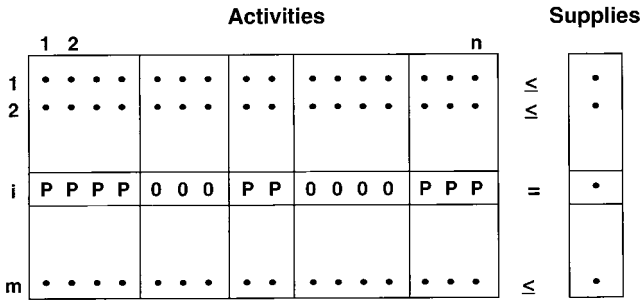

Figure 1. Example of the critical resource in the resource-activity tableau.

coefficients in the remaining $m - 1$ rows and the resource supplies in the right-hand-side column are shown as dots. None of the levels determined for activities with a P in row i_c can be feasibly changed without increasing the minimax objective value. To see this, recall that by definition of the minimal solution, none of the activity levels can be decreased without increasing the objective function value. Furthermore, if the level of some activity with a P in row i_c is increased, the level of another activity with a P in row i_c must be decreased. We conclude that the minimal solution determines the unique optimal levels of all activities with a P in row i_c. Once this minimal solution is found, a new tableau, which excludes row i_c and all activities with a P in that row, can be defined. In the new tableau, the resource supplies are updated to reflect the amounts used by the activities whose levels were fixed. We can now allocate these leftover resources among the remaining activities, by optimizing the same minimax objective over the remaining activities. The procedure can be repeated until all activity levels are fixed, at which point the lexicographic minimax solution is obtained. Luss and Smith (1986) propose the lexicographic minimax objective to solve resource allocation problems with multiple knapsack resource constraints.

A sample of applications for minimax and lexicographic minimax resource allocation problems is presented in §1. Although the discussions in this paper are in terms of the minimax and lexicographic minimax objectives, analogous models with maximin and lexicographic maximin objectives can readily be formulated and solved by similar algorithms. Note that the lexicographic minimax (maximin) objective determines equitable solutions for problems where a smaller (larger) performance function value is considered better. The relevant models are often solved in a real-time computing environment, either as stand-alone models or as part of more complex systems. Hence, it is important to exploit the mathematical structure of these models and provide efficient algorithms, capable of solving large-scale problems in seconds. Given the central role of the minimax problems in the lexicographic minimax solution approach, a significant part of the presentation is devoted to the underlying minimax problems.

Section 2 considers the basic resource allocation problem with multiple knapsack-type resource constraints, and where each term in the objective function represents a performance function for a single activity. Section 3 extends the basic resource allocation problem to a multiperiod setting. Section 4 examines resource allocation problems with substitutable resources. Section 5 highlights various resource allocation problems whose mathematical structure is more complex. Final remarks are given in Section 6.

1. APPLICATIONS

We present below a sample of applications for which resource allocation models with minimax and lexicographic minimax (or, equivalently, maximin and lexicographic maximin) objective functions are appealing. In some applications, it is only important to protect the worst-off activities, in which case, any of the minimax solutions can be selected. However, in many applications, selecting the lexicographic minimax solution from among the many minimax solutions provides a significant enhancement.

Distribution of Strategic Resource

Distribution of strategic resources among competing demand locations requires equitable allocation methods, particularly during critical periods of expected shortages. Consider, for example, distribution of spare parts for military airplanes during a war. A reasonable approach would consist of estimating relative importance parameters for different airplane types at different locations. These parameters are then fed into a model with a lexicographic minimax objective function that provides an equitable spare part distribution plan. Similar examples include the distribution of energy sources, water supply, specialized high-tech integrated circuits, etc. Brown (1979a, 1983) provides an example of distributing coal among power companies during periods of shortage of coal supply. Agnihothri et al. (1982) describe a model for the allocation of a critical product among competing locations under stochastic demand.

Production Planning

An important function in the manufacturing of high-tech products consists of the assembly of numerous (thousands) components, like integrated circuits, onto a variety (hundreds) of circuit boards. Because of rapid changes in technology and the large number of components involved, shortages are often incurred. Effective allocation of resources (components) among competing activities (circuit boards) is critically important in order to minimize penalties due to loss of current and future sales. The lexicographic minimax objective function is quite appealing for such production planning problems as it proposes equitable allocation of components among all products, while taking into consideration the relative importance of the different products. King (1989) presents a multiperiod production planning model that uses this approach. The model serves as an enhancement to the popular Material Requirement Planning (MRP) systems.

Production Scheduling

In the manufacturing of customized end-products, such as cars and telecommunications systems, multiple feeder shops supply subassemblies to the final assembly shop. The daily demands imposed on the feeder shops depend on the sequence in which the final assembly shop plans to assemble the customized products, as each of these requires a different amount of each of the subassemblies. In order to facilitate efficient manufacturing of the subassemblies, while keeping their inventories at a minimum, it is important that the final assembly sequence impose smooth daily demands for subassemblies on the feeder shops. Monden (1983) describes a final assembly sequencing model for the Toyota production system. Luss et al. (1990) describe a final assembly sequencing model for private-branch exchange telecommunications systems that uses a lexicographic minimax approach.

Emergency Services

Location of facilities for emergency services—like police, fire department, and emergency medical facilities—is an important and sensitive issue. Typically, such facilities should be located so as to provide roughly the same response time to all neighborhoods within a metropolitan area. Such problems are often modeled as covering problems, i.e., locating facilities so that each demand location will be served within a specified time, or as minimax problems, where the largest response time experienced by any of the demand locations is as small as possible. Kolesar and Walker (1974) describe, in their Lanchester prize-winning paper, a model for relocating fire companies. Daskin (1995) describes a variety of facility location models. Ogryczak (1997) proposes a lexicographic minimax approach to location problems.

Telecommunications Network Design

Modern telecommunications networks can provide instantaneous restoration service, in case of a link or node failure, due to the deployment of SONET (Synchronous Optical Network) standards and network topologies with multiple internetworked rings. The required capacity of each ring is determined by the maximal load that may be encountered by any of the links along the ring. Thus, minimizing the required capacity of a ring can be formulated as a minimax problem. Cosares et al. (1995) describe a ring-based network design tool used by multiple telecommunications companies.

As a second example, telecommunications networks are expected to accommodate multimedia services. Hence, it is important to allocate network resources, such as available bandwidth, so as to provide equitable performance in terms of, say, expected delays to all services at numerous destination nodes. Such problems can be formulated as

minimax network flow problems; see, for example, Brown (1983).

Workforce Management

Consider crew scheduling problems for transportation companies. Once a crew is assigned to a trip, it is not available for another assignment for a specified time. Trips not assigned to any crew are served through outsourcing to an external crew or to another company, thus incurring significant penalties. In order to manage these shortages effectively, it is desired to spread them equitably over time. Meketon (1996) describes a lexicographic minimax model for crew scheduling in a rail company.

As a second example, consider call service centers that handle large volumes of calls, serving requests for information, reservations, etc. It is challenging to design long employee shifts that provide acceptable service levels during, say, every 15-minute time interval throughout the day. The schedules need often address complicating issues, such as multiple teams of employees that overlap only in some of the required skills. Munson and Kodialam (1995) and Gawande (1996) describe variants of such problems.

Other Applications

A variety of other potential applications for minimax and lexicographic minimax resource allocation models have been mentioned by different authors. A few of these are mentioned below.

Chaddha et al. (1971) describe assignment of sample sizes for a stratified sample survey of different types of telephone switchboards. Kaplan (1974) discusses the allocation of the right mix of supplies, such as ammunition, food and fuel for a military mission. Garfinkel and Rao (1976) examine bottleneck problems. Brown (1979b) describes a minimax model for salary administration for a large number of job classifications. Mendelson et al. (1980) describe allocation of computer storage among numerous files so as to maximize the time until any of the application programs runs out of space. Mjelde (1983a) presents an example of allocating inspection and maintenance efforts among multiple components that are part of the same system. Eiselt (1986) discusses allocation of government funds fairly among nationally important projects. Ibaraki and Katoh (1988, Ch. 7) describe apportionment problems, concerned with the allocation of seats among electoral districts, so that the number of seats given to each district is as proportional to its population as possible. Tang (1988) presents several manufacturing applications, including storage space allocation for high-speed component insertion machines. The objective there is to maximize the time until any of the components needs to be replenished. Kouvelis and Yu (1997) describe robust optimization for various applications, such as project portfolio selection. A robust solution selects the portfolio whose return is maximized under the worst possible scenario.

2. EQUITABLE RESOURCE ALLOCATION PROBLEMS WITH MULTIPLE RESOURCES

In this section, equitable resource allocation problems with multiple resource constraints are reviewed. The presentation provides the foundation for more complex resource allocation problems described in subsequent sections.

2.1. The Minimax Problem

The minimax problem considers the allocation of multiple resources among competing activities so as to minimize the largest performance function value associated with any of the activities.

We use the following notation:

- i = Index for resources; $i = 1, 2, \ldots, m$, where m is the number of resources. Let I be the set of indices i of all resources.
- j = Index for activities; $j = 1, 2, \ldots, n$, where n is the number of activities. Let J be the set of indices j of all activities.
- a_{ij} = Amount of resource i needed for each unit of activity j; $a_{ij} \geqslant 0$.
- b_i = Amount available of resource i; $b_i > 0$.
- x_j = Level selected for activity j. Let the vector $x = (x_1, x_2, \ldots, x_n)$.
- $f_j(x_j)$ = Performance function associated with activity j. It depends only on x_j.
- $f_j^{-1}(\cdot)$ = Inverse function of $f_j(\cdot)$; that is, if $f_j(x_j) = \mu$, then $f_j^{-1}(\mu) = x_j$.
- $*$ = Superscript that denotes optimal values.

The resource allocation problem with multiple resources, referred to as PROBLEM RESOURCE, is formulated as follows:

PROBLEM RESOURCE

$$V^* = \min_x \Big[\max_{j \in J} f_j(x_j) \Big] \tag{1.1}$$

so that

$$\sum_{j \in J} a_{ij} x_j \leqslant b_i, \quad i \in I, \tag{1.2}$$

$$x_j \geqslant 0, \quad j \in J. \tag{1.3}$$

We are interested in finding not merely an optimal solution to PROBLEM RESOURCE, but rather the minimal solution. The latter is denoted as x^*. The performance functions $f_j(x_j)$ are assumed to be continuous, strictly decreasing, and invertible functions. Hence, the inverse functions $f_j^{-1}(\cdot)$ are also continuous and strictly decreasing. Resource constraints (1.2) are inequality constraints of the knapsack type ($a_{ij} \geqslant 0$). Various extensions are described in Subsection 2.3.

Several key properties of PROBLEM RESOURCE are exploited to develop efficient algorithms:

(a) Objective function (1.1) is separable. That is, each performance function depends only on the level selected for the corresponding activity.

(b) All $a_{ij} \geq 0$. That is, activities do not generate resources.
(c) All resource constraints (1.2) are inequality constraints.

Suppose the activities are indexed so that

$$f_{j+1}(0) \geq f_j(0), \quad j \in J. \tag{2}$$

Intuitively, if the minimal level of activity j_1 is positive, then the minimal level for any activity $j > j_1$ must also be positive since its performance function value at 0 is larger than the performance function of activity j_1 at any positive value. We conclude that there is an optimal set $J^* = \{j^*, j^* + 1, \ldots, n\}$ such that

$$x_j^* > 0, \quad j \in J^*, \tag{3.1}$$

$$x_j^* = 0, \quad j \notin J^*. \tag{3.2}$$

Furthermore, the minimal solution x^* satisfies:

$$f_j(x_j^*) = V^* \text{ (i.e., } x_j^* = f_j^{-1}(V^*)), \quad j \in J^*, \tag{4.1}$$

$$f_j(x_j^*) \leq V^* \text{ (i.e., } x_j^* \geq f_j^{-1}(V^*)), \quad j \notin J^*. \tag{4.2}$$

Thus, from (3) and (4), the minimal solution x^* can be expressed as:

$$x_j^* = \max[0, f_j^{-1}(V^*)], \quad j \in J. \tag{5}$$

Since the performance functions are strictly decreasing, at least one of the resource constraints (1.2) would be fully used by the minimal solution.

2.2. Minimax Algorithms

As a prelude to solving PROBLEM RESOURCE, consider a relaxed version, where only a subset of the activities $J_s = \{j_s, j_{s+1}, \ldots, n\} \subseteq J$ are considered and the nonnegativity constraints (1.3) are deleted. Also, suppose momentarily that the problem has only a single resource constraint, namely, constraint i with a supply of b_i, and let V_i^R be the minimal value of this relaxed problem. V_i^R is obtained by equating $f_j(x_j) = V_i^R$ for all $j \in J_s$ and solving for $\sum_{j \in J_s} a_{ij} x_j = b_i$ which implies:

$$\sum_{j \in J_s} a_{ij} f_j^{-1}(V_i^R) = b_i. \tag{6}$$

The minimax value V^R for the relaxed problem with all resource constraints (and activity set J_s) is simply obtained by computing V_i^R for all $i \in I$ and setting

$$V^R = \max_{i \in I} V_i^R. \tag{7}$$

The corresponding minimal decision variables are $x_j^* = f_j^{-1}(V^R)$ for all $j \in J_s$.

Consider, for example, linear performance functions $f_j(x_j) = p_j - r_j x_j$ $(r_j > 0)$ for all $j \in J$. Such functions may represent weighted shortfall from given targets. Let d_j be the target for activity j, and let α_j be the relative importance of activity j. Then, $f_j(x_j) = \alpha_j(d_j - x_j)/d_j$ is the weighted (normalized) shortfall from the target; i.e., $p_j = \alpha_j$ and $r_j = \alpha_j/d_j$. From (6), V_i^R is given by the closed-form expression:

$$V_i^R = \frac{\sum_{j \in J_s} a_{ij} p_j / r_j - b_i}{\sum_{j \in J_s} a_{ij} / r_j}. \tag{8}$$

We present below several approaches that use the solutions V_i^R and V^R to solve PROBLEM RESOURCE. We assume that the activities are reindexed according to (2).

Activity-Deletion Algorithm for PROBLEM RESOURCE

The activity-deletion algorithm is proposed in Luss and Smith (1986) and Luss (1987) for linear and certain nonlinear performance functions for which the values V_i^R are given in closed-form. The idea is to start with the initial set $J_s = J$ as a candidate set for the optimal set J^*. The relaxed problem is then repeatedly solved, where at each iteration some activities are excluded from the set J_s. The algorithm is outlined as follows:

Step 1. Initialize $J_s = J$.

Step 2. Obtain the minimal value V^R for the relaxed problem with activity set J_s.

Step 3. If $f_j(0) \geq V^R$ for all $j \in J_s$, STOP; $V^* = V^R$, J^* includes all activities in J_s for which $f_j(0) > V^R$, and x^* is given by (5). Otherwise, exclude from J_s all activities with $f_j(0) < V^R$ and return to Step 2.

Activity-Addition Algorithm for PROBLEM RESOURCE

The activity-addition algorithm is proposed in Tang (1988). The idea is to start with an initial set $J_s = \{n\}$. The relaxed problem is then repeatedly solved, where at each iteration a single activity is added to the set J_s. The algorithm is outlined as follows:

Step 1. Initialize $J_s = \{n\}$.

Step 2. Obtain the minimal value V^R for the relaxed problem with activity set J_s.

Step 3. If $f_{j_s - 1}(0) \leq V^R$, STOP; $V^* = V^R$, $J^* = J_s$, and x^* is given by (5). Otherwise, add $j_s - 1$ to J_s and return to Step 2.

Hybrid Algorithm for PROBLEM RESOURCE

As proposed by Luss (1992), the two algorithms above can be combined by starting with an initial set of activities J_s that includes about half of the activities. Based on the initial solution of the relaxed problem, the algorithm proceeds with either the activity-deletion or activity-addition algorithm. One can also modify the algorithm by adding in the activity-addition algorithm multiple activities to J_s at a time, and resorting to the activity-deletion algorithm if too many activities were added.

The algorithms above are especially attractive when the values V_i^R are computed as closed-form expressions. Consider the expression in (8) for linear performance functions. Not only are the values V_i^R easy to compute, but

their updates from one iteration to the next simply require subtracting or adding terms to the summations. Although all three approaches above solve large-scale problems efficiently, the activity-deletion approach performs best when most optimal activity levels are positive, the activity-addition approach performs best when most of these are zero, and the hybrid approach performs best when about half of the optimal levels are positive.

Now, suppose the performance functions are such that the values V_i^R are not available as closed-form expressions. Although they can be derived from (6) through numerical methods, computing their values may be computationally costly. Fortunately, in order to find J^*, we do not have to compute the values V_i^R. Note that if $j_s \in J^*$, then $f_{j_s}(0) > V^*$, implying $x_j^* = f_j^{-1}(V^*) > f_j^{-1}(f_{j_s}(0))$ for $j \in J^*$. Hence, as shown in Luss (1991), the following property can be used:

$$\text{If } \sum_{j \in J_s} a_{ij} f_j^{-1}(f_{j_s}(0)) \geqslant b_i \text{ for some } i \in I,$$
$$\text{then } J^* \subset J_s; \text{ otherwise } J^* \supseteq J_s. \tag{9}$$

Property (9) can be used to find J^* by applying a bisection search on the activities in J. Note that J^* is then derived through function evaluations, without resorting to numerical methods. Once J^* is found, V^* is computed through numerical methods by solving equations in the format of (6), for each $i \in I$.

Among other noteworthy references, Chaddha et al. (1971), Brown (1979b), Mjelde (1983), and Czuchra (1986) examine PROBLEM RESOURCE with a single resource constraint. We conclude this subsection by mentioning the well-studied resource allocation problem of minimizing $\Sigma_{j \in J} f_j(x_j)$ for convex functions $f_j(x_j)$, subject to a single linear resource constraint and nonnegativity constraints on the activity levels. As shown by Czuchra (1986), the Kuhn-Tucker conditions for this problem lead to optimality conditions that have similar structure to (3) and (4). Not surprisingly, the solution approaches for this problem are similar to those presented here. Relevant references include Luss and Gupta (1973), Zipkin (1980), Bitran and Hax (1981), Vidal (1984), Ibaraki and Katoh (1988), and Kodialam and Luss (1998).

2.3. Extensions to Problem Resource

Lower and Upper Bounds. Suppose nonnegativity constraints (1.3) are replaced by lower and upper bound constraints:

$$l_j \leqslant x_j \leqslant u_j, \quad j \in J, \tag{10}$$

where $u_j \geqslant l_j$ for all $j \in J$.

Lower bounds can be handled by an analysis similar to that presented above for $l_j = 0$ for all $j \in J$, or simply by a linear transformation of variables $x_j \leftarrow x_j - l_j$.

Upper bounds can be handled by solving PROBLEM RESOURCE without the upper bounds, resulting in an objective value V^*. The minimax objective value V^{**} for PROBLEM RESOURCE with the upper bounds is $V^{**} = \max[V^*, \max_{j \in J} f_j(u_j)]$. The minimal solution is $x_j = \max[l_j, f_j^{-1}(V^{**})]$ for all $j \in J$.

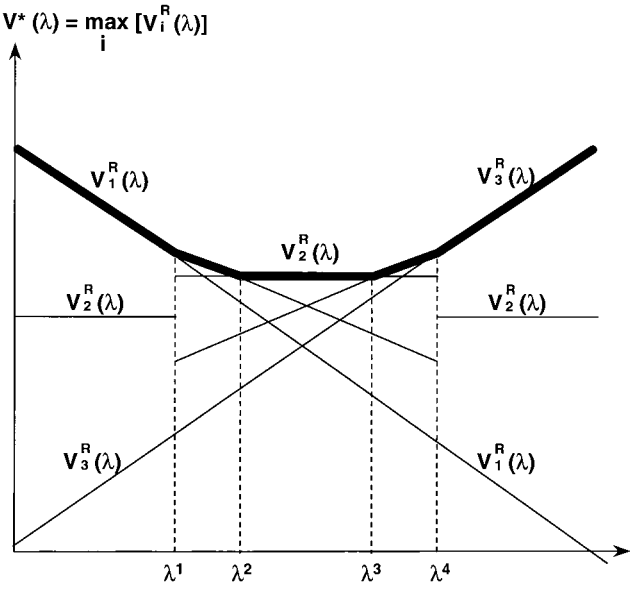

Figure 2. Example of parametric analysis.

Sensitivity Analysis. Sensitivity analysis is an extremely important feature of practical optimization software. Given the simple nature of the algorithms for PROBLEM RESOURCE, such capabilities can be readily provided, especially, when the values V_i^R are given as closed-form expressions. For example, consider PROBLEM RESOURCE with linear performance functions, so that the values V_i^R are given by (8). Suppose the resource supplies are given as $b_i + \lambda b_{0i}$, where the b_{0i}'s are constants and $\lambda \geqslant 0$ is a changing parameter. Parametric analysis then provides the optimal solutions for all $\lambda \geqslant 0$. Figure 2 exhibits an example with three resource constraints. The first constraint has $b_{01} > 0$, the second has $b_{02} = 0$, and the third has $b_{03} < 0$. The figure shows for the three constraints $V_i^R(\lambda)$, that is, the value of V_i^R as a function of λ, where J_s is the optimal set J^* for that λ. The bold line is $V^*(\lambda) = \max_i V_i^R(\lambda)$. Initially, at $\lambda = 0$, constraint 1 is the critical one. At $\lambda = \lambda^1$, an additional variable is added to the optimal set J^*. At $\lambda = \lambda^2$, constraint 2 becomes the critical one. At $\lambda = \lambda^3$, constraint 3 becomes the critical one, and at $\lambda = \lambda^4$, the variable added before to the set J^* is excluded from this set. Note that $V^*(\lambda)$ is a piecewise linear, convex function. Luss (1992) presents details on post-optimization and parametric analysis for PROBLEM RESOURCE with linear performance functions.

General Performance Functions. Brown (1991, 1994) presents detailed analysis for variants of PROBLEM RESOURCE with more general performance functions. Consider the performance function depicted by the solid line in Figure 3. This function has both increasing and decreasing segments. However, since all a_{ij}s in PROBLEM RESOURCE are nonnegative, each increasing segment may be replaced by a segment with zero slope (depicted by

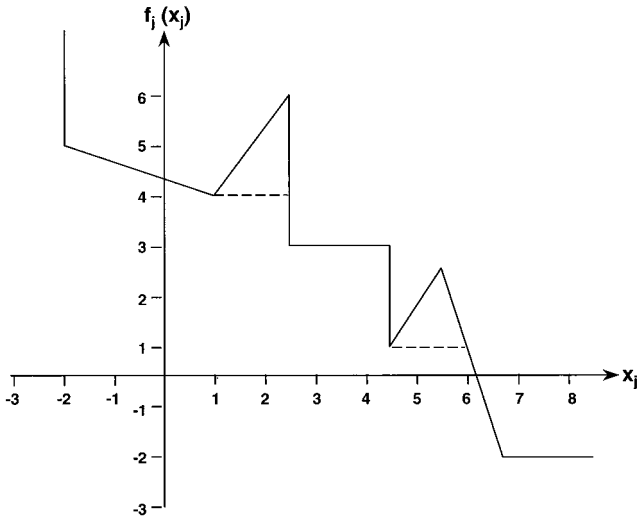

Figure 3. A more general performance function transformed to a descending function.

dashed lines) without affecting the minimax solution. The resulting performance function is called a descending function, as it is nonincreasing. It may have segments with an infinite slope, representing multiple values for the same activity level, as well as segments with a zero slope in which the function value does not change. Thus in Figure 3, the descending performance function has an infinite slope at $x_j = -2$, a decreasing segment from $x_j = -2$ to $x_j = 1$, a zero-slope segment from there to $x_j = 2.5$, etc. The algorithms for solving PROBLEM RESOURCE with descending performance functions use optimality properties similar to (9). However, special care should be devoted to using the proper function value (the smallest one) in segments with infinite slope, and the proper inverse value (again, the smallest one) in segments with zero slope. Due to performance functions with zero-slope segments, the minimal solution may be such that no resource constraint is fully used. A resource constraint is then called critical if the minimax objective value cannot be improved even if all other resource constraints are deleted.

Integer Solutions. Suppose constraints (1.3) are replaced by $x_j \geq 0$ and integer for all $j \in J$. Brown (1979b) proposes an algorithm that finds an integer solution to PROBLEM RESOURCE with a single resource constraint, and Tang (1988) extends the algorithm to problems with multiple resource constraints. Their algorithms repeatedly solve problems in the format of PROBLEM RESOURCE with continuous activity levels. Consider the minimal noninteger solution x^* for the original PROBLEM RESOURCE. Activities with $x_j^* = 0$ are fixed and excluded from further consideration. Note that $\lfloor x^* \rfloor$ is a feasible integer solution. A new integer solution (perhaps infeasible) is constructed by increasing by one unit the levels of all activities whose performance function value is the largest at $\lfloor x^* \rfloor$. If the new integer solution does not satisfy all resource constraints, the algorithm terminates with $\lfloor x^* \rfloor$ as the integer optimal solution. If it satisfies all resource constraints, the activities whose levels were increased are excluded from further consideration and their levels are fixed at those determined by the new integer solution. This may lead to fixing more activity levels at zero, and to deleting constraints whose leftover supplies are not needed by any activity that was not fixed. A new problem in the format of PROBLEM RESOURCE is then solved.

Other papers that present marginal allocation algorithms for the problem with a single resource constraint include Jacobsen (1971), Porteus and Yormark (1972), Zeitlin (1981), and Ichimori (1984).

2.4. The Lexicographic Minimax Problem and Its Solution

The solution of PROBLEM RESOURCE determines the minimax objective value and the minimal solution x^*. It also identifies activities whose corresponding level x_j^* cannot feasibly be changed without increasing the objective function value. However, the minimal solution may provide leftover resources that can be allocated equitably among activities that do not need the critical resources.

In order to address this important issue, Luss and Smith (1986) extend the minimax solution for PROBLEM RESOURCE to the lexicographic minimax solution. Given a vector of activity levels x, let $f^{(n)}(x)$ be a vector composed of the n elements $f_j(x_j)$, where these elements are sorted in a nonincreasing order. That is, $f^{(n)}(x) = [f_{j_1}(x_{j_1}), f_{j_2}(x_{j_2}), \ldots, f_{j_n}(x_{j_n})]$, where $f_{j_1}(x_{j_1}) \geq f_{j_2}(x_{j_2}) \geq \ldots \geq f_{j_n}(x_{j_n})$. The lexicographic minimax problem, referred to as PROBLEM L-RESOURCE, is formulated as follows:

PROBLEM L-RESOURCE

$$V^L = \operatorname{lexmin}_x [f^{(n)}(x)] \tag{11.1}$$

so that

$$\sum_{j \in J} a_{ij} x_j \leq b_i, \quad i \in I, \tag{11.2}$$

$$l_j \leq x_j \leq u_j, \quad j \in J. \tag{11.3}$$

The objective function minimizes lexicographically the vector $f^{(n)}(x)$. Let $x^L = (x_1^L, \ldots, x_n^L)$ be the optimal vector of activity levels; then $V^L = f^{(n)}(x^L)$. The vector x^L provides an equitable solution to all activities, in the sense that no performance function value can be decreased without either violating feasibility or increasing an already equal or larger performance function value, associated with a different activity. The performance functions are assumed to be continuous, strictly decreasing, and invertible (the discussion can readily be extended to descending functions). Constraints (1.3) are replaced by general lower and upper bound constraints (11.3).

The solution of PROBLEM L-RESOURCE is obtained by the repeated solution of PROBLEM RESOURCE. After each solution of the latter problem, some activity levels are fixed at their minimal values and deleted from the problem. Some resources may be deleted as well from

further consideration. Consider the solution of PROBLEM RESOURCE without the upper bounds, with activity set J and resource set I. Let V^* be the optimal objective value and let x^* be the corresponding minimal activity levels. The following properties provide the foundation for the lexicographic minimax solution approach:

(a) Suppose $x_j^* \geq u_j$ for one or more activities. Then, $x_j^L = u_j$ for these activities.

(b) Suppose i_c is a critical resource (i.e., it is fully used by the solution x^*) and $x_j^* \leq u_j$ for all activities j with $a_{i_c j} > 0$. Then, $x_j^L = x_j^*$ for all activities with $a_{i_c j} > 0$.

Algorithm for PROBLEM L-RESOURCE

Step 1. Formulate the initial PROBLEM RESOURCE with activity set J and resource set I, and lower and upper bounds l_j and u_j.

Step 2. Solve PROBLEM RESOURCE without the upper bounds. Determine the minimal solution x_j^*.

Step 3. Using properties (a) and (b) above, fix activity levels at $x_j^L = u_j$ and $x_j^L = x_j^*$, as appropriate.

Step 4. Delete from J all activities whose values were fixed. If J is now empty, STOP; the lexicographic minimax solution was obtained.

Step 5. Update the values b_i. Delete from I all resources that are not used by any of the activities in the updated set J. Return to Step 2.

The algorithms described for PROBLEM RESOURCE and PROBLEM L-RESOURCE can solve large-scale problems, especially when the expressions for V_i^R are available in closed form. The computational effort for PROBLEM RESOURCE is then on the order of $O(mn)$; in practice, however, the effort is on the order of $O(\theta mn)$, where θ is the fraction of positive a_{ij}s. The computing effort for PROBLEM L-RESOURCE is on the order of $O(mn^2)$, as the number of iterations is $O(n)$. Lennon (1991) experimented with Luss and Smith's Activity-Deletion Algorithm to solve problems with linear performance functions. Examples for PROBLEM RESOURCE with 3,000 activities, 3,000 resources, and θ ranging from 0.01 to 0.04 were solved in a fraction of a second on an IBM 3090 computer. Typically, these computation times were hundreds of times faster than those obtained by a simplex-based linear programming software (interior point methods are significantly slower than the simplex method for these problems). Examples for PROBLEM L-RESOURCE with $\theta = 0.01$ were solved in 200 to 300 iterations in 6 to 30 seconds. Since the number of lexicographic iterations decreases when θ increases, the computing time for PROBLEM L-RESOURCE did not change much with θ; the added effort required to solve each PROBLEM RESOURCE is compensated for by the need to solve fewer of these problems.

3. MULTIPERIOD RESOURCE ALLOCATION

In this section, PROBLEM RESOURCE and PROBLEM L-RESOURCE are extended to a multiperiod setting, primarily for storable resources.

3.1. The Minimax Multiperiod Problem

The minimax multiperiod problem extends PROBLEM RESOURCE to a multiperiod setting with a finite planning horizon. We use the same notation as in §2, with the following additions:

- t = Index for time periods; $t = 1, 2, \ldots, p$, where p is the planning horizon. Let T be the set of indices t of all time periods.
- (j, t) = Referred to as activity-period pair.
- (i, t) = Referred to as resource-period pair.
- b_{it} = Incremental amount of resource i made available at period t; $b_{it} \geq 0$.
- B_{it} = Cumulative amount of resource i made available up to period t; $B_{it} = \sum_{\tau=1}^{t} b_{i\tau}$.
- x_{jt} = Level selected for activity j at period t; $x_{jt} \geq 0$.
- X_{jt} = Cumulative level selected for activity j up to period t; $X_{jt} = \sum_{\tau=1}^{t} x_{j\tau}$, implying $X_{jt} \geq X_{j,t-1}$ where $X_{j0} = 0$. Let X be the vector of np variables X_{jt}; $X = (X_{11}, \ldots, X_{1p}, \ldots, X_{n1}, \ldots, X_{np})$.
- $f_{jt}(X_{jt})$ = Performance function of activity j at period t. It depends only on X_{jt}.
- L_{jt} = Lower bound for X_{jt}; $L_{jt} \geq L_{j,t-1}$ ($L_{j0} = 0$).
- U_{jt} = Upper bound for X_{jt}; $U_{jt} \geq U_{j,t-1}$ ($U_{j0} = 0$) and $U_{jt} \geq L_{jt}$.

We formulate below the multiperiod problem, referred to as PROBLEM MULTIPERIOD, for storable resources, where unused resources in one period can be used in subsequent periods. Storable resources are common in many of the application areas described in §1.

PROBLEM MULTIPERIOD

$$M^* = \min_X [\max_{j \in J, t \in T} f_{jt}(X_{jt})] \tag{12.1}$$

so that

$$\sum_{j \in J} a_{ij} X_{jt} \leq B_{it}, \quad i \in I \text{ and } t \in T, \tag{12.2}$$

$$X_{jt} \geq X_{j,t-1} (X_{j0} = 0), \quad j \in J \text{ and } t \in T, \tag{12.3}$$

$$L_{jt} \leq X_{jt} \leq U_{jt}, \quad j \in J \text{ and } t \in T. \tag{12.4}$$

Each performance function in (12.1) is assumed to be continuous, strictly decreasing, and invertible. Resource constraints (12.2) are expressed in cumulative terms, thus allowing for usage of surplus resources at period t in subsequent periods. Ordering constraints (12.3) are equivalent to constraints $x_{jt} \geq 0$. Constraints (12.4) impose lower and upper bounds on cumulative activity levels. PROBLEM MULTIPERIOD is feasible if and only if the lower bounds satisfy all the resource constraints. Then, an upper bound on M^* is given by $M_U = \max_{j \in J, t \in T} f_{jt}(L_{jt})$, and a lower bound is given by $M_L = \max_{j \in J, t \in T} f_{jt}(U_{jt})$. The

mathematical structure of PROBLEM MULTIPERIOD is quite similar to that of PROBLEM RESOURCE, with the added complexity because of ordering constraints (12.3).

Suppose the minimax value M^* for PROBLEM MULTIPERIOD is known. The minimal solution, as a function of M^*, satisfies the following recursive equations:

$$X^*_{jt}(M^*) = \max[L_{jt}, X^*_{j,t-1}(M^*), f^{-1}_{jt}(M^*)], \quad j \in J \text{ and } t \in T, \tag{13}$$

where $X^*_{j0}(M^*) = 0$ for all $j \in J$. This recursive equation is an extension of the minimal solution (5) for PROBLEM RESOURCE, as it assures that the ordering constraints will be satisfied. Recursion (13) forms the basis for a solution approach that implements a bisection search for finding M^*. For any trial value M, recursion (13) is used to compute the corresponding minimal solution $X^*(M)$. If that solution satisfies all resource constraints, then $M^* \leqslant M$; otherwise, $M^* > M$.

Betts et al. (1994) present a solution approach for PROBLEM MULTIPERIOD that is particularly attractive for cases in which the values V^R_i, used to solve PROBLEM RESOURCE, are available in closed form. Their algorithm solves at each iteration a problem in the format of PROBLEM RESOURCE (i.e., the ordering constraints are deleted), where the performance function of each activity-period pair (j_1, t_1) is represented by one of the original performance functions $f_{j_1\tau}(\cdot)$ for some $\tau \leqslant t_1$. Based on the solution at each iteration, either optimality is established, or the number of possible choices for τ in subsequent iterations is reduced for at least one activity-period pair (j, t).

3.2. Linear Performance Functions

We explore below PROBLEM MULTIPERIOD with certain linear performance functions. Let

α_{jt} = Weight (e.g., relative importance) of activity-period pair (j, t),
D_{jt} = Target for cumulative level of activity j up to period t.

We assume that $D_{jt} \geqslant D_{j,t-1}$ $(D_{j0} = 0)$ for all $j \in J$ and $t \in T$, that is, the incremental changes in the targets from one period to the next are nonnegative for each activity j.

The problem, referred to as PROBLEM MPL (MultiPeriod-Linear), is formulated as follows:

PROBLEM MPL

$$M^* = \min_X \Big[\max_{j\in J, t\in T} \alpha_{jt}(D_{jt} - X_{jt})/D_{jt} \Big] \tag{14.1}$$

so that

$$\sum_{j\in J} a_{ij}X_{jt} \leqslant B_{it}, \quad i \in I \text{ and } t \in T, \tag{14.2}$$

$$X_{jt} \geqslant X_{j,t-1} (X_{j0} = 0), \quad j \in J \text{ and } t \in T, \tag{14.3}$$

$$X_{jt} \leqslant D_{jt}, \quad j \in J \text{ and } t \in T. \tag{14.4}$$

The objective function (14.1) minimizes the largest weighted shortfall from its target among all activity-period pairs. The upper bounds (14.4) state that no level can exceed its target. Indeed, the minimal solution is either $X^*_{jt} = D_{jt}$ for all $j \in J$ and $t \in T$, or $X^*_{jt} < D_{jt}$ for all $j \in J$ and $t \in T$. Note that if $X^*_{jt} = D_{jt}$ for all $j \in J$ and $t \in T$, and excess supplies are still available, a similar maximin problem can be formulated to allocate these supplies equitably above the specified targets.

Suppose the optimal solution of PROBLEM MPL satisfies $X^*_{jt} < D_{jt}$ for all $j \in J$ and $t \in T$. Consider a relaxed version of PROBLEM MPL, where constraints (14.3) and (14.4) are deleted. To simplify the presentation, the parameters a_{ij} are replaced below by parameters a_{ijt}, allowing for dependency on t. The relaxed problem has the same mathematical structure as the relaxed problem for PROBLEM RESOURCE, and, hence, its solution is the same as (7) and (8). Let $J_{s(t)}$ be the set of activities considered by the relaxed problem at period t. In the initial relaxed problem, $J_{s(t)} = J$ and $a_{ijt} = a_{ij}$ for all $i \in I, j \in J$ and $t \in T$. Let M^R_{it} be the solution of the relaxed problem if only the single resource constraint for resource-period pair (i, t) is considered, and let M^R be the solution when the constraints for all resource-period pairs are considered. With this notation the minimax solution of the relaxed problem is:

$$M^R_{it} = \frac{\sum_{j\in J_{s(t)}} a_{ijt}D_{jt} - B_{it}}{\sum_{j\in J_{s(t)}} a_{ijt}D_{jt}/\alpha_{jt}}, \tag{15.1}$$

and

$$M^R = \max_{i\in I, t\in T} M^R_{it}. \tag{15.2}$$

Given the solution of the relaxed problem M^R and $X^R_{jt} = f^{-1}_{jt}(M^R)$ for all $j \in J_{s(t)}$ and $t \in T$, the following optimality properties form the basis for an algorithm for PROBLEM MPL:

(a) Suppose, for a given j, $\alpha_{jt} < M^R$ (note that $\alpha_{jt} = f_{jt}(0)$) for $t = 1, 2, \ldots, t_0$. Then the minimal solution of PROBLEM MPL satisfies $X^*_{jt} = 0$ and $f_{jt}(X^*_{jt}) < M^*$ for $t = 1, 2, \ldots, t_0$.

(b) Suppose, for a given j, $X^R_{jt} < X^R_{jt_1}$ for $t = t_1 + 1, t_1 + 2, \ldots, t_2$ $(t_2 > t_1)$. Then,

(i) The minimal solution of PROBLEM MPL satisfies $X^*_{jt} = X^*_{jt_1}$ for $t = t_1 + 1, t_1 + 2, \ldots, t_2$.

(ii) For any feasible solution of PROBLEM MPL with $X_{jt} = X_{jt_1}$ for $t = t_1 + 1, t_1 + 2, \ldots, t_2$, either $f_{jt}(X_{jt_1}) \leqslant f_{jt_1}(X_{jt_1})$ or $f_{jt}(X_{jt_1}) < M^*$ for $t = t_1 + 1, t_1 + 2, \ldots, t_2$.

Property (a) allows for fixing some variables at zero, while part (i) of Property (b) allows for the collapsing of strings of variables to a single variable. Part (ii) of Property (b) assures that the performance function terms of the collapsed variables can indeed be deleted from the objective function. Luss and Smith (1988) propose an efficient algorithm for PROBLEM MPL. The algorithm, outlined below, is an extension of the activity-deletion algorithm presented for PROBLEM RESOURCE.

Algorithm for PROBLEM MPL

Step 1. Check whether $X_{jt} = D_{jt}$ for all $j \in J$ and $t \in T$ is feasible. If so, STOP; $M^* = 0$ and $X^*_{jt} = D_{jt}$ for all (j, t).

Step 2. Initialize $J_{s(t)} = J$ and $a_{ijt} = a_{ij}$ for all $i \in I, j \in J$ and $t \in T$.

Step 3. Solve the relaxed problem with sets $J_{s(t)}$.

Step 4. Use Property (a) above to fix activity-period pairs at zero.

Step 5. Use Property (b) above to collapse strings of activity-period pairs into a single pair.

Step 6. If some activity-period pair was set to zero or collapsed into another pair, update the sets $J_{s(t)}$ and parameters a_{ijt}, as needed, and return to Step 3. Otherwise, STOP; $M^* = M^R$ and $X^*_{jt} = X^R_{jt}$ for all $j \in J_{s(t)}$ and $t \in T$.

Klein et al. (1992) extend the algorithm to handle arbitrary lower bounds on variables X_{jt}. This extension is needed to obtain the lexicographic minimax solution. Although the analysis of the extended problem is quite involved, the resulting algorithm is similar to that presented for PROBLEM MPL.

3.3. The Lexicographic Minimax Multiperiod Problem

Let $f^{(np)}(X)$ be the vector of the np elements $f_{jt}(X_{jt})$, sorted in nonincreasing order. The lexicographic minimax extension of PROBLEM MULTIPERIOD, referred to as PROBLEM L-MULTIPERIOD, is formulated as follows:

PROBLEM L-MULTIPERIOD

$$M^L = \operatorname{lexmin}_X [\, f^{(np)}(X)] \tag{16}$$

so that (12.2)–(12.4) are satisfied.

The methodology for solving PROBLEM L-MULTIPERIOD is similar to that for PROBLEM L-RESOURCE. It repeatedly solves PROBLEM MULTIPERIOD, where at each iteration some of the variables are fixed at their lexicographically optimal value, denoted as X^L_{jt}, and excluded from further consideration. The following properties are used to determine which variables can be fixed:

(a) Suppose that the minimal solution X^* of PROBLEM MULTIPERIOD without the upper bounds has $X^*_{jt} \geqslant U_{jt}$ for one or more activity-period pairs. Then, $X^L_{jt} = U_{jt}$ for these pairs.

(b) Suppose $X^*_{jt} \leqslant U_{jt}$ for all activity-period pairs, and suppose (i_c, t_c) is a critical resource-period pair constraint. Then, $X^L_{jt} = X^*_{jt}$ for all activity-period pairs (j, t_c) with $a_{i_c j} > 0$.

Suppose that at some iteration activity-period pair (j_1, t_1) is fixed. As this activity-period pair is deleted from the minimax problem that will be solved at the next iteration, the following lower and upper bounds need to be revised:

$$L_{j_1 t} \leftarrow \max[L_{j_1 t}, X^*_{j_1 t_1}], \quad t = t_1 + 1, t_1 + 2, \ldots, p, \tag{17.1}$$

$$U_{j_1 t} \leftarrow \min[U_{j_1 t}, X^*_{j_1 t_1}], \quad t = 1, 2, \ldots, t_1 - 1. \tag{17.2}$$

Property (b) above is slightly more restrictive than that shown for PROBLEM L-RESOURCE. An outline of the algorithm, described in Klein et al. (1992) and Betts et al. (1994), is as follows:

Algorithm for PROBLEM L-MULTIPERIOD

Step 1. Formulate the initial PROBLEM MULTIPERIOD with all activity-period pairs (j, t), resource-period pairs (i, t), and lower and upper bounds L_{jt} and U_{jt}.

Step 2. Solve PROBLEM MULTIPERIOD without the upper bounds. Determine the minimal solution X^*.

Step 3. Using properties (a) and (b) described above, fix activity-period pairs $X^L_{jt} = U_{jt}$ or $X^L_{jt} = X^*_{jt}$, as appropriate.

Step 4. Delete activity-period pairs fixed at Step 3. If all activity-period pairs were fixed; STOP.

Step 5. Update lower and upper bounds according to (17) and the supplies B_{it}. Delete resource-period pair constraints not needed by any remaining activity-period pair in the updated formulation. Return to Step 2.

Luss and Smith (1988) and Klein et al. (1992) provide computational results for PROBLEM MPL and its lexicographic extension. For PROBLEM MPL, most of the computational effort is spent on computing the initial values of M^R_{it}. This requires an effort of $O(\theta mnp)$, where θ is the fraction of positive a_{ij}'s, and m, n and p are the number of resources, activities and periods, respectively. The minimax solutions for problems with 100 activities, 100 resources, 10 time periods, and $\theta = 0.5$ were obtained in less than one second on an Amdahl 5890-300E computer. The computing times were, on average, about 200 times faster than those required by linear programming software. The lexicographic minimax solution is obtained by repeatedly solving (up to np) problems in the format of PROBLEM MPL with general lower and upper bounds. The number of iterations depends on θ and was, in practice, quite small. The lexicographic solutions for these problems required 30–100 iterations and were obtained in about three seconds.

3.4. Nonstorable Resources

Although many resource allocation applications consider only storable resources, this is not always the case. For example, idle machine time, or idle people's time, in one period is a lost resource that cannot be carried over to the next period, unless it is used to generate inventory.

Resource constraints for nonstorable resources are expressed in terms of noncumulative decision variables x_{jt} and noncumulative resource supplies b_{it}. Likewise, each performance function $f_{jt}(\cdot)$ depends on the sum of noncumulative decision variables, namely, $\sum_{\tau=1}^{t} x_{j\tau}$. Alternatively, nonstorable resource constraints can be expressed as $\sum_{j \in J}$

$a_{ij}(X_{jt} - X_{j,t-1}) \leq b_{it}$, in which case each performance function $f_{jt}(\cdot)$ would depend on the single variable X_{jt}. Either formulation violates one of the key properties (separable objective function or knapsack resource constraints) used to develop the efficient algorithms presented in this paper. Section 5 provides a brief review of resource allocation problems with more general structures, which include multiperiod problems with nonstorable resources.

4. RESOURCE ALLOCATION WITH SUBSTITUTABLE RESOURCES

In this section we review resource allocation models in which certain substitutions among resources are allowed. Indeed, in many practical resource allocation problems, effective management of substitutable resources is essential. Consider, for example, the assembly of diverse high-tech circuit boards, where each of these is composed of many different components like integrated circuits. Prudent allocation of components among circuit boards should consider possible substitutions among the components. Myopic allocation of substitutes to relatively more important boards can lead to severe, unacceptable shortages for many other boards. Thus, models that seek optimal allocation of substitutes among all boards are needed.

4.1. Framework for Problems with Substitutable Resources

We extend PROBLEM RESOURCE (and PROBLEM L-RESOURCE) to models that explicitly consider substitutions among resources. Each resource is classified into one of several groups, where resources in one group cannot substitute for, or be substituted by, any resource that is not in the same group. Within each group, certain substitutions among resources are allowed, as described below.

We use the same notation as before with the following additions:

- q = Index for groups of substitutable resources $q = 1, 2, \ldots$. Let Q be the set of indices q of all groups.
- i, k = Indices for resources; $i, k = 1, 2, \ldots, m$.
- I_q = Set of resources in group q.
- I_0 = Set of singleton resources that cannot substitute for, or be substituted by, any other resource (note that $0 \notin Q$).
- y_{ik} = Amount of resource i used as a substitute for resource k, where y_{ii} is the amount of resource i used directly—not as a substitute.
- $\gamma(i)$ = Set of resources that can be substituted by resource i (we include i in $\gamma(i)$).
- $\delta(i)$ = Set of resources that can substitute for resource i (we include i in $\delta(i)$).

The minimax resource allocation problem with substitutable resources, referred to as PROBLEM SUBRES (for SUBstitutable RESources), is formulated as follows.

PROBLEM SUBRES

$$S^* = \min_x [\max_{j \in J} f_j(x_j)] \tag{18.1}$$

so that

$$\sum_{j \in J} a_{ij} x_j \leq b_i, \quad i \in I_0, \tag{18.2}$$

$$\sum_{k \in \gamma(i)} y_{ik} \leq b_i, \quad i \in I_q \text{ and } q \in Q, \tag{18.3}$$

$$\sum_{k \in \delta(i)} y_{ki} = \sum_{j \in J} a_{ij} x_j, \quad i \in I_q \text{ and } q \in Q, \tag{18.4}$$

$$l_j \leq x_j \leq u_j, \quad j \in J, \tag{18.5}$$

$$y_{ik} \geq 0, \quad k \in \gamma(i), i \in I_q \text{ and } q \in Q. \tag{18.6}$$

The performance functions $f_j(x_j)$ are assumed to be continuous, strictly decreasing, and invertible. For each group $q \in Q$, constraints (18.3) ensure that the total amount used of resource i does not exceed its supply. Constraints (18.4) ensure that the amount used of resource i, or its substitutes, suffices to sustain the selected activity levels for all $j \in J$. For simplicity, we assume that a single unit of one resource can substitute for a single unit of another resource. However, the formulation can readily handle, through change of units, cases where $\beta_{ik} > 0$ units of resource i can substitute for one unit of resource k for $k \in \gamma(i)$, provided the factors β_{ik} are consistent across all possible substitutions within each group q.

An effective solution approach repeatedly solves relaxations of PROBLEM SUBRES, where each relaxed problem is in the format of PROBLEM RESOURCE. For each group $q \in Q$, the initial relaxed problem is formulated by assuming that all resources $i \in I_q$ can substitute for each other. This results in an aggregation of constraints (18.3)–(18.4) for all $i \in I_q$ to a single knapsack resource constraint,

$$\sum_{j \in J} \Big(\sum_{i \in I_q} a_{ij} \Big) x_j \leq \sum_{i \in I_q} b_i, \quad q \in Q. \tag{19}$$

The relaxed problem can therefore be solved by any of the algorithms for PROBLEM RESOURCE.

Let x^1 be a (not necessarily feasible) vector of selected activity levels. Consider a group $q \in Q$ of resources. A *self-sufficient subset* F_q for activity levels x^1 is defined as a subset of resources in I_q that satisfies the following conditions:

(a) Demands by activity levels x^1 for any resource $i \in F_q$ can be met by resource i or possible substitutes for i within the subset F_q.

(b) Resources in F_q cannot substitute for any resources that are not in F_q.

Once a solution x^1 is obtained for the relaxed problem with aggregated constraints (19), we need to check for each group $q \in Q$ whether the solution is feasible or not for PROBLEM SUBRES. If the solution is feasible for group q, then the set I_q itself is self-sufficient for x^1. For each group q for which feasibility is violated, a self-sufficient subset of resources $F_q \subset I_q$ is identified and

excluded from further consideration. As a result, a new relaxed problem is formulated with tighter resource constraints in the format of (19), whose solution has a larger objective function value. An outline of the algorithm is given below.

Algorithm for PROBLEM SUBRES

Step 1. Formulate the relaxed problem, in the format of PROBLEM RESOURCE, with resource constraints (18.2), a single aggregated constraint (19) for each $q \in Q$, and constraints (18.5).

Step 2. Find the minimal solution to the relaxed problem by one of the algorithms for PROBLEM RESOURCE.

Step 3. Find whether the solution for the relaxed problem is feasible for PROBLEM SUBRES. If feasible, it is optimal; determine feasible (and, hence, also optimal) values for the variables y_{ik} that satisfy constraints (18.3)–(18.4) and STOP.

Step 4. If not feasible, find for each violated group q a self-sufficient subset F_q, and formulate a new aggregated constraint (19) with resources $I_q \leftarrow I_q \backslash F_q$.

Step 5. Formulate a new relaxed problem with the new aggregated resource constraints for the violated groups. (All other resource constraints, including (18.2), are deleted.) Return to Step 2.

This algorithm is effective if the following conditions hold:
(a) An efficient algorithm exists for solving the relaxed problem, PROBLEM RESOURCE.
(b) An efficient method is available for checking feasibility and for identifying self-sufficient subsets.
(c) The number of iterations is small.

As shown in §2, PROBLEM RESOURCE can be solved efficiently; and, as shown in subsequent subsections, conditions (b) and (c) are satisfied for a variety of substitutional relations. The solution approach can be extended to handle the more general descending performance functions.

The lexicographic minimax extension of PROBLEM SUBRES, referred to as PROBLEM L-SUBRES, is formulated as follows.

PROBLEM L-SUBRES

$$S^L = \operatorname*{lexmin}_x [\, f^{(n)}(x)\,] \tag{20}$$

so that (18.2)–(18.6) are satisfied.

Extension of the algorithm for PROBLEM SUBRES to PROBLEM L-SUBRES is similar to the extension of the algorithm for PROBLEM RESOURCE to PROBLEM L-RESOURCE.

Algorithm for PROBLEM L-SUBRES

Step 1. Formulate the initial PROBLEM SUBRES with activity set J and resource constraint sets I_0 and I_q for each $q \in Q$.

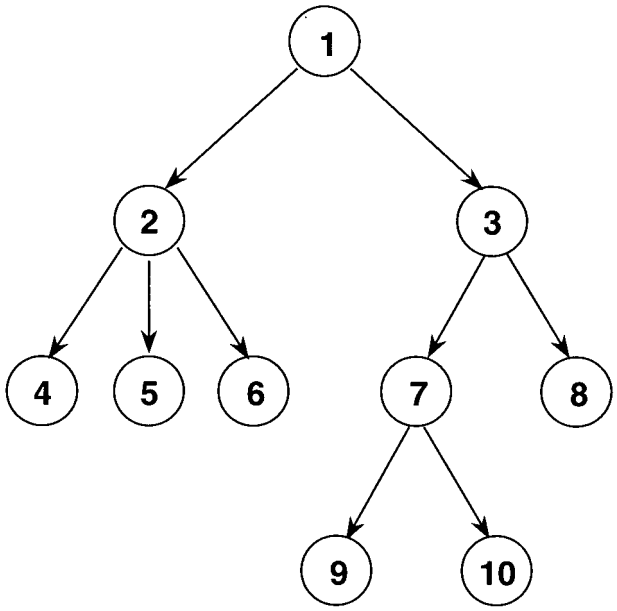

Figure 4. Substitutional relations represented by a tree.

Step 2. Solve PROBLEM SUBRES. Determine the minimal solution x^*.

Step 3. (i) Fix $x_j^L = u_j$ for any activity with $x_j^* = u_j$.
(ii) Suppose constraint $i_c \in I_0$ is critical. Fix $x_j^L = x_j^*$ for all j with $a_{i_c j} > 0$.
(iii) Suppose the aggregated constraint for group q is critical. Fix $x_j^L = x_j^*$ for all activities which use one or more of the resources that form the aggregated constraint.

Step 4. Delete from J all activities whose values were fixed. If J is now empty, go to Step 5. Otherwise, delete the critical resources from sets I_0 and I_q, update parameters for the next PROBLEM SUBRES to be solved, and go to Step 2.

Step 5. The lexicographic minimax solution x^L is obtained. Solve the feasibility problem for the initial formulation of PROBLEM SUBRES to determine feasible (and optimal) values for the variables y_{ik}. STOP.

4.2. Transitive Substitutional Relations

We present below resource allocation models, where the substitutional relations among the resources are transitive. That is, if resource i_1 is a substitute for resource i_2 and resource i_2 is a substitute for resource i_3, then resource i_1 is also a substitute for resource i_3.

Resource Substitutions Represented by Trees. Consider a group of resources where the substitutional relations can be represented by a tree. An example is shown in Figure 4. Each node represents a resource, and a link from node i to k indicates that resource i is a direct substitute for resource k. In addition, by transitivity, resource i is a substitute for all descendants of node k. Thus, for example,

resource 3 is a direct substitute for resources 7 and 8, and a substitute by transitivity for resources 9 and 10.

Klein and Luss (1991) describe an algorithm for PROBLEM SUBRES for substitutional relations that are represented by a tree for each group $q \in Q$. The relaxed problem in each iteration is in the format of PROBLEM RESOURCE. Let $x^1 = (x_1^1, x_2^1, \ldots, x_n^1)$ be the minimal solution of the relaxed problem at the first iteration. Taking advantage of the tree structure, it is easy to determine, through a backtracking procedure, whether the resources in any group have sufficient supplies to sustain activity levels x^1. For convenience, the resources in each group are indexed so that each node has a larger index than its unique predecessor. Starting at the node with the largest index (in Figure 4, node 10), we check whether its supply b_{10} is sufficient to sustain x^1. If so, we mark it by a plus sign, and if not, we mark it by a minus sign and debit its predecessor (node 7) by the shortage. We repeat this procedure with every unmarked node in decreasing order of the indices until the root is marked. If the root is marked by a plus sign, the resources in this group can sustain activity levels x^1. If the root is marked by a minus sign, the associated resource supplies cannot sustain x^1 and at least one subtree within the tree can be identified with its root (the node with the smallest index) marked by a plus sign. The resources associated with each such subtree, as well as the union of these subtrees, form a self-sufficient subset of resources for activity levels x^1.

In the next iteration, all self-sufficient subsets are dropped from the trees and a new relaxed problem is formulated. The minimax objective value of the new relaxed problem will be larger than that of the current problem. The values y_{ik}^* are also determined by a simple backtracking procedure. First, the resource with the largest index (resource 10) is allocated, then the resource with the next largest index, etc. This scheme ensures that a substitute is used only if the supplies of all its descendants have been exhausted. Klein and Luss (1991) present computational results for problems with linear performance functions. Problems with 1,000 resources, 1,000 activities, and a fraction of $\theta = 0.015$ of positive a_{ij}s were solved in less than a second on an Amdahl 5890 computer. Although the number of relaxed problems that need to be solved is at most n, it was less than 10 in all reported examples. The lexicographic minimax solutions were obtained for the problem sizes mentioned above in about 5 seconds.

Resource Substitutions Represented by Graphs. Substitutional relations represented by trees are somewhat restrictive as each node has only one predecessor (the root has none). A natural extension includes representation of substitutional relations by general graphs. An example for a single group is shown in Figure 5, which is the same as Figure 4 with three additional links: from node 3 to 6, from node 4 to 9, and from node 5 to 9. Thus, both resources 2 and 3 are direct substitutes for resource 6; and resources 4, 5, and 7 are direct substitutes for resource 9.

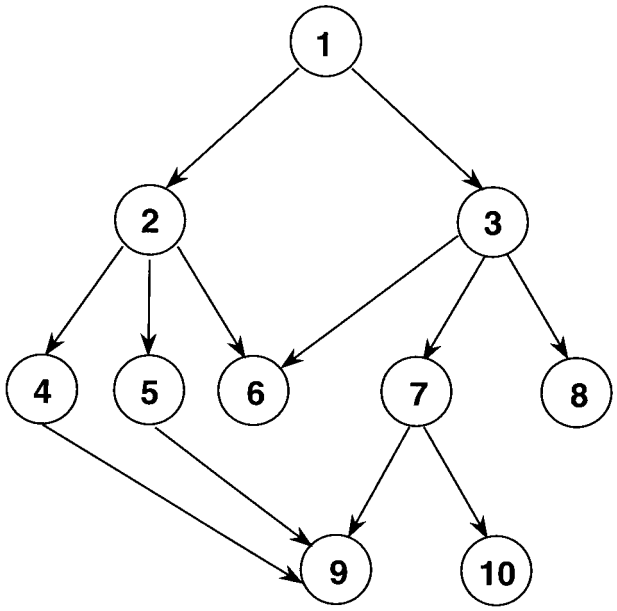

Figure 5. Transitive substitutional relations represented by a graph.

Again, substitution by transitivity is assumed, for instance, resource 2 can substitute for resources 4, 5, 6, and 9. The graphs would not contain any cycles because the resources represented by the nodes along a cycle can substitute for each other, implying that they can simply be represented as a single resource (node).

Klein et al. (1993) adapt the algorithm for PROBLEM SUBRES to such substitutional relations. The backtracking methods, used for tree-type substitutions to check feasibility of the relaxed solution (referred to as x^1) and to identify self-sufficient subsets, cannot be used for general graphs. Instead, the graph for each group $q \in Q$ is augmented to the following network flow problem. A source node s_q and a sink node t_q are added with links (s_q, i) from s_q to i for each $i \in I_q$, with upper bounds b_i, and links (i, t_q) from i to t_q for each $i \in I_q$, with upper bounds $\Sigma_{j \in J} a_{ij} x_j^1$. All links (i, k), which describe the possible substitutions within group q, have infinite capacity. It is shown that if the maximal flow from s_q to t_q is equal to $\Sigma_{i \in I_q} \Sigma_{j \in J} a_{ij} x_j^1$, then the resources in group q can sustain the selected levels x^1. However, if the maximal flow is less than the above, these resources cannot sustain x^1. In the latter case, the max-flow solution is used to identify self-sufficient subsets of resources. Specifically, all *labeled* nodes in I_q form a self-sufficient subset, where $i \in I_q$ is labeled if any of the following conditions holds: (a) The flow on the link (s_q, i) is less than b_i; (b) Node k is labeled and $k \in \delta(i)$; (c) Node k is labeled, $k \in \gamma(i)$ and the flow on link (i, k) is positive. These self-sufficient subsets are deleted from group q in the next iteration.

Extension of the minimax algorithm to derive the lexicographic minimax solution follows the algorithm described for PROBLEM L-SUBRES. Klein et al. (1993) report

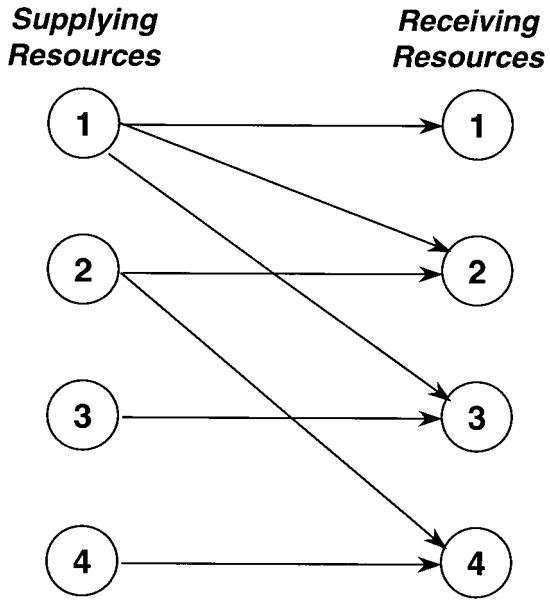

Figure 6. Nontransitive substitutional relations.

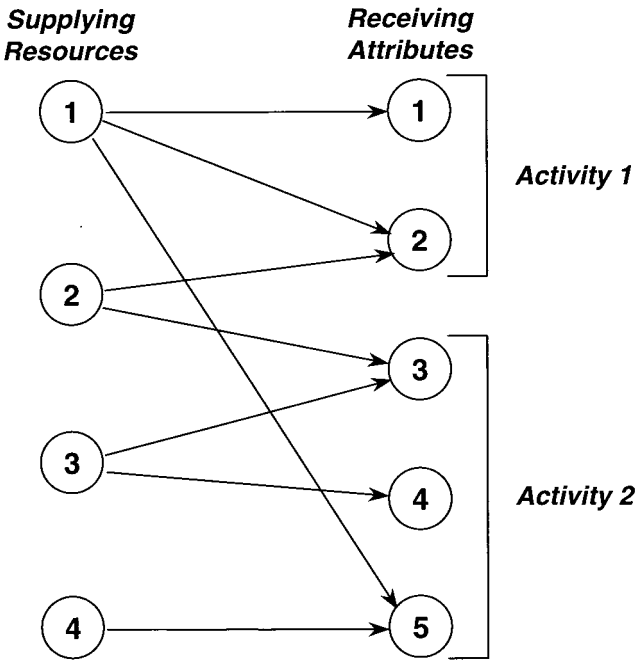

Figure 7. Activity-dependent substitutional relations.

computational results for problems with linear performance functions. Minimax problems with 1,000 resources and 1,000 activities (or even larger problems) were solved in less than one second on an Amdahl 3090 computer. Lexicographic solutions were obtained for most of the reported problems in less than 30 seconds.

4.3. More General Substitutional Relations

Figure 6 shows an example of nontransitive substitutional relations in a single group q. Each resource is represented by a "supplying" node and a "receiving" node. Substitution of resource k by resource i is indicated by a link from supplying node i to receiving node k. Thus, resource 1 can substitute for resources 2 and 3, and resource 2 can substitute for resource 4. However, resource 1 cannot substitute for resource 4. Figure 7 shows activity-dependent substitutional relations, that is, resource i_1 may substitute for resource i_2 when used for activity j_1, but not when used for activity j_2. To model these relations, the receiving nodes represent *attributes*, rather than resources, where each unit of activity j requires a_{kj} units of attribute k. The variables y_{ik} are redefined to represent the amount of resource i used for attribute k. In Figure 7, four resources supply five attributes, where attributes 1 and 2 are required by activity 1, and attributes 3–5 are required by activity 2. Thus, for example, although both resources 1 and 2 can supply attribute 2, resource 1 (but not 2) can supply attribute 5 and resource 2 (but not 1) can supply attribute 3.

As discussed in Klein et al. (1994), these more flexible models can also be solved by an adaptation of the algorithm for PROBLEM SUBRES, similar to that described for transitive substitutional relations. Finding whether a solution x^1 of a relaxed problem is feasible for group of resources $q \in Q$ and identifying self-sufficient subsets of resources is done by employing a max-flow algorithm on a network constructed from the corresponding graph that represents the substitutional relations.

Pang and Yu (1988) present a variant of these models with linear performance functions. They describe a parametric transportation solution method for their variant. Starting with a feasible solution, the minimax objective function value is reduced at each iteration, while maintaining feasibility, until optimality is achieved. The reported computational results exhibit significant reduction in computing time over those obtained by linear programming software.

Extension of the minimax solution for these models to the lexicographic minimax solution is obtained by the algorithm described for PROBLEM L-SUBRES.

4.4. Multiperiod Allocation of Substitutable Resources

The combined formulation of PROBLEM MULTIPERIOD and PROBLEM SUBRES, referred to as PROBLEM MSRA (Multiperiod Substitutable Resource Allocation), is as follows:

PROBLEM MSRA

$$W^* = \min_X [\max_{j \in J, t \in T} f_{jt}(X_{jt})] \tag{21.1}$$

so that

$$\sum_{j \in J} a_{ij} X_{jt} \leq B_{it}, \quad i \in I_0 \text{ and } t \in T, \tag{21.2}$$

$$\sum_{p=1}^{t} \sum_{k \in \gamma(i)} y_{ikp} \leq B_{it}, \quad i \in I_q, q \in Q \text{ and } t \in T, \tag{21.3}$$

$$\sum_{p=1}^{t} \sum_{k \in \delta(i)} y_{kip} = \sum_{j \in J} a_{ij} X_{jt}, \quad i \in I_q, q \in Q \text{ and } t \in T, \tag{21.4}$$

$$X_{jt} \geq X_{j,t-1} \ (X_{j0} = 0), \quad j \in J \text{ and } t \in T, \tag{21.5}$$

$$L_{jt} \leq X_{jt} \leq U_{jt}, \quad j \in J \text{ and } t \in T, \tag{21.6}$$

$$y_{ikt} \geq 0, \quad k \in \gamma(i), \ i \in I_q, \ q \in Q \text{ and } t \in T. \tag{21.7}$$

The notation is the same as before, where y_{ikp} is the amount of resource i used as a substitute for resource k at period p, and y_{iip} is the amount of resource i used directly—not as a substitute—at period p. Constraints (21.2) are the multiperiod resource constraints for resources in I_0. Constraints (21.3) and (21.4) extend constraints (18.3) and (18.4) to a multiperiod setting.

As discussed in Klein et al. (1994, 1995), the algorithm described for PROBLEM SUBRES can be readily extended to solve PROBLEM MSRA. The relaxed problem is in the format of PROBLEM MULTIPERIOD and can be solved by the algorithms described in Section 3. Substitutional relations among resources in a multiperiod setting are represented by multiperiod versions of the graphs in Figures 4 through 7. The graphs are extended to network flow problems, which, in turn, are used to determine whether a solution is feasible or not and to identify self-sufficient subsets of resources. However, unlike for PROBLEM SUBRES, a self-sufficient subset in one iteration may not be self-sufficient in subsequent iterations.

Nguyen and Stone (1993) describe a variant of PROBLEM MSRA with activity-dependent substitutions and linear performance functions. They present a primal-dual solution approach. The algorithm starts with an infeasible minimax objective value. At each iteration, a measure of infeasibility is computed by solving a max-flow problem. The measure is used to increase the estimate of the minimax objective function value for the next iteration, until optimality is established. Computational results exhibit computing times that are significantly smaller than those obtained by linear programming software.

Extension of the minimax solution for PROBLEM MSRA to the lexicographic minimax solution is similar to that described in §3 for multiperiod problems.

5. RELATED MINIMAX PROBLEMS

As already mentioned, there exist important resource allocation problems, like the multiperiod problem with nonstorable resources, that do not conform to the special structures considered in §§2 through 4. Klein et al. (1994) present the following more general formulation of a minimax optimization problem:

$$W^* = \min_x [\max_{j \in J} f_j(x_j)] \tag{22.1}$$

so that

$$Ax \leq By + b, \tag{22.2}$$

$$Cx \leq d, \tag{22.3}$$

$$x \geq l, \tag{22.4}$$

$$y \geq 0. \tag{22.5}$$

The performance functions $f_j(x_j)$ are continuous, strictly decreasing, and invertible. The vector x is composed of performance variables that appear in the objective function, whereas the vector y is composed of auxiliary variables that are not part of the objective function. A, B, and C are matrices of parameters, and b, d, and l are vectors of parameters. Constraints (22.2) are called the primary constraints and include both vectors x and y. Constraints (22.3) are called secondary constraints and include only x. Formulation (22) can be reduced to the multiperiod problem with storable and nonstorable resources and to the various resource allocation problems with substitutable resources.

Klein et al. describe a solution approach that alternates between solving a relaxed version of problem (22) and solving Phase I of the simplex algorithm for a linear programming problem to check whether the solution to the relaxed problem is feasible for (22). Brown (1984) presents a model that is similar to (22), and provides a solution approach that improves at each iteration a lower bound for the objective function value by solving a series of knapsack problems and employing a dual simplex algorithm. The solution approaches above can be extended to handle the more general descending performance functions. Although such solution approaches for problem (22) (and similar variants) are quite elegant, the computation time required to solve large-scale problems may be large. Moreover, the lexicographic minimax solution may not necessarily be obtained by the repeated solution of the minimax problem, as the minimal solution to the minimax problem may not be the right one to select or it may even not exist.

Consider now a minimax objective function, where the performance functions (indexed by h) are of the form $f_h(x) = f_h(\sum_{j \in J} w_{jh} x_j)$, where the parameters w_{jh} are given. The minimax objective function is then $\min_x[\max_h f_h(\sum_{j \in J} w_{jh} x_j)]$. Formulation (22) can handle such performance functions by defining new performance variables $z_h = \sum_{j \in J} w_{jh} x_j$, one for each h. The minimax objective function reduces then to $\min_z[\max_h f_h(z_h)]$ and the variables x_j become auxiliary variables. New equality constraints $z_h = \sum_{j \in J} w_{jh} x_j$ (formulated, for each h, as two inequalities) are added to (22.2). Recall that the multiperiod problem with nonstorable resources can be formulated with such nonseparable terms in the minimax objective function. As another example, suppose that each h represents one scenario of possible parameter values w_{jh} for all $j \in J$. Each performance function $f_h(x)$ represents the outcome, as a function of x, under scenario h. The minimax problem then determines the decision vector x, that minimizes the objective function value under the worst possible scenario. Kouvelis and Yu (1997) refer to such a model as the robust optimization problem, and provide numerous applications to such models.

Various papers examine variants of minimax (or maximin) problems with either non-separable, linear performance functions, and/or linear constraints that are not of the knapsack inequality type constraints. Kuno et al. (1991), Karabati et al. (1995), Yamada et al. (1996), Yu (1996), and Kouvelis and Yu (1997) describe algorithms

for problems with a non-separable objective function and a single knapsack resource constraint. The first reference focuses on a problem with continuous decision variables, and the others discuss algorithms for problems with integer variables. Papers that examine problems with general linear constraints include Kaplan (1974), Posner and Wu (1981), Ahuja (1985), Eiselt (1986), and Kuno et al. (1989). Marchi and Oviedo (1992) apply the lexicographic minimax approach to multiple objective linear programs. Dutta and Vidyasagar (1977), Madsen and Schjaer-Jacobsen (1978), and Bazaraa and Goode (1982) describe algorithms for more general nonlinear minimax problems. Brown (1989) classifies and describes a variety of references on minimax and maximin constrained optimization problems.

We conclude this section by discussing an important class of problems in which the constraints represent flows on networks. Various applications in logistics, transportation, and telecommunications can be modeled as minimax network problems where supplies are shipped from given sources to destinations. The flow conservation constraints are obviously not of the knapsack type. However, their special structure can still be exploited to design effective solution approaches that repeatedly employ efficient network flow algorithms. Megiddo (1974, 1977) presents a lexicographic minimax network flow model for distributing flows from multiple sources or among multiple destinations. Brown (1979a) provides an algorithm for a minimax model which is concerned with allocation of supplies among competing destinations, where each destination is associated with a performance function. Brown (1983) extends the previous model by including performance functions for a designated subset of links. Ahuja (1986) presents a minimax algorithm for the more restricted transportation problem. Betts and Brown (1997) present a proportional equity network flow problem. The flow on a designated link is maximized under constraints that other links will receive flows that are within a proportional range of that received by the designated link. This objective is, though, quite different from the minimax and lexicographic minimax objectives.

6. FINAL REMARKS

In this expository paper, we concentrated on resource allocation problems in which it is desirable to allocate numerous resources among competing activities in an equitable way. This led to defining the lexicographic minimax objective that emphasizes equitable resource sharing among all activities. This objective is particularly of interest in large-scale problems, with numerous resources and activities, where typically, each activity uses only a small subset of the resources.

Since the lexicographic minimax solution approach often requires solving the minimax problem many times, it is imperative that the latter problem be solved very quickly. This, in turn, led to the imposition of certain modeling assumptions that facilitated the design of efficient algorithms, and, at the same time, were not too restrictive for diverse application areas. The exposition covered primarily three classes of problems: the basic problem with multiple resource constraints, multiperiod problems, and problems with substitutable resources. Other minimax problems that do not conform to the imposed assumptions were briefly mentioned with appropriate references.

Although the lexicographic minimax objective appears to be intuitively appealing for resource allocation problems, it is not well-known or understood. It is hoped that this paper will popularize the approach among practitioners. The classes of problems presented are quite general, and may be used as stand-alone models or integrated as modules within more complex models. The results of the models are easy to explain to decision makers, which enhances credibility. Furthermore, post-optimization and parametric analysis, which are critically important for real-life applications, can be easily implemented.

We have also found that the nice mathematical structures facilitate the design of efficient, elegant algorithms. It is hoped that the exposition provides insights to various resource allocation problems, so that it will encourage researchers to explore new, related problems and propose effective solution approaches. Not only are the presented algorithms efficient, but they are also easy to implement as effective software tools. Modern computer science methods, including efficient data structures and parallel computations, can be used to further enhance implementation of these algorithms.

ACKNOWLEDGMENTS

Work for this paper was done while the author was employed at AT&T Labs. The author thanks John G. Klincewicz, Muralidharan S. Kodialam, Beth S. Munson, and Moshe B. Rosenwein for their helpful comments and constructive suggestions.

REFERENCES

Agnihothri, S., U. S. Karmarkar, and P. Kubat. 1982. Stochastic allocation rules. *Oper. Res.* **30** 545–555.

Ahuja, R. K. 1985. Minimax linear programming problem. *Oper. Res. Letters* **4** 131–134.

——. 1986. Algorithms for the minimax transportation problem. *Naval Res. Logistics Quarterly* **33** 725–739.

Bazaraa, M. S., and J. J. Goode. 1982. An algorithm for solving linearly constrained minimax problems. *European J. Oper. Res.* **11** 158–166.

Betts, L. M., and J. R. Brown. 1997. Proportional equity flow problem for terminal arcs. *Oper. Res.* **45** 521–535.

——, ——, H. Luss. 1994. Minimax resource allocation problems with ordering constraints. *Naval Res. Logistics* **41** 719–738.

Bitran, G. R., and A. C. Hax. 1981. Disaggregation and resource allocation using convex knapsack problems with bounded variables. *Management Sci.* **27** 431–441.

Brown, J. R. 1979a. The sharing problem. *Oper. Res.* **27** 324–340.

——. 1979b. The knapsack sharing problem. *Oper. Res.* **27** 341–355.

——. 1983. The flow circulation sharing problem. *Math. Programming* **25** 199–227.

——. 1984. The linear sharing problem. *Oper. Res.* **32** 1087–1106.

——. 1989. Sharing (maximin and minimax) constrained optimization. Working Paper. Graduate School of Management, Kent State University, Kent, OH.

——. 1991. Solving knapsack sharing problems with general tradeoff functions. *Math. Programming* **51** 55–73.

——. 1994. Bounded knapsack sharing. *Math. Programming* **67** 343–382.

Chaddha, R. L., W. W. Hardgrave, D. J. Hudson, M. Segal, and J. W. Suurballe. 1971. Allocation of total sample size when only the stratum means are of interest. *Technometrics* **13** 817–831.

Cosares, S., D. N. Deutsch, I. Saniee, and O. J. Wasem. 1995. SONET toolkit: a decision support system for designing robust and cost-effective fiber-optic networks. *Interfaces* **25**(1) 20–40.

Czuchra, W. 1986. A graphical method to solve a maximin allocation problem. *European J. Oper. Res.* **26** 259–261.

Daskin, M. S. 1995. *Network and Discrete Location: Models, Algorithms, and Applications*. John Wiley, New York.

Dutta, S. R. K., and M. Vidyasagar. 1977. New algorithms for constrained minimax optimization. *Math. Programming* **13** 140–155.

Eiselt, H. A. 1986. Continuous maximin knapsack problems with GLB constraints. *Math. Programming* **36** 114–121.

Garfinkel, R. S., and M. Rao. 1976. Bottleneck linear programming. *Math. Programming* **11** 291–298.

Gawande, M. 1996. Workforce scheduling problems with side-constraints. Presented at INFORMS Meeting, Washington, DC, Spring.

Ibaraki, T., and N. Katoh. 1988. *Resource Allocation Problems: Algorithmic Approaches*. MIT Press, Cambridge, MA.

Ichimori, T. 1984. On min-max integer allocation problems. *Oper. Res.* **32** 449–450.

Jacobsen, S. 1971. On marginal allocation in single constraint min-max problems. *Management Sci.* **17** 780–783.

Kaplan, S. 1974. Application of programs with maximin objective functions to problems of optimal resource allocation. *Oper. Res.* **22** 802–807.

Katoh, N., and T. Ibaraki. 1998. Resource allocation problems. D.-Z. Du and P. M. Pardolos, eds. *Handbook of Combinatorial Optimization* (Vol. 2). Kluwer Academic Publishers, Dordrecht, The Netherlands, 159–260.

King, J. H. 1989. Allocation of scarce resources in manufacturing facilities. *AT&T Technical J.* **68**(3) 103–113.

Klein, R. S., and H. Luss. 1991. Minimax resource allocation with tree structured substitutable resources. *Oper. Res.* **39** 285–295.

——, ——, U. G. Rothblum. 1993. Minimax resource allocation problems with resource-substitutions represented by graphs. *Oper. Res.* **41** 959–971.

——, ——, ——. 1994. Relaxation-based algorithms for minimax optimization problems with resource allocation applications. *Math. Programming* **64** 337–363.

——, ——, ——. 1995. Multiperiod allocation of substitutable resources. *European J. Oper. Res.* **85** 488–503.

——, ——, D. R. Smith. 1992. A lexicographic minimax algorithm for multiperiod resource allocation. *Math. Programming* **55** 213–234.

Kodialam, M. S., and H. Luss. 1998. Algorithms for separable nonlinear resource allocation problems. *Oper. Res.* **45** 274–286.

Kolesar, P., and W. E. Walker. 1974. An algorithm for the dynamic relocation of fire companies. *Oper. Res.* **22** 249–274.

Kouvelis, P., and G. Yu. 1997. *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Kuno, T., H. Konno, and E. Zemel. 1991. A linear-time algorithm for solving continuous maximin knapsack problems. *Oper. Res. Letters* **10** 23–26.

——, K. Mori, and H. Konno. 1989. A modified GUB algorithm for solving linear minimax problems. *Naval Res. Logistics* **36** 311–320.

Lennon, T. M. 1991. Comparison of the Luss-Smith algorithm for resource allocation with two linear programming codes. Unpublished Manuscript. AT&T Bell Laboratories.

Luss, H. 1987. An algorithm for separable nonlinear minimax problems. *Oper. Res. Letters* **6** 159–162.

——. 1991. A nonlinear minimax allocation problem with multiple knapsack constraints. *Oper. Res. Letters* **10** 183–187.

——. 1992. Minimax resource allocation problems: optimization and parametric analysis. *European J. Oper. Res.* **60** 76–86.

——, S. K. Gupta. 1975. Allocation of effort resources among competing activities. *Oper. Res.* **23** 360–366.

——, M. B. Rosenwein, and E. T. Wahls. 1990. Integration of planning and execution: final assembly sequencing. *AT&T Technical J.* **69**(4) 99–109.

——, D. R. Smith. 1986. Resource allocation among competing activities: a lexicographic minimax approach. *Oper. Res. Letters* **5** 227–231.

——, ——. 1988. Multiperiod allocation of limited resources: a minimax approach. *Naval Res. Logistics* **35** 493–501.

Madsen, K., and H. Schjaer-Jacobsen. 1978. Linearly constrained minimax optimization. *Math. Programming* **14** 208–223.

Marchi, E., and A. Oviedo. 1992. Lexicographic optimality in the multiple objective linear programming: the nucleolar solution. *European J. Oper. Res.* **57** 355–359.

Megiddo, N. 1974. Optimal flows in networks with multiple sources and sinks. *Math. Programming* **7** 97–107.

——. 1977. A good algorithm for lexicographically optimal flows in multi-terminal networks. *Bull. American Math. Soc.* **83** 407–409.

Meketon, M. S. 1996. Crew planning models at CONRAIL. Presented at INFORMS Meeting, Washington, DC, Spring.

Mendelson, H., J. S. Pliskin, and U. Yechiali. 1980. A stochastic allocation problem. *Oper. Res.* **28** 687–693.

Mjelde, K. M. 1983a. Max-min resource allocation. *BIT* **23** 529–537.

——. 1983b. *Methods of Allocation of Limited Resources*. John Wiley, New York.

Monden, Y. 1983. *Toyota Production System*. Institute of Industrial Engineers, Industrial Engineering and Management Press, Norcross, GA.

Munson, B. S., and M. S. Kodialam. 1995. Workforce scheduling across multiple teams. Presented at INFORMS Meeting, Los Angeles, Spring.

Nguyen, Q. C., and R. E. Stone. 1993. A multiperiod minimax resource allocation problem with substitutable resources. *Management Sci.* **39** 964–974.

Ogryczak, W. 1997. On the lexicographic minimax approach to location problems. *European J. Oper. Res.* **100** 566–585.

Pang, J.-S., and C.-S. Yu. 1989. A min-max resource allocation problem with substitutions. *European J. Oper. Res.* **41** 218–223.

Porteus, E. L., and L. S. Yormark. 1972. More on min-max allocation. *Management Sci.* **18** 502–507.

Posner, M. E., and C.-T. Wu. 1981. Linear max-min programming. *Math. Programming* **20** 166–172.

Tang, C. S. 1988. A max-min allocation problem: its solutions and applications. *Oper. Res.* **36** 359–367.

Vidal, R. V. V. 1984. A graphical method to solve a family of allocation problems. *European J. Oper. Res.* **17** 31–34.

Yamada, T., M. Futakawa, and S. Kataoka. 1996. Some exact algorithms for the knapsack sharing problem. Unpublished manuscript. Department of Computer Science, The National Defense Academy, Yokosuka, Kanagawa 239, Japan. Also, *IFORS* 14th Triennial Conference, Vancouver, Canada, July.

Yu, G. 1996. On the max-min 0-1 knapsack problem with robust optimization applications. *Oper. Res.* **44** 407–415.

Zeitlin, Z. 1981. Integer allocation problems of min-max type with quasiconvex separable functions. *Oper. Res.* **29** 207–211.

Zipkin, P. H. 1980. Simple ranking methods for allocation of one resource. *Management Sci.* **26** 34–43.