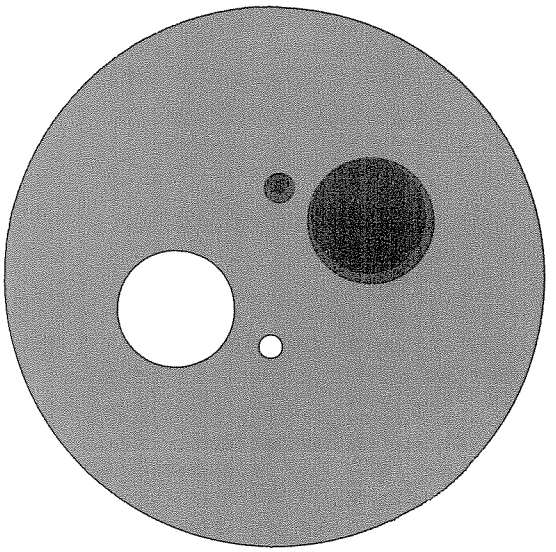
COMPUTER SCIENCES
DEPARTMENT

University of Wisconsin-
Madison



ALGORITHMS FOR A CLASS OF "CONVEX" NONLINEAR INTEGER PROGRAMS

by

R. R. Meyer and M. L. Smith

Computer Sciences Technical Report #274

June 1976

ALGORITHMS FOR A CLASS OF "CONVEX" NONLINEAR INTEGER PROGRAMS*

by

R. R. Meyer and M. L. Smith†

Abstract

Algorithms are given for the efficient solution of the class of nonlinear integer programs with separable convex objectives and totally unimodular constraints. Because of the special structure of this problem class, the integrality constraints on the variables can be easily handled. In fact, the integrality constraints actually make the problem "easier" than its continuous version, for in the case that bounds are available on the problem variables, the first of the proposed algorithms yields the optimal solution by the solution of a single, easily-generated linear program. For the cases in which bounds are not available for the variables or the sum of the variable ranges is very large, other algorithms are discussed that yield the solution after a finite number of linear programs and require less storage than the first algorithm.

---

*Supported in part by the National Science Foundation under Contract No. DCR74-20584 and in part by the United States Army under Contract No. DAAG-29-75-C-0024.

†Mathematics Research Center

1. Introduction

Consider the nonlinear integer program*

$$\min_{x} \sum_{i=1}^{n} f_i(x_i)$$

(1.1) s.t. $Ax = b,\ x \geq 0,\ x$ integer

where $x = (x_1,\ldots,x_n)^T$, each f_i is convex on a closed interval $B_i \subseteq [0,+\infty)$ and A is a totally unimodular $m \times n$ matrix. The intervals B_i, which are assumed to be given, are assumed to cover the feasible set of (1.1) in the sense that $\Omega \equiv \{x|Ax=b, x\geq 0, x \text{ integer}\} \subseteq \{x|x_i \in B_i;\ i=1,\ldots,n\}$. (No differentiability or continuity properties are needed or assumed for the f_i, but convexity of f_i on B_i implies continuity on the interior of B_i.)

Problems of the class (1.1) arise in logistic and personnel assignment applications, and have been the subject of a number of studies [1], [4], [6], [7], [8] that deal with the special case in which $Ax = b$ consists of the single constraint $\sum_{i=1}^{n} x_i = K$ (see also [5] for an algorithm for this special case). Dantzig [3, p. 498] considers the case in which the constraints $Ax = b$ correspond to supply-demand constraints in a bi-partite network.

The problem class (1.1) has the remarkable property

*When modified in the obvious fashion, the algorithms to be described below can be used to solve more general problems in which (1) the constraints $Ax = b$ are replaced by a system of equations and inequalities with a totally unimodular constraint matrix, and (2) integer upper and lower bounds on individual variables are added. Alternatively, such constraints can be converted into the form (1.1) through the addition of integer slack variables, and the resulting coefficient matrix will be totally unimodular.

(shown in [5]) that the integrality constraints actually make the problem easier to solve than its corresponding continuous version. Specifically, it was shown in [5], that if there exist known non-negative integers ℓ_i, u_i such that $B_i = [\ell_i, u_i]$ for $i = 1,\ldots,n$ (i.e., there exist known bounds for the feasible set of (1.1)), then (1.1) may be solved by solving the single linear program constructed by (1) replacing each f_i by an appropriate piecewise linear "approximation" and (2) deleting the integrality constraints. It was also shown in [5] that, if the intervals B_i are infinite (or if the sum of the ranges of the x_i is very large), an algorithm based on the solution of a finite number of similarly constructed linear programs is guaranteed to yield the optimal solution within a finite number of iterations, under the assumption that the given problem of class (1.1) has an optimal solution. In this report we consider in more depth the question of the trade-offs between number of LP's solved, storage required, and number of function evaluations for four specific algorithms of the general types described previously. A numerical example is discussed in Section 4.

2. Basic Algorithms

In order to describe in a compact manner algorithms for the problem (1.1) we will introduce some notation to represent certain piecewise-linear approximations to (1.1). Letting R_i $(i=1,\ldots,n)$ be finite sets of non-negative integers, we define a linear programming approximation to (1.1) as the problem $P(R_1,\ldots,R_n)$ given by:

$$\min_{x,\lambda} \sum_{i=1}^{n} \sum_{j\in R_i} f_i(j)\lambda_{i,j}$$

(2.1) s.t. $Ax = b,\ x \geq 0$

$$\sum_{j\in R_i} j\lambda_{i,j} = x_i,\quad \sum_{j\in R_i} \lambda_{i,j} = 1 \quad (i=1,\ldots,n)$$

$$\lambda_{i,j} \geq 0 \quad \forall\ i,j$$

The problem $P(R_1,\ldots,R_n)$ can be thought of as a separable programming approximation to (1.1) in which the integrality constraints are deleted and the breakpoint-sets are given by the sets R_i $(i=1,\ldots,n)$. The significance of this LP approximation, as shown in [5], is that (1) an extreme point of (2.1) will have x_i integer-valued for $i = 1,\ldots,n$, and (2) if x_i^* $(i=1,\ldots,n)$ is part of an optimal solution (x^*,λ^*) of (2.1) and the breakpoint-sets R_i have the property that (for $i=1,\ldots,n$)

$$[\{x_i^*-1, x_i^*, x_i^*+1\} \cap B_i] \subseteq R_i, \tag{2.2}$$

then an optimal solution to (1.1) is obtained by setting $x_i = x_i^*$ $(i=1,\ldots,n)$. (It should be noted that it is essential to employ a "separable programming approximation" to (1.1), since other piecewise-linear approximations that agree with f_i at the integer points in $B_i (i=1,\ldots,n)$ may not have the required extreme point integrality property - see [5].)

We will now consider three algorithms for (1.1) corresponding to three different procedures for constructing the R_i.

Algorithm 1 (Single LP, maximal R_i)

This algorithm is applicable only if there are known bounds ℓ_i, u_i (which will, without loss of generality, be assumed to be non-negative integers) such that $B_i = [\ell_i,u_i]$, i.e. if x is feasible for (1.1), then $\ell_i \le x_i \le u_i$ for $i = 1,\ldots,n$. It has the advantage that it yields a solution to (1.1) via the solution of a single easily-constructed linear program. Specifically, let each R_i consist of the integers in the interval $[\ell_i,u_i]$, and solve the LP (2.1), then note that the optimality condition (2.2) is satisfied by x^* if (x^*,λ^*) is an optimal extreme point of the LP (if (2.1) is infeasible, then so is (1.1)). ∎

Algorithm 1 provides a very efficient, one-step approach to the solution of (1.1) as long as the problem $P([\ell_1,u_1],\ldots,[\ell_n,u_n])$, which will have $2n + \sum_{i=1}^{n} (u_i-\ell_i)$ variables and $m + 2n$ constraints, does not exceed the capacity of the available linear programming algorithm. In this regard, it should be noted that many commerical LP "packages" have separable programming and/or generalized upper bound capabilities that take advantage of the special structure of (2.1). It is also possible to modify the data handling in the simplex algorithm so that a distinct column is not need for each $\lambda_{i,j}$, since, for a given i, the columns corresponding to the $\lambda_{i,j}$ can differ only in two entries. Furthermore, if the constraints $Ax = b$ can be given a network interpretation, efficient algorithms for network optimization can be applied rather than the ordinary simplex algorithm (see, for example, [2]).

However, if known bounds ℓ_i,u_i are not available or if the attempted implementation of Algorithm 1 would lead to storage problems, then alternative approaches are possible because the optimality conditions (2.2) do not require a "full grid" of points. The next two algorithms take advantage of this fact. To get starting "grids" for the next two algorithms, two cases should be considered: (1) if bounds ℓ_i,u_i are known, then the initial breakpoint-sets $R_i^{(0)}$ needed for the algorithms can be taken as any integer sets containing at least ℓ_i and u_i, and (2) if finite bounds are not available for all variables, let $\bar{x}$ be an extreme point of $\{x|Ax=b,x\ge 0\}$ (such an $\bar{x}$ may be determined by the simplex method, and it will be integer), and set $R_i^{(0)}$ to $\{\ell_i,\bar{x}_i,u_i\}$, where ℓ_i and u_i $(i=1,\ldots,n)$ are estimates (which need not be rigorous) for lower and upper bounds on an optimal solution of (1.1). (If a good guess is available for the optimal solution of (1.1), the corresponding

breakpoints should also be included in the $R_i^{(0)}$.) In both cases, the initial LP considered will be infeasible if and only if (1.1) is infeasible, so we will assume in the algorithms below that feasibility has been established.

Algorithm 2 (Multiple LP's, intermediate R_i's)

At the kth iteration (k=0,1,2,...), we assume that a feasible solution $x^{(k)}$ to (1.1) has been obtained by solving the problem $P(R_1^{(k)},\ldots,R_n^{(k)})$. If the optimality conditions (2.2) are satisfied by $x^* = x^{(k)}$ and $R_i = R_i^{(k)}$, then terminate with $x^{(k)}$ as an optimal solution of (1.1). Otherwise, obtain new breakpoint-sets $R_i^{(k+1)}$ by adding to each $R_i^{(k)}$ the points of $[\{x_i^{(k)}-1, x_i^{(k)}, x_i^{(k)}+1\} \cap B_i]/R_i^{(k)}$, (these are the points "missing" from the optimality conditions) and let the (k+1)st iterate $x^{(k+1)}$ be the value of x in an optimal extreme point of $P(R_1^{(k+1)},\ldots,R_n^{(k+1)})$ ($x^{(k+1)}$ will be integer).

We also assume that the algorithm tests $x^{(k)}$ for optimality in $P(R_1^{(k+1)},\ldots,R_n^{(k+1)})$ and terminates by declaring $x^{(k)}$ optimal for (1.1) if this test is satisfied.

(Since $(x^{(k)},\lambda^{(k)})$ would normally be used as a starting basic feasible solution for $P(R_1^{(k+1)},\ldots,R_n^{(k+1)})$, this is a natural assumption.) ∎

Algorithm 3 (Multiple LP's, minimal R_i's)

Proceed as in Algorithm 2, except that if $x^{(k)}$ and the $R_i^{(k)}$ do not satisfy the optimality conditions of (2.2), then the breakpoint-sets $R_i^{(k+1)}$ for the next iteration are taken to be the sets $\{x_i^{(k)}-1, x_i^{(k)}, x_i^{(k)}+1\} \cap B_i$ $(i=1,\ldots,n)$. ∎

Algorithms 2 and 3, which represent opposite ends of the column-generation spectrum, will converge to an optimal solution of (1.1) in a finite number of iterations, provided that (1.1) actually has an optimal solution. This result follows from a finiteness theorem [5] for integer programs with separable convex objective functions. These algorithms also in general avoid the problem generation and storage problems that may occur in some cases in Algorithm 1. In certain instances, however, difficulties with storage limitations and/or speed of convergence might also arise with Algorithms 2 and 3, and some additional computational refinements are described in the following section.

3. Some Computationally Useful Modifications

Although the algorithms of the previous section are guaranteed to display either one-step or finite convergence under rather weak hypotheses, theoretical finite convergence of course does not necessarily imply that an optimal solution will be obtained within the time or storage available to solve the problem.

If Algorithm 1 can be employed without exceeding the capacity of the available LP or network code, then time and storage will not be problems unless $\sum_{i=1}^{n} (u_i - \ell_i)$ turns out very large. On the other hand, Algorithm 2, starts out with relatively small breakpoint-sets, but there is no control on the size of these sets, and thus no guarantee that problem size limits might not be exceeded.

Although Algorithm 3 employs the minimal breakpoint-sets needed to establish optimality, one would not expect it to be very efficient, since the value of a variable can change by at most one unit at each iteration, and since much of the computed information on values of the f_i may be discarded at each iteration. In addition, slow convergence might also occur in Algorithm 2 in the case that lower and upper bounds are not known and at least one of the estimates ℓ_i, u_i is consistently violated

by the iterates, since Algorithm 2 allows only a single unit change beyond the estimated bounds at each iteration.

Algorithm 4, to be described below (see also Figures 1 and 2) avoids these potential problems, and also makes use of the possibility of computing a lower bound on the optimal solution of (1.1). To set up the linear program for determining a lower bound, let R_i' be the subset of the most recently generated breakpoint-set $R_i^{(k)}$ defined by $R_i' = \{j | j \in R_i^{(k)}, (j+1) \in B_i\}$. (Lower bounds can also be obtained by working with sets smaller than R_i', but smaller sets will yield cruder bounds.)

A lower bound on the optimal value of (1.1) may be obtained by solving the following LP, since the objective function of the LP is no greater than the objective function of (1.1) on the feasible set Ω:

$$
\begin{aligned}
&\min_{x,y,\delta} \ \sum_{i=1}^{n} y_i \\
&\text{s.t.,} \quad Ax = b,\ x \geq 0 \\
&\qquad y_i \geq f_i(j) + \delta_{i,j}(f_i(j+1) - f_i(j)) \qquad (3.1) \\
&\qquad x_i = j + \delta_{i,j} \\
&\qquad (j \in R_i', i = 1, \ldots, n)
\end{aligned}
$$

(Of course, the lower bound generated by solving (3.1) may turn out to be $-\infty$, in which case it would not be useful. In many cases, however, additional information such as non-negativity can be included in the lower-bounding LP in order to prevent unboundedness. For example, if the functions $f_i(x_i)$ are known to be non-negative for non-negative x_i (in many applications the f_i are exponentials or posynomials), then the additional constraint $y_i \geq 0$, when added to the constraints of (3.1) will prevent unboundedness of the objective function.) Since Algorithms 2 and 3 (and Algorithm 4 below) generate feasible solutions to (1.1), it is possible to use a lower bound on the optimal value of (1.1) in a termination criterion if one sets an optimality tolerance on the gap

between the lower bound and the value of the best feasible solution. (In general, the feasible solution with the best objective value will be the last iterate, but an exception to this might occur if the optimal solution $(\underline{x},\underline{y},\underline{\delta})$ to (3.1) had the property that $\underline{x}$ was integer. In general, a solution of (3.1) will not have this integrality property, but if it does, then $\underline{x}$ will be a feasible solution of (1.1), and may have an objective function value better than that of the last iterate, in which case this iterate should be replaced by $\underline{x}$. Note that if $\underline{x}$ is integer and $\sum_{i=1}^{n} f_i(\underline{x})$ coincides with the optimal value of (3.1), then $\underline{x}$ solves (1.1). See Figure 2 for the details of how these possibilities may be taken into account.)

The statement of Algorithm 4 assumes that the following parameters have been supplied: (a) a bound on the total number of $\lambda_{i,j}$ variables that can be handled by the LP to be used, (b) increments α_i, β_i to be used for revising lower and upper bound estimates (these are not used if rigorous lower and upper bounds are known), (c) an optimality tolerance ε and a parameter establishing the frequency of the lower bound computations (this could be based on clock time or number of iterations).

Algorithm 4 (Multiple LP's, bounded storage, optimality tolerance)

Proceed as in Algorithm 3, except that

(a) when the update of the $R_i^{(k)}$ would violate the capacity of the LP, remove from the breakpoint sets $R_i^{(k)}$, prior to the update, all indices other than those corresponding to the current values of the upper and lower bound estimates;

(b) when an iterate violates the current value ℓ_i' of the lower bound estimate, ℓ_i' is updated to $\max\{0, \ell_i' - \alpha_i\}$, and when an iterate violates the current value u_i' of the upper bound estimate, u_i' is updated to $u_i' + \beta_i$;

(c) a lower bound is periodically computed by solving a problem of the form (3.1), and the algorithm is terminated if the objective function value of the best feasible solution obtained thus far lies within ε of the lower bound (if the t^{th} problem of the form (3.1) has an optimal value $\underline{z}^t > -\infty$, the constraint $\sum_{i=1}^{n} y_i \geq \underline{z}^t$ should be added to the (t+1)st problem of the form (3.1) in order to guarantee monotonicity of the lower bounds; note that the dual simplex algorithm may be used with the optimal dual solution from the t^{th} problem being used as the initial solution of the dual of the (t+1)st problem). ∎

For notational convenience in the flowchart for Algorithm 4, $\sum_{i=1}^{n} f_i(x_i)$ is denoted by $f(x)$ and $P(R_1^{(s)},\ldots,R_n^{(s)})$ is denoted by $p^{(s)}$.

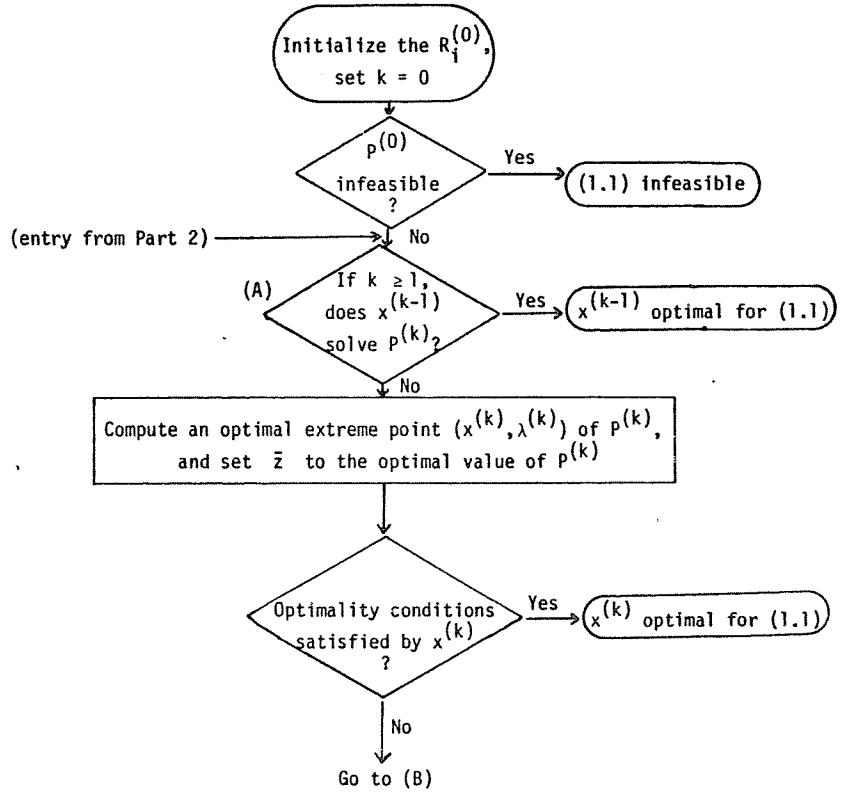


FIGURE 1. Part 1 of the Flowchart for Algorithm 4

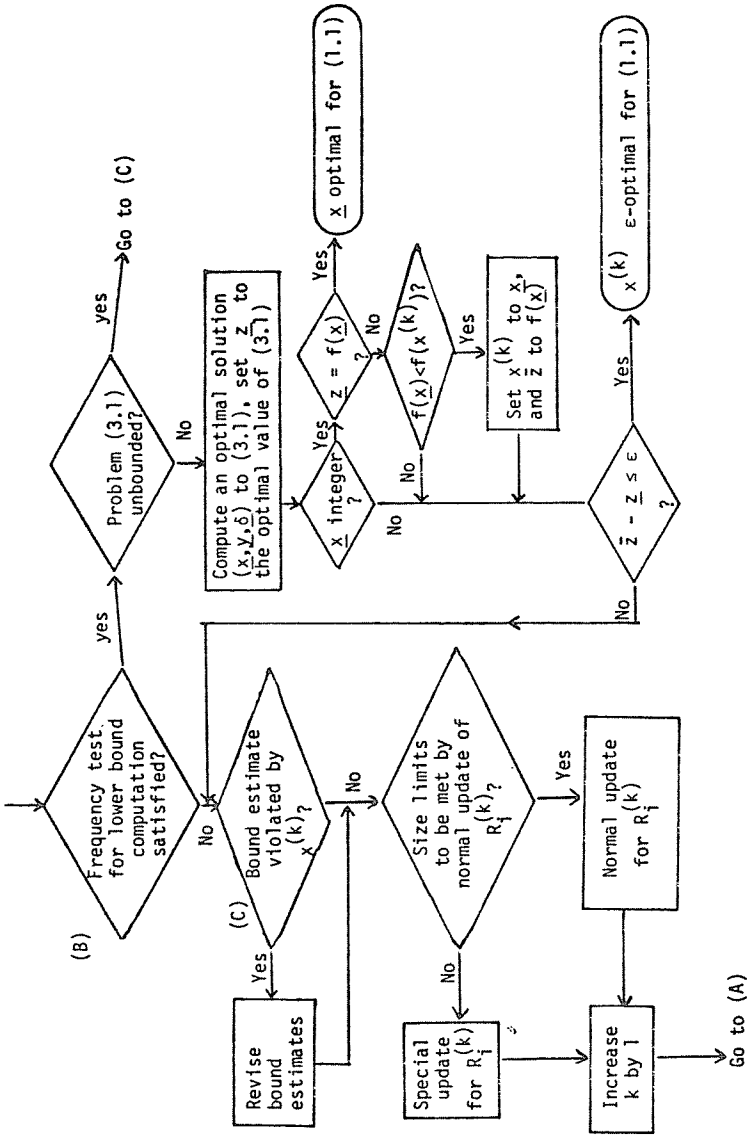


FIGURE 2. Part 2 of the Flowchart for Algorithm 4

4. Numerical Example

In this section we present a numerical example to illustrate the algorithmic ideas introduced in the previous sections.

The problem dealt with has the form

$$\min \sum_{i=1}^{15} w_i (1-q_i)^{x_i}$$

$$\text{(4.1)} \qquad \text{s.t.} \sum_{i=1}^{10} x_i = b_1, \quad \sum_{i=5}^{15} x_i = b_2$$

$$0 \leq x \leq u, \; x \text{ integer}$$

where the data and optimal solution, x^* are given in Table 1. Using the "complete grid" approach of Algorithm 1, the resulting LP has 17 equations and 247 variables, and the use of a small, locally-written LP package yielded the optimal solution (x^* of Table 1) in 148 pivots.

By contrast, a column-generation procedure of the type described in Algorithm 2 started with 45 variables and terminated with an optimal solution (x^* of Table 1) in 15 iterations, the final LP solved having 152 variables.

i	w_i	q_i	u_i	x_i^*
1	9.2	0.31	16	12
2	1.0	0.45	16	4
3	7.6	0.23	19	14
4	0.6	0.09	10	2
5	8.8	0.15	10	10
6	4.2	0.21	11	11
7	3.2	0.15	17	12
8	3.4	0.01	20	0
9	8.8	0.79	16	3
10	6.6	0.41	15	7
11	1.2	0.71	17	3
12	4.6	0.77	12	4
13	0.8	0.79	13	2
14	3.0	0.21	20	13
15	1.2	0.07	20	12
$b_1 = 75$, $b_2 = 67$, optimal value = 7.7586				

Table 1. Data for Numerical Example

References

1. L. B. Boza, "The Interactive Flow Simulator: A System for Studying Personnel Flows," paper presented at the ORSA/TIMS Joint National Meeting, San Juan, Puerto Rico, October 1974.
2. A. Charnes, Fred Glover, David Karney, D. Klingman, Joel Stutz, "Past, Present and Future of Large Scale Transshipment Computer Codes and Applications," Research Report CS 131, Center for Cybernetic Studies, The University of Texas, Austin, October 1974.
3. George B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, 1963.
4. O. Gross, "Class of Discrete Type Minimization Problems," RM-1644, Rand Corp., Santa Monica, 1956.
5. R. R. Meyer, "A Class of Nonlinear Integer Programs Solvable by a Single Linear Program," Computer Sciences Technical Report #267, University of Wisconsin--Madison, February 1976.
6. Thomas L. Saaty, Optimization in Integers and Related Extremal Problems, McGraw-Hill, New York, 1970.
7. R. E. Schwartz and C. L. Dym, "An Integer Maximization Problem," Opns. Res. 19 (1971), 548-550.
8. Iram J. Weinstein and Oliver S. Yu, "Comment on an Integer Maximization Problem," Opns. Res. 21 (1973), 648-650.