Discrete Applied Mathematics 2 (1980) 21–25

A BETTER STEP-OFF ALGORITHM FOR THE KNAPSACK PROBLEM

Harold GREENBERG and Israel FELDMAN

Statistics Department, Tel Aviv University, Ramat-Aviv, Israel

Received 29 March 1978
Revised 10 January 1979 and 5 July 1979

The knapsack problem, maximize $\sum_{i=1}^m c_i x_i$ when $\sum_{i=1}^m a_i x_i \leqslant b$ for integers $x_i \geqslant 0$, can be solved by the classical step-off algorithm. The algorithm develops a series of feasible solutions with ever-increasing objective values. We make a change in the problem so that the step-off algorithm produces a series of solutions of not necessarily increasing objective values. A point is reached when no better solutions can be found and the calculation is stopped.

1. Introduction

The knapsack problem is: find integers $x_i \geqslant 0$, $i = 1, \ldots, m$, that maximize z when

$$\sum_{i=1}^{m} c_i x_i = z,$$

$$\sum_{i=1}^{m} a_i x_i \leqslant b;$$

the c_i are given positive reals and the a_i, b, m are given positive integers. We assume without loss of generality that the greatest common divisor $\gcd(a_1, a_2, \ldots, a_m) = 1$.

The popular step-off algorithms [1, 2] for the knapsack problem succeed in enumerating all values of $\sum_{i=1}^m a_i x_i$ until the value of b is reached. At the same time the algorithms produce the maximum of $\sum_{i=1}^m c_i x_i$. These algorithms are efficient for small b. For large b there exists a periodicity in the computation; the enumeration may be stopped before the value of b is reached. The periodicity is recognized when any further step-off will enumerate values for one x_i variable only. This recognition occurs, however, only after extraneous computation.

In this paper we change the knapsack problem in order to improve the efficiency of step-off methods. Problems can now be solved with a smaller number of step-offs.

2. A step-off algorithm

In this section we consider the knapsack problem with the indices ordered so that index m satisfies $c_m/a_m > c_i/a_i$ for $i \neq m$. We have $x_m \leqslant [b/a_m]$, where $[y]$

denotes the greatest integer less than or equal to y. If b/a_m is an integer, then the maximum z is given by $x_m = b/a_m$, $x_i = 0$ *for* $i \neq m$. If b/a_m is not an integer, we make the change $x_m + k = [b/a_m]$, and obtain the equivalent problem: maximize z where

$$\sum_{i=1}^{m-1} c_i x_i = z - c_m[b/a_m] + c_m k, \tag{1}$$

$$\sum_{i=1}^{m-1} a_i x_i \leq b - a_m[b/a_m] + a_m k, \tag{2}$$

$k = 0, 1, \ldots, [b/a_m]$.

We need to solve only the equivalent problem for each k, $0 \leq k \leq [b/a_m]$, and select the solution that gives the largest z value. In fact, we can step-off using the left-side of (2) until the value b is reached. We can do even better. We rely on the likelihood of x_m having a positive optimal value for the knapsack problem,[1] with optimal k being less than $[b/a_m]$. Hence, the solution will be found before b is reached in the enumeration of (2). We are able to determine if the enumeration may be stopped. Since the objective value is $z^* = c_m[b/a_m]$ for $k = 0$, $x_1 = 0, \ldots, x_{m-1} = 0$, we want a solution where $z > c_m[b/a_m]$. Similarly, given any feasible solution $k^*, x_1^*, \ldots, x_{m-1}^*$ to (2) with resultant objective value z^*, we want a solution with $z > z^*$.

Any z larger than z^* clearly must also satisfy $z \geq z^* + d$, where

$$d = \begin{cases} \gcd(c_1, c_2, \ldots, c_m) & \text{all } c_i \text{ integer,} \\ \varepsilon > 0 & \text{otherwise,} \end{cases} \tag{3}$$

where ε is sufficiently small. We then have the

Theorem. *A solution to the knapsack problem with* $z \geq z^* + d$, *where* d *is defined by* (3) *and* z^* *is the value of* (1) *given by feasible values* $k^*, x_1^*, \ldots, x_{m-1}^*$ *to* (2), *requires that the* k *value of the solution be* $k \leq k_{\max}$, *where*

$$k_{\max} = \left[\frac{c_r\lambda - a_r\delta}{b_r}\right] \tag{4}$$

for

$$\lambda = b - [b/a_m]a_m, \qquad \delta = z^* - c_m[b/a_m] + d$$

and index r *is defined by* $c_r/b_r = \max(c_j/b_j \mid j \in I)$ *for the set* $I = \{i \mid b_i \leq c_m b - a_m(z^* + d),\ 1 \leq i \leq m-1\}$ *for* $b_i = c_m a_i - c_i a_m$. *If* I *is empty, then* $\max z \leq z^*$.

Proof. If feasible k, x_i values to (2) produce $z \geq z^* + d$, then from (1)

$$k \leq \left(\sum_{i=1}^{m-1} c_i x_i + c_m[b/a_m] - z^* - d\right)/c_m \tag{5}$$

[1] E.g., optimal $x_m \geq 1$ for b sufficiently large.

and from (2)

$$k \geq \left(\sum_{i=1}^{m-1} a_i x_i - b + a_m[b/a_m]\right)/a_m \tag{6}$$

Combining (5) and (6), we obtain

$$\sum_{i=1}^{m-1} b_i x_i \leq c_m b - a_m(z^* + d) \tag{7}$$

and the maximum value possible for the right side of (5) subject to (7) is given by $x_r = (c_m b - a_m(z^*+d))/b_r$, $x_i = 0$ for $i \neq r$; (4) and the rest of the theorem follow rapidly.

Corollary. *If, for* $k = 0, 1, \ldots, k_{max}$, $\max z$ *subject to* (1) *and* (2) *occurs for* $z^*, x_1^*, \ldots, x_{m-1}^*, k^*$, *where* k_{max} *is given by* (4), *then* z^* *is maximal.*

Remark. We require $c_m/a_m > c_i/a_i$. Otherwise, $b_i = 0$ and we are unable to find a bound for k smaller than $[b/a_m]$. See Section 3.

Remark. For ease of computation, it appears best to assume the index ordering given by $c_1/a_1 \leq \cdots \leq c_{m-1}/a_{m-1} < c_m/a_m$. We then have $c_1/b_1 \leq \cdots \leq c_{m-1}/b_{m-1}$ and the index r in the theorem is the largest index in set I.

The theorem lets us solve (1) and (2) by enumerating $\sum_{i=1}^{m-1} a_i x_i$ while producing the maximum of $\sum_{i=1}^{m-1} c_i x_i$. We obtain maximum z as a function of k for increasing values of k starting with $k = 0$. We begin with $z^* = c_m[b/a_m]$ and save any larger z value as new z^*, obtaining decreasing values for k_{max} and decreasing bound for the right side of (2). We stop the enumeration long before b is reached whenever optimal k is small. We base our algorithm on the step-off algorithm of Gilmore and Gomory [1].

We define $F(x)$ as

$$F(x) = \max\left(\sum_{i=1}^{m-1} c_i x_i \;\middle|\; \sum_{i=1}^{m-1} a_i x_i \leq x, \quad \text{integer} \quad x_i \geq 0\right).$$

$I(x)$ is the usual index function; the variable with index $I(x)$, increased by one in the step-off to x, produces $F(x)$. Hence, a backtrack procedure, using $I(x)$, will produce the optimal x_i values at the conclusion of the algorithm.

Algorithm. k_{max} and next z are determined by their appropriate subroutines. The indices are ordered so that $c_1/a_1 \leq \cdots \leq c_{m-1}/a_{m-1} < c_m/a_m$. $d = \gcd(c_1, c_2, \ldots, c_m)$ if all c_i are integers; $d = \varepsilon > 0$ otherwise.

Step 1. Initialize $F(x) = 0$ for $0 \leq x \leq b$, $y = 0$, $\lambda = b - [b/a_m]a_m$, $L = \lambda$, $I(0) = 1$, $z = c_m[b/a_m]$ and $k = 0$. Set $b_i = c_m a_i - c_i a_m$ for $i = 1, \ldots, m-1$. Determine k_{max}. Go to step 2a.

Step 2a. Let $j = I(y)$.

Step 2b. If $y + a_j \leq \lambda + a_m k_{\max}$, then let $v = c_j + F(y)$ and go to Step 2c. Otherwise, go to Step 2d.

Step 2c. If $v \geq F(y + a_j)$, then let $F(y + a_j) = v$, $I(y + a_j) = j$ and go to Step 2d. Otherwise, go to Step 2d.

Step 2d. If $j < m - 1$, then let $j = j + 1$ and go to Step 2b. Otherwise, go to Step 3a.

Step 3a. Let $y = y + 1$.

Step 3b. If $F(y) > F(y - 1)$, go to Step 3c. Otherwise, let $F(y) = F(y - 1)$ and $I(y) = m + 1$; go to Step 3d.

Step 3c. If $y = L$, obtain next z. Go to Step 2a. Otherwise, go to Step 2a.

Step 3d. If $y = L$, obtain next z. Go to Step 3a. Otherwise, go to Step 3a.

Routine $k_{\max}$.

Step 1. If $I = \{i \mid b_i \leq c_m b - a_m(z + d),\ 1 \leq i \leq m - 1\}$ is non-empty, determine r, the largest index in I and go to Step 2. Otherwise, stop.

Step 2. Set $\delta = z - c_m[b/a_m] + d$ and $k_{\max} = \min([(c_r\lambda - a_r\delta)/b_r], [b/a_m])$. If $k > k_{\max}$, stop. Otherwise, return.

Routine next z.

Step 1. If $F(y) + c_m[b/a_m] - c_m k \leq z$, go to Step 2. Otherwise, set $z = F(y) + c_m[b/a_m] - c_m k$ and determine $k_{\max}$; go to Step 2.

Step 2. Let $k = k + 1$. If $k > k_{\max}$, stop. Otherwise, set $L = L + a_m$ and return.

That completes the algorithm.

3. The $c_m/a_m = c_i/a_i$ case

Consider the knapsack problem with $c_j/a_j < c_s/a_s = \cdots = c_m/a_m$, $j < s$. We have $a_s x_s + \cdots + a_m x_m \leq p[b/p]$ where $p = \gcd(a_s, \ldots, a_m)$. Making the change

$$(1/p) \sum_{i=s}^{m} a_i x_i + k = [b/p], \tag{8}$$

we obtain the equivalent problem: maximize z where

$$\sum_{i=1}^{s-1} c_i x_i = z - p(c_m/a_m)[b/p] + p(c_m/a_m)k, \tag{9}$$

$$\sum_{i=1}^{s-1} a_i x_i \leq b - p[b/p] + pk, \tag{10}$$

$k = 0, \ldots, [b/p]$. We then follow the same approach as in the theorem and find an upper bound for k. The algorithm, then, is almost the same—stepping-off in (9) and (10). When each k value arises we need to solve (8) before any z value becomes allowable. Again $k_{\max}$ is decreasing and we stop when $k > k_{\max}$. The solution of (8) for nonnegative x_i is one of classical number theory and any good method may be used. We use the method of [3].

References

[1] P.C. Gilmore and R.E. Gomory, The theory and computation of knapsack functions, Operations Res. 14 (1966) 1045.
[2] J.F. Shapiro and H.M. Wagner, A finite renewal algorithm for the knapsack and turnpike models, Operations Res. 15 (1967) 319.
[3] H. Greenberg, An algorithm for a linear diophantine equation and a problem of frobenius, Technical Report, Tel Aviv University.