AN O(n) ALGORITHM FOR THE LINEAR MULTIPLE CHOICE KNAPSACK PROBLEM AND RELATED PROBLEMS

Eitan ZEMEL

J.L. Kellogg Graduate School of Management, Northwestern University, Evanston, IL 60201, U.S.A.

Communicated by J. Nievergelt
Received 1 November 1982
Revised 2 November 1983

We present an O(n) algorithm for the Linear Multiple Choice Knapsack Problem and its d-dimensional generalization which is based on Megiddo's (1982) algorithm for linear programming. We also consider a certain type of convex programming problems which are common in geometric location models. An application of the linear case is an O(n) algorithm for finding a least distance hyperplane in R^d according to the rectilinear norm. The best previously available algorithm for this problem was an O(n log^2 n) algorithm for the two-dimensional case. A simple application of the nonlinear case is an O(n) algorithm for finding the point at which a 'pursuer' minimizes its distance from the furthest amongst n 'targets', when the trajectories involved are straight lines in R^d.

Keywords: Linear programming, multiple choice knapsack, linear algorithms, least distance hyperplane

1. Introduction

Consider the following linear programming problem

\[(P): \max \sum_{j \in N} c_j x_j\]

subject to

\[\sum_{j \in N} a_j x_j = a_0, \quad (1)\]

\[\sum_{j \in I_k} b_j x_j = b_0^k, \quad k = 1, \ldots, r, x_j \geq 0, j \in N, \quad (2)\]

where the sets \(I_k\), \(k = 1, \ldots, r\), are mutually disjoint.

Without loss of generality, we can assume that \(a_0\) and \(b_0^k\), \(k = 1, \ldots, r\), are nonnegative and that \(b_j, j \in I_k, k = 1, \ldots, r\), are not zero. We let \(J_0 = N \setminus \bigcup_{k=1}^r I_k\) be the set of variables which do not appear in any of the multiple choice constraints (2). In order to achieve the equality relation in (1) and (2), slack or surplus variables may have been required. Such a convention is imposed here for notational convenience only. As will become obvious shortly, the algorithm can be easily adapted to handle inequality constraints explicitly.

Problem (P) and various of its special cases are well studied. If the constraints (2) reduce to simple upper bounds on the variables \(x_j, j \in N\), the problem becomes the well-known linear knapsack problem (KP). Linear time algorithms for this problem, using the well-known linear time algorithm for median finding, appeared independently in [1], [2] and [7]. Another important subcase is the Multiple Choice Knapsack Problem (MCKP) obtained when the coefficients \(b_j, j \in I_k, k = 1, \ldots, r\), as well as the right-hand sides, \(b_0^k, k = 1, \ldots, r\), are all equal to one. An algorithm for MCKP is given

---

* Supported in part by NSF Grant No. ECS-8121741 to Northwestern University and by the Israel Institute of Business Research, Tel Aviv University. Part of this work was done while the author was a visiting professor at Tel Aviv University.

** On leave with Chulalongkorn University, Graduate Institute of Business Administration, Training and Continuing Education Center, Bangkok 10500, Thailand.
in [17,16,5,6,3] with the last two reporting running time of $O(n \log n)$. In [18], a unified algorithm is presented for MCKP and KP whose running time is $O(n) + O(|J| \log J_{\text{max}})$, where

$$J = \bigcup_{i=1}^{r} K_{i} \quad \text{and} \quad J_{\text{max}} = \max_{k=1,\ldots,r} |j_k|.$$  

In case of KP, $J_{\text{max}} = 2$, and the algorithm reduces to the $O(n)$ algorithm for KP. We note that, in the worst case, $J$ and $J_{\text{max}}$ are both of order $n$ and thus the complexity of the algorithm is still $O(n \log n)$. It is shown in [18] that under certain probabilistic assumptions on the data, the expected running time of the algorithm is $O(n)$. It is also shown that, if the data of the problem arise from piecewise linearization of convex functions, the problem can be sometimes solved in sublinear time.

We note that the bottleneck phase in the algorithm for MCKP presented in [3,6,18] is the identification of the convex hull of the points $(c_j, b_j)$, $j \in J_k$, $k = 1,\ldots,r$. This is achieved by sorting the variables in each of these subsets according to increasing $b_j$ values. Thus, the complexity term ($J \log J_{\text{max}}$) is a lower bound on the complexity of any algorithm which attempts to solve MCKP via that route. It was left as an open question in [18] whether MCKP can be solved faster without explicitly identifying the convex hulls of each of the sets $(c_j, b_j)$, $j \in J_k$, $k = 1,\ldots,r$. In this article we settle this question in the affirmative and present an $O(n)$ algorithm for (P).

Our result is based on the work of Megiddo [9,10]. In these papers it is shown how a linear program with a fixed number of variables can be solved in linear time (with a constant which depends on the dimension in a very serious way). His algorithm can be used without any modification in the case where $r = 1$, since, in this case, the dual program contains two variables only (and $n$ constraints). However, in general, the dual of (P) may contain up to $O(n)$ variables and hence, the results of [9] and [10] do not apply. Nevertheless, we show that a straightforward adaptation of Megiddo's algorithm solves (P) in $O(n)$ steps.

We note another result of Megiddo which is of some relevance here. The dual of (P) is a linear program with two variables per constraint. Such systems are treated in [8], where an $O(mn^3 \log m)$ algorithm for finding a feasible solution is presented ($m$ is the number of constraints). It is not known whether the optimal solution of such a system can be identified in time polynomial with $n$ and $m$. In any case, the problem treated here is simpler than the general case of two variables per constraint and can be solved much more easily.

2. The dual problem

Consider the linear programming dual of (P):

$$\text{(D)}: \quad \min a_0 \cdot w + \sum_{k=1}^{r} b_k v^k$$

subject to

$$a_j w + b_j v^k \geq c_j, \quad j \in J_k, \quad k = 1,\ldots,r,$$

$$a_j w \geq c_j, \quad j \in J_0.$$

Without loss of generality, we can assume that $a_j \neq 0$, $j \in J_0$. Let

$$J^+_k = \{ j \in J_k \mid b_j > 0 \}, \quad k = 1,\ldots,r,$$

$$J^-_k = \{ j \in J_k \mid b_j < 0 \}, \quad k = 1,\ldots,r,$$

$$J^+_0 = \{ j \in J_0 \mid a_j > 0 \},$$

$$J^-_0 = \{ j \in J_0 \mid a_j < 0 \}.$$

Thus, we can write the dual constraints in the form

$$v^k \geq \frac{c_j}{b_j} - \frac{a_j}{b_j} \cdot w, \quad j \in J^+_k, \quad k = 1,\ldots,r,$$

$$v^k \leq \frac{a_j}{b_j} \cdot w - \frac{c_j}{b_j}, \quad j \in J^-_k, \quad k = 1,\ldots,r,$$

with

$$w \leq \frac{c_j}{a_j}, \quad j \in J^-_0,$$

$$w \geq \frac{c_j}{a_j}, \quad j \in J^+_0.$$

Let

$$w = \max \left\{ \frac{c_j}{w_j} \mid j \in J^+_0 \right\}, \quad \bar{w} = \min \left\{ \frac{c_j}{w_j} \mid j \in J^-_0 \right\}.$$
and let
\[ \alpha_j = \frac{c_j}{b_j}, \quad \beta_j = \frac{a_j}{b_j}, \quad j \in \mathbb{N} \setminus J_0. \]

Our problem (D) reduces to the following optimization problem:
\[ \min a_0 \cdot w + \sum_{k=1}^{r} b_0^k \cdot v^k \]  \hspace{1cm} (6)

subject to
\[ v^k \geq \alpha_j - \beta_j w, \quad j \in J_k^+, \quad k = 1, \ldots, r, \]
\[ v^k \leq \beta_j w - \alpha_j, \quad j \in J_k^-, \quad k = 1, \ldots, r, \]
\[ w \leq w \leq \bar{w}, \]  \hspace{1cm} (7)

where \( w \) may be \( -\infty \) and \( \bar{w} \) may be \( +\infty \) if the sets \( J_k^- \) or \( J_k^+ \) are empty, respectively. We recall that the coefficients of the objective row are non-negative.

Problem (6), (7) has a great deal of similarity to the two-dimensional problem studied by Megiddo, namely,
\[ \min v \]
subject to
\[ v \geq \alpha_j + \beta_j w, \quad j \in J^+, \]
\[ v \leq \alpha_j + \beta_j w, \quad j \in J^-, \]
\[ w \leq w \leq \bar{w}. \]  \hspace{1cm} (8)

The new feature here is the existence of many (up to \( O(n) \)) sets \( J_k, \quad k = 1, \ldots, r \), and a corresponding number of variables \( v^k \). Nevertheless, a straightforward generalization of Megiddo's algorithm solves the problem in \( O(n) \) steps. Below, we give the key elements of the algorithm. We also point out the existence of an extremely simple algorithm which solves this problem in \( O(n \log d_{\text{max}}) \) operations where \( d_{\text{max}} \) is the largest size of a coefficient and the data is expressed as integers. We then point out the generalization of the algorithm to \( d \) dimensions, and present an example. Finally, we present a generalization to the nonlinear case and consider an example for this case too.

3. The algorithm

Consider a given value of \( w \), say \( w^0 \). It is easy to check whether there exists a feasible value \( v^k \) for \( w^0 \). All we have to do is find the maximum value of the numbers \( \alpha_j - w^0 \beta_j, \quad j \in J_k^+, \) say \( \bar{a} \) and verify that this maximum does not exceed \( \bar{a} \), the minimum over \( J_k^- \) of the numbers \( \beta_j w^0 - \alpha_j \). If this condition is met, then the optimal value of \( v_k \) is obviously \( \bar{a} \) (since \( b_0^k \geq 0 \) and we are minimizing).

Graphically, for each \( k = 1, \ldots, r \), consider the two functions
\[ f_k^+(w) = \max_{j \in J_k^+} \alpha_j - w \beta_j, \quad f_k^-(w) = \min_{j \in J_k^-} w \beta_j - \alpha_j. \]

Then \( f_k^+ \) and \( f_k^- \) are piecewise linear convex and concave functions respectively as depicted in Fig. 1.

The feasible set of \( w \), for each individual set \( J_k, \) \([w_k, \bar{w}_k]\), is the set of \( w \) for which \( f_k^+(w) \leq f_k^-(w) \) and, for each \( w \) in this set, \( v_k \) should be set to \( f_k^+(w) \). Thus, our problem reduces to a problem with one variable, \( w \):
\[ \min a_0 w + \sum_{k=1}^{r} b^k f_k^+(w) \]
subject to
\[ f_k^+(w) \leq f_k^-(w), \quad k = 1, \ldots, r, \]
\[ w \leq w \leq \bar{w}. \]

It is clear that, for each \( k \), \( f_k^+ \) and \( f_k^- \) can be found in \( O(|J_k|) \) time, provided \( w \) is specified. As a by-product of this computation, we can also calculate the left and right derivative of \( f_k^+ \) at \( w^0 \), namely, the appropriate slopes of the linear pieces which make up this function at \( w^0 \). (The left or

Fig. 1.
right derivative is set to $+\infty$ if a move in that direction makes $w$ infeasible.) Thus, we can find in $O(n)$ steps the objective function value at $w^0$ together with very important information on the location of the optimal value, $w^*$, namely, we know if $w^*$ is to the left or right of $w^0$. Hence, we can locate $w^*$ within its feasible set by a binary search. If the data $c_j, a_j, b_j, j \in \mathbb{N}$, are expressed as integers, then $w^0$ can be identified in $O(\log d_{\max})$ iterations $[14,15]$, where

$$d_{\max} = \max_{j \in \mathbb{N}} \{ |c_j|, |a_j|, |b_j| \}$$

(see [19] for details). This yields an overall complexity of $O(n \log d_{\max})$. To get an algorithm with $O(n)$ running time, we need to add one additional element to the algorithm. This element is the key element in Megiddo’s linear time algorithms $[9,10]$. Specifically, we now show how to eliminate at each iteration a fixed fraction (in our case $\frac{1}{2}$) of the variables. Once the number of variables in each subset is reduced to a (fixed) pre-terminal level, say 2, we can finish the algorithm by the method of the present author [18]. This yields an overall total effort of $O(n)$.

Assume that at a given iteration, $J^+ = \bigcup_{k=1}^{r} J_k^+$ consists of at least one half of the outstanding variables. Divide the members of $J_k^+$, $k = 1, \ldots, r$, to pairs in an arbitrary fashion. Consider the condition

$$f^+_j = a_j - w \beta_j, \quad j \in J_k^+.$$ 

Clearly, for each pair $i, j$, there exists a cut-off value of $w$, say $w_{ij}$ such that $a_i - w \beta_i \geq a_j - w \beta_j$ to the left of $w_{ij}$ and conversely to the right. Thus, if we know that $w^* \leq w_{ij}$, then variable $i$ can be eliminated. Let $w_m$ be the median point of all the cut-off values $w_{ij}$. Since there are at most $\frac{1}{2}n$ pairs in $J^+$, $w_m$ can be found in $O(n)$ effort. But, since $|J^+| \geq \frac{1}{2}n$, there exists at least $\frac{1}{2}n$ $w_{ij}$ values to the left of $w_m$ and a similar number to its right. We now evaluate $f^+_k$ and $f^-_k$, $k = 1, \ldots, r$, at $w = w_m$. This is done in $O(n)$ effort and allows us to assert whether $w^* > w_m$ or vice versa. In any case, we can eliminate one member from each pair which lies on the appropriate side of $w_m$. Thus, the total number of variables eliminated is at least $\frac{1}{2}n$. A similar reduction of variables can be achieved in the case $J^+ < \frac{1}{2}n$ by forming pairs among the variables in $J_k^-$, $k = 1, \ldots, r$. (Note that $J^+ < \frac{1}{2}n$ implies $J^- \geq \frac{1}{2}n$ since the variables in $J_0$ are no longer needed once the upper and lower bounds $w$ and $\bar{w}$ have been computed.) In fact, we can combine the two cases by forming pairs in $J^-$ and $J^+$ simultaneously and testing at the mean of the combined set. This yields a factor of $\frac{1}{2}$ of the variables eliminated at each iteration. Repeating the process iteratively, we achieve the desired $O(n)$ bound for the solution of (D). Given the optimal value of $w$, the optimal solution for (P) can now be found by solving one problem of type (8) for each set $J_k$. This amounts to additional $O(n)$ effort since $\sum_{k=1}^{r} |J_k| = n$.

A straightforward generalization is the d-dimensional Multiple Choice Linear Programming Problem (MCLPP) obtained when we replace the single constraint (1) by a set of $d$ constraints. In the dual formulation this corresponds to replacing each of the right-hand side expressions $a_j - \beta w$ or $\beta w - a_j$ by a d-dimensional analog $a_j^0 + \sum_{i=1}^{d} \beta_i w_i$. This problem is related to (D) in the same way as the general d-dimensional linear program $[10]$ to the two-dimensional case $[9]$. Thus, we can search for the optimal $w$ value in $\mathbb{R}^d$ using the methods of Megiddo $[10]$. This yields an $O(n)$ algorithm for MCLPP for any fixed dimension $d$. (We recall that $n$ in this case stands for the number of constraints while the number of variables is $r + d$ which can be $O(n)$ if $r$ is large.) One important application of MCLPP which is relevant both to statistics and to location theory is given below.

Let $X = \{x_1, \ldots, x_r\}$ be a set of points in $\mathbb{R}^d$. We wish to find an hyperplane in this space which minimizes the weighted sum of distances from the points of $X$, when distances are measured by the $\ell_1$ (rectilinear) norm. An $O(n \log^2 n)$ algorithm for this problem in $\mathbb{R}^2$ is given by Megiddo and Tamir $[11]$. However, the problem can be solved in $O(n)$ time for every fixed dimension $d$. Indeed let $a_0 + a_1x_1 + \cdots + a_dx_d = 0$ be an equation of an hyperplane $H$ in $\mathbb{R}^d$. The rectilinear distance of $H$ from a point $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ is given by the expression

$$\max |a_1|, \ldots, |a_d|$$
Thus, the problem of finding $H$ reduces to $d$ problems of the form

$$ \min_{\alpha=(\alpha^0,\ldots,\alpha^d)} \sum_{i=1}^{n} w_i |a^0 + a^1 x_i^0 + \cdots + a^d x_i^d|,$$

where, in each case, one of the $a^i$ is set to 1. It is easy to check that by replacing each of the absolute value terms in this expression by the appropriate maximum of two linear terms, we obtain the formulation (say for the case $a_1 = 1$)

$$ \min_{\mu_i} \sum_{i=1}^{n} \mu_i $$

subject to

$$ \mu_i \geq w_i \left( a^0 + x_i^1 + a^2 x_i^2 + \cdots + a^d x_i^d \right), $$

$$ \mu_i \geq -w_i \left( a^0 + x_i^1 + a^2 x_i^2 + \cdots + a^d x_i^d \right). $$

This problem, which has $n + d$ variables ($\mu$, $a$), and $n$ constraints, clearly has the format of the $(d-1)$-dimensional dual of MCLPP and thus can be solved in linear time. The algorithm can be adapted without any modification in its complexity to solve the following generalization. Let $S = \{S_1, \ldots, S_k\}$ be a partition of our $n$ points to $k$ subsets. We wish to find the hyperplane which minimizes the weighted sum of distances to $S_1, \ldots, S_k$, where the weighted distance to a set is defined as the maximum of the weighted distance to its elements. Obviously, this problem falls into the same class as well and can be solved in $O(n)$ time.

4. The nonlinear case

The algorithm of the previous section relies on the convexity, rather than on the linearity, of the functions involved. In fact, it can be generalized to yield an $O(n)$ algorithm for the following nonlinear analog of D:

$$(\text{NLP}) : \min g_0(w) + \sum_{k=1}^{r} g_k(v^k)$$

subject to

$$v_k \geq f_j(w), \quad j \in J_k^+, \quad k = 1, \ldots, r,$$

$$v_k \leq f_j(w), \quad j \in J_k^-, \quad k = 1, \ldots, r,$$

where the functions $f_j(\cdot)$, $j \in J^+$, are convex, $f_j(\cdot)$, $j \in J^-$, are concave, and $g_k(\cdot)$, $k = 0, \ldots, r$, are convex increasing. In order to handle the nonlinearities of the functions within a computational complexity study, we make the following assumptions:

(i) For any given value of its argument, each function $g_k(\cdot)$, $f_j(\cdot)$, can be evaluated together with its left and right derivatives, in constant time.

(ii) For each pair $(i, j)$ of constraints, the equation $f_i(w) = f_j(w)$ has at most $q$ roots, all of which can be found in constant time.

(iii) For each selection of indices $j(k) \in J_k^+$, the (convex) function $g_k(w) + \sum_{i \in J_k^+} g_f_j(x_i(w))$ of the variable $w$ can be optimized in $O(n)$ steps.

Let

$$f_k^+(w) = \min_{j \in J_k^+} f_j(w), \quad f_k^-(w) = \max_{j \in J_k^-} f_j(w);$$

then $f_k^+$ and $f_k^-$ are convex and concave functions respectively. Furthermore, for each value $w$, we can compute the values of these functions in $O(n)$ time together with their left and right derivatives. This enables us to conduct a binary search for the optimal $w$ value, in a similar fashion to the linear case. We now show how a fixed fraction of the functions, namely $\frac{1}{2}$, can be eliminated at each iteration. (In fact, we can eliminate a larger fraction than this, getting arbitrarily close to $\frac{1}{2}$. We omit the details.)

Divide each set $J_k^+$, $k = 1, \ldots, r$, into pairs and similarly divide the sets $J_k^-$, for a total of $m = \frac{1}{2} n$ pairs. For each pair $\alpha = (i, j)$, consider the equation $f_i(w) = f_j(w)$ and let the set of distinct roots of this equation in the relevant interval $[x_{k1}, w_k]$ be $x_{1k} < x_{2k} < \cdots < x_{nk}$ with $p(\alpha) \leq r$. These roots define a partition of $[w_k, \bar{w}_k]$ such that $f_i$ and $f_j$ alternate being above and below each other in adjacent intervals of this partition. Thus, if we know the location of $w^*$ relative to these roots, we can discard one of the variables $i$ or $j$. Let $X = \{x_1, \ldots, x_t\}$ be the set of all these roots over all pairs, with $t < q \cdot m$. Perform a test for $w^*$ at the median element of $X$, then at the median of the remaining points, and so on for a total of $1 + \log q$ tests. This amounts to $O(n \log q)$ effort after which we can establish for at least $\frac{1}{2} m$ pairs $\alpha$ = (i, j) whether or not $f_i(w^*) \geq f_j(w^*)$. This ena-
bles us to eliminate one element from each such pair for a total of \( \frac{1}{4} m = \frac{1}{2} n \) variables. This amounts to \( O(n \log q) \) total effort. For fixed \( q \), it is \( O(n) \) effort.

Functions which satisfy the conditions of the generalized algorithm abound, for instance, in various geometric location problems. An application of these ideas to the one center problem will be discussed in a separate paper [13]. An interesting example is the following. Let \( L_i(t) \) be a set of \( n + 1 \) lines in \( \mathbb{R}^d \). For instance, \( L_i(t) \) can represent the position, at time \( t \), of a particle moving in a straight line through \( \mathbb{R}^d \). Assume that particle 0 is a ‘pursuer’, while particles 1, ..., \( n \) are ‘targets’. At time \( t \), let \( \lambda(t) \) denote the maximum Euclidean distance of the pursuer from the targets, i.e.,

\[
\lambda(t) = \max_{i=1, ..., n} \{d(L_i(t), L_0(t))\}.
\]

We wish to find the point at which \( \lambda(t) \) is minimized. To that end, let

\[
g_i(t) = d(L_i(t), L_0(t))^2.
\]

It is straightforward to verify that \( g_i(t) \) satisfies the conditions of (NLP) with \( q = 2 \). Thus, the problem can be solved in \( O(n) \) time provided quadratic equations can be solved in constant time.

**Note added in proof**

After this work was completed, it was brought to my attention that M.E. Dyer [4] has independently derived a linear time algorithm for MCKP.

**Acknowledgment**

I wish to acknowledge Arie Tamir for several conversations and for bringing Megiddo’s article [12] to my attention. I also wish to thank Nimrod Megiddo for several comments and for pointing out to me another \( O(n) \) algorithm for the Least Distance Line problem mentioned in this paper. Also, a more complicated ‘pursuer-and-target’ problem is mentioned in [11].

**References**

[7] D.B. Johnson and T. Mizoguchi, Selecting the \( k \)th element in \( X + Y \) and \( X_1 + X_2 + \cdots + X_m \), SIAM. J. Comput. 7 (1978) 147–153.
[10] N. Megiddo, Solving linear programming in linear time when the dimension is fixed, Unpublished manuscript, Dept. of Statistics, Tel Aviv University, 1982.