Carnegie-Mellon University
PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION
WILLIAM LARIMER MELLON, FOUNDER

This document has been approved for public release and sale; its distribution is unlimited.
THE LINEAR MULTIPLE CHOICE

KNAPSACK PROBLEM

by

Eitan Zemel*

October 1978
Revised April 1979

The research underlying this reported was supported by Grant MCS76-12026 A02 of the National Science Foundation. Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

Management Science Research Group
Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

*Graduate School of Management, Northwestern University
Abstract

We discuss a fast algorithm for the linear programming relaxation of the Multiple Choice Knapsack Problem. Let $N$ be the total number of variables in this problem and let $J$ and $J_{\text{max}}$ denote the total number of multiple choice variables and the cardinality of the largest multiple choice set, respectively. The running time of the algorithm is then bounded by $O(J \log J_{\text{max}}) + O(N)$. Under certain conditions it is possible to reduce this bound to $O(N)$ steps on the average. Possible further improvements are also discussed.
I. Introduction

Consider the following LP/Multiple-Choice Knapsack Problem (LMCK)

\[
\begin{align*}
(LMCK) \quad Z^* = \text{Minimize} & \quad \sum_{j \in \mathbb{N}} c_j x_j \\
\text{subject to} & \quad \sum_{j \in \mathbb{N}} a_{kj} x_j = a_0 \quad k \in K \\
& \quad \sum_{j \in J_k} x_j = 1, \quad k \in K \\
& \quad x_j \geq 0, \quad j \in \mathbb{N} \\
& \quad x_j \leq 1, \quad j \in I' \subseteq \mathbb{N} \setminus \bigcup_{k \in K} J_k
\end{align*}
\]

where the multiple choice sets, \( J_k, k \in K \), are mutually disjoint. Let \( J = \bigcup_{k \in K} J_k, I = \mathbb{N} \setminus J \). We refer to the variables of \( J \) as multiple choice (or GUB) variables and to those of \( I' \) as simple upper bounded ones.

(LMCK) is a special case of the general (LP) problem with generalized GUB or VUB constraints which has been studied extensively (see, for instance, a recent paper by Schrage [11]). Its main application is as a relaxation for the integer multiple choice knapsack problem [9], [13], which is a useful model for various real life problems. In addition, as pointed out by Witzgal, [14], an efficient algorithm for (LMCK) can be used to accelerate the solution of ordinary LP/GUB problems by the dual simplex algorithm. The reader may note that several generalizations of (LMCK) fall within the scope of the model presented here. For instance, arbitrary positive upper bounds in (4), as well as arbitrary coefficients in the multiple choice constraints (2), can be handled by normalization. \(^1\)

\(^1\) Negative coefficients in (2) can be handled by complementing the variable in question relative to an artificially set large upper bound. The optimal solution of (LMCK) must be checked in such cases for non-boundedness of the original problem.
Thus the two constraint linear programming problem can be viewed as a special case of (LMCK).

(LMCK) is equivalent, but not identical, to the problem treated recently by Glover and Klingman [6], which in turn is a slight generalization of the problems studied by Sinha and Zoltner [13], and Witzgal [14]. The difference between the model presented here and the one of [6] is in the introduction of the individual upper bounds (4). [13] and [14] do not allow for any variables which are outside of the multiple choice constraints. (Using the notation introduced earlier, the model presented in [6] corresponds to the case \( I' = \emptyset \) while those of [13] and [14] to \( I = \emptyset \).)2 Upper bounded variables can be accommodated by the algorithms of [6], [13] and [14] by treating each such variable, together with its slack, as a multiple choice constraint. However, this convention tends to increase both \( N \) and \( K \) and with them the computational effort. In contrast, the algorithm proposed here works in the opposite direction, i.e. it converts multiple choice variables into simple upper bounded ones. Thus, it takes full advantage of the existence of variables in \( I \).

The algorithms of [6], [13], and [14], as well as the one presented here, are basically two-phase procedures. The role of Phase I is to identify the lower convex boundary of each of the multiple-choice sets (see Propositions 1 and 2 below). It is well known that this task can be accomplished in \( O(J \log J_{\text{max}}) \) steps, where \( J_{\text{max}} \) denotes the cardinality of the largest multiple choice set.3,4

---

2The distinction between \( I \) and \( I' \) is of little consequence from a computational point of view, since, as pointed out by Glover and Klingman [6], all but at most 2 variables of \( I \setminus I' \) can be trivially eliminated at the outset.

3Throughout this paper we let the same symbol stand for a set and for its cardinality.

4Throughout this paper we take \( \log x \) to mean \( \max\{\log x, 1\} \)
Phase II of the above mentioned algorithms is an iterative improvement procedure. As mentioned earlier, the algorithms of [6], [13], and [14] treat the variables of I as additional multiple choice constraints. Let \( K' = K + I \) be their total number under this convention. The complexity of Witzgal's Phase II is then \( O(K'(N-K)) \) while Glover and Klingman's is \( O(N \log K') \). Sinha and Zoltners do not give complexity estimates but their procedure is of a similar type. Its precise performance may equal one or the other of these bounds depending on some unspecified details of implementation.

A particularly interesting special case of (LMCK) is the knapsack problem (LKP), which corresponds to the case \( J = \emptyset \). An \( O(N) \) algorithm for this problem is given in [1],[2],[8]. Although any of the algorithms of [6], [13], and [14] can be specialized to solve (LKP), the resulting procedure is not an \( O(N) \) algorithm for (LKP). This discrepancy brings to mind the question, posed by Glover and Klingman, [6], as to the possibility of designing an algorithm for (LMCK) which retains (or even exceeds) the efficiency achieved by the algorithms [6], [13] or [14], while specializing to an \( O(N) \) procedure when the instance of (LMCK) corresponds to (LKP). In this paper we settle this question in the affirmative. More specifically, our algorithms use the calculations of Phase I in order to convert (LMCK) into (LKP). This conversion is done without increasing any of the problem parameters (such as the number of variables or the size of the coefficients) and takes a negligible computational effort. Since (LKP) is known to be of complexity \( O(N) \), the overall complexity of our algorithm is
(5) \[ P = O(J \log J_{\text{max}}) + O(N) \]

Expressed in terms of \( N \) alone, it is quite clear that there exist positive constants, \( c_1, c_2 \), such that

\[ c_1 N \leq P \leq c_2 N \log N \]

For instance, \( P \approx O(N) \) if there exist \( c_3 \) such that \( J_{\text{max}} \leq c_3 \), or if for every positive constant, \( c_4 \), there exists \( N^* \) such that \( c_4 \cdot N > J \) for \( N \geq N^* \).

We present the transformation of (LMCK) into a knapsack problem in section II. In section III we discuss two possibilities for improvements on the bound (5). The first one is the "Divide and Conquer" algorithm of Bentley and Shamos, [3], which can be used to speed up the computation of phase I. Under certain conditions this approach yields an algorithm for (LMCK) whose expected running time is \( O(N) \). The second improvement, directed at phase II, is based on a generalization of ideas due to Jefferson, Shamos and Tarjan, [12], Johnson and Mizoguchi, [8] and Galil and Megiddo, [5]. It is particularly relevant for cases in which Phase I can be avoided entirely. This may arise, for instance, if each of the sets \( J_k \), \( k \in K \), arises from piecewise linearization of a certain (one dimensional) convex function. In such cases, it is sometimes possible to solve (LMCK) in sublinear (worst case) effort.

II. The Transformation

Several properties of an optimal solution for (LMCK) are discussed in [5], [13] and [14]. Propositions 1 - 3 below are straightforward generalizations of these properties.
Proposition 1 [6], [13], [14] Let \( i, j \in J_k \) for some \( k \in K \) with

\[
\begin{align*}
 a_i &= a_j \\
 c_i &\geq c_j
\end{align*}
\]

then there exists an optimal solution to (LMCK) with \( x_i = 0 \).

Proposition 2 [6], [13], [14] Let \( i, j, k \in J_k \) for some \( k \in K \) with

\[
\begin{align*}
 a_i < a_j < a_k
\end{align*}
\]

and

\[
\begin{align*}
 \frac{c_i - c_j}{a_i - a_j} &\geq \frac{c_k - c_j}{a_k - a_j}
\end{align*}
\]

then there exists an optimal solution for (LMCK) with \( x_j = 0 \).

It will be convenient to think on a given multiple choice set, as a set of points, \( \{(a_j, c_j) \mid j \in J_k \} \) in a two dimensional space. Using Proposition 2 we can eliminate from such a set all but those variables which define its lower convex boundary. Let \( J'_k \subseteq J_k \) be the set of remaining variables and let \( J' = \bigcup_{k \in K} J'_k \), \( N' = I \cup J' \). Proposition 1 ensures that \( a_i \neq a_j \) for \( i, j \in J'_k \), \( i \neq j \).

There are several available algorithms which can be used to identify the sets \( J'_k \), \( k \in K \) (e.g., [7], [10]). The approach taken in [6], [13] and [14] is based on first sorting the variables of each set \( J_k \) according to increasing \( a_j \) values. The sorted sets are then scanned and variables which violate Propositions 1 or 2 are purged. The computational complexity of this procedure, as is the case for most other techniques which identify the convex hulls of a set of points in a plane, is determined by the sorting phase. Since sorting a set of size \( J_k \) requires \( O(J_k \log J_k) \) operations, the overall complexity of phase I is given by

\[
O\sum_{k=1}^{k} J_k \log J_k \leq O(J \log J_{\text{max}}).
\]
Denote by \((\text{LMCK})'\) the (equivalent) problem obtained from \((\text{LMCK})\) by replacing the sets \(J_k\) with \(J'_k\), \(k \in K\) and \(N\) with \(N'\). We assume that the multiple choice sets are indexed such that
\[
J'_k = \{j_k, j_k+1, \ldots, j_k + J'_k - 1 = i_k\}
\]
and
\[
a_{j_k} < a_{j_k+1} < \ldots < a_{i_k}
\]
W.L.O.G. we can assume \(J'_k \geq 2\), \(k \in K\). By convexity,
\[
\frac{(c_{j+1} - c_j)}{(a_{j+1} - a_j)} < \frac{(c_{j+2} - c_{j+1})}{(a_{j+2} - a_{j+1})} \quad \text{for} \quad j = j_k, \ldots, i_k, \quad k \in K
\]

**Proposition 3** [6], [13], [14]

Any basic optimal solution for \((\text{LMCK})'\), \(\overline{x}\), satisfies

1. \(\overline{x}\) has at most two fractional variables.
2. if \(\overline{x}\) has two fractional variables they must be adjacent variables within one of the multiple choice constraints.
3. if \(\overline{x}\) has a unique fractional variable it must be a non-multiple choice variable.

We now define a knapsack problem, \((\text{LKP})\), which is equivalent to \((\text{LMCK})'\).

For each multiple choice set, \(J'_k\), let
\[
J''_k = J'_k \setminus \{j_k\} \quad \text{and let} \quad J'' = \bigcup_{k \in K} J''_k, \quad N'' = I \cup J''.
\]
Define
\[
d_j = \begin{cases} 
  c_j - c_{j-1} & \text{if } j \in J'' \\
  c_j & \text{if } j \in I
\end{cases}
\]
\[
e_j = \begin{cases} 
  a_j - a_{j-1} & \text{if } j \in J'' \\
  a_j & \text{if } j \in I
\end{cases}
\]
and consider the following knapsack problem:

\[
(\text{LKP}) \quad W^* = \minimize \sum_{j \in J''} d_j y_j + \sum_{k \in K} c_j
\]
\[
\text{subject to} \quad \sum_{j \in N''} e_j y_j + \sum_{k \in K} a_j = a_0
\]
In words, we have eliminated the first variable, $x_j$, from each of the multiple choice constraints by setting this variable to 1. The objective and constraint rows have been re-adjusted to record this elimination. The remaining multiple choice variables are then replaced by "difference variables", $y_j = x_j - x_{j-1}$, whose role is to enable one to shift from $x_j$ to other variables of $J_k$ as possible representatives of this set. It is quite apparent that the variables $y_j$ will function properly only under certain conditions. Indeed, there is no obvious correspondence between the feasible solution set of (LKP) and that of (LMCK'). However,

**Theorem 1**

(LKP) is equivalent to (LMCK)' in the following sense:

(i) $w^* = Z^*$

(ii) Let $\bar{y}$ be a basic optimal solution for (LKP) and let $f$ denote the index of its basic variable, $0 \leq y_f < 1$. Then a basic optimal solution for (LMCK)', $\bar{x}$, can be defined as follows:

$$
\begin{align*}
\bar{x}_j &= \bar{y}_j & j \in I \\
\end{align*}
$$

$$
\begin{align*}
\text{(14)} & \quad \bar{x}_j = y_j & j \in I \\
\text{(15)} & \quad \text{let } J_k' \text{ be such that } f \notin J_k'' . \\
\text{Define} & \quad h_k = \begin{cases} 
J_k & \text{if } \bar{y}_j = 0 \forall j \in J_k'' \\
\max\{j | j \in J_k'', \bar{y}_j = 1\} & \text{otherwise}
\end{cases} \\
\text{then} & \quad \bar{x}_j = \begin{cases} 
1 & j \in J_k', j = h_k \\
0 & j \in J_k', j \neq h_k
\end{cases}
\end{align*}
$$
Let \( J'_r \) be the unique set such that \( f \in J'_r \) (if indeed such a set exists). Set

\[
\begin{cases}
  -y_f & j = f \\
  1 - y_f & j = j-1 \\
  0 & j \in J'_r, j \neq f, f-1
\end{cases}
\]

Proof. Call a basic feasible solution to (LMCK)' \( \bar{x} \), acceptable if it satisfied (6), (7) and (8). Call a basic feasible solution to (LKP), \( \bar{y} \), acceptable if it satisfies the following condition:

\[
\bar{y}_j > 0, j \in J'_k \implies \bar{y}_i = 1 \quad \forall i \in J'_k, i < j
\]

The theorem follows from the following three facts:

(i) The transformation defined by (14), (15) and (16) is a one to one mapping from acceptable solutions of (LMCK)' onto those of (LKP).

(ii) For any acceptable solution for (LMCK)', the transformation of (i) preserves the objective function value.

(iii) The optimal basic solutions to both (LKP) and (LMCK)' are acceptable for the corresponding problems. Q.E.D.

III. Improvements

Let us reconsider the bound (5). As we have already noted, the overall complexity of the algorithm presented in the previous section may get as high as \( O(N \log N) \). The bulk of this effort is spent on the execution of phase I, i.e., on the identification of the sets \( J'_k \), \( k \in K \). It is well known (e.g., [3]) that a lower bound on the effort involved with this task is given by
Thus, any algorithm which is based on phase I (or its equivalent) will require at least \( O(N \log N) \) steps under adverse conditions. The question of whether \((LMCK)\) can be solved without explicitly identifying the sets \( J'_k, k \in K \), is open.

The foregoing discussion relates to worst case analysis only. Under certain conditions, one may do better on the average. For instance, Bentley and Shamos, \([3]\), have developed a "Divide and Conquer" algorithm which is particularly efficient if \( J'_k \ll J_k, k \in K \). More precisely, let \( L_k \subseteq J_k \) be a random subset of \( J_k \) and let \( L'_k \) stand in the same relation to \( L_k \) as \( J'_k \) does to \( J_k \). Since \( L_k \) is a random set so is \( L'_k \). Denote by \( E(L'_k) \) the expected size of this set. If there exists a constant \( p < 1 \) such that

\[
E(L'_k) < L_k^p, \quad k \in K
\]

Then Bentley and Shamos' algorithm finds \( J'_k \) in expected time which is bounded by \( O(J_k) \). This implies, of course, that phase I, and hence the algorithm as a whole, can be solved in (expected) \( O(N) \) steps. (It is noted in \([3]\), that a "Divide and Conquer" approach can yield an algorithm for the two variable linear programming problem whose expected and worst case running time are bounded by \( O(N) \) and \( O(N \log N) \) respectively. As noted earlier, the dual of this problem, i.e., the two constraint linear programming problem is a special case of \((LMCK)\).)

Condition (18) is known to hold under quite a variety of situations. For instance, it suffices to assume that the data \( \{(c_j, a_j)\}, j \in J_k \) are independent and identically distributed and that for any given variable the cost and weight coefficients are independent of each other. For details and references see \([3]\).

A final remark concerns the case where each of the sets \( J_k, k \in K \),
arises from a process of piecewise linearization of a certain (one dimensional) function. Such a process often yields the coefficients \(a_j\), \(j \in J_k\), already sorted in a natural way. This reduces the computational complexity of phase I to \(O(J)\), and of (LMCK) to \(O(N)\).

If the nonlinear functions referred to in the previous paragraph are known to be convex, one can sometimes do even better. We note that in such cases \(J_k' = J_k, k \in K\), and one can start the computation directly at phase II (as in [4], p.p. 484-486). The knapsack problem that results in such a case has a special structure that can be exploited by generalizing some ideas brought forward by Jefferson, Shamos and Tarjan, [12], Johnson and Mizoguchi, [8] and Galil and Megiddo [5]. The resulting algorithm is of complexity

\[O(K \log^2 (J/K)) + O(1 \log (N/I))\]

which may be less than linear in \(N\). This would mean that such problems can be solved in less time than is needed to read in the totality of the data \(\{(c_i, a_i)\}, i \in N\). Such sublinear behavior is possible, for instance, in an on-line environment where the convex functions which are subject to the process of piecewise linearization are only evaluated when needed.

The algorithm of [1], [2], [8], for (LKP) is based on an iterative step in which the median element of the set \(\{d_i/e_i\}, i \in N\), is used to reduce the size of \(N\) by a factor of at least \(1/2\). The lion's share of the computational effort is spent on the operation of identifying the median. The procedure below uses a certain approximation for the median which is cheaper to calculate but which still guarantees that a significant portion of \(N\) (at least \(1/4\)) will be disposed of at each iteration. For a statement of the algorithm it is convenient to rename the set \(I\) as \(J'_0\). At each iteration, for \(k = 0, \ldots, K\) let
$J_k^+ = \{j \in J_k^* | y_j \text{ was already set to } 1\}$

$J_k^- = \{j \in J_k^* | y_j \text{ was already set to } 0\}$

$J_k^* = \{j \in J_k^* | y_j \text{ is as yet unassigned}\}$

and for $\# = +, -, \text{ or } *$ let

$N^\# = \bigcup_{k=0}^{K} J_k^\#$

For a given scalar $\lambda$ consider the following partition of $J_k^*$, $k = 0, \ldots, K$.

$J_k^1(\lambda) = \{j \in J_k^* | d_j/e_j > \lambda\}$

$J_k^2(\lambda) = \{j \in J_k^* | d_j/e_j = \lambda\}$

$J_k^3(\lambda) = \{j \in J_k^* | d_j/e_j < \lambda\}$

For $i = 1, 2, 3$ let

$N^i(\lambda) = \bigcup_{k=0}^{K} J_k^i(\lambda)$

and

$S^i(\lambda) = \sum_{j \in N^i(\lambda)} e_j$

Algorithm KNAPSACK

0. Set $J_k^+ = J_k^- = \emptyset$, $J_k^* = J_k^*$, $k = 0, \ldots, K$.

1. Choose $\lambda$ as follows:

   (a) Let $r_k$ be the median index of each set $R_k = \{d_j/e_j\}$, $j \in J_k^*$, $k = 0, \ldots, K$.

   (b) Let $r$ be the weighted median index of the set

   $R = \{d_k/e_k\} \quad k = 0, \ldots, K$, where the weight associated

   with the $k$th element is the cardinality of the set $J_k^*$.

   (c) Let $\lambda = d_r/e_r$. 
2. Calculate $S^1(\lambda)$ and $S^2(\lambda)$.

(a) If $S^1(\lambda) < a'_0 \leq S^1(\lambda) + S^2(\lambda)$ stop, $\lambda$ is optimal. An optimal solution, $\bar{y}$, can be found by setting $\bar{y}_j = 1$, $j \in N^+ \cup N^1(\lambda)$, $\bar{y}_j = 0$, $j \in N^- \cup N^3(\lambda)$, and then "filling the knapsack" with variables $y_j$, $j \in N^2(\lambda)$ (any, possibly including one at a fractional value).

(b) If $S^1(\lambda) \geq a'_0$ set $J^* = J^- \cup J^2_k \cup J^3$, $J^*_k = J^1_k$, $k = 0, \ldots, K$.

(c) If $S^1(\lambda) + S^2(\lambda) < a'_0$ set $J^* = J^+ \cup J^2_k \cup J^3$, $J^*_k = J^1_k$, $k = 0, \ldots, K$.

3. If $N^* > I + K$ Go to 1.

Otherwise

4. Solve the remaining knapsack problem using the linear time algorithm of [1], [2], [8].

To assess the computational complexity of algorithm KNAPSACK we note that each iteration reduces the size of $N^*$ by a factor of at least $1/4$. The number of iterations through steps 1, 2 and 3 is then bounded by $O(\log(N/(K+I)))$.

The effort involved in each iteration is as follows:

(i) For each of the sets $J^*_k$, $k = 1, \ldots, K$, one can find $r_k$, calculate the contribution of $J^*_k$ to $S^1(\lambda)$, $S^2(\lambda)$, and form the sets $J^i_k$, $i = 1, 2, 3$, in an effort bounded by $O(\log J^*_k) \leq O(\log J^i_k)$.

(ii) The same tasks can be accomplished for $J_0$ in $O(J_0^*) \leq O(I)$ steps.

(iii) The weighted median $r$ can be identified in $O(K)$ steps.

In addition one may spend $O(k + I)$ operations going through step 4.

Thus, the overall complexity of algorithm KNAPSACK is bounded by

$O[\log(N/(K+I))(K \log(J/K) + I)] \leq O(K \log^2(J/K)) + O(I \log(N/I))$. 
References


We discuss a fast algorithm for the linear programming relaxation of the Multiple Choice Knapsack Problem. Let $N$ be the total number of variables in this problem and let $J$ and $J_{\text{max}}$ denote the total number of multiple choice variables and the cardinality of the largest multiple choice set, respectively. The running time of the algorithm is then bounded by

$$O(J \log J_{\text{max}}) + O(N).$$

Under certain conditions it is possible to reduce this bound to $O(N)$ steps on the average. Possible further improvements are also discussed.