SHORTEST PATH ALGORITHMS FOR KNAPSACK TYPE PROBLEMS

A.M. FRIEZE
University of London, London, England

Received 13 February 1975
Revised manuscript received 16 January 1976

The group knapsack and knapsack problems are generalised to shortest path problems in a class of graphs called knapsack graphs. An efficient algorithm is described for finding shortest paths provided that arc lengths are non-negative. A more efficient algorithm is described for the acyclic case which includes the knapsack problem. In this latter case the algorithm reduces to a known algorithm.

1. Introduction

The name group knapsack problem has been given to

\[
\text{minimise } \sum_{j=1}^{n} c_{j}x_{j}, \quad (1.1)
\]

subject to \( \sum_{j=1}^{n} x_{j}g_{j} = g_{0}, \quad (1.2) \)

\[x_{1}, \ldots, x_{n} \text{ non-negative integers.}\]

The elements \( g_{0}, \ldots, g_{n} \) are a subset of the elements of a finite additive abelian group \( H \) and \( c_{1}, \ldots, c_{n} \) are non-negative reals.

This problem was first considered by Gomory [4] and arises in a pure-integer programming problem when the non-negativity constraints are relaxed on an optimal set of basic variables for the associated LP problem.

Algorithms for solving this problem have been described by Gomory [5], Shapiro [8, 9], Hu [6] and others.

It can be formulated as a shortest path problem in the following way:

Let \( G_{1} \) be the graph with nodes \( H \) and arcs of the form \((h, h + g_{j})\) \( h \) an arbitrary element of \( H \) and \( j = 1, \ldots, n \). The length of such an arc is \( c_{j} \). Let \( P \) be a path from 0 to \( g_{0} \) in \( G_{1} \) then if \( x_{j} \) is the number of arcs of the form \((h, h + g_{j})\) in \( P \) then \((x_{1}, \ldots, x_{n})\) is a solution to (1.2) and the length of \( P \) is (1.1). Conversely if \((x_{1}, \ldots, x_{n})\) satisfies (1.2), then one may construct a set of paths from 0 to \( g_{0} \) of the same length. Thus the problem becomes that of finding a shortest path from 0 to \( g_{0} \). In this paper we give a new algorithm for solving this problem.

The name knapsack problem applies to

\[
\text{maximise } \sum_{j=0}^{n} c_{j}x_{j}, \quad (1.3)
\]
subject to $\sum_{j=0}^{n} w_j x_j = W$, \hspace{1cm} \text{(1.4)} \\
where $c_0 = 0$, $c_1, \ldots, c_n$ are positive reals, $w_0 = 1$ and $w_1, \ldots, w_n$, $W$ are positive integers.

One can formulate a knapsack problem as a longest path problem defining the graph $G_2$ with nodes $0, 1, \ldots, W$ and arcs of the form $(w, w + w_j)$ of length $c_j$. The knapsack problem is then equivalent to that of finding a longest path from $0$ to $W$.

Gilmore and Gomory [2] describe an algorithm for solving trim loss problems which solve a sequence of knapsack problems. A more efficient algorithm for solving the knapsack sub-problems is given in Gilmore and Gomory [3].

2. An algorithm

The graphs $G_1$ and $G_2$ of the previous section are examples of a class of graphs which for the purposes of this paper we call knapsack graphs.

Definition. A graph $G$ with nodes $N$ and arcs $A$ is a knapsack graph if:

(2.1) The arcs $A$ can be partitioned into $n$ disjoint sets $A_1, \ldots, A_n$;
(2.2) the length of each arc belonging to $A_i$ is $l_i$;
(2.3) let $P = (i_0, i_1, \ldots, i_p)$ be a path between an arbitrary pair of nodes $i_0, i_p$. Suppose that $(i_{t-1}, i_t) \in A_{m_t}$ for $t = 1, \ldots, p$. Then for any re-ordering $n_1, \ldots, n_p$ of the indices $m_1, \ldots, m_p$ there exists a path $Q = (j_0, j_1, \ldots, j_p)$ where $j_0 = i_0, j_p = i_p$ and $(j_{t-1}, j_t) \in A_{n_t}$ for $t = 1, \ldots, p$.

For shortest path problems with non-negative arc lengths an efficient algorithm is that described by Dijkstra [1]. We describe a modification of this algorithm applicable to a group knapsack problem which takes advantage of property (2.3) of knapsack graphs. The algorithm finds a shortest path from an origin node $s$ to all other nodes.

Algorithm 1

The algorithm uses a set of labels $(d_j, p_j)$ for each node $j$ such that when a label is made 'permanent' by the algorithm $d_j$ is the length of a shortest path $TP_j$ from $s$ to $j$ and $p_j$ is the predecessor of $j$ on $TP_j$. Define $a_j$ by arc $(p_j, j) \in A_{a_j}$ and note that for a group knapsack problem one can dispose with $p_j$ and use labels $(d_j, a_j)$. Finally if a label is not currently permanent it is referred to as temporary.

Step 0. Put $(d_s, p_s) = (0, s)$, $a_s = n$ and $(d_j, p_j) = (\infty, s)$ for $j \neq s$.

Step 1. If all labels are now permanent terminate, otherwise let $d_k = \min(d_j \mid j$ has a temporary label) make the label $(d_k, a_k)$ permanent.

Step 2. For $r \leq a_j$ and $(k, j) \in A_r$, calculate $d_k + l_r$ and if $d_k + l_r < d_j$ replace the label of $j$ by $(d_k + l_r, k)$. Go to step 1.

The improvement of the above algorithm over the more general Dijkstra