1970

A Branch Search Algorithm for the Knapsack Problem

Greenberg, Harold; Hegerich, Robert L.

INFORMS

Harold Greenberg, Robert L. Hegerich, (1970) A Branch Search Algorithm for the Knapsack Problem. Management Science 16(5):327-332.
http://hdl.handle.net/10945/70157

MANAGEMENT SCIENCE
Vol. 16, No. 5, January, 1970
Printed in U.S.A.

A BRANCH SEARCH ALGORITHM FOR THE KNAPSACK PROBLEM*

HAROLD GREENBERG AND ROBERT L. HEGERICH

Naval Postgraduate School

This paper presents an algorithm for the solution of the knapsack problem. The method involves searching the nodes of a tree along a single branch at a time. The algorithm eliminates the computational drawbacks inherent in the usual branch and bound schemes.

I. Introduction

This paper presents a new method for the solution to the knapsack problem:

$$\text{(1)} \qquad \text{maximize } \sum_{i=1}^{N} v_i x_i$$

$$\text{subject to} \qquad \sum_{i=1}^{N} w_i x_i \leq W,\ x_i = 0, 1 (i = 1, \cdots, N).$$

We can assume without any loss in generality that the constants v_i and w_i are positive integers. Fractions can be handled by multiplying through by a proper factor. Nonpositive constants are handled as in Glover [2]. We also assume that the indices have been arranged so that $v_1/w_1 \geq v_2/w_2 \geq \cdots \geq v_N/w_N$.

For physical interpretations to the knapsack problem and further references, see Kolesar [3] who presents a branch and bound algorithm for the solution to (1).

We develop first a branch and bound algorithm that is computationally more efficient than the one in [3]. However, branch and bound methods usually have very large computer memory and time requirements. We then present a branch search procedure that eliminates these serious computational drawbacks. Computational results are presented for our two algorithms and for Kolesar's method.

II. A Branch and Bound Algorithm

We are able to solve (1) by enumerating solutions in a tree search procedure. As shown by Dantzig [2], the optimal fractional solution to

$$\text{(2)} \qquad \text{maximize } \sum_{i=1}^{N} v_i x_i$$

$$\text{subject to} \qquad \sum_{i=1}^{N} w_i x_i \leq W$$

$$0 \leq x_i \leq 1 \ (i = 1, \cdots, N)$$

$$\text{is given by} \qquad x_i = 1 \text{ if } i < r$$

$$x_i = 0 \text{ if } i > r$$

$$x_r = (W - \sum_{i<r} w_i)/w_r$$

where r is the least integer $(0 \leq r \leq N)$ for which $\sum_{i \leq r} w_i \geq W$. If no r exists we have all $x_i = 1$. If $x_r = 0$, we have the optimal solution to (1).

If x_r is fractional the value of the objective function is $z(1) = \sum_{i<r} v_i + v_r x_r$. We consider $z(n)$, the value of the objective function at node n of the tree, as the solution

* Received March 1968; revised January 1969.

Published in Management Science on January 01, 1970 as DOI: 10.1287/mnsc.16.5.327.
This article has not been copyedited or formatted. The final version may differ from this version.

to (2) with assigned variables added as constraints. A branch and bound scheme is essentially as follows:

1. Label node 1 with $z(1)$. Go to 2.

2(a). Find the terminal node of the tree with the largest value of $z(n)$. This is the node at which the next branching will take place. Any node ($\neq 1$) contains the effect of assigning values to variables and solving (2) with the assigned values of the variables added as constraints.

(b). If the solution at node n has all integer variables, we have achieved an optimal solution to (1). Stop. If not, go to 3.

3(a). Set $n = n + 1$ and some unassigned variable, say x_t, equal to zero. Solve (2) with all assigned variables added as constraints. Label node n with the value $z(n)$. Go to 3(b).

(b). Set $n = n + 1$ and $x_t = 1$. Solve (2) with all assigned variables added as constraints. Label node n with the value $z(n)$. Go to 2.

Kolesar's algorithm consists in taking as x_t in step 3(a) the unassigned variable with smallest index (i.e., $t = i$ for which v_i/w_i is the maximum for unassigned variables x_i).

We propose instead to follow [4] and to take as x_t in step 3(a) the variable that is fractional at node n.

To illustrate our branch and bound algorithm, we solve the problem given in [3]:

Index	w_i	v_i
1	30	60
2	50	60
3	40	40
4	10	10.
5	40	20
6	30	10
7	10	3

and $W = 100$. Solving (2), we obtain $z(1) = 140$ with $x_1 = 1$, $x_2 = 1$, $x_3 = \frac{1}{2}$. We label node 1 with $z(1) = 140$. Since x_3 is fractional we branch from node 1 and proceed to step 3(a). We solve (2) with $x_3 = 0$. This produces $z(2) = 135$ with $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$, $x_5 = \frac{1}{4}$. We label node 2 with $z(2) = 135$. In step 3(b) we set $x_3 = 1$. This produces $z(3) = 136$ with $x_1 = 1$, $x_2 = \frac{3}{5}$, $x_3 = 1$. We go to step 2 in the algorithm. We see that $z(3)$ is the maximum for all terminal nodes. Since the solution at node 3 has x_2 fractional, we branch from node 3 and proceed to 3(a). We solve (2) with $x_3 = 1$ and $x_2 = 0$ added as constraints. This produces $z(4) = 120$, with $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, $x_5 = \frac{1}{2}$. We proceed to 3(b) and solve (2) with $x_3 = 1$ and $x_2 = 1$. This produces $z(5) = 120$ with $x_1 = \frac{1}{3}$, $x_2 = 1$, $x_3 = 1$. Returning to step 2 we see that $z(2)$ is the maximum for all terminal nodes. Since the solution at node 2 has x_5 fractional we must branch from node 2. The method continues easily; the complete tree is shown in Figure 1. The optimal solution is given at node 8 with $z(8) = 133$, $x_1 = 1$, $x_2 = 1$, $x_4 = 1$, $x_7 = 1$ and all other $x_i = 0$.

The algorithm achieves solution with the generation of 9 nodes. The same problem is solved in [3] and requires 15 nodes. The choice rule used here for the selection of x_t is a clear improvement. Further computational experience is presented in §IV.

III. A Branch Search Algorithm

Branch and bound techniques normally require large amounts of computer storage and time because of the necessity to store information and make comparisons for all

Published in Management Science on January 01, 1970 as DOI: 10.1287/mnsc.16.5.327.
This article has not been copyedited or formatted. The final version may differ from this version.

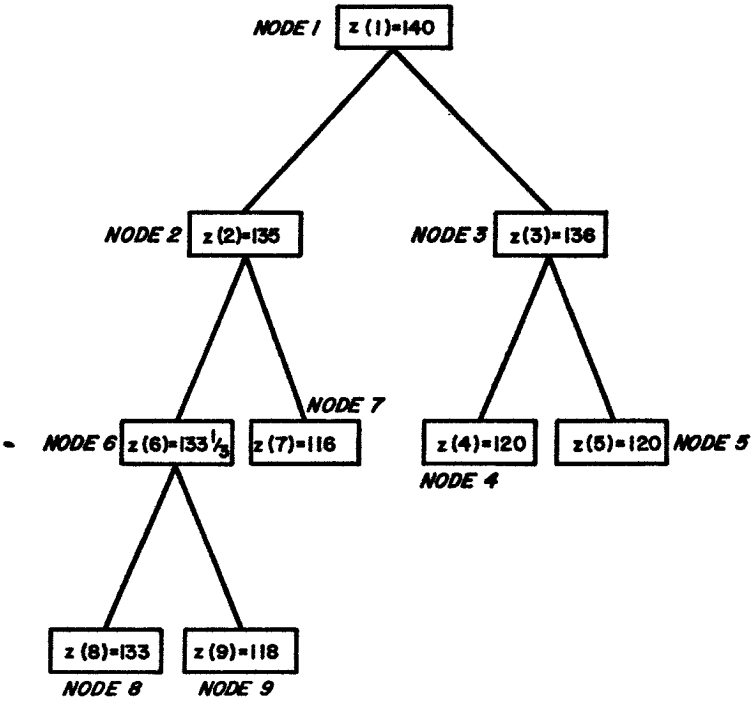

FIGURE 1. Complete tree for the example.

terminal nodes. In this section, we present a branch search algorithm for solving (1) that eliminates these drawbacks.

Our branch search algorithm first finds an obvious integer solution to the constraints of (1). This solution is a lower bound to the optimal solution. We develop a branch of a tree and explore each part of the branch until the lower bound is reached or until a new feasible solution is found that represents a larger lower bound. We then backtrack and develop new branches of the tree with possibly larger lower bounds. Further branching is excluded when the lower bound is reached. The algorithm stops when all new branches are excluded. The lower bound solution is optimal. The only information that is stored is the current lower bound solution and the branch routing.

To initiate the algorithm:

Define $[x]$ as the greatest integer less than or equal to x.

Define $S(x_0, x_1, \cdots, x_N)$ as the current lower bound solution where the x_i are all given and $x_0 = \sum_{i=1}^{N} v_i x_i$.

Define $X(x_1, x_2, \cdots, x_N)$ to indicate assigned variables. A value $x_i = 2$ (or any number not equal to zero or one) indicates that the variable is unassigned. An assigned variable will have the value zero or one.

Define $R(j)$ as the index of the j^{th} assigned variable.

The solution to (2), with L assigned components of X added as constraints, is

$$
\begin{aligned}
(3) \qquad & x_i = 1 \quad \text{if} \quad i < r, \quad i \neq R(j) \ (j = 1, \cdots, L) \\
& x_i = 0 \quad \text{if} \quad i > r, \quad i \neq R(j) \ (j = 1, \cdots, L) \\
& x_{R(j)} = 0 \quad \text{or} \quad 1 \quad (j = 1, \cdots, L) \text{ depending on the assignment} \\
& x_r = (W - \textstyle\sum_{i=1}^{L} w_{R(i)} x_{R(i)} - \sum_{i \in M(L)} w_i)/w_r \\
& Z(L+1) = \textstyle\sum_{i=1}^{L} v_{R(i)} x_{R(i)} + \sum_{i \in M(L)} v_i + v_r x_r
\end{aligned}
$$

where the set $M(L)$ is given by

$$M(L) = \{i \mid i < r, \quad i \neq R(j) \quad (j = 1, \cdots, L)\}$$

and r is the least integer $(0 \leq r \leq N)$ for which

$$\sum_{i \in M(L)} w_i + w_r \geq W - \sum_{i=1}^{L} w_{R(i)} x_{R(i)} .$$

If no r exists we have all $x_i = 1$ for $i \neq R(j)$ $(j = 1, \cdots, L)$. A lower bound to the solution of (1) is given by

$$x(L) = \sum_{i=1}^{L} v_{R(i)} x_{R(i)} + \sum_{i \in M(L)} v_i .$$

The algorithm follows:

1. Set $L = 1$ and all components of X to two. Go to 2.
2. Solve (2). If the solution is all integer we have the optimal solution. Stop. If not the solution is $x_1 = 1, x_2 = 1, \cdots, x_{r-1} = 1$ and x_r is fractional. We calculate $x_0 = \sum_{i<r} v_i$ and form $S(x_0, 1, 1, \cdots, 1, 0, 0, 0)$ as a lower bound to the optimal solution, where the zero components in S represent $x_i = 0$, $i \geq r$. Set $R(1) = r$ and the r^{th} component of X to zero. Go to 3.

3(a). Solve (2) with the L assigned components of X added as constraints. We obtain (3). If $[Z(L + 1)] \leq x_0$ go to 4. If $Z(L + 1) > x_0$ and we have an integer solution, take $x_0 = Z(L + 1)$, form a new $S(x_0, x_1, x_2, \cdots, x_N)$ from (3), and go to 4. If $[Z(L + 1)] > x_0$ and we do not have an integer solution, go to 3(b).

(b). If $x(L) > x_0$, take $x_0 = x(L)$ and form a new $S(x_0, x_1, \cdots, x_N)$ from (3) with $x_r = 0$. In any case, Set $L = L + 1$, take $R(L) = r$, and set the $R(L)$ component of X to zero. Go to 3(a).

4(a). If the $R(L)$ component of X is equal to zero change the component to one and go to 3(a). If the $R(L)$ component of X is one go to 4(b).

(b). If $L = 1$ the optimal solution is $S(x_0, x_1, \cdots, x_N)$. Stop. If $L \neq 1$ change the $R(L)$ component of X to two, set $L = L - 1$, and go to 4(a).

This completes the algorithm.[1] $z(L)$ represents the value of the objective function at the L^{th} level of a branch. $R(L)$ represents a routing of assignments along the branch. Only one branch is studied at a time. Preference is given to the branch with the fractional variable assigned the value of zero. This allows lower bounds to be achieved more rapidly. The only permanent storage information required is the current lower bound solution S, the assigned variables $R(L)$ at level L of the branch, and the assignment vector X.

In the problem given above, the algorithm as viewed in Figure 1 is as follows:

1. At node one the optimal solution to (2) is fractional. $x_0 = 120$.
2. Branch to node two, assigning $x_3 = 0$.
3. Branch to node six, assigning $x_5 = 0$.
4. Branch to node eight, assigning $x_6 = 0$. This produces an all integer result with $x_0 = 133, x_1 = 1, x_2 = 1, x_4 = 1, x_7 = 1$.
5. Back-track to node six; branch to node 9, assigning $x_6 = 1$.
6. Back-track to node 6, removing the assignment on x_6.
7. Back-track to node 2; branch to node 7, assigning $x_5 = 1$.
8. Back-track to node 2, removing the assignment on x_5.

[1] This type of branch search is similar to that in [2]. The algorithm here simplifies [2] by retaining only one of its tests and applying a different choice rule in the assignment of values to the variables.

Published in Management Science on January 01, 1970 as DOI: 10.1287/mnsc.16.5.327.
This article has not been copyedited or formatted. The final version may differ from this version.

9. Back-track to node 1; branch to node 3, assigning $x_3 = 1$. Calculate $x(1) = 100$; do not change the current lower bound.
10. Branch to node 4, assigning $x_2 = 0$.
11. Back-track to node 3; branch to node 5, assigning $x_2 = 1$.
12. Back-track to node 3, removing the assignment on x_2.
13. Back-track to node 1; end.
14. The optimal solution is given at node 8.

In this example, the total tree investigated has the same number of nodes as the tree developed by the algorithm in §II.

IV. Computational Results

We have programmed the two branch and bound schemes and the branch search algorithm in Fortran IV and have run test problems on the IBM 360/67. The coefficients v_i and w_i were generated as random integers and the three methods were then tried on the same problem. Let

BK = Kolesar's branch and bound algorithm,
BB = the branch and bound algorithm presented here,
BS = the branch search algorithm presented here.

The results for the number of nodes generated and the time to achieve solution are as follows:

		Average number of nodes generated.				Average time in Milliseconds.			
W	N	20	30	40	50	20	30	40	50
25	BS	28.8	28.0	28.8	35.7	61	87	120	164
	BB	26.4	20.6	26.6	28.6	201	282	395	494
	BK	56.2	49.6	48.0	50.0	259	345	447	552
50	BS	34.5	38.7	36.1	54.5	75	111	145	235
	BB	38.6	48.2	42.0	51.8	236	361	453	581
	BK	62.2	78.4	78.0	92.6	276	434	553	723
75	BS	19.8	39.0	27.8	18.5	58	116	128	135
	BB	31.0	30.2	27.2	25.0	222	312	404	485
	BK	62.6	101.2	84.4	61.6	280	523	578	616
100	BS	7.2	24.3	46.8	29.0	36	97	181	163
	BB	16.6	24.8	64.0	50.4	191	301	560	598
	BK	56.6	65.4	136.6	134.2	273	403	806	978

(10 problems run to obtain each average)

In these test problems the superiority of BB and BS over BK is clearly evident with regard to number of nodes generated. As for time considerations, the BS method is markedly faster than the other two methods, while BB is faster than BK. Various other problems than those listed were run, and in general, the BS method achieved solution in about one-half to one-third the time required by the BK and BB methods.

The branch search method was tried on several problems with 5000 variables and solution was achieved in all cases in approximately 4 minutes. The other two methods failed because of storage limitations.

In general, the branch search algorithm as given in this paper can solve extremely

large problems. The procedure requires minimal computer storage and time. The choice rule used here to assign a value to a variable is seen to be of distinct advantage.

References

1. DANTZIG, G. B., "Discrete Variable Extremum Problems," *Operations Research*, Vol. 5 (1957), pp. 266–277.
2. GLOVER, F., "A Multiphase Dual Algorithm for the Zero-One Integer Programming Problem," *Operations Research*, Vol. 13 (1965), pp. 879–919.
3. KOLESAR, P. J., "A Branch and Bound Algorithm for the Knapsack Problem," *Management Science*, Vol. 13 (1967), pp. 723–735.
4. LAND, A. H., AND DOIG, A. G., "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, Vol. 28 (1960), pp. 497–520.

Published in Management Science on January 01, 1970 as DOI: 10.1287/mnsc.16.5.327.
This article has not been copyedited or formatted. The final version may differ from this version.