The Change-Making Problem

J. W. WRIGHT

University of Manchester Institute of Science and Technology, Manchester, England

ABSTRACT. A cashier has a number of coins of different denominations at his disposal and wishes to make a selection, using the least number of coins, to meet a given total The solution given here employs dynamic programming Suggestions are made which reduce the volume of computation required in handling the recursive equations The method can be applied to the one-dimensional cargo-loading and stock-cutting problem, and it can be extended to the two-dimensional problem.

KEY WORDS AND PHRASES change-making, dynamic programming, knapsack problem, cargo loading problem

CR CATEGORIES 5 41, 5.42

1. *Introduction*

The change-making problem may arise in the following way. A cashier has a number of coins of different denominations at his disposal and wishes to make a selection, using the least number of coins, to meet a given total. It is assumed that the number of coins available in each denomination is not limited.

An algorithmic solution to this problem appeared in [1], but an alternative method is proposed here which is obtained by the use of dynamic programming. The change-making problem is a special case of the one-dimensional cargo-loading problem [2], often referred to as the knapsack problem. It has been used with linear programming in the solution of the trim-loss-cutting problem.

In the general case each item carries a utility value, but in the change-making problem these values are made equal to unity, which ensures that the variable to be minimized will be the total number of coins.

This paper suggests an improvement in the computational efficiency in solving the problem, which also may be readily extended to the more general case.

2. *Formulation*

An unlimited number of coins of denominations $w_1, w_2, \cdots, w_n$ are made available. For convenience without loss of generality, these may be ordered so that $w_1 < w_2 < w_3 < \cdots < w_n$.

Assuming that x_i coins of denominations w_i are selected to meet a total C, the problem to be solved then becomes:

$$\text{Minimize } Z = \sum_{i=1}^{n} x_i, \qquad i = 1, 2, \cdots, n,$$

$$\text{subject to } \sum w_i x_i = C,$$

$$x_i \geq 0 \text{ and integer.}$$

Author's address. Department of Management Sciences, University of Manchester Institute of Science and Technology, P. O Box 88, Manchester, M60, 1QD, England

3. *Solution by Dynamic Programming*

The recursive equations for solving this problem may be written as

$$f_N(z) = \min(x_N + f_{N-1}(z - x_N.w_N)), \tag{1}$$

where x_N is allowed to range over the values 0, 1, 2, $\cdots$, $[z/w_N]$, and where $[z/w_N]$ is the greatest integer less than or equal to z/w_N. For $N = 1$, $f_1(z) = [z/w_1]$. N is thus the number of stages in the multistage decision process such that, in turn, $N = 1, 2, \cdots, n$, where n is the number of different types of coin.

At each stage z is ranged from 0 to C in incremental steps, where C is the total sum to which the change must be totaled. To illustrate the use of this and later modified equations, values in [1] where $n = 5$, $w_i = (1,4,5,6,7)$, and $C = 352$, will be employed.

Thus the range of x_N at stage $N = 3$ (item 3) with $W_3 = 5$ requires at $z = 18$ the selection of the minimum value from

$$0 + f_2(18), \quad 1 + f_2(13), \quad 2 + f_2(8), \quad 3 + f_2(3).$$

This involves the calculation of four values and comparison of them to select the minimum value.

The number of calculations may be reduced to one with one comparison by replacing eq. (1) by the following:

$$f_N(z) = \min(f_{N-1}(z), \quad 1 + f_N(z - w_N), \tag{2}$$

provided $f_N(z - w_N)$ has previously been optimized. Since both z and w_i are positive, $(z - w_N)$ will be less than z, which permits the earlier optimization.

Table I illustrates the process using the value in [1].

It will be observed that the calculations at each stage s need not proceed beyond the point where $z = w_s \times w_{s+1}$. Since w_s coins of value w_{s+1} will match, in total, w_{s+1} coins of value w_s, the former requires fewer coins.

z will be allowed to range from 0 to C, which in large problems may result in a severe computational problem. It should be noted, however, that a computer program could

TABLE I SECTION OF TABLE SHOWING VALUES OF $f_N(z)$ FOR FIVE ITEMS

	Item									
	1	2		3		4		5		
	$f_1(z)$	$1 + f_2(z - 4)$	$f_2(z)$	$1 + f_3(z - 5)$	$f_3(z)$	$1 + f_4(z - 6)$	$f_4(z)$	$1 + f_5(z - 7)$	$f_5(z)$
0	0		0		0		0		0
1	1		1		1		1		1
2	2		2		2		2		2
3	3		3		3		3		3
4	4	1 + 0 = 1	1		1		1		1
5		1 + 1 = 2	2	1 + 0 = 1	1		1		1
6		1 + 2 = 3	3	1 + 1 = 2	2	1 + 0 = 1	1		1
7		1 + 3 = 4	4	1 + 2 = 3	3	1 + 1 = 2	2	1 + 0 = 1	1
8		1 + 1 = 2	2	1 + 3 = 4	2	1 + 2 = 3	2	1 + 1 = 2	2
9		1 + 2 = 3	3	1 + 1 = 2	2	1 + 3 = 4	2	1 + 2 = 3	2
10		1 + 3 = 4	4	1 + 1 = 2	2	1 + 1 = 2	2	1 + 3 = 4	2
11		1 + 4 = 5	5	1 + 2 = 3	3	1 + 1 = 2	2	1 + 1 = 2	2
12		1 + 2 = 3	3	1 + 3 = 4	3	1 + 1 = 2	2	1 + 1 = 2	2
13		1 + 3 = 4	4	1 + 2 = 3	3	1 + 2 = 3	3	1 + 1 = 2	2
14		1 + 4 = 5	5	1 + 2 = 3	3	1 + 2 = 3	3	1 + 1 = 2	2
15		1 + 5 = 6	6	1 + 2 = 3	3	1 + 2 =			3
16		1 + 3 = 4	4	1 + 3 = 4	4				
17		1 + 4 = 5	5	1 +					
18		1 + 5 = 6	6						
19		1 + 6							

take advantage of the fact that each value of w_i at each stage can be used as an address modifier to extract the appropriate value of f_N.

Again, z would range in the usual dynamic program formulation over the values 0 to C at each of the n stages, resulting in $n \cdot C$ calculations and comparisons, the optimum value of f_n being stored at each stage to be available at the next.

The following modification will considerably reduce the volume of computation.

4. *An Improved Method*

Instead of having z range from 0 to C in each of the n stages, it is advantageous to reverse this procedure by having N range from 1 to n for each value of z, as Table II shows.

The optimum value of f^* is obtained from the equation

$$f^*(z) = \min_{y - w_i > 0} f^*(z - w_i) + 1, \qquad i = 1, 2, \cdots, n. \tag{3}$$

It should be noted that intervening values of f_N need not be stored, although for explanatory purposes they are included in Table II.

It may also be observed that the table terminates at the value $z = 17$, assuming in this case that alternative optima are not required. The circled entries are those values that have the highest value of w_i which are used to determine $f^*(z)$. Whereas in the case of item 4, the preceding w_i (= 6) entries are not circled in this sense, it is apparent that such an item can, for subsequent values of z, no longer contribute to the optimum and the calculation may be terminated for that item. When this condition arrives in the case of the penultimate item the table may be closed. It is then a simple calculation to derive the residual number of coins required for the last item.

5. *Decoding the Result and Conclusion*

To keep track of the patterns, a column headed "decode" may be appended, which contains the item number of the circled entry. Using this value as an address modifier, the pattern of items representing the optimum is easily ascertained.

Where alternative optima are required, modification of the rules requiring closure of an item will be necessary, but they present no difficulty. The decode column would then require some listing procedure, with the address of the starting position for each value of z being stored in the decode column.

TABLE II DERIVATION OF $f^*(N)$

z	1	2	3	4	5	$f^*(z)$	Decode
0	0					0	0
1	1					1	1
2	2					2	1
3	3					3	1
4	4	①				1	2
5	close	2	①			1	3
6		3	2	①		1	4
7		4	3	2	①	1	5
8		close	4	3	②	2	5
9			②	4	3	2	3
10			2	②	4	2	4
11			2	2	②	2	5
12			2	2	②	2	5
13			3	2	②	2	5
14			3	3	②	2	5
15			close	3	③	3	5
16				3	③	3	5
17				close			

Modification of eq. (2) to accommodate the knapsack problem requires the substitution of v_i for the value of 1, where v_i is the value of the item. This again presents no difficulty.

Extension to the two-dimensional number can also be effected.

REFERENCES

1 CHANG, S K., AND GILL, A Algorithmic solution of the change-making problem. *J. ACM 17*, 1 (Jan 1970), 113–122

2 BELLMAN, R E , AND DREYFUS, S E *Applied Dynamic Programming* Princeton U Press, 1962.

RECEIVED DECEMBER 1973, REVISED FEBRUARY 1974