1. Publication Nº	2. Version	3. Date	5. Distribution
INPE-3898-PRE/940		*May., 1986*	☐ Internal ☒ External
4. Origin: *DIN*	Program: *POEPS/INFOR*		☐ Restricted

6. Key words - selected by the author(s)
DYNAMIC PROGRAMMING *PLANAR PROCEDURE*
ENUMERATION SCHEME *IMPLEMENTATION*

7. U.D.C.:

8. Title *INPE-3898-PRE/940*

10. Nº of pages: *17*

11. Last page: *15*

SOME ASPECTS TO BE CONSIDERED IN THE ATTEMPT TO FIND MORE EFFICIENT METHODS FOR SOLVING OPTIMIZATION PROBLEM BY ENUMERATION

12. Revised by

Paulo Renato de Morais

13. Authorized by

Marco Antonio Raupp
Diretor Geral

9. Authorship *Nei Yoshihiro Soma (*)*
Horacio Hideki Yanasse

Responsible author

14. Abstract/Notes

In a new enumeration scheme to solve the unidimensional knapsack problem, two characteristics called our attention: a) the reduction of an N-dimensional problem to one in the plane; b) visiting states and stages in a different sequence than the traditional dynamic programming improved memory and computational requirements. The implementation resulted in a more efficient algorithm compared with others using dynamic programming. In this work we focus on these particularities, making comparisons with the traditional methods. We believe that such observations are potentially useful to other researches in developing new and more efficient methods for solving some optimization problems.

15. Remarks *(*) Instituto Tecnológico da Aeronáutica*
This paper was accepted for presentation at the III Latin Ibero American Congress on Operations Research, to be held in Santiago, Chile, Aug 18 through 22, 1986.

RESUMO

Em um novo esquema enumerativo para resolver o problema da mochila unidimensional, duas características chamaram a nossa atenção: a) a redução de um problema N-dimensional para um, no plano; b) percorrendo estados e estágios em uma sequência diferente daquela da programação dinâmica tradicional ocasionou melhoras nos requisitos computacionais e de memória. A implementação resultou em um algoritmo mais eficiente comparado com outros que usam programação dinâmica. Neste trabalho são focalizadas estas particularidades fazendo comparações com os métodos tradicionais. Acredita-se que tais observações sejam potencialmente úteis a outros pesquisadores para desenvolver nossos métodos mais eficientes de resolver alguns problemas de otimização.

ABSTRACT

In a new enumeration scheme to solve the unidimensional knapsack problem, two characteristics called our attention: a) the reduction of an N-dimensional problem to one in the plane; b) visiting states and stages in a different sequence than the traditional dynamic programming improved memory and computational requirements. The implementation resulted in a more efficient algorithm compared with others using dynamic programming. In this work we focus on these particularities, making comparisons with the traditional methods. We believe that such observations are potentially useful to other researchers in developing new and more efficient methods for solving some optimization problems.

1. INTRODUCTION

Let us define the unidimensional knapsack problem (KP)

$$\text{Max } Z = \sum_{j=1}^{N} C_j x_j$$

$$\text{Subject to } \sum_{j=1}^{N} A_j x_j = B;$$

$$x_j \varepsilon \mathbb{N} \quad , j=1,\dots, N;$$

$$A_j, B \varepsilon \mathbb{N}, j=1,\dots, N;$$

This problem, although simple, is quite representative of the class of integer linear problems (see SALKIN [9]). Recall that there are results on aggregation methods for discrete problems (see Onyekwelu

[8]; Kendall and Zionts [6]; Kannan [5]) where systems of equations are aggregated to a single equation. Therefore, many integer problems can be reduced in theory to a KP of the previous form.

This KP is considered in Yanasse and Soma [10]. There, they propose on $(N(B-A_1)- \sum_{j=1}^{N} A_j)$ algorithm, assuming without loss of generality that $A_1 < A_2 < \ldots < A_n$.

The Yanasse and Soma's algorithm presents some particularities that we think would be worth discussing.

To be self-contained we present the Yanasse and Soma's algorithm in Section 2. In Section 3 two important features of this method are discussed. In Section 4 we present some final comments.

2. THE YANASSE AND SOMA'S ALGORITHM

The Yanasse and Soma's algorithm for solving (KP) is based on Mignosi [7] work and can be formulated as:

Algorithm

Step 0 [Initialization]

Make a list $Z(A_1)$, $Z(A_1+1)$, $Z(A_1+2)$,..., $Z(B-A_1)$ and $Z(B)$ and set

$$Z(I)= \begin{cases} C_j, \text{ for } I=A_j;\ j= 1, 2,\ldots, N \\ -1, \text{ otherwise} \end{cases}$$

Set POINTER $\leftarrow A_1$

Step 1 DO FOR J←1 TO N

```
        BEGIN
          IF POINTER + A_j ≤ B-A_1 OR
             POINTER + A_j=B
               BEGIN
                 ZLIN(POINTER+A_j)= Z(POINTER) + C_j
                 IF ZLIN(POINTER+A_j) > Z(POINTER+A_j)
                   THEN
                     Z(POINTER+A_j)= ZLIN(POINTER+A_j)
               END
          IF POINTER+A_j > B, THEN GO TO step 2
      END
```

Step 2: POINTER $\leftarrow$ POINTER+1

IF POINTER > $B-A_1$, THEN GO TO Step 3

IF Z(POINTER) < 0, THEN GO TO Step 2

ELSE GO TO Step 1

Step 3: If Z(B) < 0, THEN the problem is infeasible, STOP

ELSE the optimal value is Z(B).

A few comments are necessary at this point. It is assumed in this algorithm that all C_j's are nonnegative. Adequate modifications can be made to include any real C_j.

The variable Z(k) carries the best objective value encountered so far at each step of the algorithm for a right-hand side equal to k of the KP. If Z(K) is negative for some K at the end of the algorithm this implies that the problem is infeasible for the right-hand side equal to K.

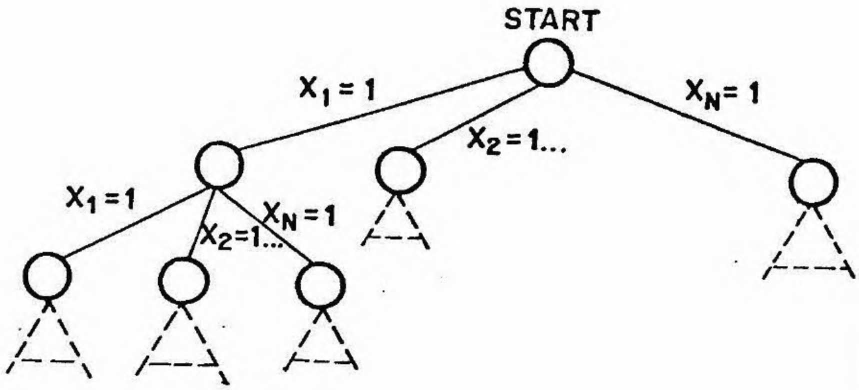
The algorithm as previously formulated arrives only at the optimal value. To also obtain the optimal solution, we must define a new auxiliary variable. For details see Yanasse and Soma [10].

The variable POINTER indicates the position of the right-hand side value below which all pptimal values have already been computed.

The variable ZLIN(K) keeps the objective value relative to the feasible solution obtained in that step for the right-hand side equal to K.

As can be seen the algorithm is quite simple. It enumerates feasible solutions with right-hand side starting from A_1 to B, always keeping the best value encountered so far.

One can say that the algorithm makes the following enumeration which, at first glance, does not seem that would have a good performance:



The better performance of the algorithm (as compared with other dynamic programming methods) is due to the implementation of such enumeration.

We present now some interesting features of this algorithm.

3. FEATURES OF THE ALGORITHM

With N decision variables, it would be difficult to represent the (KP) graphically when N is greater than 3. The traditional graphical way of solving linear programs can be applied to integer ones, but only to problems with a small number of variables, two or at most three.

In the Yanasse and Soma [10] algorithm, the N-dimensional problem is solved by an enumeration scheme that can be interpreted, under a geometric point of view, as solving a problem in the plane.

To illustrate this, we will consider the following example.

$$\text{Max.} \quad 5x_1 + 7x_2 + 9x_3$$

$$\text{Subject to} \quad 5x_1 + 7x_2 + 9x_3 = 32, \quad x_1, x_2, x_3 \in \mathbb{N} \tag{1}$$

That is, we just want to know if the linear diophantine equation (1) has a solution or not.

It is possible to draw the feasible region correspondent to equation (1) in a three-dimensional space. However if we have more variables this task would become difficult or impossible.

If one follows the Yanasse and Soma's algorithm to solve the example, one can see that what is done is equivalent to some specified operations over a grid of size 32 as shown in figures 1 to 4.

We build a square grid of size B and draw diagonals in the positions corresponding to $A_1, A_2, \ldots, A_n$. We also draw a guideline which is a secondary diagonal as shown in figure 1.

Starting from the first black dot from the top left we draw a horizontal line that crosses the guideline at A (see figure 1). From A we draw a vertical line that crosses the diagonal lines at B, C and D, respectively. From B, C, D we draw horizontal lines that cross the vertical scale at 10, 12 and 14. So, these positions are marked with black dots and they indicate values for which equation 1 has a solution for the right-hand side equal to that value.

We proceed to the immediately near black dot and perform these same operations. This is schematized in figure 2.

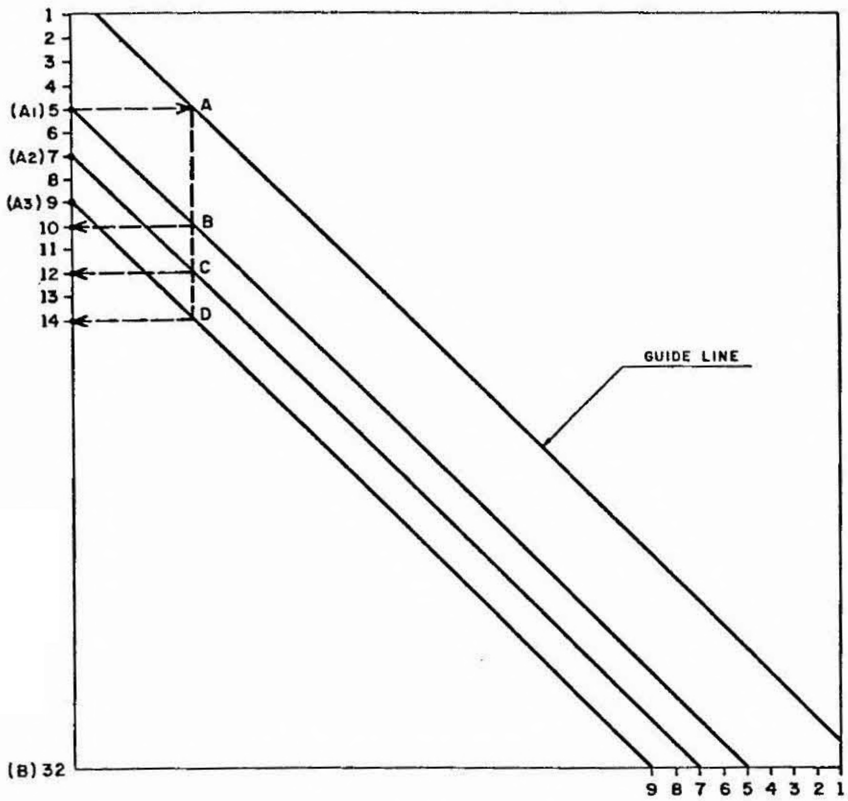
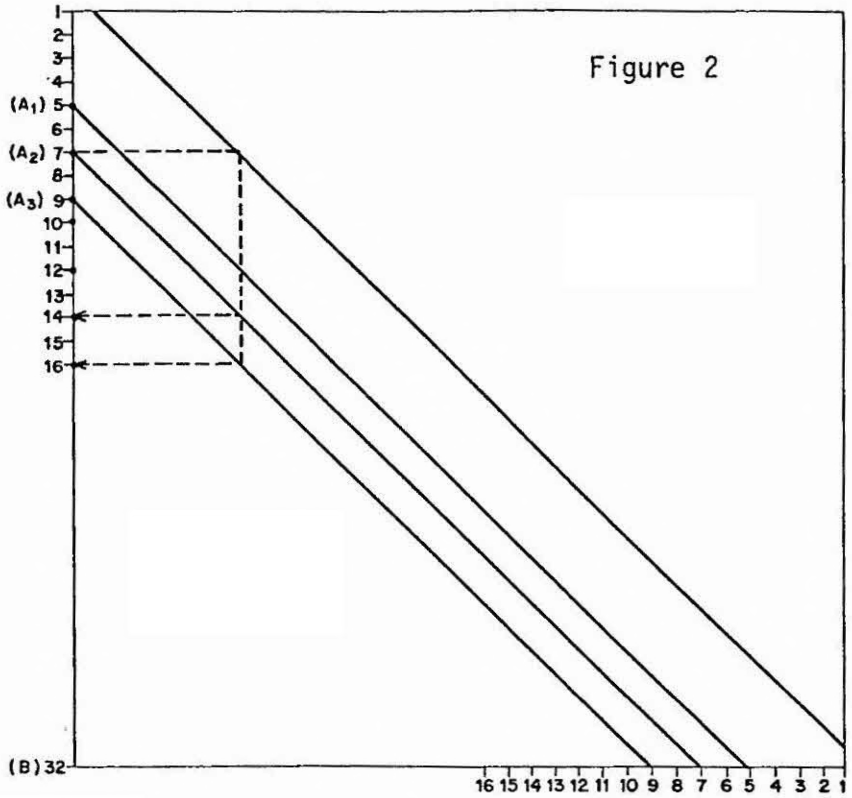
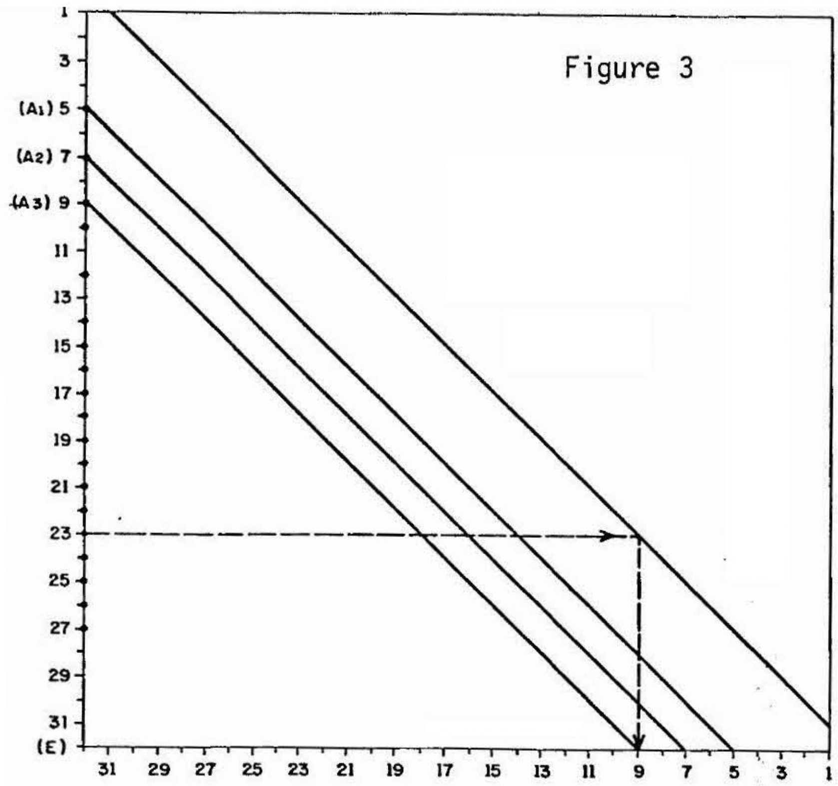


Figure 1



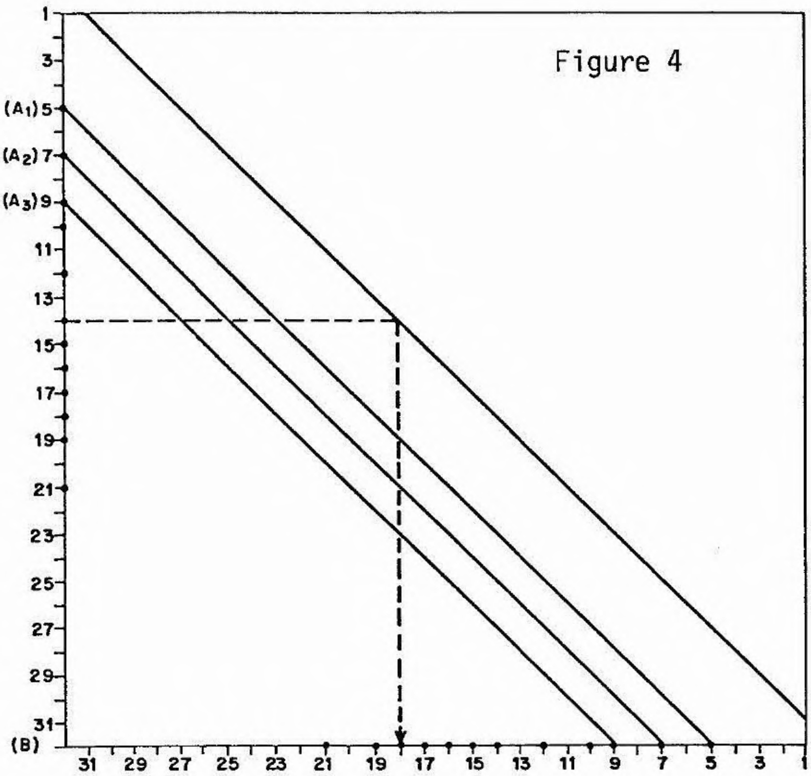
After a few iterations we arrive at the position shown in figure 3 indicating in this example that equation 1 has a feasible solution.



The same steps would be performed in the case where the objective function is of a general form. The only difference would be the necessity of keeping the objective value in correspondence with the black dots in the vertical scale.

Variations can also be suggested, for instance, making black dots in the horizontal scale in correspondence with the black dots in the vertical scale, so that when one draws the vertical line from a point in the guideline, if one reaches a black dot in the horizontal scale one could stop immediately.

In figure 4 we illustrate what we would achieve with this variation.



As can be seen from the previous example, the problem was reduced to one in the plane. We should only work with a grid of size B, draw diagonals corresponding to the values A_1, A_2,..., A_n and mark dots conveniently, according to some specified rules. Observe that these operations can be done for <u>any</u> KP of <u>any</u> size (at least in theory) of the form presented, which is quite interesting.

Considering the Yanasse and Soma's algorithm under another point of view, let us make a comparison with a dynamic programming method.

If one solves KP using dynamic programming we would end up, for instance, with the following recursion (see Garfinkel and Nemhauser [1] Gilmore and Gomory [2], [3], [4]):

$$f(k,g) = \max_{x_k = 0,\ldots,\lfloor g/A_k \rfloor} (C_k x_k + f(k-1,\ g-A_k x_k)) \qquad (2)$$

for each $k=1,\ldots,N$, with $f(0,0) \triangleq 0$ and $f(0,g) \triangleq -\infty$ for $g=1,\ldots,B$.

$f(k,g)$ is the best objective value using the first $\underline{k}$ items, with the right-hand side equal to g and $\lfloor g/A_k \rfloor$ is the greatest integer less than or equal to (g/A_k).

We have then N+1 stages and B+1 states for each stage.

The recursion (2) implies that we have to find all values for the states in stage k-1 before going to stage k. In the Yanasse and Soma's algorithm this does not happen. They do some computation in stage k even if all the computations in stage k-1 are not completed. Let us illustrate this with a small example. Consider the problem:

$$\begin{aligned} &\text{Max } 5x_1 + 7x_2 + 11x_3 \\ &\text{Subject to: } 2x_1 + 3x_2 + 5x_3 = 11 \\ &\qquad x_1, x_2, x_3 \in N \end{aligned}$$

If we solve this problem using recursion 2 we would have:

$f(0,0)= 0$, $f(0,g)= -\infty$, $g= 1,\ldots, 11$,

$f(1,0)= \max(0+f(0,0))= 0$,

$f(1,1)= \max(0+f(0,1))= -\infty$,

$f(1,2)= \max(0+f(0,2),\ 5+f(0,0))= 5$,

$f(1,3)= \max(0+f(0,3),\ 5+f(0,1))= -\infty$,

$f(1,4)= \max(0+f(0,4),\ 5+f(0,2),\ 10+f(0,0))= 10$,

$f(1,5)= \max(0+f(0,5),\ 5+f(0,3),\ 10+f(0,1))= -\infty$,

$f(1,6)= \max(0+f(0,6),\ 5+f(0,4),\ 10+f(0,2),\ 15+f(0,0))= 15$,

$f(1,7)= \max(0+f(0,7),\ 5+f(0,5),\ 10+f(0,7),\ 15+f(0,1))= -\infty$,

$f(1,8)= \max(0+f(0,8),\ 5+f(0,6),\ 10+f(0,4),\ 15+f(0,2),\ 20+f(0,0))=20$

$f(1,9)= -\infty$,

$f(1,10)= 25$,

$f(1,11)= -\infty$,

$f(2,0)= \max(0+f(1,0))= 0,$

$f(2,1)= \max(0+f(1,1))= -\infty,$

$f(2,2)= \max(0+f(1,2))= 5,$

$f(2,3)= \max(0+f(1,3),\ 7+f(1,0))= 7,$

$f(2,4)= \max(0+f(1,4),\ 7+f(1,1))= 10,$

$f(2,5)= \max(0+f(1,5),\ 7+f(1,2))= 12,$

$f(2,6)= \max(0+f(1,6),\ 7+f(1,3),\ 14+f(1,0))= 15,$

$f(2,7)= \max(0+f(1,7),\ 7+f(1,4),\ 14+f(1,1))= 17,$

$f(2,8)= \max(0+f(1,8),\ 7+f(1,5),\ 14+f(1,2))= 20,$

$f(2,9)= \max(0+f(1,9),\ 7+f(1,6),\ 14+f(1,3),\ 21+f(1,0))= 22,$

$f(2,10)= \max(0+f(1,10),\ 7+f(1,7),\ 14+f(1,4),\ 21+f(1,1))= 25,$

$f(2,11)= \max(0+f(1,11),\ 7+f(1,8),\ 14+f(1,5),\ 21+f(1,2))= 27,$

$f(3,0)= \max(0+f(2,0))= 0,$

$f(3,1)= \max(0+f(2,1))= -\infty,$

$f(3,2)= \max(0+f(2,2))= 5,$

$f(3,3)= \max(0+f(2,3))= 7,$

$f(3,4)= \max(0+f(2,4))= 10,$

$f(3,5)= \max(0+f(2,5),\ 11+f(2,0))= 12,$

$f(3,6)= \max(0+f(2,6),\ 11+f(2,1))= 15,$

$f(3,7)= \max(0+f(2,7),\ 11+f(2,2))= 17,$

$f(3,8)= \max(0+f(2,8),\ 11+f(2,3))= 20,$

$f(3,9)= \max(0+f(2,9),\ 11+f(2,4))= 22,$

$f(3,10)= \max(0+f(2,10),\ 11+f(2,5),\ 22+f(2,0))= 25,$

$f(3,11)= \max(0+f(2,11),\ 11+f(2,6),\ 22+f(2,1))= 27.$

The dynamic programming steps can be schematized as in figures 5, 6 and 7.

If we solve this problem using Yanasse and Soma's algorithm we would have:

<u>Step (0)</u> $Z(2)= 5,$

$Z(3)= 7,$

$Z(5)= 11,$

$Z(4)= Z(6)= Z(7)= Z(8)= Z(9)= Z(11)= -1;$

POINTER= 2

Step (1) $ZLIN(2+2) = Z(2)+5 \rightarrow ZLIN(4) = 10$,
$ZLIN(4) > Z(2+2) \rightarrow Z(4) = ZLIN(4) = 10$,
$ZLIN(2+3) = Z(2)+7 \rightarrow ZLIN(5) = 12$,
$ZLIN(5) > Z(5) \rightarrow Z(5) = ZLIN(5) = 12$,
$ZLIN(2+5) = Z(2)+11 \rightarrow ZLIN(7) = 16$,
$ZLIN(7) > Z(7) \rightarrow Z(7) = ZLIN(7) = 16$;

Step (2) $POINTER = 3$, $POINTER \leq 9$,
$Z(3) = 11 > 0$;

Step 1 $ZLIN(3+2) = Z(3)+5 \rightarrow ZLIN(5) = 12$,
$ZLIN(5) = Z(5)$
$ZLIN(3+3) = Z(3)+7 \rightarrow ZLIN(6) = 14$,
$ZLIN(6) > Z(6) \rightarrow Z(6) = ZLIN(6) = 14$,
$ZLIN(3+5) = Z(3)+11 \rightarrow ZLIN(8) = 18$,
$ZLIN(8) > Z(8) = 18$,

Step 2 $POINTER = 4$, $POINTER \leq 9$,
$Z(4) = 10 > 0$,

Step 1 $ZLIN(4+2) = Z(4)+5 \rightarrow ZLIN(6) = 15$,
$ZLIN(6) > Z(6) \rightarrow Z(6) = 15$,
$ZLIN(4+3) = Z(4)+7 \rightarrow ZLIN(7) = 17$,
$ZLIN(7) > Z(7) \rightarrow Z(7) = 17$,
$ZLIN(4+5) = Z(4)+11 \rightarrow ZLIN(9) = 21$,
$ZLIN(9) > Z(9) \rightarrow Z(9) = 21$;

Step 2 $POINTER = 5$, $POINTER \leq 9$,
$Z(5) > 0$;

Step 1 $ZLIN(5+2) = Z(5)+5 = 17$,
$ZLIN(7) = Z(7)$,
$ZLIN(5+3) = Z(5)+7 = 19$,
$ZLIN(8) > Z(8) \rightarrow Z(8) = 19$,
$POINTER + 5 > 9$,

Step 2 $POINTER = 6$, $POINTER \leq 9$,
$Z(6) > 0$;

Step 1 $ZLIN(6+2)= Z(6)+5= 20,$

$ZLIN(8) > Z(8) \rightarrow Z(8)= 20,$

$ZLIN(6+3)= Z(6)+7= 22,$

$ZLIN(9) > Z(9) \rightarrow Z(9)= 22,$

$ZLIN(6+5)= Z(6)+11= 26,$

$ZLIN(11) > Z(11) \rightarrow Z(11)= 26;$

Step 2 $POINTER= 7, \quad POINTER \leq 9,$

$Z(7) > 0;$

Step 1 $ZLIN(7+2)= Z(7)+5= 22,$

$ZLIN(9)= Z(9),$

$POINTER + 3 > 9,$

$POINTER + 5 > 11,$

Step 2 $POINTER= 8, \quad POINTER \leq 9,$

$Z(8) > 0;$

Step 1 $POINTER + 2 > 9,$

$ZLIN(8+3)= Z(8)+7= 27,$

$ZLIN(11) > Z(11) \rightarrow Z(11)= 27$

$POINTER + 5 > 11;$

Step 2 $POINTER= 9, \quad POINTER \leq 9,$

$Z(9) > 0;$

Step 1 $ZLIN(9+2)= Z(9) + 5= 27,$

$ZLIN(11)= Z(11),$

$POINTER + 3 > 11,$

$POINTER + 5 > 11,$

Step 2 $POINTER > 9$ stop.

In figures 8, 9, 10 we schematize the previous steps.

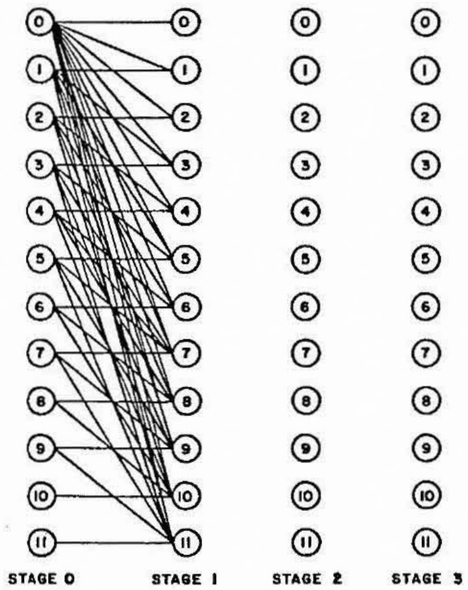


Figure 5

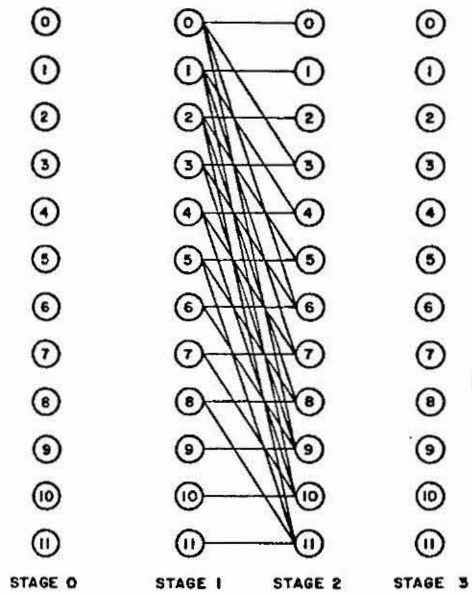


Figure 6

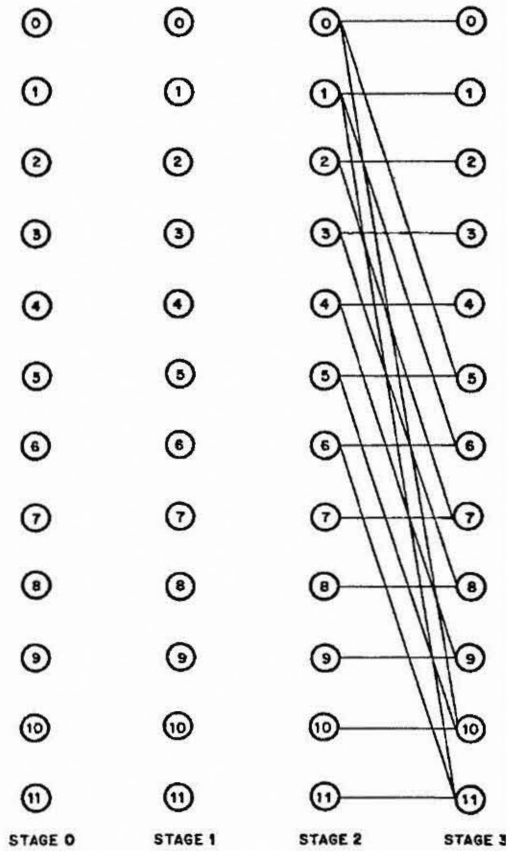


Figure 7

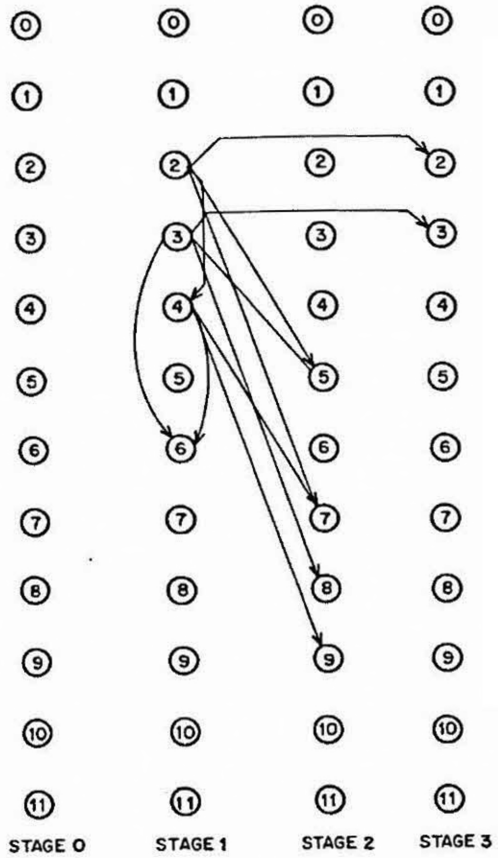


Figure 8

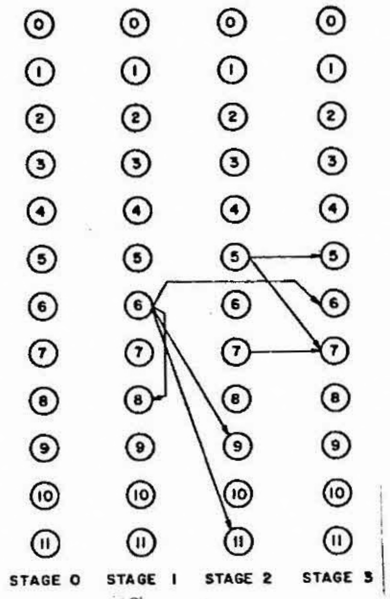


Figure 9

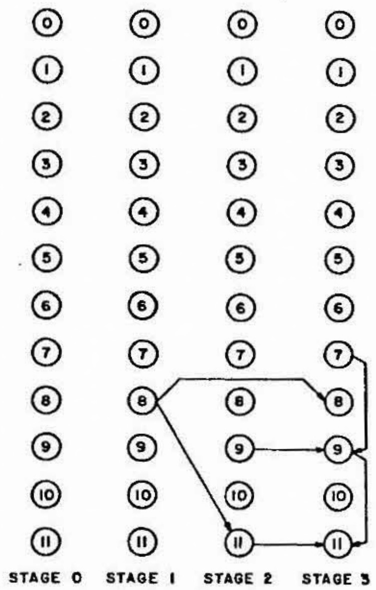


Figure 10

As can be seen, there are economies in computation since only the relevant states to the optimal solution in each stage are visited.

We can see that for this problem it was possible to follow a different sequence of states and stages to compute the objective values of interest. We conjecture that this might be true for other problems that are solved by dynamic programming.

This different approach might lead to improved algorithms for such problems.

4. FINAL COMMENTS

We first presented here a "planar" solution procedure for solving an N-dimensional integer problem. The example shown is particular but perhaps it might be extended to other "planar" solution procedures for more general N-dimensional integer problems.

The second aspect we tried to show was the different enumeration scheme as compared with dynamic programming. This led to savings in computation and memory requirements.

We believe that these aspects discussed are interesting and potentially useful to other researchers in developing new and more efficient methods for solving some optimization problems by enumeration. It would be interesting if we could establish the general conditions under which one can follow a different order than the "serial" one, stage after atage, used in dynamic programming methods. This is a topic that still needs further research.

REFERENCES

[1] R. Garfinkel; G.L. Nemhauser, "Integer Programming", John Wiley and Sons, New York, 1972.

[2] P. Gilmore; R. Gomory, "A linear programming approach to the cutting stock problem II", Operations Research, 11(6), 863-888, 1963.

[3] P. Gilmore; R. Gomory, "Multistage cutting stock problems of two and more dimensions", Operations Research, 13, 94-120, 1965.

[4] P. Gilmore; R. Gomory, "The theory and computation of knapsack functions", Operations Research, 14, 1045-1074, 1966.

[5] R. Kannan, "Polynomial time aggregation of Integer Programming Problems", Journal of ACM, 30(1), 133-145, 1983.

[6] K. Kendall; S. Zionts, "Solving Integer Programming aggregation constraints", Operations Research, 25(2), 346-351, 1977.

[7] G. Mignosi, "Sulla Equazione Lineare Indeterminata", Periodico di Matematica, 23, 173-176, 1908.

[8] D.C. Onyekwelu, "Computational viability of a constraint aggregation scheme for Integer Linear Programming Problems", Operations Research, 31(4), 795-861, 1983.

[9] H. Salkin, "Integer Programming", Addison-Wesley, Reading, 1975.

[10] H.H. Yanasse; N.Y. Soma, "A new enumeration scheme for the knapsack problem". Presented at the school of Combinatorial Optimization, 8-19 July 1985, UFRJ, Rio de Janeiro, RJ. (INPE-3563-PRE/769 - June 1985).