procedure of dynamic programming algorithms are varied depending on the structure of models to be applied on, a knapsack type model is used as a prototype model for the development of the Hybrid algorithm.

REFERENCES

1- *Ahmad , N. V., " Optimization of Large Scale 0-1 Integer Linear Programming Problems With Multiple Choice Constraints", Ph.D. Dissertation, Texas A&M University, Texas, 1978.*

2 - *Brooks, R., and Geoffrion, A., " Finding Evertt's Lagrangian Multiplier by Linear Programming ", Operations Research, Vol. 14 , No. 6 (1966) , 1149-1152.*

3- *Fisher, M.L., " The Lagrangian Relaxation Method for Solving Integer Programming Problems" , Management Science, Vol. 27 (1981), 1-18.*

4- *Ghassemi - Tari, F., " An Algorithm for Large Scale Nonlinear Knapsack Problems and Extensions", Ph.D. Dissertation, Texas A&M University, Texas 1980.*

5- *Glover , F., " Surrogate Constraint Duality in Mathematical Programming ", Operations Research, Vol. 23 , No. 5 (1970), 434451*

6- *Greenberg, H.J., and Pierskalla , W.P. , " Surrogate Mathematical Programming ", Operations Research, Vol. 18 No.5 (1970), 924939.*

7- *Greenberh, H.j., " The Generalized Penalty Function / Surrogate Model", Operations Research, Vol. 18 (1973), 162-178.*

8- *Hallefjord, A. and Story, S., " Aggergation and Disaggergation in Integer Programming Problem", Operations Research, Vol. 38 (1990) , 619-623.*

9- *Marsten, R.E., and Morin, L.T., " A Hybrid Approach to Discrete Mathematical Programming " , Mathematical Progrmming, Vol. 14 (1987) , 21-40.*

10- *Mathur K., Salkin, H.M ., and Marito S., " A Branch and Search Algorithm for a Class of Nonlinear Knapsack Problem" , Operations Research Letters, Vol. 2 , No. 4 (1983) , 155 - 160.*

11- *Mathur K., Salkin, H.M., and Mohanty, B.B., " A Note on a General Nonlinear Knapsack Problem", Operations Research Letters, Vol. 5 , No. 2 (1986) . 79-81.*

12- *Mizukami, K., and Sikrorski, J., " Three Algorithm for Calculating Surrogate Constraints in integer Programming Problems" , Control and Cybernetics, Vol. 13 , No. 4 (1984), 375- 397.*

13- *Morin, L.T. , and Esogbue, M.u., " The Imbedded State Space Approach to Reducing Dimensionality in Dynamic Progrms of Higher Dimensions", Journal of Mathematical Analysis and Applications, Vol. 48 (1974), 801-810.*

14- *Morin, T.L., and Marsten, R.E. , " Branch and Bound Strategies for Dynamic Prgoramming " , Operations Research.Vol. 24, No.4 (1976), 611-627.*

15- *Morin, T.L and Marsten, R.E,. " An Algorithm for Nonlinear Knapsack Problems" , Management Science, Vol. 22, No. 10 (1976) , 1147-1158.*

16- *Morin, T.L., " Computational Advances in Dynanic Programming" , Research No. 78-1 Purdue University, Indiana, 1974.*

17- *Nauss , R.M., " The 0-1 Knapsack Problem With Multiple Choice Constraaints", Univrsity of Missouri, St. Louis, Missouri, 1976.*

18- *Silvano, M., and Paolo, T., " A Program for the 0-1 Multiple Knapsak Problem" , ACME Transactions, Vol. 11, No. 2 (1985), 135-140.*

19- *Sinha, P., and Zoltners, A.A, " The Multiple Chioce Knapsak Problem" , Management Science, Vol. 27 , No. 3 (1979), 503-515.*

20- *Toyoda, Y., and Senju, S., " An Approach to Linear Programming With 0-1 Variables", Management Science, Vol. 15, No. 4 (1968), 196-207.*

21- *Toyoda, T., " A Simplifed Algorithm for Obtaining Approximate Solutions to Zero - One Programming Problems" , Management Science , Vol. 21, No. 12(1975), 1417- 1427.*

computations are conducted in four stages. It is to be noted that, the information regarding the state and return values at each stage are stored in a matrix called TS. This matrix consists of M+1 columns and a number of rows which varies for each stage. In addition another matrix Tj is defined composed of two columns and the same number of rows as TS Matrix, where the rows of Tj posses a one to one correspondence to TS. The first column of Tj Matrix stores the values of the decision variables, Xj s , and the second column stores an index number showing the row number of previous stage from which the current state is obtained. This column facilitates the backtracking procedure to obtain the optimal values of decision variables at each stage.

The flowing are the results obtained by computational procedure of the hybrid algorithm using the above mentioned lower and upper bounds:

STAGE 1

TS			T1	
0	0	0	1	1
6	3	2	2	1
8	4	3	3	1
9	5	5	4	1
11	7	8	5	1

STAGE 2

TS			T2	
0	0	0	1	1
6	3	2	1	2
7	4	3	2	1
9	5	5	1	4
11	7	8	1	5
18	11	11	2	5
21	13	12	3	5

STAGE 3

TS			T3	
15	12	13	5	1
19	13	14	3	4
21	14	16	4	4
21	15	17	3	5
23	16	19	4	5
26	19	21	5	5

STAGE 4

TS			T4	
27	23	24	3	4

The computation is terminated in step 28. The optimal return is obtained as $R^* = 24$ and the optimal solution of variables recorded as $X^* = (5 , 1 , 3 , 3)$.

The solution obtained by this procedure is one of the alternative sloutions to this problem. As it is seen in table (1) , there are two other optimal decision variables with the same amount of return. They are; $X^* = (1 , 1 , 5 , 5)$, and $X^* = (1 , 2 ,4 , 4)$.

CONCLUSION

In this paper an efficient algorithm is developed to solve the nonlinear discrete optimization problems. The concept of imbedded state approach, surrogate constraint technique, branch and bound procedure, and updating routine for lower and upper bounds are incorporated in a dynamic programmming algorithm.

Due to this integeration an efficient algorithmic procedure called the Hybrid algorithm is developed. The developed algorithm is capable of reducing the state space and limits the growth of solution space of dynamic programming in each iteration. This provides an extenisve saving in computational time and the required memory storage of computer to solve a large problem. Since the behavior and the computational

Step 25 - If $LB_j = UB_j$ gb to step 27, otherwise go to step 26

Step 26 -If j=N go to step 28 , otherwise go to step 11

Step 27 - If j<N , set X_{j+1} , X_{j+2} , ... , X_N = o go to step 29, otherwise go to step 28

Step 28 - Set R^* = Max . R (g^i_j) and recover X^*_k for K= 1 , 2,... N and stop.

Step 29 - Set $R^* = UB_j$ and recover X^*_k for k = 1 , 2 , ... j and stop.

Step 30 - Indicate that the solution is obtained by surrogate problem , Set $R^* = UB_1$ and recover X^*_k for k = 1 , 2 , ... N , and stop.

EXAMPLE

Consider the general nonlinear knapsack problem with the following data:

R1 (X1) = (0 , 2 , 3 , 5 , 8)
R2 (X2) = (0 , 3 , 4 , 5 , 6)
R3 (X3) = (0 , 6 , 9 , 11 , 13)
R4 (X4) = (0 , 4 , 7 , 10 , 11)
Xj = { 1 , 2 , 3 , 4 , 5)
A11 (X1) =(0 , 6 , 8 , 9 , 11)
A21 (X1) = (0 , 3 , 4 , 5 , 7)
A12(X2) = (0 , 7 , 10 , 12 , 14)
A22(X2) = (0 , 4 , 6 , 8 , 10)
A13(X3) = (0 , 8 , 10 , 12 , 15)
A23(X3) = (0 , 6 , 8 , 9 , 12)
A14(X4) = (0 , 5 , 6 , 9 , 10)
A24(X4)= (0 , 4 , 8 , 12 , 15)
B1 = 28
B2 = 28

Solution:

The surrogate problem with the equal weight of each constraint can be written as below:

RSP = Max. Rj (Xj)

Subject to :

Aj (Xj) < B
Xj Sj

Where

A1 (X1) = (0 , 4 , 6 , 7 , 9)
A2 (X2) = (0 , 5 , 8 , 10 , 12)
A3(X3) = (0 , 7 , 9 , 10 , 13)
A4(X4) = (0 , 4 , 7 , 10 , 12)
Bj = (1 , 2 , 3 , 4 , 5) for j = 1 , 2 , 3 , 4
B = 28

The solution to the surrogate prblem is RSP(28)=27 with the value of X = (5 , 1 , 3 , 4)for the decision variables. However , this solution is not feasible with respect to the original problem. Thus, a search must be conducted to obtain the highest RSP value which is feasible with respect to the original problem. The result of this search is RSP (25) = 24 which is used as the initial lower bound for the hybrid algorithm; and thus we have LB = 24 and UB = 27 . Table (1)presents the final solution of the initial surrogate problem.

Table (1) . Solution of the surrogate problem

STATE	RETURN	X1	X2	X3	X4
0	0	1	1	1	1
4	4	1	1	1	3
7	7	1	1	1	3
9	9	1	1	3	1
10	11	1	1	4	1
13	13	1	1	5	1
13	13	1	1	3	2
14	15	1	1	4	2
16	16	1	1	3	3
17	18	1	1	4	3
19	19	5	1	4	1
19	19	1	1	3	4
20	21	1	1	4	4
22	22	1	1	4	5
23	23	5	1	4	2
25	24	5	1	3	3
25	24	1	1	5	5
25	24	1	2	4	4
26	26	5	1	4	3
28	27	5	1	3	4

Using the above initial lower and upper bounds, the statgewise computations of the algorithm is started. Since there are four decision variables, the

resource type i if alternative Xj of object j is selected.

The nonlinear knapsack model reduces the number of variables and eliminates all the multiple choice constraints. A promising solution approach to handel nonlinear version of the knapsack model is dynamic programming technique. However, this technique performs very poorly when it is faced with the well-known problem of dimensionality of state variables.

In this research effort an algorithm is developed which is capable of solving the general version of nonlinear knapsack models. The steps to be taken by the developed algorithm is presented in the next section.

DEVELOPMENT OF AN ALCORITHM

A General description of the hybrid algorithm is given below. Steps 1-8 of the algorithm comprise the solution of the surrogate problem to obtain an initial lower and upper bounds. If the optimal solution of the surrogate problem satisfies the feasibility condition of the main problem, steps 9-29 are discarded and the final solution will be obtained by step 30; otherwise, further computations will take place starting from step 9. The computations regarding the first stage are performed through steps 10-11. Steps 11 and 12 comprise the constraction of the imbedded state space. The reductions in the imbedded state space through feasibility and dominancy tests are performed by steps 18-22. The lower and upper bounds are updated by steps 23-24 and the remaining steps comprise the required tests for termination of the algorithm.

Step 1 - Set $R=0$, and $\alpha_i = \frac{1}{MM}$ for $i = 1, 2, ..., M$,

Step2-Set $A_j(X_j) = \left[\sum_{i=1} \alpha_i \, a_{ij}(X_j) \right]$ for $j=1,2,...,N$

Step3-set $B = \begin{cases} \sum_{i=1}^{M} \alpha_i \, b_i & \text{if } \sum_{i=1}^{M} \alpha_i \, b_i = \left[\sum_{i=1}^{M} \alpha_i \, b_i \right] \\ \left[\sum_{i=1}^{M} \alpha_i \, b_i \right] + 1 & \text{otherwise} \end{cases}$

Step 4 - Find the optimal solution to the following problem ;

$$\text{Max. } R_{SP}(B) = \sum_{j=1}^{N} r_j(X_j)$$

Subject to :

$$\sum_{j=1}^{N} A_j(X_j) \le B$$

Call the solution as $\underline{X^*}$ and $R^*_{SP}(B)$, and let $UBo = R^*_{SP}(B)$.

Step 5 - If $\sum_{j=1}^{N} a_{ij}(X^*_j) \le b_i$ for $i = 1,2, ... M$ go to step 8, otherwise go to step 6.

Step 6 - Set $l = l+1$

Step 7 - Let $\hat{B} = B-l$ and find the optimal solution of $R_{SP}(\hat{B})$, call this solution X^* ,$R^*_{SP}(\hat{B})$,and go to step 5.

Step 8 - If $l > 0$, set $LB_0 = R^*_{SP}(\hat{B})$, go to step 9 , otherwise set $LB_0(B) = UB_0(B)$, go to step 30.

Step 9 - Let $j=1$, $S=S_j$, $F^e_j = G_j = \{a_{j1}(X_j)$, and $a_{j2}(X_j)$, ... ,

$a_{jM}(X_j)$; for $X_j \varepsilon S\}$ and $r(g_j) = \{ r_j(X_j) \mid X_j \varepsilon S \}$

Step 10 - Go to step 16.

Step 11- Let $j = j+1$, $S=S_j$, $K_j=K$ and $G_j = \{ a_{j1}(X_j)$, $a_{j2}(X_j)$, ... , $a_{jM}(X_j)$; for $X_j \varepsilon S\}$

Step 12 - $F_j = G_j \, o \, F^e_{j-1}$

Step 13 - $F^f_j = \{ g^i_j \mid g^i_j \varepsilon F_j$ and $g^i_j \le (b_1 , b_2 , .. B_M)\}$

Step 14 - $R_j = \{ r^i_j \mid r^i_j = R_j(g^i_j) + r_j(\underline{X}_j)$ and $\underline{X}_j \in S_j \}$

Step 15 - Let $F^e_j = \{ F^f$ - all the points dominated by better points$\}$

Step 16- Let the number of points in F^e_j be KK

Step 17 - Set $i=1$

Step 18 - Let $UB1^i_j = R(g^i_j) + \sum_{t=j+1}^{N} r_t(K)$

Step 19 - Let $UB2^i_j = R(g^i_j) + R_{SP}(B -[B'(g^i_j)])$

Step 20 - Let $R(g^i_j) = -1$ for those i which statisfy $UB1^i_j < LB_{j-1}$ or $UB2^i_j < LB_{j-1}$

Step 21 - If $i=KK$ go to step 22 otherwise set $i=i+1$ and go to step 18

Step 22 - Redefine $R(g_j)$ by elimination of all negative $R(g_j)$

Step 23 - Let $UB_j = \text{Min.} \{\text{Min}_i . UB2^i_j , UB_{j-1} \}$

Step 24 - Set $LB_j = \text{Max} . \{ \text{Max}_i . R(g^i_j) , LB_{j-1} \}$

approach is usually less compared to lagrangian method, however a unified theory to provide a basis for algorithmic procedures using surrogate constraint methods has not emerged and general computational applications are waiting a unified theory.

The third attempt is the approach of Morin and Esogbue [13], who developed an algorithmic procedure for solving dynamic programming problems by searching over an imbedded state space at any stages. Latter Morin and Marsten have shown how the use of different methods, such as branch & bound, and imbedded state approaches can be implemented in dynamic programming for reducing its computational requirements [14] [15] [16].

In this paper an efficient algorithm to reduce the dimensionality of state space solution of a dynamic programming model when it is applied to a discrete nonlinear knapsack problem is presented. The developed algorithm is essentially a form of dynamic programming method. Several algorithmic procedures such as the concepts of imbedded state, surrogate constraint, and branch and bound, are incorporated in the algorithm to increase its computational capability through the reduction of state space solution vector. A numerical example is also presented.

THE SCOPE OF MATHEMATICAL MODEL

the simplest version of the knapsack model is constructed when we have n objects to be selected but, because of some limitations, not all can be selected [20] , [21]. A more general version of the knapsak model is the case where there are several kinds of limitations (mostly resources), several sets of mutually exclusive alternatives for each object, and we are interested to select one alternative from each set [1] , [17] , [18], [19]. To be more precise, it is assumed that " no selection" of an object is included in each set of alternatives. The restriction of having only one alternative to be selected from each set adds a set of constraints, called multiple choice constraints, to the original model.

A zero - one mathematical model for the later case can be presented as below:

$$\text{Max } Yo = \sum_{i=1}^{M} \sum_{j=1}^{N} Uij \; Yij$$

subject to

$$\sum_{i=1}^{M} \sum_{j=1}^{N} Cijk \; Yij < Bk \quad \text{for } k = 1, 2, ..., M$$

$$\sum_{i=1}^{M} Y\,ij = 1 \quad \text{for } j = 1, 2, ..., N$$

Where

N = total number of objects

T = total number of alternative objects in a set

M = total number of resources

Uij = utility of selecting alternative i of object j

Cijk = amount of usage of resource k when alternative i of object j is selected

Ek = total amount of resource k

Yo = total utility earned

Yij = a zero - one variable having value one if the alternative i of object j is selected, and zero otherwise

An alternative mathematical model for this problem is a discrete nonlinear knapsack model [4] , [11]. This model can be presented in a form of discrete nonlinear knapsack model as below:

$$\text{Max } Xo = \sum_{j=1}^{n} Rj\,(Xj)$$

subject to

$$\sum_{j=1}^{n} Aij\,(Xj) < Bi \text{ for } i = 1, 2, ..., M$$

$$Xj = 1, 2, ..., T \text{ for } j = 1, 2, ..., N$$

Where

Xj = an integer variable indicating the alternative number for selection of object j

Rj = an integer valued function indicationg the amount of return if alternative Xj of object j is selected

Aij = an integer valued function indicating the amount of the resource consumption of

A HYBRID ALGORITHM FOR DISCRETE NONLINEAR OPTIMIZATION

FARHAD GHASSEMI TARI, Ph.D.

DEPARTMENT OF INDUSTRIAL ENGINEERING
SHARIF UNIVERSITY OF TECHNOLOGY
TEHRAN, IRAN

ABSTRACT

Optimization of a discrete nonlinear model suffers from several obstacles, namely the discontinuity of objective function and constraints, nonconvexity of mathematical relations, and the existence of inequality constraints. Dynamic programming technique may consider the only solution approach to this type of models. The use of this powerful technique however is limited since the growth of the number of variables and constraints requires an extensive computer memory storage and computational time.

In this research effort a hybrid algorithm is developed to overcome the mentioned difficulties. The developed algorithm is basically a dynamic programming solution approach. The imbedded state technique is incorporated to the algorithm to solve the problem of dimensionality of dynamic programming approach. To limited the growth of state space solution the concept of branch and bound is employed. The lower and upper bound for the optimal solution is calculated by the mean of imbedded state approach and is updated through the progress of algorithm in each iteration. An illustrative example is also presented.

INTRODUCTION

Dynamic programming is a well known solution approach for optimization of a separable function which provides a global optimal solution even in case of non convex programming problems. However the use of this powerful technique for discrete variable problems has been limited by its excessive computer memory storage and computational requirements [7]. These computational difficulties become more sever whenever the state variables are defined by a vector of more than three dimensions and / or the state variables are in a lower dimension but the number of discontinuities of state grows exponantionally in the algorithmic solution procedures.

Considerable research has deen devoted to overcome the problem of dimensionality in dynamic programming techniques. Earlier attempts were made through employing the concept of lagrangian multipliers [2] [3]. Another attempts in this area was based on the concept of surrogate constraints [8] [9] [10]. Both lagrangian and surrogate constraint framework suffer from existence of the duality gap. Although the preformed studies by Greenberg [6] [7] , Glover [5], Mizukami [12], have shown that the occurrence and the size of the gap in surrogate