Journal of Industrial Engineering International *Islamic Azad University, South Tehran Branch*
March 2006, Vol. 2, No. 1, 31 - 37

A dynamic programming approach for solving nonlinear knapsack problems

E. Jahangiri

Assistant Professor, Islamic Azad University, Science and Research Branch, Tehran, Iran

F. Ghassemi-Tari*

Associate Professor, Sharif University of Technology, Tehran, Iran

Abstract

Nonlinear Knapsack Problems (NKP) are the alternative formulation for the multiple-choice knapsack problems. A powerful approach for solving NKP is dynamic programming which may obtain the global optimal solution even in the case of discrete solution space for these problems. Despite the power of this solution approach, it computationally performs very slowly when the solution space of the problems grows rapidly. In this paper the authors developed a procedure for improving the computational efficiency of the dynamic programming for solving KNP. They incorporate three routines; the imbedded state, surrogate constraints, and bounding scheme, in the dynamic programming solution approach and developed an algorithmic routine for solving the KNP. An experimental study for comparing the computational efficiency of the proposed approach with the general dynamic programming approach is also presented.

Keywords: Discrete optimization; Multiple-choice knapsack; Imbedded state; Surrogate constraint

1. Introduction

A Nonlinear Knapsack Problem (NKP) is an alternative formulation for a multiple-choice knapsack problem. A powerful approach for solving the nonlinear knapsack problems is Dynamic Programming (DP) which may obtain the global optimal solution even in the case of discrete solution space for these problems. Despite the power of this approach, it computationally performs very slowly when the solution space of the problems grows rapidly.

Mathematically KNP are presented in a form of mathematical programming models. Depending on the specific properties of the objective function and/or constraints, many versions of the nonlinear knapsack problem have been addressed in the literature. Variations include continuous or integer variables, convex or non-convex functions, separable or non-separable functions, and, in some cases, additional specially structured constraints such as bounds on the variables or generalized upper bound constraints [4,5].

Of course, any nonlinear optimization method developed for problems with multiple constraints could be used to solve the nonlinear knapsack problem. However, due to the combinatorial nature of the mathematical model, the computation time grows very rapidly as the number of the decision variables increases. This is from the fact that, the computational time in combinatorial optimization varies exponentially as the variation of the number of decision variables. From this fact it is realized that the specialized algorithms are much faster and more reliable than standard nonlinear programming software. Of course in case of continuous nonlinear knapsack problem we do not encounter with this computational difficulty. For example, Bretthauer and Shetty [3] developed a specialized algorithm for a class of continuous quadratic problems and found their method to be up to 4000 times faster than the general purpose reduced gradient software LSGRG [14].

The NKP have a variety of applications, including financial models [13], production and inventory man-

* *Corresponding author. E-mail:ghasemi@sharif.edu*

agement [6,11], stratified sampling [7], the optimal design of queuing network models in manufacturing [1,2], computer systems [8] and health care.

Although there are large bodies of literature addressing a variety of applications of NKP, most of the research efforts have been focused on knapsack problems with a linear objective and a linear constraint [12] and knapsack problems with a separable convex nonlinear objective function and a simple linear equality constraint [9,10].

Bretthauer and Shetty [4] developed a pegging algorithm for the nonlinear resource allocation problems. In another research attempt they conducted a thorough survey of the nonlinear knapsack model [5]. In their survey they categorized the nonlinear knapsack models according to the structure of the models and then they presented the most recent advancements for each model.

In this paper the authors consider a specific form of the NKP which fall in the category of non-convex separable integer. More specifically we consider a type of the NKP which is an alternative formulation of a general type of the Linear Multiple-Choice Knapsack Problems (LMCKP) with the multiple recourse constraints.

2. Mathematical models for the problem

The multiple choice knapsack problems may be formulated in a linear from as follows:

Let us first define the parameters used for developing the linear form of the mathematical model,

X_o : The total value of items selected to be put in the knapsack.

x_{jk} : $\begin{cases} 1 & \text{if the alternative } k \text{ of item } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$

n : The total number of items.

K_j : The total number of alternative item j.

M : The total number of resources.

r_{jk} : The value of alternative k for item j.

b_{ijk} : The amount of the consumption of resource i of alternative k of item j.

B_i : The level of available resource for resource i.

The mathematical model for LMCKP is as follows:

$$\text{Max } X_0 = \sum_{j=1}^{n}\sum_{k=1}^{K_j} r_{jk}x_{jk}$$

Subject to:

$$\sum_{j=1}^{n}\sum_{j=1}^{K_j} b_{ijk}x_{jk} \le B_i \quad \text{for} i = 1,2,\cdots,m \tag{1}$$

$$\sum_{k=1}^{K_j} x_{jk} = 1 \quad \text{for } j = 1,2,\cdots,n$$

$$x_{jk} = 0 \, or \, 1 \quad \text{for } j = 1,2,\cdots,n, \text{and} k = 1,2,\cdots,K_j .$$

An alternative form for formulating the multiple choice knapsack problems is a nonlinear form as follows:

$$\text{Max } X_0 = \sum_{j=1}^{n} R_j(x_j)$$

Subject to:

$$\sum_{j=1}^{n} c_{ij}(x_j) \le B_i \qquad \forall\, i = 1,2,\cdots m \tag{2}$$

$$x_j \in S = \{1,2,\cdots K_j\}$$

$$B_i = \{b_1, b_2, \cdots, b_m\}.$$

In the latter mathematical formulation, x_j is defined as the decision variable representing the j^{th} project, which its value represents the alternative number of the j^{th} project; i.e., when it takes value of k, it means that the k^{th} alternative of project j is selected. $R_j(x_j = k)$ is defined as a discrete non-linear function of variable x_j, representing the return of selecting the alternative of k of project j. $c_{ij}(x_j = k)$ is defined as a discrete non-linear function of project x_j, representing the consumption of resource i when alternative k of project j, is selected.

3. Development of the algorithm

Basically the developed algorithm is a dynamic programming approach in which three routines are incorporated for decreasing the solution space and hence for increasing the computational efficiency of the DP approaches.

The researchers employed the forward computation of the DP approach for developing of the proposed algorithm. First, the concept of the imbedded state approach is employed in each stage of the DP in or-

der to limit the state space solution only to those feasible solution points which cause a jump in the objective function value. Second, the authors employed the concept of surrogate constraint to reduce the m dimensional state pace vector to a single dimension sate pace vector. By incorporating the surrogate constraint, it should first be proved that any feasible solution of the original problem constitutes a feasible solution the surrogate constraint problem. The authors prove this fact through stating a theorem as follow:

Theorem. any feasible solution of the original problem is a feasible solution of the surrogate constant problem.

Proof. Let us consider $X^k = (x_1^k, x_2^k, \cdots, x_n^k)$ as a feasible solution point of the original problem. Therefore we have:

$$\begin{aligned} &\sum_{j=1}^{n} c_{1j}(x_j^k) \le B_1 \\ &\sum_{j=1}^{n} c_{2j}(x_j^k) \le B_2 \\ &\quad\vdots \\ &\sum_{j=1}^{n} c_{mj}(x_j^k) \le B_m . \end{aligned} \tag{3}$$

Now by multiplying a non-negative value such as $\alpha_i \ge 0$, the above inequalities remain unchanged and we have:

$$\begin{aligned} &\alpha_1 \sum_{j=1}^{n} c_{1j}(x_j^k) \le \alpha_1 B_1 \\ &\alpha_2 \sum_{j=1}^{n} c_{2j}(x_j^k) \le \alpha_2 B_2 \\ &\quad\vdots \\ &\alpha_m \sum_{j=1}^{n} c_{mj}(x_j^k) \le \alpha_m B_m . \end{aligned} \tag{4}$$

By adding up each side of these inequalities we will have:

$$\sum_{i=1}^{m} \alpha_i \sum_{j=1}^{n} c_{ij}(x_j^k) \le \sum_{i=1}^{m} \alpha_i B_i . \tag{5}$$

The last inequality reveals that the point which was feasible to the original problem satisfies the constraint of the surrogate problem.

Therefore the authors can solve the surrogate problem instead of the original problem and they are assured that by searching through the feasible space of the surrogate problem to obtain the optimal solution of the original problem they are considering all the solution points without losing even a single point.

Finally a bounding routine is employed for discarding those points of the state space solution which may lead to a solution worse than the existing lower bound. In order to incorporate the latter routine, it is essential to have an algorithm to find a lower bound for the optimal value of the objective function. Determination and use of a good lower bound can efficiently reduce the state space solution of the problem. The authors developed a heuristic algorithm which may obtain a good solution of the problem. To develop a powerful heuristic algorithm, the authors first developed several routines based on the concept of the constrained gradient. Since there are more than one resource constraints in our model, it is essential to develop a combined form of the constrained gradient. To choose the most efficient heuristic, the authors conducted an experimental study through which they solved several randomly generated test problems by the developed routines and they selected the heuristic routines which provided the best average objective function value.

Having a lower bound, at each stage (let us assume stage *j*), a bounding test is performed. To perform this test, it is assumed that in further stages the best alternative of the remaining projects are to be selected (for stage *j+1, j+2, …, J*). The authors then add the return value of these alternatives to the objective value of each point in the state space solution, and consider them as the best possible objective function values for these points. Those solution points which have the best possible objective value less than the lower bound are also discarded from the state space solution. Since the solution space grows exponentially as the computation progresses, discarding some solution points in the early stage of the computation efforts will result in a considerable elimination of the solution space of the problem at the end. The proposed algorithm has been summarized in Section 4.

4. Algorithm

Step 1. Obtain the surrogated problem by substituting all the resource constraints of the original

problem by a single constraint which is determined by the summation of all the resource constraints.

Step 2. Select the best alternative from each project as an initial solution.

Step 3. If this solution is a feasible solution to the original problem go to Step 8, otherwise go to Step 4.

Step 4. Determine the gradient of each alternative for each project by dividing the objective function return corresponding to that alternative by its associated resource consumption in surrogated constraint.

Step 5. Flag the gradient values of the selected alternatives, and consider the selected alternative as an initial solution.

Step 6. Select the largest un-flagged gradient value and recognize its associated alternative of that project as a new solution which substitutes the previous alternative to form the new solution.

Step 7. If the new solution is a feasible solution to the original problem go to Step 8, otherwise flag this gradient value and go to Step 6.

Step 8. Consider the solution obtained as a lower bound of the optimal solution, and let $j=1$.

Step 9. For the stage j, determine all the combinations of the state variables and calculate their associated returns and sort them in non-decreasing order.

Step 10. Add the value of $\sum_{i=j+1}^{n} R_i(x_i = K_j)$ to the return values of each state variable and consider them as the highest possible values that a state variable may be reached. Now discard those state variables which have the highest possible values less than the lower bound of the objective function.

Step 11. If $j=n$ go to Step 12, otherwise let $j=j+1$ and use the state variable vector (after discarding the non-promising state variables) for obtaining all possible combinations of the next stage state variables vector and go to Step 9.

Step 12. Select the largest return among the state variables vector as the optimal solution of the surrogated constraint. If this solution is feasible to the original problem stop, otherwise search through the state variables vector to find the largest value which is feasible to the original problem and recognize it as the optimal solution and then stop.

5. Experimental study

The authors conducted an extensive computational experiment for verifying the efficiency of the proposed algorithm. A set of test problems which are classified according to different values of *J's, K's*, and *M's* is randomly generated. To solve these problems, the authors defined a factor by which they could change the tightness of the generated problems constraints. They used TF symbols for this factor. For TF equal to one, they would have the original generated problems. Then they used the values of 0.90, 0.95, 1.00, 1.05, and 1.10 for the TF to generate additional different set of problems.

As the TF becomes larger the constraints become less tight and as it becomes smaller the constraint becomes tighter. Through this factor, the authors could evaluate the efficiency of our algorithm more thoroughly.

In each class, five problems with different input values were generated. The authors solved these problems by the regular DP algorithm and by their proposed algorithm and they determined the amount of state space reductions which are achieved through the use of their proposed algorithm. The computational result reveals a considerable reduction in state space solution.

Table 1. State space solution points which are enumerated by the proposed algorithm for different values of TF in problems with 4 projects.

TF Values	Problem number				
	1	2	3	4	5
0.90	172	263	182	120	69
0.95	172	246	182	34	69
1.00	172	200	182	17	69
1.05	51	200	182	17	69
1.10	51	115	182	17	41

Table 2. State space solution points which are enumerated by the proposed algorithm for different values of TF in problems with 5 projects.

TF Values	Problem number				
	1	2	3	4	5
0.90	107	186	339	109	16
0.95	107	186	177	27	16
1.00	56	91	177	27	16
1.05	56	91	177	27	13
1.10	56	91	177	27	13

Table 3. State space solution points which are enumerated by the proposed algorithm for different values of TF in problems with 6 projects.

TF Values	Problem number				
	1	2	3	4	5
0.90	144	539	733	1540	751
0.95	56	246	335	1398	240
1.00	56	246	244	860	240
1.05	22	246	244	860	240
1.10	4	246	10	860	92

Table 4. State space solution points which are enumerated by the proposed algorithm for different values of TF in problems with 7 projects.

TF Values	Problem number				
	1	2	3	4	5
0.90	696	1246	1062	3143	739
0.95	98	1246	1062	3143	739
1.00	98	150	345	440	115
1.05	48	150	345	440	109
1.10	48	84	345	440	119

Table 5. The average number of state space solution points for different problem size and according to the values of TF.

TF Values	Average number of points enumerated			
	For 4 projects	For 5 projects	For 6 projects	For 7 projects
0.90	140	151	741	1377
0.95	128	102	455	1257
1.00	103	73	329	229
1.05	81	72	322	218
1.10	140	72	242	207
number of points enumerated by classical DP	**1020**	**4092**	**16380**	**65532**

Table 6. The computational efficiency of proposed algorithm (state space reduction) comparing to the classical DP.

TF Values	The problems size			
	For 4 projects	For 5 projects	For 6 projects	For 7 projects
0.90	84.20	96.30	95.47	97.90
0.95	86.22	97.49	97.19	98.08
1.00	87.45	98.21	97.99	99.65
1.05	89.82	98.22	98.03	99.67
1.10	92.04	98.22	98.52	99.68

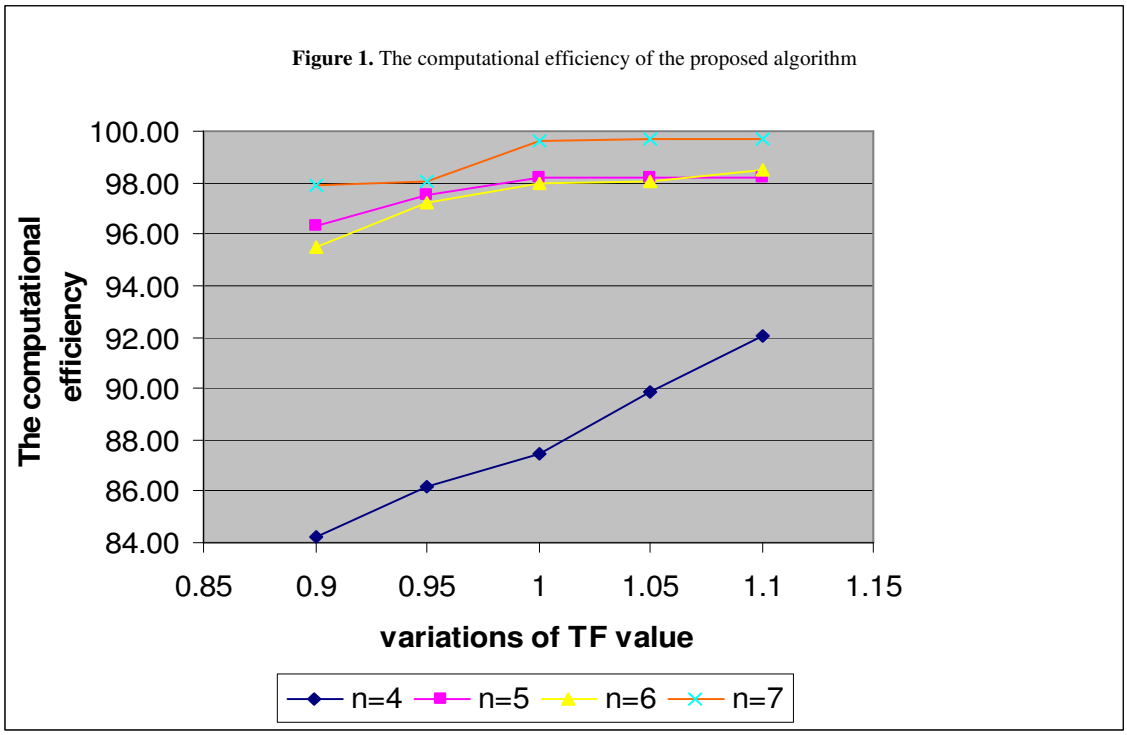

Figure 1. The computational efficiency of the proposed algorithm

6. Conclusion

In this paper, the researchers developed an efficient algorithm for solving the multiple choice knapsack problems. The experimental study reveals that the proposed algorithm powerfully reduces the state space solution and hence provides an efficient solution approach. The interesting fact is that, the efficiency of the proposed algorithm increases as the size of problem becomes larger.

For further research, one may evaluates the efficiency of surrogate problems when they are defined by the use of different coefficients. To perform this, the duality concept can be used and for those constraints which are less tighter, one can define smaller coefficients, while he can define the larger coefficient for those constraints which are tighter.

Another research that is proposed is the development of a set of new heuristic methods for obtaining the tighter binding for the optimal solution which might lead to the more reduction of state space solution.

References

[1] Bitran, G. R. and Tirupati, D., 1989a, Tradeo. Curves, targeting and balancing in manufacturing queuing networks. *Operations Research,* 37, 547–564.

[2] Bitran, G. R. and Tirupati, D., 1989b, Capacity planning in manufacturing networks with discrete options. *Annals of Operations Research,* 17, 119–135.

[3] Bretthauer, K. M. and Shetty, B., 1997, Quadratic resource allocation with generalized upper bounds. *Operations Research Letters*, 20, 51–57.

[4] Bretthauer, K. M. and Shetty, B., 2001, A pegging algorithm for the nonlinear resource allocation problem. *Computers and Operations Research,* 29(5), 505–527.

[5] Bretthauer, K. M. and Shetty B., 2002, The nonlinear knapsack problem - algorithms and applications. *European Journal of Operational Research,* 138, 459–472.

[6] Bretthauer, K. M., Shetty, B., Syam, S. and White, S., 1994, A model for resource constrained production and inventory management. *Decision Sciences,* 25, 561–580.

[7] Cochran, W. G., 1963, *Sampling Techniques,* second edition, John Wiley & Sons, New York.

[8] Gerla, M. and Kleinrock, L., 1977, On the topological design of distributed computer networks. *IEEE Transactions on Communications,* 25, 48–60.

[9] Hochbaum, D. S., 1994, Lower and upper bounds for the allocation problem and other nonlinear optimization problems. *Mathematics of Operations Research,* 19, 390–409.

[10] Ibaraki, T. and Katoh, N., 1988, *Resource Allocation Problems.* MIT Press, Cambridge, MA.

[11] Maloney, B. M. and Klein, C. M., 1993, Constrained multi-item inventory systems: an implicit approach. *Computers and Operations Research,* 20, 639–649.

[12] Martello, S. and Toth, P., 1990, *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, New York.

[13] Mathur, K., Salkin, H. M. and Morito, S., 1983, A branch and search algorithm for a class of nonlinear knapsack problems. *Operations Research Letters,* 2, 155–160.

[14] Smith, S. and Lasdon, L., 1992, Solving large sparse nonlinear programs using GRG. *ORSA Journal on Computing,* 4, 2–15.