A Novel Local Search Algorithm for Knapsack Problem

Mostafa Memariani
Department of Electrical Engineering,
Ferdowsi University of Mashhad
Mashhad, Iran
E-mail: mostafa.memariani@stu-mail.um.ac.ir

Kambiz Shojaee Ghandeshtani
Department of Electrical Engineering,
University of Tehran
Tehran, Iran
E-mail: k.shojaee@ece.ut.ac.ir

Ahmad Madadi
Department of Computer Engineering,
University of Amir Kabir
Tehran, Iran
E-mail: a.madadi@gmail.com

Mohammad Mohsen Neshati
Department of Electrical Engineering,
Ferdowsi University of Mashhad
Mashhad, Iran
E-mail: mohsen_neshati@stu-mail.um.ac.ir

Abstract— **Knapsack problem is an integer programming that is generally called "Multidimentional Knapsack". Knapsack problem is known as a NP-hard problem. This paper is an introduction to a new idea for solving one-dimentional knapsack that with defining the "Weight Value Index", "Sorting" and "Smart local search" forms a new algorithm. This algorithm is mathematically formulated and has run on 5 sample problems of one-dimentional knapsack, that in most of them the result is close to the optimum. The results show that this method by comparison with the others recently published in this field, despite of its simplicity, has enough required functionality in order to get the result on the tested items.**

Keywords-Artificial intelligence; NP-hard; Knapsack problem; Combinational optimization.

I. INTRODUCTION

Knapsack problem is an integer programming that is in general called "Multidimentional Knapsack". Knapsack problem is known as a NP-hard problem [1]. One-dimentional knapsack problem with "constant weight group" is a special form of multidimensional knapsack. For one-dimensional knapsack in comparison with multidimensional knapsack, more precise evolutionary algorithms have been studied. Most of the researches is regarding to one-dimentional knapsack problem. For further information about knapsack problem and different precise algorithms, please refer to [2]-[4].

The reason for naming this problem to "knapsack" is because of its similarity to making decision for a mountain climber to pack his knapsack. The person should decide the optimum combination in choosing his accessories for knapsack in a way that according to the knapsack capacity, he should select items with more value (profit). This kind of problems is of combinational optimization problems family.

For several past years, precise methods such as Branch and Bound have used for solving knapsack problem [22]. In recent years, and with the development of smart optimization and evolutionary algorithms, solving more difficult problems is now possible, such that in addition to reducing the time for achieving results close to the optimum, it has increased the accuracy in solving knapsack problem. Therefore evolutionary algorithms and more definitely decoder-based evolutionary algorithms are widely used in solving knapsack problem [5], [6]. Their advantage over the more traditional direct representation of the problem is their ability to always generate and therefore carry out evolution over feasible candidate solutions, and thus focus the search on a smaller more constrained search space.

Many researchers have struggled in developing evolutionary methods for knapsack problems. From them, we can name some modern evolution methods like tabu search [7], [8], genetic algorithm [9], [10] and simulated annealing [11], [12] that in most cases show good results. In recent years, genetic algorithms show that it is the best method for solving large knapsack problems and in general 0-1 integer programming problems [13], [14].

The knapsack repeatedly is used in different processing models like processor allocation in distributed systems [15], manufacturing in-sourcing [16], asset-backed securitization [17], combinatorial auctions [18], computer systems design [19], resource-allocation [20], set packing [21], cargo loading [22], project selection [23], cutting stock [24] and capital budgeting (where project has profit and consume units of resource. The goal is to determine a subset of the projects such that the total profit is maximized and all resource constraints are satisfied) [25].

Another type of knapsack is Quadratic Knapsack Problem (QKP) [26]. In the Quadratic Knapsack Problem, an object’s value density is the sum of all the values associated with it divided by its weight. It can be used in finance [27], VLSI design [28] and location problems [29].

In the second part of this paper, we will describe the knapsack problem; in third part, the proposed algorithm will be introduced. In the forth part, algorithm simulation and comparison of results have been presented and we will conclude in the final part.

II. PROBLEM DESCRIPTION

Suppose that some items are available and each item has a weight of 'w_i' and a value of 'v_i'. In knapsack problem, weight restriction is defined in a way that the total weight of selected items should be less than knapsack capacity. The goal in this problem is finding a subset of items in a way that they have the most total value and also satisfy the knapsack capacity constraint.

For mathematically defining the mentioned concepts, we have:

$$\max\left\{\sum_{i=1}^{n} v_i x_i : \sum_{i=1}^{n} w_i x_i \leq b, x_i = 0 \text{ or } 1\ , i = 1,...,n\right\} \quad (1)$$

In formula (1), 'n', 'v_i' and 'w_i' are number of items, value of item 'i' and weight of item 'i', respectively. In the above formula, 'b' is the knapsack capacity and x_i is the algorithm input array. If the element is chosen, the x_i is 1 and otherwise is 0.

As formula (1) shows, the goal is to maximize the goal function with the given conditions. In the next section, the proposed algorithm for solving the knapsack problem will be introduced.

III. PROPOSED ALGORITHM

The presented method for solving the knapsack problem is based on statistical operations on data and combining it with artificial intelligence methods. In this method we have a set of weight and value data groups that are related in pairs and each of data shows the weight and the value of an item. The goal of this method in first stage is introducing each item with a new coefficient that would be a combination of its value and weight. With the help of this new index, the chance of selecting an item will be defined. The proposed algorithm with enough experience and iteration in changing the method of selecting based on the weight-value index and in a converged evolutionary process will provide results close to optimum. The stage of process on data for achieving a real close result to optimum will be as follows:

- According to the point that the goal of knapsack problem is to take the sum of values to the maximum and satisfy the weight constrain of knapsack, for converting 2 item dependents to one dependent, we will use the general form of (Value ^{p1} / Weight ^{p2}) that the p1 and p2 are the power of values and weights, respectively. The best value of them will be different depending on the number of items and their dispersion that with scanning the power of values and weights in the above combinational index and calculating the sum of selected item values until satisfaction of the weight constrain, we can have the best selection for the powers of mentioned formula in the beginning of the algorithm. This value would be the "weight-value index" of items.
- Next step of solving the problem is sorting items based on their weight-value index and generating initial result that would be close to optimum. In this selection, the items with higher weight-value prioritized for selection and selection of items will continue until the knapsack capacity is full.
- Because of the used method in first stage for generating weight-value index is not precise. The probability of error in the second stage would be existent as well. It is important to know that the probability of the error in selecting items based on proposed priority that is weight-value index would increase as we get closer to final stages. The probability of such errors is in the moment that the knapsack is getting filled with lower weight-value index of items. Therefore in this stage that is the main part of algorithm, we will replace the items with similar weight-value index in the final stages of selecting items. In this stage we will gradually increase the boundary of searching. In this part of algorithm we will study different results to achieve the best one.

In this intelligence searching algorithm, in addition to previous stages, we achieved the better results by the helping of some sort of modifications and corrections. For instance we can find the minimum of the selected items by dividing the knapsack capacity to the item with the highest weight. We can get to the scope of weight-value index results or in fact, items that their probability of being among the optimum answer is very high.

The main foundation of this method has been introduced above in 3 steps and the algorithm pseudocode would be as follows.

IV. ALGORITHM FORMULA

s1- Determining optimum powers for achieving optimum weight-value index by scanning from 0 to 2 with the step of 0.1 and selecting the best powers for the proportion of value to weight of items by selecting items until the knapsack is completely full. This selection is based on a way that the weight-value index priority, selected items value should be higher than the other powers that has been scanned for the proportion of value to weight.

s2- Random search around the selected power from s1 with the Radius boundary of $\alpha = 0.5$.

s3- Sorting and selecting items based on weight-value of s2 until the completion of knapsack capacity sequence length accepting items l1 and rejected items the l2

s4- Fixing items from vector value of s3 that is higher than Mean and standard deviation values of weight vector elements as selected items and random replacement of the rest selected items from s3 and rejected items as well around the last selection of s3 with the radius of 0.1 items and l1 and l2.

s5- Studying selection rule of selecting minimum items equal to dividing the maximum capacity of knapsack to the highest weight of items value and increasing the length of sequence of accepted items (l1) until satisfying the minimum selection rule.

s6- comparing the answers and the results of the current selections with the best achieved result and replacing it with the previous if that is a better answer.

s7- α=0.5 +α

s8- reduce the radius boundary of optimum power index with a coefficient of 0.9.

s9- repeating s1 to s9 while α=1 and radius boundary has reached to boundary interval.

V. RESULTS AND COMPARISON

In this part, the result of running algorithm on set of data that was given in [32]-[33] is analyzed. Five sample problems are proposed in [30]-[32] for testing the algorithm. In [30], e2, e3 and e5 samples have been solved with the different methods.

In [31], samples e1 to e4 and in [32], samples e1 and e2 have been studied. The samples e1 to e5 have 10, 20, 50,100 and 100 objects respectively. It is obvious that the samples with a greater number of objects are more complicated than samples with less number of objects and they are more difficult and more time consuming to solve.

In Table 1, the best obtained results in the relevant papers have been compared with the results of our proposed algorithm. As it is clear in Table 1, the proposed algorithm that is called Wise Experiencing Knapsack Problem (WEKP) has resulted acceptable answers.

The algorithm that has been introduced in this paper has improved the results of greedy and simple evolutionary algorithms by rate of 0.9 and 1.9 percent to the best answer. The algorithm of [31], which is a combination of greedy and genetic algorithms, has been improved the results of e1-e4 problems by rate of 0.7 and 0.2. The algorithm mentioned in [32] is an enhanced form of ACO that the results shows 0.2 percent improvement in e1 and e2 problems as well.

The results after simulating the proposed algorithm by this paper show that the results have been improved by 0.16 percent regarding to [30], 0.05 percent to [31] and 0.9 % regarding to [32].

In Table 1, we can see that for the 3rd sample problem we have achieved a result that was never achieved in other papers up to now.

In Table 2, the best, average and the worst answers for 20 times run for every sample has been given. Also, the sequence of the best obtained results for every sample has been determined as a string containing 0 and 1, where 0 means no selection and 1 stands for selecting the i^{th} object.

As it is illustrated in Table 2, even the average of the responses is very close to the optimum response and these responses acquire in an acceptable time period.

The mean time for running every problem on a pentium4 and a processor of 1.8GHz speed and 512MB of ram with the MATLAB 7.7 software is given answers.

VI. CONCLUSION

This paper is an introduction to a novel idea for solving one-dimensional knapsack problem by defining weight-value index and sorting; as a consequence, a new algorithm was proposed. This algorithm is mathematically formulated and has run on 5 samples regarding to one-dimensional knapsack that in most of them the answers are near to optimum.

The results shows that this method in comparison with the recent works published in this field, despite of its simplicity is functional enough to achieve acceptable results in tested problems.

ACKNOWLEDGMENT

This work is supported by Nano-Age Technology Group in Mashhad, Iran (www.nanoage.ir).

REFERENCES

[1] M. Garey and D. Johnson, "Computers and intractability: a guide to the theory of NPcompleteness," San Francisco: W. H. Freeman, 1979.

[2] S. Martello and P. Toth, “Knapsack Problems: Algorithms and Computer Implementations,” Wiley, New York, 1990.

[3] S. Martello, D. Pisinger, and P. Toth, “New trends in exact algorithms for the 0–1 knapsack problem,” European Journal of Operational Research, vol. 123,No. 2, pp. 325–336, 1999.

[4] D. Pisinger, “Contributed research articles: a minimal algorithm for the bounded knapsack problem”, ORSA Journal on Computing, vol. 12, No. 1, pp. 75–84, 2000.

[5] J. Gottlieb, “Permutation-Based evolutionary algorithms for multidimensional knapsack problem,” Proc. of ACM Symp. on Applied Computing, 2000.

[6] G. R. Raidl, “An improved genetic algorithm for the multiconstrained 0-1 knapsack problem,” Proc of 1998 IEEE Congress on Evolutionary Computation, pp. 207 – 211, 1998.

[7] F. Glover and G. A. Kochenberger, "Critical event tabu search for multidimensional knapsack problems," Kluwer Academic Publishers, pp. 407–427, 1996.

[8] S. Hanafi and A. Fréville, "An efficient tabu search approach for the 0-1 multidimensional knapsack problem," European Journal of Operational Research, vol. 106, pp. 659–675, 1998.

[9] Chu and J. Beasley, "A genetic algorithm for the multiconstrained knapsack problem," Journal of Heuristics, vol. 4, pp. 63–86, 1998.

[10] G. R. Raidl, “Weight-Codings in a genetic algorithm for the multiconstraint knapsack problem,” Proc of 1999 IEEE Congress on Evolutionary Computation, pp. 596-603, 1999.

[11] C. Reeves, “Modern Heuristic Techniques for Combinatorial Problems," McGraw-Hill Book Company Europe, 1995.

[12] A. Drexl. "A simulated annealing approach to the multiconstraint zero-one knapsack problem". Computing, vol. 40, pp. 1–8, 1988.

[13] Y. Sun and Z. Wang, “The Genetic Algorithm for 0–1 Programming with Linear Constraints,” Proc. of the 1st ICEC’94, Orlando,FL, edited by D. B. Fogel, pp. 559–564, 1994.

[14] R. Hinterding, “Mapping, order-independent genes and the knapsack problem”, Proc. of the 1st IEEE International Conference on Evolutionary Computation 1994, Orlando, FL, edited by D. B. Fogel, pp. 13–17, 1994.

[15] B. Gavish and H. Pirkul, “Allocation of data bases and processors in a distributed computing system Management of Distributed Data Processing,” vol. 31, pp. 215–231, 1982.

[16] N. S. Cherbaka, R. D. Meller, and K. P. Ellis, “Multidimensional knapsack problems and their application to solving manufacturing insourcing problems,” Proc. of the Annual Industrial Engineering Research Conference, Houston,TX, May 16-19, 2004.

[17] R. Mansini and M. Speranza, “A multidimensional knapsack model for the asset-backed securitization,” Journal of Operational Research Society, vol. 53, pp. 822-832, 2002.

[18] S. DeVries, and R. Vohra, “Combinatorial Auctions: A Survey,” Northernwestern University Technical Report, Evanston, IL (2000).

[19] C. Ferreira, M. Grotschel, S. Kiefl, C. Krispenz, A. Martin, and R. Weismantel., “Some integer programs arising in the design of mainframe computers,” ZORMethods Models Operations Research, vol. 38, No. 1, pp. 77-110, 1993.

[20] E. Johnson, M. Kostreva, and U. Suhl, “Solving 0 – 1 integer programming problems arising from large scale planning models,” Operations Research, vol. 33, pp. 805-819, 1985.

[21] G. Fox and G. Scudder, "A heuristic with tie breaking for certain 0 – 1 integer programming models," Naval Research Logistics, vol 32, No. 4, pp. 613-623, 1985.

[22] W. Shih, "A branch and bound method for the multiconstraint zero-one knapsack problems," Journal of the Operations Research Society, vol. 30, pp. 369-378, 1979.

[23] C. Peterson, "Computational experience with variants of the balas algorithm applied to the selection of research and development projects," Management Science, vol. 13, pp. 736-750, 1967.

[24] P. Gilmore and R. Gomery, "The theory and computation of knapsack functions," Operations Research, vol. 14, pp. 1045-1074 1966.

[25] J. Lorie and L. Savage, "Three problems in capital rationing," journal of business, vol. 28, pp. 229-239, 1955.

[26] B. A. Julstrom," Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem," GECCO'05, Washington, DC, USA, pp.607-614, June 25–29, 2005.

[27] D. L. Laughhunn, "Quadratic binary programming with applications to capital budgeting problems", Operations Research, vol. 18, pp. 454–461, 1970.

[28] C. E. Ferreira, A. Martin, C. C. de Souza, R. Weismantel, and L. A. Wolsey," Formulations and valid inequalities for node capacitated graph partitioning," Mathematical Programming, vol. 74, pp. 247–266, 1996.

[29] J. Rhys, "A selection problem of shared fixed costs and network flows," Management Science, vol. 17, pp. 200–207, 1970.

[30] K. Li, Y. Jia, W. Zhang, and Y. Xie, "A new method for solving 0-1 knapsack problem based on evolutionary algorithm with schema replaced," Proceedings of the IEEE, International Conference on Automation and Logistics Qingdao, China, pp. 2569-2571, Sep. 2008.

[31] Y. Shao, H. Xu, and W. Yin, "Solve zero-one knapsack problem by greedy GA," IEEE 2009 -International Workshop on Intelligent Systems and Applications.

[32] P. Zhao, P. Zhao, and X. Zhang, "A new ant colony optimization for the knapsack problem," Computer-Aided Industrial Design and Conceptual Design, 2006, CAIDCD '06, 7th International Conference on 17-19 Nov. 2006.

TABLE 1. COMPARATIVE RESULTS BY OTHER HEURISTIC METHODS

	[30]			**[31]**			**[32]**		**WEKP**
	Greedy algorithm	***Simple evolutionary algorithm***	***evolutionary algorithm with schema replace***	***Greedy algorithm (GA)***	***Standard genetic algorithm (SGA)***	***Greedy genetic algorithm (GGA)***	***Basic ACO***	***Improved ACO***	***Proposed method (Best result)***
e1	-	-	-	295	295	295	292	295	295
e2	1023	1042	1042	1024	1037	1042	1022	1024	1042
e3	3095	3077	3103	3077	3103	3112	-	-	**3119**
e4	-	-	-	5372	5365	5375	-	-	5372
e5	26380	25848	26559	-	-	-	-	-	26553

TABLE 2. BEST RESULTS FOR FIVE SAMPLE PROBLEM

	Example 1	**Example 2**	**Example 3**	**Example 4**	**Example 5**
No. of Objects	10	20	50	100	100
Best	295	1042	3119	5372	26553
Mean (20 runs)	295	1040.5	3106	5367.4	26553
worst	295	1037	3115.1	5360	26553
Mean time (s)	7.1	22.8	13.08	23.07	45.33
Best chromosome	111000111	101111110 10111100000	110101011110100110 110111111111000010 11011000000010	111111111011111111111001110111011 000101001110111110010110101000001 0000100001100100101000011000000000	1111111111111111111111111111111111 1111111111110111111110100010110110 11111110001110111000000000000000