A Dynamic Programming based GA for 0-1 Modified Knapsack Problem

Zaheed Ahmed
PMAS-AAUR,
Rawalpindi, Pakistan

Irfan Younas
HITEC University,
Taxila Cantt. Pakistan

ABSTRACT
The classical 0-1 knapsack problem is one of the more studied combinatorial optimization problem which belong to the NP class of algorithms. A number of its generalized forms have been addressed by various researchers using different designing techniques. In this paper, we design and analyze the Multiple Knapsack Problems (MKP) by using genetic algorithms. A modified Genetic Algorithm (mGA) is developed with the key focus on efficient encoding scheme for binary string representation and a competent dynamic programming based method for initial population generation. Furthermore transposition is applied in mGA instead of crossover for maintaining the population diversity. Performance analysis of the mGA, justifies our claims that the population incorporates adequate quality and diversity to reach a near optimal solution and transposition reduces the overall computation time.

General Terms
Dynamic Programming, Genetic Algorithms

Keywords
Multiple knapsack problem, Genetic algorithm, Dynamic programming and transposition.

1. INTRODUCTION
Genetic algorithms (GAs) have been used to tackle many NP problems since they evaluate the space of possible solutions and provide a reliable result where often conventional algorithms do not. In 1970s, John Holland and his students, sets the theoretical foundations of GAs that work on the basis of evolutionary biology. The standard GAs starts with the randomly generated set of individuals called initial population. Initial population evolves through two principles i.e. survival of the fittest and natural selection [1]. Several mechanisms are adopted to maintain genetic diversity in the space of possible solutions. Conventionally crossover and mutation are two standard and most widely applied operators in GAs to gain diversity in the solution space [2, 3], and an objective function is used to evaluate the fitness of each individual. Best fit chromosomes can be used in the mating process to produce offspring or they can be included in next generation unchanged. Reproduction and evaluation of individuals is an iterative process that continuous until acceptable solution produced.

The classical 0-1 knapsack problem involves the selection subset of available items having maximum profit so that the total weight of the chosen subset does not exceed the knapsack capacity. 0-1 knapsack is one of the most studied optimization problem. During its extensive study, a number of extensions and variants have evolved: Multiple knapsack problems (MKP), multidimensional knapsack problems (MDKP), multi choice knapsack problems (MCKP) and multiple choice multidimensional knapsack problems (MMKP) [4].

In this research we have addressed the MKP extension of the 0-1 knapsack problem. Given n items, it is required to pack k knapsacks having weight capacity of each knapsack W_j where $j \in \{1, 2 \ldots k\}$, each item i has a profit $p_{i,j}$ and weight w_i. Note that profit of each item varies according to knapsacks while weight is constant. The problem calls for selecting the disjoint subset of items for each knapsack with the objective to maximize the total profit of all the items placed in all knapsacks. Mathematically this problem can be formulated as follows.

Maximize

$$P = \sum_{j=1}^{k} \sum_{i=1}^{n} p_{i,j} . x_{i,j} \quad (1)$$

Subject to

$$\sum_{i=1}^{n} w_i . x_{i,j} \leq W_j \quad (2)$$

where

$$\sum_{j=1}^{k} x_{i,j} \leq 1 \quad (3)$$

and

$$x_{i,j} \in \{0,1\} \; \forall \; i, j \quad (4)$$

Equation (1) is used to sum the profit of all the knapsacks, equation (2) is making ensure that the total weight of items selected for a particular knapsack must be in its limit and equation (3) guarantees that one item can belong to at most one knapsack during packing.

At the abstract level, focus of this paper is on three components of a GA i.e. efficient encoding strategy, dynamic programming based method for initial population generation and use of transposition instead of crossover as the genetic operator.

The knapsack problems have many real world applications i.e. project selection, cargo loading, industrial production, capital budgeting, menu planning and processor allocation in distributed environment [1].

The rest of this paper is organized as follows. A brief outline of GAs and existing techniques for 0-1 knapsack using GAs is presented in section 2. We propose mGA for undertaken study in section 3. Performance analysis of the newly designed

approach is described in section 4. Finally we present conclusions about this research and direction for future work.

2. RELATED WORK: EA AND MKP

2.1 Overview of Evolutionary Algorithms in MKP Context

Evolutionary algorithms (EA) are search paradigms based on the concept of natural selection and evolution which can be applied to a number of optimization problems. Conventional EA framework consists on fairly simple steps like definition of encoding scheme, population generation method, objective function, selection strategy, crossover and mutation.

Two types of encoding schemes or solution representation methods are most widely used in EA for MKP i.e. bit string and ordered based representation [5, 6]. Capacity constraints are ensured by penalizing are repairing the solution in binary representation while special purpose heuristic decoders are utilized in case of numeric representation [6].

Even though, recently in some EA for MKP the pre-optimized initial population generation methods [1, 5, 7] are devised but the most common method utilized in existing approaches is random population generation [8-15]. The use of classical population generation method provides less computational burden and sufficient diversity in solution space but this method can't guarantee of the generation of even a single feasible solution. Infeasible dominated population can lead to slow inclusion of feasible solution in search space during genetic operations as well as slow convergence towards the optimal solution [7]. Moreover, a strategy to handle the infeasible solutions must be defined like reject, repair or penalized strategy [16]. Hence, execution time for overall heuristic will increase while using the conventional method.

Numerous mutation operators for knapsack problems are available in literature. Mutation per bit, swap mutation and inverted mutation are commonly used. Hoff et al. [17] suggest that inverted mutation, with 1/N mutation rate, is superior performer.

Crossover is known as the power of GAs [18]. As a major factor in maintaining the population diversity the crossover is utilized. The performance of GAs depends on the particular choice of the crossover strategy [1]. A number of crossover operators were developed and applied in MKP and other combinatorial optimization problems i.e. one-point, two-point, m-point and cyclic crossover. Since the introduction of GAs crossover and mutation operators are widely used for population diversity but several other operators are used in nature for diversity i.e. inversion, conjugation, transduction, transformation, translocation and transposition [19]. Simoes and Costa [2, 20] explores transposition as an alternative to crossover. The obtained results show that transposition outperforms crossover. Moreover, for knapsack problem transposition based GA is compared with classical GA [21] and reported results are encouraging for transposition utilization. To enhance the transposition performance a guideline for its proper parameterization is given by Simoes and Costa [22]. To the best of our knowledge transposition in history is never applied in any EA for MKP.

2.2 MKP using Evolutionary Algorithms

Since the introduction of 0-1 knapsack problem in the late fifties, a number of algorithms using different designing techniques like brute force, conventional algorithms, dynamic programming, greedy approach and approximation algorithm have been proposed. For larger knapsack problems, the efficiency of approximation algorithms is limited in both solution quality and computational cost [5]. MKP is considered as NP-hard problem hence any dynamic programming solution will produce results in exponential time [23]. In recent times, the GAs techniques have emerged as strong search procedures for large-sized knapsack problems [5, 17].

In the past various researchers employed different terminologies for 0-1 multiple knapsack problem. Here existing techniques have been reviewed in which the knapsack problem is studied involving GAs.

An improvement in multiple constrained knapsack is presented in [5] by using linear programming. Pre-optimization of initial population, competent repair procedure and local improvements operators are major contributions of this approach. Better results were obtained for large-sized knapsack problem in both optimality and efficiency as compared to existing heuristics.

An evolutionary approach was adapted by [24], where the search space was transformed to another space and search for final solution was made in transformed space. For space transformation purpose Lagrange multiplier was utilized. Fitness function was adjusted properly to achieve better results. Their heuristic was tested on benchmark data but encouraging results were not achieved in comparison with existing techniques.

Another genetic algorithm for multi-objective 0-1 knapsack problem was proposed by [8]. For comparison purpose ε-dominance relation was utilized in this algorithm. Implementation results show that proposed algorithm is better performer. In [17] it was suggested that if GA is parameterized properly then quality results can be obtained. Their tuned GA for multidimensional knapsack was tested on standard test data and reported results are relatively better.

In [7] they developed and tested a novel method for generating competent diverse and quality initial population. For the sake of performance analysis, the developed method was applied on two dimensional knapsack problems. Implementation results depict that less computational effort is required in achieving acceptable solution.

The GAs involve another type of solution for multiple knapsack problems as developed by [13]. Vectors are used to encode the solution for the easy application of genetic algorithms. Infeasible solutions are allowed to exist in population for diversity purpose and fitness function uses graded penalty to penalize the infeasible solutions. This GA was applied on well known set of test problems and encouraging results were reported. The researchers in [12] have proposed a most effective genetic algorithm which is based loosely on the theory of natural selection and genetic process. The outcome of the experiments shows dominating results as compared to the existing heuristics.

Recently in [15] evolutionary algorithms for MKP were presented. To approximate the frontiers of MKP they utilized favorable weight based evolutionary algorithm. A Fitness function assigns the value to each chromosome on the basis of strength of each individual. Binary chromosome representation is used and to repair infeasible solution which included round-and-repair method in their approach.

Implementation results show that this algorithm outperforms the best heuristics.

In literature almost every evolutionary approach for MKP uses bit strings for encoding purpose. This scheme is fairly simple but requires more space as we keep record of each item that is included or excluded. Most of the discussed GAs use randomly generated initial populations except [1, 5, 7], these require more computational effort to converge on a desired solution as compared to some pre-planned method based population. Furthermore, traditional crossover operator is utilized for generating and maintaining the population diversity while in literature a number of other operators are available for same purpose with less computational burden. Having the aforementioned points in mind, an efficient evolutionary algorithm is developed in the next section.

3. MODIFIED GENETIC ALGORITHM (MGA) FOR MKP

3.1 Solution representation

The way to represent the solutions in population of evolutionary approach for a particular problem is the key issue and fundamental step that need to be addressed. Adaptation of existing or development of new genetic operators depends on the representation scheme. Reportedly the most common and suitable encoding scheme for MKP problems is the binary representation. In mGA the n-bit binary strings are used to represent the individuals in solution set. Every bit of the string indicates the inclusion (1) or exclusion (0) of the certain item in knapsack. Thus, a typical representation of an individual for a problem having n items is depicted below.

Items	1	2	3	…	n-1	n
Solution	0	1	0	…	1	1

3.2 Dynamic Programming based initial population Method

The idea is to (i) apply items partitioning into k equal size arrays with respect to profit/weight ratio as formulated in pseudo code below:

```
Set i=1
Set j=1
While (i<n)
  While (j<k)
    If (j or j+1 ∉ Skip [])
        If (p_{i,j}>p_{i,j+1})
          Keep (current_item=i, kp = j)
        Else
          Keep (current_item=i, kp = j+1)
Insert (current_item, KP_j)
If (KP_j==Full)
  Skip [] = (Skip [] U KP_j)
```

(ii) Then to apply classical dynamic programming application on each array separately:

$V[i, w] = \max (V[i-1, w], v_i + V[i-1, w - w_i])$
For $1 =< i =< n$, $0 =< w =< W$

On the basis of above formula table 1 is computed.

Table 1. Entire solution representation for knapsack *m*

V[i, w]	w=0	w=1	w=2	…	w=W
i=0	V[0, 0]	V[0, 1]	V[0, 2]	V[0, …]	V[0, W]
i=1	V[1, w]	V[1, w]	V[1, w]	V[1, …]	V[1, W]
i=2	V[2, w]	V[2, w]	V[2, w]	V[2, …]	V[2, W]
…	V[…, w]	V[…, w]	V[…, w]	V[…, …]	V[…, W]
i=n/k	V[n/k, w]	V[n/k, w]	V[n/k, w]	V[n/k, …]	V[n/k, W]

(iii) Construction of individuals from dynamic programming results table:

```
Population_Construction (DPm)
{
  K=Wm;
  i=n;
    Next K=0;
   Next i=0;
   For (j=1 up to items)
   {
      Solution_Construction (i, K);
      i=i-1;
      K=Wm;
   }
}
```

```
Solution_Construction (i, K)
{
Counter=0;
For (i down to 1)
{
  If keep [i, K] == 1
          Output i;
          K=K-w_i;
            If (counter==0)
               Next K = W - w_i – 1;
  Next i=i;
  Counter++;
  }
Solution_Construction (Next i, Next K);
}
```

3.3 Fitness Function

Selection and reproduction need some way to evaluate the fitness of each individual so that the best chromosome can be reproduced or selected for mating process. Some researchers integrate penalty with fitness functions to handle the infeasible solutions while others use some repair strategy for the same purpose. In proposed approach repair strategy is incorporated. Hence, evaluation function is defined as below:

Evaluate (solution) = $\sum_{i=1}^{n} p_{i,j} . x_{i,j}$ where j is the knapsack.

Note that the higher the evaluation value indicates that the solution quality is better.

3.4 Selection

For reproduction purpose every GA include some strategy to select the appropriate parents. In this study, the simplest selection scheme is chosen that is known as stochastic universal sampling. For detail description of stochastic universal sampling see [1].

3.5 Transposition

The bit string representation is one of the most simple and commonly used representations with its problem-independent nature. Due to its nature user has freedom to adopt standard genetic operators. Evolutionary approaches for combinatorial optimization problem are relatively intensive to the crossover in context of efficiency. To lessen the computational load, standard transposition is adopted as an alternative to crossover in mGA for MKP.

We adopted tournament based transposition due to the observation that this type of transposition is closer to the biological transposition mechanism. The tournament based transposition uses two parents who compete with each other. The transposon will be formed in winner candidate and it will replace the same amount of genetic material in loser candidate because the chromosome length (CL) is fixed (equal to the number of items) in our representation.

The detail functioning mechanism of tournament based transposition consists on following steps:

1. Define the flanking sequence length (FSL) and select a bit T randomly in winner parent.
2. Compute the FSL bits immediately before the T which is known as first flanking sequence.
3. Search second flanking sequence that will be identical or inverse sequence of first flanking sequence from bit T in a cyclic fashion.
4. Build transposon which is bits from T to the end of second flanking sequence.

Look for the identical or inverse flanking sequence that is replacement point in loser parent. If found, replace the genetic material after that with transposon in cyclic fashion otherwise randomly define replacement point and replace.

Example

10011101111010110010011100: In winner parent underlined is flanking sequences, bold is T and italic is transposon.

11010001110011110011010011: Loser, underlined flanking sequences shows possible insertion points.

Let suppose the replacement point is first underlined flanking sequence in loser parent then the siblings will be as:

10011101111010110010011100: sibling 1

11010011011110101100101001 1: sibling 2

If there is a visible gap in selection probability of the two siblings then best fit child participate in next generation while other one will be knocked out.

3.6 Optimized Transposition Parameters

Usually GA performance basis on the computational cost of crossover and as here we are using transposition as an alternative to crossover so the performance of our evolutionary approach is depending on transposition's performance. The major performance contributor parameter in transposition is FSL which is depending on the chromosome length. Reportedly [22] in tournament based transposition the chromosomes can be divided into two classes on the basis of their size i.e. small: 19-38 genes and large 39-max genes. According to this division, if chromosome size is small then FSL is (18% * chromosome size) ± 1 otherwise FSL will be (5% * chromosome size) ± 1. Another parameter that has impact on transposition performance is population size but it has secondary importance as compared to FSL. In larger population size there is inherently higher the computational burden while in smaller size population one can use heuristic to adjust the appropriate FSL.

3.7 Greedy Invert Mutation

A new mutation operator is utilized which is designed on the basis of a reportedly superior performer invert mutation operator. Inverted mutation selects a gene randomly in individual and assigns an opposite value of current value to that gene. The slight modification that has been done is the injection of greed instead of randomly mutating the gene. After transposition offspring pass through the computation of its feasibility which is the underflow or overflow value of the knapsack in [-W … -1, 0, 1 … $\sum_{i=1}^{n} w_{i,m}$]. Feasibility is computed as follows: f (s) = Cm - $\sum_{i=1}^{n} w_{i,m}$

Mutation only takes place if the knapsack for that particular solution is under flowing. Best fit strategy is applied for the gene selection instead of random selection where underflow value is compared with the weight of excluded items. Mutation probability is adjusted 1/number of items. Table 2 shows that how greedy invert mutation operates.

Table 2. Greedy invert mutation mechanism

	Feasibility	Solution									
Original	**-30**	1	0	0	1	0	**0**	0	1	1	0
	$w_{i,j}$	31	12	23	5	31	28	22	19	7	26
Mutated	**-2**	1	0	0	1	0	**1**	0	1	1	0

3.8 Greedy-repair strategy

Remove items, every time the minimum with respect to profit/weight ratio with believe that in subsequent generation

it will get an individual again and if the generation is last then remove the item having minimum profit.

3.9 Pseudo-Code of Proposed Evolutionary Approach

The detail description of our evolutionary approach is given in pseudo code format here:

```
DPEA for MKP
{
Input C [m x n], W [1 x n], P [m x n];   //required test problem data
Set Gen=0;                               //count the number of generations for exit criteria
Max-Gen=1000;
T-r=0.9;                                 // Transposition rate
FSL = 3;                                 // Flanked Sequence Length
M-r=0.8;                                 // optimized mutation rate
Set Pop-Size = (n (n+1))/2;              // each table row can have as many individuals as n
Item-Partition ();                       //see section xxx
Population-Construction (DP_m);          //see section xxx

While (Not Exit Criteria)                // for exit criteria see section xxx
{
Evaluate Fitness ();                     //see section xxx
Selection ();                            //selection procedure is defined in section xxx
Transposition ();                        //working of transposition is given in section xxx
If (solution = feasible)
{
Mutation ();                             // for details see section xxx
        }
Else
        {
        Repair ();                       //for details of repair strategy see section xxx
        Mutation ();
}
Gen++
}
}
```

4. PERFORMANCE ANALYSIS OF MGA

4.1 Development of Transposition

Djannaty and Doostdar [1] suggested that crossover is major contributor in computational cost of genetic approaches for any combinatorial optimization problem. Simeos and Costa [2, 21-22] proves that transposition is an efficient alternative to crossover. In existing evolutionary approaches for MKP [1, 5, 7, 8-15, 17] the traditional crossover operators are utilized. In proposed approach first time transposition is introduced for MKP. Hence, mGA reduces the computational burden as compared to existing heuristics.

4.2 Pre-Optimized Initial Population

Do Classically, randomly generated population is used in existing approaches [8-15] except some recent approaches [1, 5, 7] where methods are devised for effective initial population generation. In this research, dynamic programming based method for pre-optimized initial population is developed. As afore mentioned that the DPIP generates the population having sufficient diversity and 100% feasibility which requires less computational effort to converge on optimal solution.

Lemma 1: Best case DPIP versus random

In best case it might be possible that each V [n/k, W] (see table 1) remains as the final solution. So in very efficient manner DPIP can handle the MKP problem. Best case for DPIP in comparison to random method for population generation is much more efficient. The probability of having best case is high in DPIP as every time it generates feasible solutions from partitioned items with respect to profit/ weight ratio for each knapsack.

Lemma 2: Generally DPIP versus random

Even though items partitioning method can decrease the fitness of individuals if suitable division does not exist there. But at least all the individuals in DPIP fulfill the feasibility criteria while in worst case of random population can lead to include all the infeasible solutions in population. Naturally the infeasible individuals require much more computational cost to converge to a final acceptable solution as compared to feasible one.

4.3 Greedy Invert Mutation

A novel method for mutation is developed in this approach which enhances the fitness value of each individual in a greedy fashion. In existing genetic algorithms for MKP, conventional mutation operators were utilized which can increase of decrease the fitness value but greedy mutation guarantees the increment in fitness if it occurs.

5. CONCLUSION AND FUTURE WORK

In this article we have developed an innovative evolutionary approach for MKP in which the total profit is maximized without violating the capacity constraint for each knapsack. The algorithm uses dynamic programming based method for pre-optimized initial population having sufficient diversity and quality in it. In addition, to reducing the execution time the traditional crossover operator is replaced by transposition. Both the DPIP and transposition achieve significant performance in computational burden and solution optimality. Comparative analysis is presented to prove the claims and it is shown that proposed algorithm is better performer than the existing approaches.

Although current research is developed and applied for the particular variant of the knapsack problem, but it can incorporate other extensions of the knapsack with slight modifications. This aspect has been left for future works.

6. REFERENCES

[1] Farhad Djannaty and Saber Doostdar, A Hybrid Genetic Algorithm for the Multidimensional Knapsack Problem, Int. J. Contemp. Math. Sciences, Vol. 3, 2008, no. 9, pp. 443 – 456.

[2] Anabela Simões Ernesto Costa, "Using Genetic Algorithms with Asexual Transposition," Proceedings of the Genetic and Evolutionary Computation Conference (GECCO' 2000), Las Vegas, USA, 2000, pp. 323-330.

[3] A. Simoes and E. Costa, "On Biologically Inspired Genetic Operators: Transformation in the Standard Genetic Algorithm," Proceedings the Genetic and Evolutionary Computation Conference, GECCO-2001, San Francisco, USA, 2001, pp.584-591.

[4] Raymond R. Hill, Chaitr S. Hiremath, "Generation Methods for Multidimensional Knapsack Problems and their Implications", Journal of Systemics, Cybernetics and Informatics, 2007, pp. 59-64.

[5] Gunther R. Raidl, "An Improved Genetic Algorithm for the Multi-constrained 0–1 Knapsack Problem," Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, 1998, pp.207 - 211.

[6] Christine L. Mumford, "Comparing Representations and Recombination Operators for the Multi-Objective 0/1 Knapsack Problem," The 2003 Congress on Evolutionary Computation, IEEE, 2003, pp.854- 861.

[7] Hill, R.R. and Hiremath, C. 'Improving genetic algorithm convergence using problem structure and domain knowledge in multidimensional knapsack problems', Int. J. Operational Research, Vol. 1, Nos. 1/2, 2005, pp.145–159.

[8] Crina Groşan, Mihai Oltean, D. Dumitrescu, "A new evolutionary algorithm for the multi-objective 0/1 knapsack problem," Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics – ICTAMI, 2003, pp. 233-237.

[9] Crina Grosan, "Improving the performance of evolutionary algorithms for the multi-objective 0/1 knapsack problem using ε-dominance," 2003.

[10] Rajeev Kumar, Nilanjan Banerjee, Analysis of a Multiobjective Evolutionary Algorithm on the 0–1 knapsack problem, Theoretical Computer Science, Elsevier, 2006, pp. 104 – 120.

[11] Yehoon Kim, Jong-Hwan Kim, and Kuk-Hyun Han, Quantum-inspired Multiobjective Evolutionary Algorithm for Multiobjective 0/1 Knapsack Problems, IEEE Congress on Evolutionary Computation, 2006, pp. 2601-2606.

[12] Sachi Nandan Mohanty, Rabinarayan Satapathy, An Evolutionary Multi-Objective Genetic Algorithm To Solve 0 / 1 Knapsack problem, IEEE, 2009.

[13] Khuri, S., Back, T. and Heitkotter, J., "The zero/one multiple knapsack problem and genetic algorithms," Proceedings of the 1994 ACM Symposium on Applied Computing, SAC '92, ACM Press, 1994, pp.188–193.

[14] GEN M, CHENG R, IDA K, JIN Y, "Multiple Objective 0-1 Knapsack Problem Using Genetic Algorithms," Research Reports Ashikaga Institute of Technology, 1998, pp.295-301.

[15] Koksalan, M. and B. Soylu, "An Evolutionary Algorithm for the Multi-objective Multiple Knapsack Problem", 20th International Conference on Multiple Criteria Decision Making, China, July 2009, pp. 1-8.

[16] Michalewicz, Z. (1996). Genetic Algorithm + Data Structures = Evolution Programming, Springer Verlag, 3rd edition.

[17] Arild Hoff, Arne Løkketangen, Ingvar Mittet, "Genetic Algorithms for 0/1 Multidimensional Knapsack Problems," Proceedings of Norsk informatikk konferanse, NIK'96, 1996, pp. 291-301.

[18] Z. Ezziane, Solving the 0/1 knapsack problem using an adaptive genetic algorithm, Cambridge Journal, 2002, pp 23-30.

[19] J. L. Gould and W. T. Keeton (1996). Biological Science. W. W. Norton & Company.

[20] A. Simões and E. Costa. "Transposition: A Biologically Inspired Mechanism to Use with Genetic Algorithms," In the Proceedings of the Fourth International Conference on Neural Networks and Genetic Algorithms (ICANNGA'99), Springer-Verlag, 1999, pp. 612-619.

[21] Anabela Simoes, Ernesto Costa, An Evolutionary Approach to the Zero/One Knapsack Problem: Testing Ideas from Biology, International Conference on Neural Networks and Genetic Algorithms(ICANNGA'01), Prague, Czech Republic, Springer,2001, pp. 236-239.

[22] A. Simões and E. Costa, Enhancing Transposition Performance. Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99), Piscataway, NJ: IEEE Press, 1999, pp.1434-1441

[23] Chanin Srisuwannapa and Peerayuth Charnsethikul,"An Exact Algorithm for the Unbounded Knapsack Problem with Minimizing Maximum Processing Time," Journal of Computer Science, 2007, 3 (3): 138-143.

[24] Y. Yoon, Y.-H. Kim, B.-R. Moon, "An Evolutionary Lagrangian Method for the 0/1 Multiple Knapsack Problem" ACM, GECCO'05, Washington, DC, USA, 2005