A hybrid genetic algorithm for solving 0/1 Knapsack Problem

Laabadi Soukaina
ENS- Hassan II University
Laboratory of Mathematics and Applications
Casablanca, Morocco
soukainalaabadi@gmail.com

Naimi Mohamed
EST- Hassan I University
Laboratory of Analysis, Modeling Systems and Decision Support
Berrechid, Morocco
mohamed.naimi@uhp.ac.ma

El Amri Hassan
ENS- Hassan II University
Laboratory of Mathematics and Applications
Casablanca, Morocco
hassanelamri@gmail.com

Achchab Boujemâa
EST- Hassan I University
Laboratory of Analysis, Modeling Systems and Decision Support
Berrechid, Morocco
achchab@estb.ac.ma

ABSTRACT

The Knapsack problem is an integer programming problem. This well-known problem in the field of operations research, is considered as a NP-hard problem. In this paper, we present a new hybrid genetic algorithm (HGA) to address one-dimensional knapsack problems, by using a sexual selection operator and a specific-knowledge crossover operator. Empirical results show the feasibility and effectiveness of our new algorithm HGA, by comparing with some selection and crossover mechanisms commonly used for solving the knapsack problems.

CCS CONCEPTS

• **Operations research** → **Combinatorial Optimization problem** • **Metaheuristics** → Genetic algorithms.

KEYWORDS

Hybrid genetic algorithm, sexual selection, crossover operator, knapsack problem.

ACM Reference format:

S. Laabadi, M. Naimi, H. El Amri, B. Achchab. 2018. In *Proceedings of ACM LOPAL conference, Rabat, Morocco, May 2018 (LOPAL'18)*, 6 pages.
https://doi.org/10.1145/3230905.3230907

1 INTRODUCTION

The optimization algorithm plays an important role in solving the complex problems, and many complex problems can be modeled as a combinatorial optimization problem. The knapsack problem is a kind of typical combinatorial optimization problem. In this paper, we are interested to binary version of KP, often called 0/1 Knapsack Problem (0/1-KP). The current problem can be described as follows: given a knapsack with limited capacity and a set of items, each item having a profit and weight. The goal is to select a sub-set of items to be included in the knapsack, so that the profit is as large as possible, while the total weight must not exceed the capacity of knapsack. There is a variety of real-world applications for 0/1-KP. It has appeared first in capital budgeting [1]. Since then, many practical problems are stated as 0/1-KP, such as cargo loading [2], resource allocation [3], combinatorial auctions [4], Available-to-Promise problem [5], etc. Recently, the 0/1-KP is used to model the real estate property maintenance problem [6], which consist to achieve multiple operations (e.g. facade renovating, change of heating system or boiler replacement), using limited budget in a limited period. To adapt this problem to those fields, main variations occur by changing the number of parameters, such as the number of items, the number of objectives, the number of constraints and / or the number of knapsacks.

Exact and approximate algorithms are two main approaches to solving 0/1-KP. For several past years, a variety of exact methods including dynamic programming [7], branch and bound [8], and enumeration techniques [9], have been used for solving small-to-moderate knapsack instances. However, when the instance increases, the problem cannot be solved in a reasonable time by using exact algorithms. So, the approximate methods are employed, that leads to a good compromise between quality of solution and time consumption. Therefore, a lot of papers have addressed these competitive alternatives, see [10] that represent a comprehensive overview of 0/1-KP solving methods. In fact, we distinguish two classes of approximate algorithms for solving KP. The first class is heuristics that includes greedy heuristics [11], and heuristics based on relaxation [12]. The second one is advanced heuristics, often called metaheuristics, that includes the single-

LOPAL '18, May 2–5, 2018, Rabat, Morocco

ACM ISBN 978-1-4503-5304-5/18/05…$15.00
https://doi.org/10.1145/3230905.3230907

based algorithms (e.g. tabu search [13], variable neighborhood search [14] …), as well as population-based algorithms (e.g. genetic algorithms [15], ant colony algorithms [16], swarm particular algorithms [17],…). Among these algorithms, the genetic algorithm is suitable for solving 0/1-KP. Furthermore, it is easy to understand and easy to implement, and it achieves good results.

This paper is organized as follows. The next section formulates the 0/1-KP mathematically and presents some basic descriptions of classical genetic algorithm (GA). In section 3, a new hybrid GA (HGA) is introduced in detail. Empirical results of our HGA are presented in section 4. Finally, a conclusion and some perspectives are given in section 5.

2 CLASSICAL GENETIC ALGORITHM FOR SOLVING 0/1 KNAPSACK PROBLEM

2.1 The problem formulation

Given N items, each item i having an integer weight $w_i > 0$ and an integer profit $p_i > 0$. The 0/1-KP is to pack as many items as possible to a knapsack with capacity $C > 0$ such that the total profit of items in the knapsack is maximized.

The 0/1-KP is a general statement of any 0/1 integer problem with non-negative coefficients. It can be formulated mathematically as follows:

$$\begin{cases} \text{Max} \sum_{i=1}^{N} p_i x_i & (1) \\ s.t \sum_{i=1}^{N} w_i x_i \leq C ; & (2) \\ x_i \in \{0,1\} ; \quad \forall i \in \{1, \dots, N\} & (3) \end{cases}$$

Where (3) means that x_i is a binary decision variable, it equals to 1 if the item i is selected, and 0 otherwise.

2.2 Mechanisms of genetic algorithm for 0/1 Knapsack Problem

The genetic algorithm was introduced by John Holland and his colleagues in 1975 at university of Michigan [18]. It belongs to the larger class of evolutionary algorithms (EA). Considered as a bio-inspired algorithm, it based on evolution process of biological organisms in nature. Sequential mechanisms are used in GA, to get the next generation and enhance searching ability in a large space of multiple solutions:

- ***Initialization of population***: The 0/1-KP solution is presented by a chromosome with N genes, where N presents the number of items to be placed in the knapsack. Each gene is equal to 0 or 1: the ith gene takes the value 1 if the ith item is selected to be included in the knapsack, and takes the value 0 otherwise. Each chromosome represents an individual of population and it's evaluated with a fitness value. Generally, the fitness value is based on the objective function of the problem:

$$f(x_1, x_2, \dots, x_N) = \text{Max} \sum_{i=1}^{N} p_i x_i \ (x_i \in \{0,1\}) \quad (4)$$

- ***Parent selection***: To achieve a good loading pattern that maximizes the profit of the knapsack, it is likely necessary to select good parents. There are many methods in the literature for parent selection: tournament selection, roulette wheel selection, rank selection, truncation selection, etc. The tournament selection is often used in practice, especially the binary variant.

- ***Crossover:*** In this process, we mate and recombine selected parents, to create offsprings for the next generation. The crossover is a very crucial stage of GA since recombining good parents will improve the quality of generated individuals. There are many methods for crossover: uniform crossover, one-point crossover, multi-point crossover, etc. The likelihood of crossover generally should be between 0,50 and 0,95.

- ***Mutation:*** Sometimes, the fitness value of the obtained offsprings may seem to stagnate around the optimal point, and that leads to premature convergence phenomenon. This problem can be solved by applying the mutation operator, which helps to maintain the diversity of the population. Mutation is the operation of swapping the gene value from 1 to 0 or vice versa, by selecting the position of genes in the chromosome randomly. The likelihood of mutation is set to be extremally low (between 0.001 and 0.01).

- ***Repair operator:*** This operator is used in GAs to correct invalid offsprings which may be generated following crossover or mutation and sometimes in population initialization stage. The aim of repair operator is to obtain a feasible solution that respect the capacity constraint from infeasible one, and to maintain the quality of solutions.

- ***Termination criteria:*** It's important to determine when the GA run will end. Usually GA has to terminate either after a fixed number of generations, or when there is no improvement of solution after a certain number of iterations, or when a pre-defined value of objective function is reached.

- ***Replacement strategy:*** There is typically a close relationship between replacement strategy and crossover operation, we can distinguish three strategies:

  - Elitism strategy: that consists to keep the fittest individuals of the population in the next generation, so that the population size remains constant.
  - Generational strategy: that replaces all parents with all generated offsprings.
  - λ / μ replacement strategy: Let μ the population size, and λ the number of selected offsprings, where $\lambda \leq \mu$. In this strategy, we select λ offsprings for being

parents in the next generation. When $\lambda = \mu$ we will have generational strategy.

3 THE HYBRID GENETIC ALGORITHM

A lot of papers has addressed GA to solve 0/1-KP and its extensions. It's considered as a suitable approach to solve them. [19–20] have used the classical GA, other authors have proposed hybrid GA, for instance, [21] incorporate greedy heuristics in GA repair operator, [22] combine GA with local search heuristic. When others have introduced new operators as well as new mechanisms for individual generation, selection of couples and replacement strategies. For instance, [23] have presented a new selection based on gender of chromosomes, and [24] have proposed a new crossover operator based on two-stage recombination scheme. Motivated by the good outcome results of the sexual selection proposed by Varnamkhasti and Lee [23], as well as the two-stage recombination operator introduced by Aghezzaf and Naimi [24], we propose a new hybrid method that use both these selection and crossover operators. Then, we adapt this HGA for solving 0/1-KP.

3.1 The sexual selection operator of Varnamkhasti and Lee

The method proposed by Varnamkhasti and Lee in [23] to solve Multidimensional Knapsack Problem, is based on sexual selection: The chromosomes are divided into groups of males and females (denoted respectively *male_chro* and *female_chro*). The crossover operator takes place only between chromosomes with opposite sex. So, for choosing the couples, the authors use the tournament selection operator for selection *female_chro*, and propose a new selection procedure for choosing the *male_chro*. Their selection is based on the Hamming distance from *female_chro*, fitness value or active genes of *male_chro*. In fact, a certain number of male chromosomes are selected randomly from the male group. Then the *male_chro* is selected according to:

a) The maximum Hamming distance between *the male_chro* and the *female_chro*, or
b) the highest fitness value of *male_chro* (if more than one *male_chro* is having the maximum Hamming distance existed), or
c) the highest active genes of the *male_chro* (if more than one *male_chro* having the highest fitness value were found), or
d) random selection.

Where Hamming distance between two chromosomes of population is the number of positions at which the corresponding genes are different. And the active genes correspond to the number of genes with value 1 in one chromosome.

3.2 The two-stage recombination operator of Aghezzaf and Naimi

The proposed crossover operator by Aghezzaf and Naimi in [24] is divided in two stages and generates only one offspring from two parents. In the first stage, generated offspring inherits the similar genes that have the same position and the same value in both parents. In the second stage, the offspring chromosome is completed by non-similar genes having the same place but different values. The authors have proposed a specific version of their recombination operator for the 0/1 Multi-objective Knapsack Problem (MOKP). In this paper, we try to adapt this version to 0/1-KP.

As a reminder, selecting the similar genes in the first stage means choosing the common items of both parents, whereas in the second stage choosing non-similar genes means that the offspring receives the parent non-common items by respecting the constraint capacity. However, if we take into consideration the two-stage of crossover operator, all parent common and non-common items will be inserted in generated offspring solution. Consequently, the constraint of the problem can be violated. So, to respect the problem constraint by considering the similarity-preserving principle, the authors select all non-common items and allocate to each one a fitness value. The role of item's fitness value is to classify the parent non-common items, by given priority to those that can be maximized the objective function. So, the authors sort the parent non-common items in decreasing order of their fitness value. Then, they insert them iteratively in the chromosome. The insertion will be stopped if we cannot insert any candidate item without overloading the knapsack or if the list of parent non-common items becomes empty.

3.3 The hybrid genetic algorithm for solving knapsack problem

The proposed HGA for 0/1-KP maintains a population of binary solutions presented as a chromosome c_k with N genes, where N is the quantity of items which will be placed into knapsack. The binary encoding format of population is as follows:

{{101010110}, {010001110}, …, {001010100}}

In the first chromosome c_1 , 1st, 3rd,5th,7th and 8th items will be included in the knapsack. Since we use the sexual selection, we distinguish between female chromosome and male chromosome by their layout in the population.

$$c_k \; is \; a \rightarrow \begin{cases} female \; chromosome \, , & if \; k \; is \; even \\ male \; chromosome, & if \; k \; is \; odd \end{cases}$$

Each step of the proposed HGA is discussed below:

Step (1): For initializing the population, we use random

generation to achieve a diversity in population. The population size is kept constant through the generations. A greedy repair operator introduced by [25] is applied to invalid chromosomes for restoring their feasibility. Indeed, the repair operator consists of sorting items in increasing order of their profit / weight ratio. Then, it removes items from the knapsack iteratively until the capacity constraint is fulfilled.

Step (2): To select female chromosomes, we use binary tournament method while male chromosomes are selected randomly from the male group. Note that the number of selected male chromosomes depends on the selected female chromosomes.

Step (3): To build a pair of parent with opposite sex, we apply the above-mentioned selection operator in section 3.1. We fix a list of female chromosomes and we choose the suitable mate. We repeat the procedure until the list of female chromosomes becomes empty. Note that each chosen male chromosome is removed from the list of male chromosomes.

Step (4): For generating a new progeniture, we recombine a male chromosome with female chromosome by using the two-stage recombination operator discussed in section 3.2. At the first stage, the parent common items are copied in a new chromosome. In the second stage, we use a following fitness formula for evaluating the quality of each parent non-common item. Let j be the index of candidate item j:

$$f(j) = \frac{p_j}{w_j} \quad (5)$$

Then, we insert the sorted non-common items in decreasing order of their fitness value until the list of parent non-common items becomes empty or if all residual items violate the constraint capacity.

Step (5): To prevent a premature convergence and making a diversity in solution space, we use the Flip Bit Mutation operator, that chooses a gene randomly and inverts its value (i.e. if the value is 0, it is changed to 1 and vice versa). Then, we use repair operator to transform infeasible solution to feasible one.

Step (6): Knowing that the size of population is constant, we complete the new population with the best parents. So, we sort all parents in decreasing order of their fitness value, and we insert them until the size of the new population becomes equal to the size of parent population.

Step (7): We consider a fixed number of generations as a termination condition.

4 COMPUTATIONAL EXPERIMENT

4.1 Parameters and environment of simulations

In order to carry on the comparative analysis, variants of GA have been realized in the simulation. In three GAs, we use the same crossover operator and we vary selection operators, and so on. We choose as selection and crossover mechanisms, those available in the GA's literature. The abbreviations of this comparative methods are presented in Table 1.

Table 1: Abbreviation of selection and crossover mechanisms

Selection mechanism	Abbreviation	Crossover mechanism	Abbreviation
Binary tournament selection	BTS	One-point crossover	1P-C
Roulette wheel selection	RWS	Uniform crossover	UC
Truncation Selection	TS		

The empirical results reported in this paper were performed by using the proposed HGA as well as the comparative GAs with 100 individual number in one population, crossover probability of 0.95, and mutation probability of $\frac{1}{L}$, where L is the length of chromosome. Fourteen problem instances are generated randomly, varying between 10 items (small-scale instances) and 500 items (large-scale instances). For each instance, the new population is generated after 50 executions with 100 generations per run. All algorithms were coded in JAVA and run on Intel® Core™ i5 with 2.5 GHz and 4.0 Go of RAM.

4.2 Empirical results

Table 2 is the simulation comparison of computational results about fourteen problem instances. Column 3 displays the average fitness obtained by HGA and the computational time that needs HGA for reaching the final solution (expressed in seconds and presented in the next line). In the same manner, the columns 4, 5, and 6 tally the solutions and the runtime required given by GAs using BTS, RWS, and TS with 1-point crossover operator. The columns 7, 8, and 9 contain the average solutions and runtime required provided by GAs that use BTS, RWS, and TS with uniform crossover operator. In the Table 2, we can remark that the performance of HGA seems more efficient than the classical mechanisms of GA. The best results are shown in boldface. In fact, the HGA produces satisfactory solutions in all test instances within a shorter average runtime. It yields good solutions in terms of quality and time consuming, when comparing with the first three combinations: RWS, TS, and BTS with 1-point crossover operator. Nevertheless, it provides closely similar solutions to those obtained by

combining RWS, TS, and BTS with uniform crossover operator, but it's robust in terms of time consuming. However, it turns out to be more efficient and effective when we have more than 100 items. As a result, the HGA can be considered as an alternative to solve 0/1 KP.

Table 2: Comparison of average fitness and average time obtained by HGA and the comparative methods

Number of variables	Solution/ time	HGA	BTS + 1P-C	RWS+ 1P-C	TS + 1P-C	BTS + UC	RWS + UC	TS + UC
10	Solution	**68,84**	**68,8**	**68,82**	**68,76**	**68,92**	**68,96**	**68,88**
	Time	**0,018**	0,042	0,045	0,045	0,054	0,057	0,051
20	Solution	**166,08**	165,18	165,5	164,94	**166,12**	**166,04**	165,54
	Time	**0,011**	0,036	0,042	0,034	0,047	0,055	0,048
30	Solution	**290**	287,96	288,76	287,36	**290**	289,94	**290**
	Time	**0,014**	0,037	0,038	0,035	0,054	0,059	0,053
40	Solution	**396,5**	392,86	393,58	392,58	**396,34**	**396,32**	**396,02**
	Time	**0,018**	0,038	0,044	0,035	0,074	0,077	0,067
50	Solution	**506,32**	500,54	501,24	500,34	**506**	**506,12**	505,94
	Time	**0,016**	0,043	0,05	0,042	0,088	0,089	0,084
60	Solution	**599,22**	591,7	592,34	589,62	**599,54**	**599,46**	**599,1**
	Time	**0,018**	0,047	0,054	0,044	0,096	0,102	0,094
70	Solution	**630,54**	618,4	618,66	616,42	629,74	629,78	629,68
	Time	**0,02**	0,053	0,059	0,049	0,111	0,117	0,109
80	Solution	**897,98**	882,56	883,68	879,28	896,96	896,64	896,56
	Time	**0,023**	0,057	0,065	0,054	0,131	0,136	0,125
90	Solution	**957,52**	936,12	940,76	935,52	**957,64**	**957,42**	**957,16**
	Time	**0,026**	0,062	0,07	0,059	0,139	0,147	0,135
100	Solution	**1093**	1069,54	1070,74	1066,2	1092,52	1092,54	1092,46
	Time	**0,029**	0,068	0,076	0,065	0,152	0,161	0,152
200	Solution	**2261,78**	2148,08	2146,82	2131,94	2260,32	2257,78	2259,62
	Time	**0,071**	0,12	0,138	0,115	0,297	0,315	0,307
300	Solution	**3355,96**	3118,62	3119,76	3102,36	3351,66	3347,1	3348
	Time	**0,129**	0,188	0,215	0,182	0,456	0,494	0,477
400	Solution	**4268**	3916,5	3900,16	3883,46	4257,98	4254,32	4256,26
	Time	**0,212**	0,27	0,318	0,263	0,639	0,697	0,678
500	Solution	**5452,76**	4939,68	4942,44	4913,76	5433,98	5423,16	5434,28
	Time	**0,314**	0,353	0,43	0,349	0,841	0,905	0,893

5 CONCLUSION AND PERSPECTIVES

In this paper, we have proposed a hybrid genetic algorithm that uses selection operator for choosing parents by considering their gender. Indeed, the female chromosomes are selected by tournament selection whereas the male chromosomes are selected based on the Hamming distance from the already selected female chromosomes, fitness value or active genes. For

recombining the generated female and male chromosomes we apply a crossover mechanism that inherit the similar character from parents and uses the problem-specific knowledge. The proposed algorithm is tested on 0/1 knapsack problem. Thus, the computational experiments demonstrate that the adopted methodology can be considered as an alternative to improve the performance of GA.

The problem studied in this work is rich by its specificity. However, it can be easily extended to other applications. As a perspective, an extension of this work may concern the adaptation of the HGA for multidimensional knapsack problem or other variants of 0/1 knapsack problem. Another enhancement may be related to the steps of selecting male chromosomes to speed up the search process.

REFERENCES

[1] J. H. Lorie and L. J. Savage. 1955. Three problems in rationing capital. In *the Journal of Business*, Vol. 28, 229–239.

[2] S. Eilon and N. Christofides. 1971. The loading problem. In *Management Science,* Vol. 17, 259–268.

[3] F. Chen, T. La Porta, and M. B. Srivastava. 2012. Resource Allocation with Stochastic Demands. In *proceeding of 8th IEEE International Conference on Distributed Computing in Sensor Systems*.

[4] J. Pfeiffer and F. Rothlauf. 2007. Analysis of Greedy Heuristics and Weight-Coded EAs for Multidimensional Knapsack Problems and Multi-Unit Combinatorial Auctions. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation.* DOI: http://dx.doi.org/10.1145/1276958.1277258

[5] H. C. Lau and M. K. Lim. 2004. Multi-Period Multi-Dimensional Knapsack Problem and Its Application to Available-to-Promise. In *International Symposium on Scheduling (ISS)*, 94–99.

[6] F. Taillandier, C. Fernandez, and A. Ndiaye. 2017. Real Estate Property Maintenance Optimization Based on Multiobjective Multidimensional Knapsack Problem. In *Computer-Aided Civil and Infrastructure Engineering,* Vol. 32, 227–251.

[7] S. Martello, D. Pisinger, and P. Toth. 1999. Dynamic programming and strong bounds for the 0-1 knapsack problem. In *Management Science*, Vol. 45, 414–424.

[8] P. J. Kolesar. 1967. A branch and bound algorithm for the knapsack problem. In *Management science*, Vol. 13, 723–735.

[9] A. V. Cabot. 1970. An enumeration algorithm for knapsack problems. In *Operations Research*, Vol. 18, 306–311.

[10] H. M. Salkin and C. A. De Kluyver. 1975. The knapsack problem: a survey. In *Naval Research Logistics (NRL)*, Vol. 22, 127–144.

[11] M. J. Magazine, G. L. Nemhauser, and Jr. L. E. Trotter. 1975. When the greedy solution solves a class of knapsack problems. In *Operations Research*, Vol. 23, 207–217.

[12] M. Silvano. 1990. *Knapsack problems: algorithms and computer implementations.* Wiley Interscience series in discrete mathematics and optimization.

[13] S. Hanafi and A. Freville. 1998. An efficient tabu search approach for the 0–1 multidimensional knapsack problem. In *European Journal of Operational Research*, Vol. 106, 659–675.

[14] M. F. Tasgetiren, Q. K. Pan, D. Kizilay, and G. Suer. 2015, May. A differential evolution algorithm with variable neighborhood search for multidimensional knapsack problem. In *IEEE Congress on Evolutionary Computation (CEC)*, 2016 IEEE, 2797–2804.

[15] Z. Michalewicz and J. Arabas. 1994. Genetic algorithms for the 0/1 knapsack problem. In *International Symposium on Methodologies for Intelligent Systems*, 134–143. Springer Berlin Heidelberg.

[16] H. Shi. 2006. Solution to 0/1 knapsack problem based on improved ant colony algorithm. In *International Conference on Information Acquisition,* 2006 IEEE, 1062–1066.

[17] J. Langeveld and A. P. Engelbrecht. 2012. Set-based particle swarm optimization applied to the multidimensional knapsack problem. In *Swarm Intelligence*, Vol. 6, 297–342.

[18] D.E. Goldberg. 1989. Genetic Algorithm in Search, Optimization and Machine Learning. Addison Wesley, New York.

[19] G.R. Raidl. 1998. An improved genetic algorithm for the multiconstrained 0–1 knapsack problem. In *Proceedings of the 5th IEEE International Conference on Evolutionary Computation*, 207–211.

[20] A. Bader-El-Den M. and D. Boughaci. 2017. Guided genetic algorithm for the multidimensional knapsack problem. In *Memetic Computing*, 1–14.

[21] G. Lai, D. Yuan and S. A. Yang. 2014. new hybrid combinatorial genetic algorithm for multidimensional knapsack problems, In *Journal of Supercomputing*, Vol. 70, 930–945.

[22] A. Jaszkiewicz. 2002. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem-a comparative experiment. In *IEEE Transactions on Evolutionary Computation*, Vol. 6, 402–412.

[23] M. J. Varnamkhasti and L. S. Lee. 2012. A genetic algorithm based on sexual selection for the multidimensional 0/1 knapsack problems. In *International Journal of Modern Physics,* 422–431. World Scientific Publishing Company.

[24] B. Aghezzaf and M. Naimi. 2009. The two-stage recombination operator and its application to the multiobjective 0/1 knapsack problem: A comparative study. In *Computers & Operations Research*, Vol. 36, 3247–3262.

[25] E. Zitzler and L. Thiele. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. In *IEEE transactions on Evolutionary Computation*, Vol. 3, 257–271.