Journal of Engineering and Applied Sciences 12 (Special Issue 6): 7779-7784, 2017
ISSN: 1816-949X


# Evolutionary Computation: Characterization of the Genetic Algorithm in the Optimization of the Binary Knapsack Problem

Pedro Ramiro Brito Portero
Center for Research and Technology Transfer, International University of Ecuador,
Quito, Ecuador

**Abstract:** In the area of artificial intelligence, Genetic Algorithms (GAs) are based on the genetic process of biological evolution are adaptive logic processes that solve different problems of search and combinatorial optimization in engineering, the objective of this research is to characterize the behavior of the incomplete Genetic algorithm, in a classical NP-hard combinatorial optimization problem such as the binary Knapsack problem, establishing parameters or arguments of measurement in different probability ranges, allows us to establish analytical strategies that guarantee a range of minimum error, the proposed application allows maximizing the profit of the Knapsack and optimize the quantity of products that must be stored in it, presenting the results analytically and graphically.

**Key words:** Genetic algorithm, binary Knapsack, optimization, minimum error, graphically, parameters

## INTRODUCTION

This study describes a research in the field of evolutionary computation applied to engineering, on the use of Genetic Algorithms (GAs) for the roblem of the binary Knapsack (KP-01). The problem of the Knapsack is a method of combination and optimization (Gordon and Whitley, 1993) is based on a subset of items to be chosen such that the sum of corresponding benefits is maximized without exceeding the capacity of the Knapsack, the optimization process is a procedure to find the best optimal solution for a variety or set of solutions (Eiben and Smith, 2003). The candidates to solve are data structures such as graphs-tree of objects which can be unambiguously characterized by some attributes that are defined in an orderly way (Schleuter, 1990). In the study of the complexity of optimization problems are said to be programmable polynomically when there are resolution algorithms that give a response after performing a number of operations being this the size of the same problem (Goldberg, 1989), being a minimum requirement in the world of complexity that is if a problem is not polynomically programmable, absolutely all resolution techniques will be useless for increasing problem sizes, since, no computer would be able to confront in a minimally reasonable time. Usually, most of the optimization problems are nonpolynomically programmable. Under this conception, it is impossible or unlikely to find the global optimum in some applications, or there is no way to examine if there is a global optimum, there are cases where finding the global optimum is unnecessary but verifying or reaching a local optimum can be more convenient for the response time than finding the solution with an undesirable response time. Under this rationale, a classic optimization problem is the problem of the binary Knapsack, a strategy that allows us to find an adequate loading plan to fill it with the largest number of items, trying to achieve the maximum benefit (Fogel, 1994). For this, a set of n items is available which have a weight and an associated benefit and a subset of these items must be selected so that the sum of the benefits is maximum and the total sum of their weights does not exceed the capacity of the Knapsack.

Since the problem of the Knapsack is an NP problem, programming with Genetic algorithms are definitely the ones that govern and are the best approach to obtain solutions to problems traditionally considered to be computationally non-viable such as the problem of the Knapsack (Goldberg, 1989).

The document in its first phase describes the logical process that follows the Genetic algorithm and its application in the problem of the binary Knapsack, to be implemented by code using the scientific Software MATLAB and concludes with the results provided by the software whose name is identified as KP-01 (Chipperfield *et al*., 2003).

## MATERIALS AND METHODS

### Description of the Genetic algorithm

**Genetic algorithm:** Genetic Algorithms (GAs) are computer algorithms that look for good solutions to a

problem, John Holland was inspired by the mechanics of evolution, including survival of the fittest, reproduction and mutation (Hoffmeister and Back, 1990). These algorithms are suitable for solving a variety of problems, some applications of GAs are optimization, automation, programming, machine learning, economics, immune systems, genetic population and social systems.

**Basic idea behind Genetic algorithms:** GAs begin with a set of candidate solutions (chromosomes) called the population. A new population is created from solutions of an old population in the hope of having another optimized. The solutions that are chosen to form new solutions are selected according to their physical condition, the more suitable the solutions are, the more option they have to reproduce this process is repeated until some condition is fulfilled (Eiben and Smith, 2003).

**Basic elements of Genetic algorithms:** Most methods of GAs are based on the following elements, chromosome populations, selection according to physical state, crossover to produce new population and random mutation (Eiben and Smith, 2003).

**Chromosomes of Genetic algorithms:** The chromosomes in the GAs represent the space of candidate solutions, the possible encodings of the chromosomes are binary by permutation, value and encoding of trees. For the Knapsack problem, binary coding is used where each chromosome is a bit string of 0 or 1 (Eiben and Smith, 2003).

**Fitness function in Genetic algorithms:** GAs require a fitness function that assigns a score to each chromosome in the population. Thus, the best solution can be calculated and how well the problem is solved (Eiben and Smith, 2003).

**Selection in Genetic algorithms:** The selection process is based on physical fitness. Chromosomes that are evaluated with (adjusted) values are most likely to be selected for reproduction while those with low values will be discarded. The most suitable chromosomes can be selected several times, however, the number of chromosomes selected to reproduce is equal to the size of the population, therefore, the constant size is maintained for each generation. This phase has an element of randomness as does the survival of organisms in nature. The methods of selection are varied for this research is used the selection by tournament. In addition, to increase the performance of GAs, the selection methods are improved by elitism. Elitism is a method which first copies some of the chromosomes from the upper tip to the new population and then continues to generate the rest of the population (Eiben and Smith, 2003).

**Crossover in Genetic algorithms:** Crossover is the process of combining the bits of one chromosome with those of another. This is to create offspring for the next generation that inherits traits from both parents. Crossover randomly chooses a locus and exchanges sub-sequences before and after that locus between two chromosomes to create two descendants (Eiben and Smith, 2003). For example, consider the following parents and a crossover in position 3:

- Parents: 1 1 0 0|0 1 1 1
- Parents: 2 1 1 1|1 0 0 0
- Decendents: 1 1 0 0 1 0 0 0
- Decendents: 2 1 1 1 0 1 1 1

In this example, descendant 1 inherits bits in position 1-3 from the left side of the crossover of parent 1 and the remainder from the right side of the crossing point of parent 2, likewise, descendant 2 inherits bits in The position 1-3 from the left side of the father 2 and the rest of the right side of the father 1.

**Mutation in Genetic algorithms:** The mutation is performed after the crossing to avoid the fall of all solutions in the population in an optimal local problem solved. The mutation changes the new generation by changing the bits from 1-0 or from 0-1. The mutation can occur at each bit position in the chain with some probability, usually very small (0.001) (Eiben and Smith, 2003). For example, we consider the following chromosome with mutation point at position 2:

- Chromosome not mutated: 1 0 0 0 1 1 1
- Mutated Chromosome: 1 1 0 0 1 1 1

The 0 at position 2 changes to 1 after the mutation.

**Description of the binary Knapsack problem**

**Crossover in Genetic algorithms:** The problem of the binary Knapsack is an example of a combinatorial optimization problem which seeks a better solution among many other solutions. This is a Knapsack that has a positive volume or capacity V, there are ndifferent elements that can be potentially placed in the Knapsack. Item ihas a positive integer volume Vi and a positive integer Benefit Bi. In addition, there are copies of Qi of available item i where the Quantity Qi is a positive integer satisfying 1 = Qi = 8.

$X_i$ determines how many copies of item i are to be placed in the Knapsack, the goal is:

**Maximize:**

$$\sum_{i=1}^{n}\binom{n}{k}B_iX_i \quad (1)$$

**Subject to restrictions:**

$$\sum_{i=1}^{n}V_ix_i \leq V \quad (2)$$

And:

$$0 \leq xi \leq Qi$$

If one or more of Qi is infinite, the Knapsack is unlimited, otherwise, the Knapsack is limited. The limited Knapsack can be 0-1 or multimodal. If Qi = 1 for i = 1, 2, ..., n, the problem is a binary Knapsack 0-1.

**Example of the binary Knapsack:** It proposes a Knapsack that has a capacity of 13 cubic inches and several articles of different sizes and different benefits, it is required to include in the Knapsack only those items that have the highest total benefit within the restriction of the capacity of the Knapsack.

There are three potential elements (labeled "A", "B", "C"). Its volumes and benefits are:

- Article: A B C
- Benefit: 4 3 5
- Volume: 6 7 8

It seeks to maximize total profit:

$$\sum_{i-1}^{3}B_iX_i = 4X_1+3X_2+5X_3 \quad (3)$$

**Subject to restrictions:**

$$\sum_{i=1}^{3}v_ix_i = 46+7X_2+8X_2 \leq 13 \quad (4)$$

And:

$$x_i \in \{0,1\} \text{ for } i = 1,2,...,n$$

**Implementation of the Genetic algorithm for the binary Knapsack problem:** The implementation of the Genetic algorithm for the problem of the binary Knapsack is done with MATLAB Software from MathWorks. It works with 4 well-defined functions:

- Function population
- Function mutation

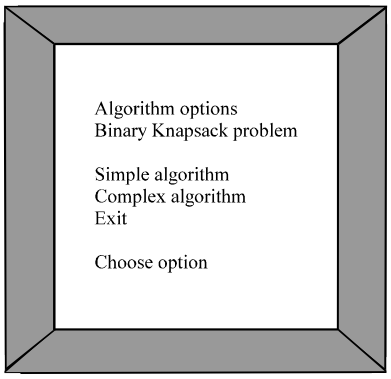


Fig. 1: Genetic algorithm menu options

- Function tournament selection
- Function recombination

In addition, you have the main program where it is dimensioned and addressed to the different functions, this program has the main menu (Fig. 1).

**Representation of items:** Two arrays are implemented for the cost or benefit denominated C and another array for the volume or weight denominated V.

C =

| | | | | | |
|---|---|---|---|---|---|
| 10 | 40 | 20 | 80 | 30 | 50 |

V =

| | | | | | |
|---|---|---|---|---|---|
| 100 | 400 | 200 | 800 | 300 | 500 |

These values are randomly generated within the set range for both the simple and the complex option [1,100], the difference lies in the number of objects in their order 100 and 10000.

**Initial population:** The implementation of the initial population is mandatory and very delicate, since, this process of deriving the whole Genetic algorithm. The coding to the binary system is done randomly for each of the products that is assigned, the way this process works is by the location in the array of the value 0-1.

X =

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 |

The binary vector **X** works with the positions of both the benefit vector C and the weight vector V, referring to the values 1 to activate both the benefit and the weight and 0 so that said referenced values are not activated. In the above example we would have the following:

C =

| | | | | | |
|---|---|---|---|---|---|
| 10 | 40 | 20 | 80 | 30 | 50 |

V =

| | | | | | |
|---|---|---|---|---|---|
| 100 | 400 | 200 | 800 | 300 | 500 |

X =

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 |

Restriction, capacity of the Knapsack 30. With these vectors, we calculate the objective function and the incumbent, the objective function is the sum of all the benefits that meet the restriction and the incumbent is the difference between the weight of the vector versus the constraint established as the capacity of the Knapsack, low. These parameters the result would be taking into account the previous vectors.

**Target function:**

| | | | | | |
|---|---|---|---|---|---|
| 10 | 40 | 20 | 80 | 30 | 50 |

**Incumbent function:**

| | | | | | |
|---|---|---|---|---|---|
| 100 | 400 | 200 | 800 | 300 | 500 |

**Benefit of 140:** This example declares an individual corresponding to a population, the same that would not be taken into account for the Knapsack for not complying with the parameter of the restriction, i.e., surpasses the capacity of the Knapsack. In the captured MATLAB binary coding is observed for 10 individuals and 10 random products, these matrix data represent the initial population.

**Binary coding is observed for 10 individuals and 10 random products:**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Evaluation:** The problems presented in the objective function are:

- Give negative values
- It may be non-discriminatory, this means that all individuals are in a very similar range
- That the problem is minimization

For the implementation of the Genetic algorithm, the minimization with the upper or the reverse dimension is verified, in this case the inverse was used.

**Selection:** Regardless of the method used to select, the process is to reduce the aspirants to parents and those with higher health have for the Genetic Algorithm implemented the model is used: parents+child, the (Nn) worst die, the best n are candidates for parents and make up the (Nn) best child of the next generation.

It is determined how many childs each individual should have in the population, for the Genetic algorithm implemented tournament selection was used, random sets of k possible parents, the star winners will be the parents in the following generations, it must be taken into account that the process of adaptation will depend on the value that is assigned to the tournament, the higher this value will have greater infeasibility. This method is the best computationally based on the whole literature of Genetic algorithms, giving quality and choice of a parent that guarantees adequate generations. In the code implemented there is a vector called des same that will contain the descendants, this vector is in the function.

**Tournament selection**

**Crossover:** The implementation of the crossover was performed in the recombination function, for this process we must take into account the vector of the descendants of, here, we ordered all possible parents according to their health and couples are formed in order, to cross a part of the paired according to an established probability, according to the literature the range of the probability of recombination or crossover is between the range of 0.5 and 1.0 for the implementation of the Genetic algorithm a probability of crossover of 0.6 was used. The same one that is within the suggested range.

**Mutation:** The mutation implementation was performed in the mutation function, this process allowed to make Genetic changes to the chromosomes, for this process a mutation rate was used which was also based on the literature establishing an established range of 0.05-0.001. The process of new scheme or mutation was worked on the recombined or crossed chromosomes worked in the

previous appendix, a gene is chosen randomly to decide if the mutation is made or not depending on the rate of mutation that is to say if in choosing the gene is less than or equal to the mutation rate the mutation is made.

## RESULTS AND DISCUSSION

The implementation of the Genetic algorithm for the binary Knapsack problem was carried out with MathWorks Software MATLAB. The Genetic algorithm used consists of a series of organized steps that are followed to give solution to a specific problem, therefore are optimization techniques that emulate the biological constitution.

The final verification process of the implemented management system was carried out taking into account some criteria:

- The time taken for the algorithm to take into account the number of generations
- Institutionalized the obligation to verify the variables population size and number of generations, i.e., increasing the number of generations within the same type of assessment, the number of individuals
- Control the number of generations where there is no improvement, there is a trigger to go through a long process of an optimal localized
- Control the optimum value to determine the nearest feasible solution

After working with instances and diagrams of tests were obtained different graphs that according to the simple or complex option were generated and was achieved at the end of several generations the following results.

For an application of a Knapsack with a capacity of 45, 10 individuals, probability of crossing or recombination of 0.6, probability of 1/n mutation and after having discriminated several experiments stability was achieved in the model, the results are indicated in figure that a specific continuation.

Figure 2 is represented all the generations that were obtained during the application of the mathematical model.

In Fig. 3, graph all the iterations both in crossover and mutation, managing to integrate the scattered spaces to reach the objective function within the generations that were obtained during the application of the model.

In Fig. 4, graph the times used in all iterations until getting the optimal function. As regards the complex option, the response time is very great, the computer to reheat the processor, it was possible to have a stability

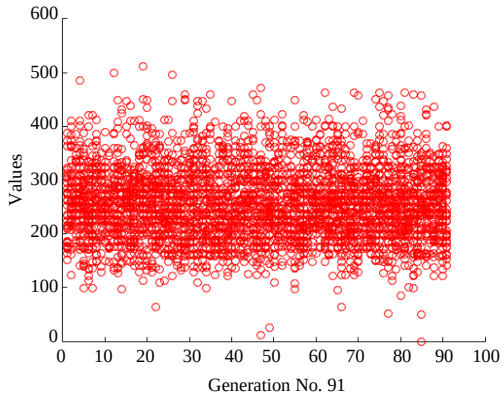


Fig. 2: Generations

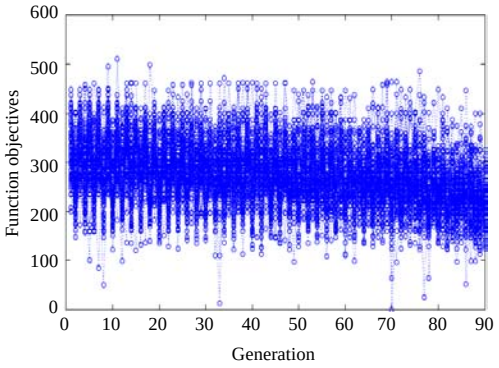


Fig. 3: Adaptation to the objective function

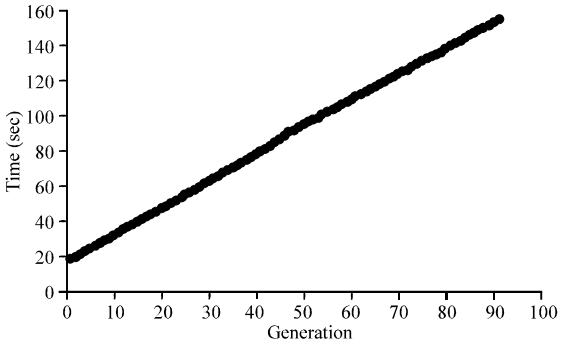


Fig. 4: Times used in iterations

but the computational results in a computer core i3 where the Genetic algorithm was implemented was very tense for the results that were not achieved but with a few hours of work and effort of the computer system.

In Fig. 5, graph the optimal function, after all the processes generated corresponding to the Genetic algorithm that is after the processes of selection, crossover and mutation, we get to obtain the optimized result function.

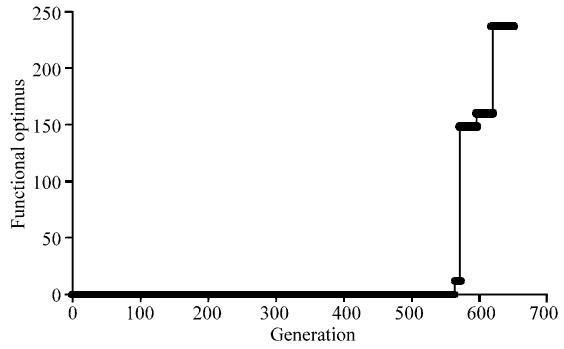


Fig. 5: Optimal function

## CONCLUSION

The optimization Genetic algorithm presents values closer to the expected ones because it searches the structure of the training data and assigns them to a different space of the other structures, once the measurement is made, it is analyzed which space belongs to throw the response that is, it searches for the center of gravity of the nearest vector space.

The solution time depends on the type of experiment being used, in this case, the simple option provides adequate processing times for the number of objects, otherwise in the complex option the time is very high.

Since, the size of the kernel is found to be adaptive, an algorithm is obtained which behaves in a stable manner for all types of instances. In fact, most cases are resolved and optimized by adapting to the classification or pre-processing of a large majority of articles.

It has been shown how Genetic algorithms can be used to find good solutions in the application of the binary Knapsack problem.

The Genetic algorithm allowed to reduce the complexity of the exponential to linear calculation which makes it possible to find approximately optimal solutions for an NP problem.

The result of the program shows that the application of a good method of selection and elitism are very important for the good performance of a Genetic algorithm.

The result of the program shows that the application of a good method of selection and elitism are very important for the good performance of a Genetic algorithm.

## RECOMMENDATIONS

The parameters used and proposed by the literature, generated adequate results, regarding the probability of crossover or recombination pr whose recommendation is to work in the range of 0.5-1.0, in addition, the propability of mutation likewise the recommendation is to work in the range of 0.001-0.05 which allowed not having remote incumbents.

## REFERENCES

Chipperfield, A.J., P.J. Fleming and H. Pohlheim, 1994. Genetic algorithm toolbox for use with MATLAB. Proceedings of the International Conference on Engineering System, September 6-8, 1994, Coventry University UK, Coventry, England, pp: 200-207.

Eiben, A.E. and J.E. Smith, 1994. Introduction to Evolutionary Computing. Springer, Berlin, Germany.

Eiben, A.E. and J.E. Smith, 2003. Introduction to Evolutionary Computing. Springer, New York, USA., ISBN-13: 9783540401841, Pages: 199.

Fogel, D.B., 1994. An introduction to simulated evolutionary optimization. IEEE Trans. Neural Networks, 5: 3-14.

Goldberg, D.E., 1989. [Genetic Algorithms in Search, Optimization and Automatic Learning]. Addison-Wesley, Boston, Massachusetts, ISBN: 9780201015768, Pages: 412 (In Spanish).

Gordon, V.S. and D. Whitley, 1993. [Serial and parallel genetic algorithms as function optimizers]. Proceedings of the 5th International Conference on Genetic Algorithms and their Application, June 11-15, 1993, Morgan Kaufmann Publishers, Burlington, Massachusetts, pp: 177-183 (In Spanish).

Hoffmeister, F. and T. Back, 1990. Genetic algorithms and evolution strategies: Similarities and differences. Proceedings of the International Conference on Parallel Problem Solving from Nature, October 1-3, 1990, Springer, Dortmund, Germany, ISBN: 978-3-540-54148-6, pp: 455-469 (In Spanish).

Schleuter, M.G., 1990. [Explicit parallelism of genetic algorithms through population structures]. Proceedings of the International Conference on Parallel Problem Solving from Nature, October 1-3, 1990, Springer, Dortmund, Germany, ISBN: 978-3-540-54148-6, pp: 150-159 (In Spanish).