Improving Performance Genetic Algorithm on Knapsack Problem by Setting Parameter

Rijois I. E. Saragih¹, Naikson F. Saragih¹, Mendarissan Aritonang¹

¹Universitas Methodist Indonesia, Jl. Hang Tuah No. 8 Medan, 20152, Indonesia
email: erwinsaragih@gmail.com

Abstract. This paper presents a solution for the knapsack problem which uses a genetic algorithm. Knapsack is combinatorial which is to find a good solution with constraint. In reality, this problem often happens. Unfortunately, making a good solution for this issue is not as easy as it is. In this research, it applied the genetic algorithm to find a good solution for that. The process is that a set of items with weight and value, then the selection of the items to be inserted into the backpack (knapsack) with limited capacity. So the items weighing should be smaller or equal to the capacity of the backpack, but the total value is as large as possible. A genetic algorithm is a heuristic searching algorithm based on natural selection of mechanism and nature genetics. The result suggests that the genetic algorithm can do a better performance than other comparable models by setting GA parameter.

1. Introduction

Knapsack's problem is a combinatorial optimization problem [1], for example, given a set of items with weight and value, then the selection of the items to be inserted into the backpack (knapsack) with limited capacity. So the items weighing should be smaller or equal to the capacity of the backpack, but the total value is as large as possible. The knapsack problem is that if a person puts things in a limited space then puts a variety of items into the room that has the maximum capacity so it does not allow to put all the items, therefore how one should put the item in the maximum possible space. The type of goods inserted into the room has weight, price and interest rate of other goods. Choosing suitable items with a place or container with weight considerations does not exceed the maximum capacity so that it can optimize the place used.

According to the theory of genetic algorithm that it forms population based on a searching technique by random [2] where nowadays more used on practical research. The genetic algorithm focus on keeping some possibility of best solutions for the problem deal with, which can be seen as process stages that known as population forming. An iterative new point in searching space that reach for evaluation and optional included in a population. Holland said [3] that the performance of the genetic algorithm is influenced by its operators such as selection, crossover, and mutation. Based on the explanation above, the researcher is interested to do research on how to apply the genetic algorithm to knapsack problem and to improve the performance of the genetic algorithm.

2. Methodology

To find a good solution for the knapsack problem and to improve the performance of the genetic algorithm are the key point of this research [4]. Therefore, it is used several steps of the process to
solve knapsack problems using a genetic algorithm. The steps are population, selection, crossover, and mutation. Those steps are continually repeated until finding the expected result [5]. The population consists of ten chromosomes or populations, the chromosome is ordered by genes which content scores.

The way genetic algorithm solves the problem through its parameters which drives an evaluation procedure, and it is also called chromosomes. These chromosomes are a simple string of data and instruction. There are steps of the genetic algorithm. The first step is chromosomes are random to form potential possible solutions. Those steps can be shown as [6];

1. Chromosomes generated randomly;
2. Apply fitness to each chromosome or genomes of the population;
3. Crossings of selected chromosomes to produce new chromosomes
4. Deleted old population members and keeping the population with the same N chromosomes;

Further explanation, a population of genetic algorithm also called chromosomes that are randomly distributed in the solution space is selected as the starting point of the search. The goal of fitness function is numerically encode the performance of the chromosomes. The crossover forms new chromosomes between two randomly selected good parents. One point crossover is applied in this research. The crossover is with some probability. The last step is mutation, this operator play important role to ensure that the probability of reaching any point in the search is never zero [7].

2.1 Population

In this research, the researcher uses the data benchmark knapsack problem. This data consists of 22 items that have value and weight. For example it will take 10 items to be explained in the following table.

| Items     | Weight | Value | Choice 1 | Choice 2 | Choice 3 |
|-----------|--------|-------|----------|----------|----------|
| Map       | 9      | 150   | N        | Y        | N        |
| Compass   | 13     | 35    | N        | Y        | Y        |
| Water     | 153    | 200   | Y        | N        | Y        |
| Sandwich  | 50     | 160   | N        | N        | Y        |
| Glucose   | 15     | 60    | Y        | N        | N        |
| Tin       | 68     | 45    | N        | Y        | N        |
| Banana    | 27     | 60    | N        | N        | Y        |
| Apple     | 39     | 40    | Y        | Y        | Y        |
| Cheese    | 23     | 30    | Y        | N        | N        |
| Beer      | 52     | 10    | N        | Y        | N        |

For example there are 10 items to be carried out in table 1, from the table the goods to be taken are marked Y and N are items that are not carried. Both marks can be encoded as 1 for Y and 0 for N marks. The chromosomes can be represented as in figure 1
2.2 Fitness

Finess value is calculated by summing the value of all selected items, but limited by maximum weight. The value of total finesses and weight can be calculated using the formula:

\[ F = \sum_{i=1}^{n} b_i v_i \]  
\[ W_{tot} = \sum_{i=1}^{n} b_i W_i \]

Where:
- \( F \) = the finess value of each chromosome
- \( b \) = bit value is 1 or 0 (value 1 if item is carried, and 0 if not carried)
- \( v \) = the value of each item
- \( n \) = the amount of all items
- \( W_{tot} \) = total Weight contained in each chromosome
- \( W \) = the weight contained in each item

The chromosomes in Figure 1 are limited to a smaller backpack capacity equal to 240 kg. The calculation of total weight uses equation 2.1 and calculates the fitness value using equation 2.2. If the calculation of the \( W_{tot} \) value exceeds the backpack capacity then the fitness value is automatically 0.

**Chromosome 1**

\[ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \]

**Chromosome 2**

\[ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \]

**Chromosome 3**

\[ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \]

Figure 1. Chromosomes

\[ \text{Chromosome 1} = 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \]

\[ F = 0 \times 150 + 0 \times 35 + 1 \times 200 + 0 \times 160 + 1 \times 60 + 0 \times 45 + 0 \times 60 + 1 \times 40 + 1 \times 30 + 0 \times 10 = 330 \]

\[ W_{tot} = 0 \times 9 + 0 \times 13 + 1 \times 153 + 0 \times 50 + 1 \times 15 + 0 \times 68 + 0 \times 27 + 1 \times 39 + 1 \times 23 + 0 \times 52 = 230 \text{ kg} \]

**Chromosome 2 = 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1**

\[ F = 1 \times 150 + 1 \times 35 + 0 \times 200 + 0 \times 160 + 0 \times 60 + 1 \times 45 + 0 \times 60 + 1 \times 40 + 0 \times 30 + 1 \times 10 = 280 \]

\[ W_{tot} = 1 \times 9 + 1 \times 13 + 0 \times 153 + 0 \times 50 + 0 \times 15 + 1 \times 68 + 0 \times 27 + 1 \times 39 + 0 \times 23 + 1 \times 52 = 181 \text{ kg} \]
Chromosome 3 = 0 0 1 0 1 0 0 1 1 0

\[ F = 0\times150 + 0\times35 + 1\times200 + 0\times160 + 1\times60 + 0\times45 + 0\times60 + 1\times40 + \\
1\times30 + 0\times10 = 330 \]

\[ W_{tot} = 0\times9 + 0\times13 + 1\times153 + 0\times50 + 1\times15 + 0\times68 + 0\times27 + 1\times39 + \\
1\times23 + 0\times52 = 230 \text{ kg} \]

3. Discussion and result

The first test was conducted using 22 item knapsack data by using several parameters of three experimental stages with different parameter values for classical genetic algorithms i.e. chromosome number, generation boundary, cross-probability probability, and mutation probability. The parameter values used for the genetic algorithm are as follows:

1. Number of population = 20
2. Generation = 80
3. Crossover probability = 0.6
4. Mutation probability = 0.1

Table 2. Test results on 20 population and 80 generations

| Testing | Generation | Fitness |
|---------|------------|---------|
| 1       | 77         | 900     |
| 2       | 50         | 855     |
| 3       | 67         | 737     |
| 4       | 44         | 810     |
| 5       | 72         | 830     |
| 6       | 80         | 860     |
| 7       | 70         | 760     |
| 8       | 55         | 615     |
| 9       | 79         | 735     |
| 10      | 69         | 680     |

In the table 2 it appears that the fitness value of each test is different, and testing is done ten times. The fitness value represents the number of items that can be chosen for inclusion in knapsack. At the test above the highest fitness value is 900 in the 77th generation, while the lowest fitness value is in the 55th generation of 615.
4. Conclusion

In this paper, the solution for the knapsack problem and improve the performance of the genetic algorithm and its application in the experiment is evaluated. The key point of this research is that is set up the GA parameter such as population, generation, crossover probability, and mutation probability. The improvement of the performance is about 75% is achieved.

Acknowledgments
Thanks to all the authors and the supervisors for all the times and supports for this research.

References
[1] R. P. Singh, “Solving 0–1 Knapsack problem using Genetic Algorithms,” 2011 IEEE
3rd Int. Conf. Commun. Softw. Networks, 2011.

[2] M. Mitchell, “An Introduction to Genetic Algorithms (Complex Adaptive Systems),” *MIT Press*, 1998.

[3] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. 1975.

[4] R. I. Erwin Saragih, M. Turnip, D. Sitanggang, M. Aritonang, and E. Harianja, “Increasing Prediction the Original Final Year Project of Student Using Genetic Algorithm,” *J. Phys. Conf. Ser.*, vol. 1007, p. 012039, Apr. 2018.

[5] J. Carr, “An Introduction to Genetic Algorithms,” *Whitman Coll. Math. Dep.*, 2014.

[6] C. R. Reeves and J. E. Rowe, *Genetic Algorithms—Principles and Perspectives*. 2002.

[7] I. De Falco, A. Della Cioppa, and E. Tarantino, “Mutation-based genetic algorithm: Performance evaluation,” *Appl. Soft Comput.*, vol. 1, no. 4, pp. 285–299, 2002.