Volume 6, Issue 1, January 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Analysis of Genetic Parameters and Operators for Knapsack Problem

Manpreet Kaur
Asst. Prof., Department of Computer Science
SRGCW, Amritsar, India

Abstract— Genetic Algorithms (GA) provide a simple method to solve complex optimization problems. The performance of a Genetic Algorithm mainly depends on the genetic parameters and operators. This paper analyzes the effect of various genetic parameters and operators on the performance of a GA. The different configurations of the algorithm are implemented on the "Knapsack Problem". The configuration of a GA is varied by using different Genetic parameters and operators. This paper also proposes an optimal parameter setting for Knapsack Problem using GA.

Keywords— Genetic Algorithm (GA), Knapsack Problem (KP), crossover, mutation, premature convergence, fitness function, Elitism.

I. INTRODUCTION

A GA is a nature-inspired algorithm that has been used to solve optimization problems. The task of optimizing a complex system presents at least two levels of problems for the system designer. First, a class of optimization algorithms must be chosen that is suitable for application to the system. Second, various parameters of the optimization algorithm need to be tuned for efficiency [15]. The optimization algorithm used here is Genetic Algorithm. A GA has many parameters that can be modified to improve the performance of particular implementations. A good selection of basic parameters of GA causes the algorithm to converge to best result in short time. Whereas, a worse setting causes the algorithm to run for a long time before finding a good solution or it might never be able to find a good solution.

According to a study, The GA parameters are divided into two categories: *Structural* and *Numerical*. Both these categories have different characteristics and thus have different impacts on GA solution. The *structural* parameters are *Coding*, *Crossover* and *Mutation operators* and its types, and *stopping criterion*. Whereas, *Numerical* parameters are *parent selection*, *population size*, *maximum generation no*. *crossover* and *mutation probabilities*. The very problem – specific nature of GA makes it difficult to specify the best combination of these parameter values. So, in this paper, the effect of using different parameters and operators is analyzed.

A) Introduction to Genetic Algorithm (GA)

The basic concept of GA's is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles of Charles Darwin of "Survival of the fittest". They represent an intelligent exploitation of a random search within a defined search space to solve a problem.

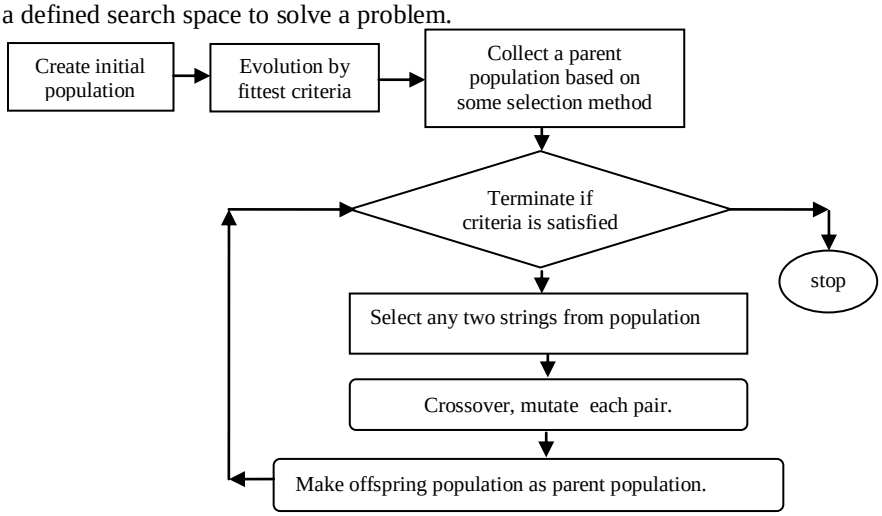

Fig. 1. A basic flowchart for a Genetic Algorithm.

The GA transforms a population of individual objects, each with an associated fitness value, into a new generation of the population using the Darwin principle of individual of reproduction and survival of the fittest and naturally occurring

genetic operation such as crossover and mutation. Each individual in the population represents a possible solution to a given problem. The GA attempts to find a very good solution to the problem by genetically breeding the population of individuals.

B) Introduction to Knapsack Problem (KP)

Knapsack problem belongs to NP difficult problem. It is a problem that some goods are loaded into a knapsack according to the capacity of the knapsack from n kinds of goods of the different weight and value. How to make the total value of the selected goods become maximal value problem. It is applied broadly to practice in resource allocation, investment decision –making, storage allocation, loading problem and so on. General knapsack problem may be described as follows. Suppose there are n different kinds of goods. The weight of goods i(i=1,2,…,n) is w_i and value is v_i. Some selected kinds of goods are loaded in a knapsack from n different kinds of goods. The capacity of a knapsack is c. Mathematical model is expressed as follows.

$$\text{Max } f(x_1, x_2, \ldots x_n) = \sum_{i=1}^{n} vixi$$

$$\sum_{i=1}^{n} wixi \leq c$$

$$x_i \in \{0,1\} \quad (i=1,2,\ldots,n)$$

The x_i of the formula is decision-making variable. x_i=1 shows that the goods *i* is loaded in the knapsack. But x_i=0 shows that the goods isn't loaded.

Genetic algorithm has become a key that solves the optimality result of the combination optimization problem. The paper adopts C-language to program for the simple and better genetic algorithm that solve knapsack problem. The experiment results show that the performance of the GA improves with the different combinations of Genetic parameters and operators.

II. RELATED STUDY

Population Size , when increased also increases the accuracy of Genetic Algorithm. According to [John J. Grefenstette, IEEE] , the optimal value of population size is between 30-100. Whereas, the standard GA based on Dejong's work has optimal value for population size to be 50-100. [Alander 91, IEEE] found that the parameter value of population size is approx. 40. [Arabas] proposed an algorithm eith varying population size. It gives the idea to adopt the parameters according to the current situation.[Zhang, Sakamoto, IEEE;2008] described that the value of population size affects the profit. According to this study, higher the population size, higher is the profit. The suggested value for knapsack problem is 200-300.

Selection, [David E. Goldberg and Kalyanmoy Deb] compared the expected behavior of four selection schemes. Proportionate selection is found to be significantly slower than the other three types. [Zhang and Kim] compared the same selection methods and reported that Tournament selection, uniform and linear ranking and genitor selection obtained comparable performance in average fitness and are significantly better than Proportional Selection. Ranking selection is best in convergence speed.

Crossover, [Spears and D. Jong] found that uniform crossover has a much higher schema disruptive rate when compared to 1- point and 2-point crossover. He further studied the effects of crossover operators on population size. He provided empirical results that uniform crossover is more suitable for small population sizes and 2- point crossover is better for large population sizes. [Boyabatli] concluded that crossover operators have significant impact on the performance on genetic algorithm. According to him, high mutation rates and low crossover rates are more suitable under some specific problem domains.

Mutation has been used by GA to maintain diversity of individuals and to prevent premature convergence to a sub-optimal solution.

Crossover Probability, the suggested settings for crossover probability are 0.6 [De Jong, 1975], 0.95 [Grefenstette,1986], [0.75,0.95] [Schaffer et al.,1989]. The common setting for crossover probability is in the range 0.7 to 1.0 [Goldberg, 1989].

Mutation Probability, the suggested settings for mutation probability are 0.001 [De Jong, 1975], 0.01 [Grefenstette, 1986], [0.005,0.01] [Schaffer et al.,1989]. The common setting for mutation probability is in the range 0.1 to 1.

III. SIMULATED EXPERIMENT

Assumed *n* of the goods number is 50, the weight of the goods is $\{w_i\}$, the value of the goods is $\{v_i\}$ and the capacity of the knapsack is *c*.

$\{w_i\}$={80,82,85,70,72,70,66,50,55,25,50,55,40,48,50,32,22,60,30,32,40,38,35,32,25,28,30,22,25,30,45,30,60,50,20,25,30,10,20,25,15,10,10,10,4,4,2,1};

$\{v_i\}$={220,208,198,192,180,180,165,162,160,158,155,130,125,122,122,120,118,115,110,105,101,100,100,98,96,95,90,88,82,80,77,75,73,72,70,69,66,65,63,60,58,56,50,30,20,15,10,8,5,3,1};

C=1000

Population size is 8.

A. The Simple Genetic Algorithm

The full process of simple genetic algorithm that solves knapsack problem is as follows: [14]

Step 1: The chromosome coding adopts binary coding. A binary string shows a chromosome that is made up of x_i whose value shows a gene of the chromosome. The chromosome is a possible result of the problem.
Step 2: Pseudo-random generator is designed that can generate the initial population.
Step 3: In the simple genetic algorithm, a fitness function is used to evaluate the superiority of the each individual and the higher score indicates that the individual is better. The fitness function reads as follows.
Weight=$\sum_{i=1}^{n} WiXi$
If weight $\leq$ c
$f(x_1,x_2,\ldots x_n) = \sum_{i=1}^{n} ViXi$;
else
$f(x_1,x_2,\ldots x_n) = \sum_{i=1}^{n} ViXi - k(Weight - c)$;
where, k is penalty coefficient, $k=\max(V_i/W_i)$
Step 4: Roulette-wheel selection operator is used.
Step 5: Crossover operator used is One-point crossover.
Step 6: Mutation operator used is Basic mutation operator. One random bit among each individual is changed. If the gene value is 1, then 1 is converted into 0, otherwise 0 is converted into 1.
Step 7: If the program dissatisfies the stopping criteria, then the next step turns into step 3 and continues to execute. Otherwise the next step turns into step 8.
Step 8: Stop with optimality results.

B. The Improved Genetic Algorithm with more suitable parameter setting
Population size=10
Termination criteria: Maximum no. of generations=1000.
Crossover operator: 1-point crossover or 2- point crossover
Mutation operator: Bitwise mutation or Swap mutation
Selection Method: Random/Roulette-wheel/Group selection
Elitism: yes/no. When creating new population by crossover and mutation, there is a big chance of losing the best chromosome. Elitism first copies the best chromosome to new population. Elitism can vary rapidly increase performance of GA, because it prevents losing the best found solution.

IV. DIFFERENT VARIATIONS OF GENETIC ALGORITHM

(i) Comparison of different crossover probabilities

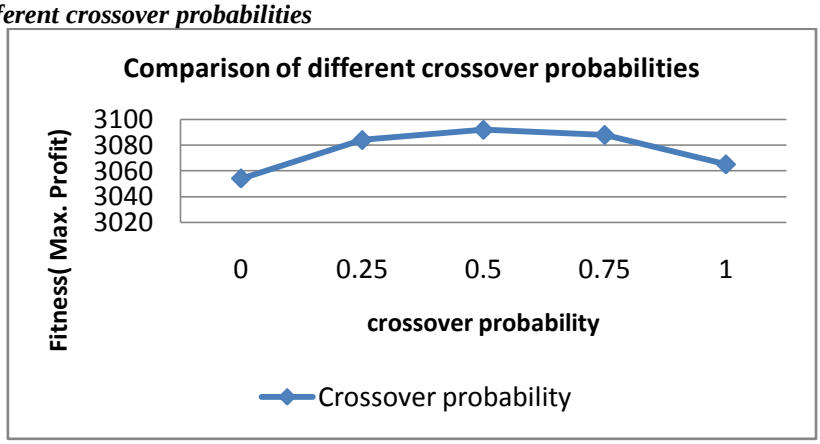

Fig 2: Effect of different crossover probabilities on the performance of GA.

According to the results, above figure presents improvement on fitness going from 0.25 to 0.50 level of crossover probability. The program gives maximum performance on level *Pc*=0.5. The maximum benefit for knapsack problem is 3092, which is far better than an existing solution found by a genetic algorithm i.e. 2879.

(ii)Comparison of different mutation probabilities

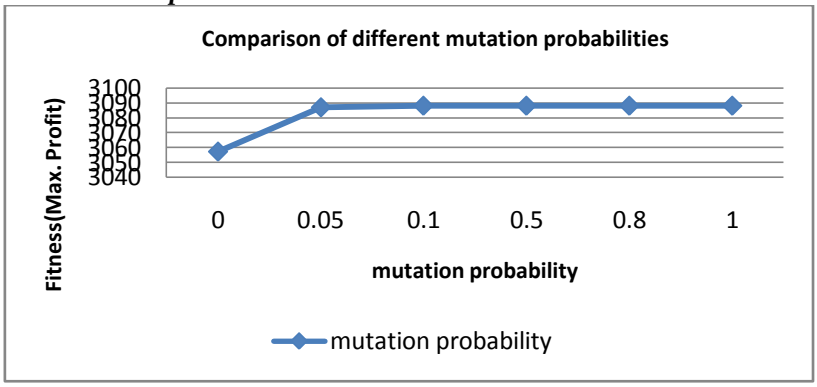

Fig 3: Effect of different mutation probabilities on the performance of GA.

The above figure shows the importance of mutation operator in terms of finding better solutions. This idea can be observed best by looking at the graph with Pm=0. There is no mutation and GA cannot find even good solution. There is significant improvement on fitness going from 0.05 to 0.1 level of mutation probability. The results show that increasing the mutation probability increases the fitness value until Pm=0.1. The other probabilities are insignificant after this level.

(iii)Comparison of different crossover methods

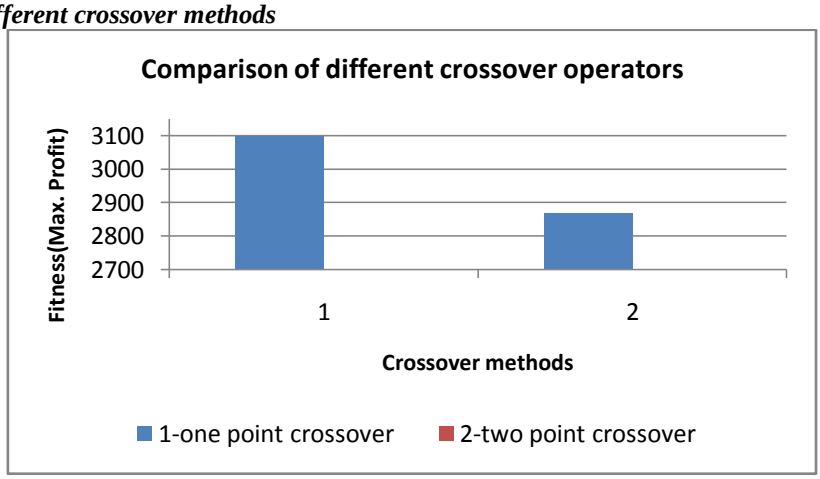

Fig. 4: Comparison of 1-point crossover and 2-point crossover operator.

The above figure shows the effect of different crossover methods on the performance of GA. The algorithm gives a better solution to knapsack problem using 1- point crossover than 2- point crossover.

(iv) Comparison of different mutation methods

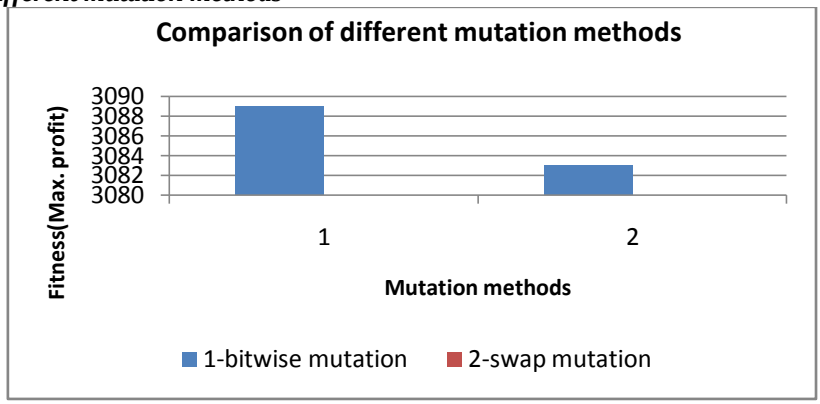

Fig 5: Effect of different mutation methods on the performance of GA.

The above results show the comparison between the bitwise mutation operator and the swap mutation method. It has been proved that bitwise mutation gives better results than swap mutation operator.

V. RESULTS AND ANALYSIS

The table given below shows the effect of different parameter and operator setting on the performance of GA. It compares the various four solutions of knapsack problem.

Parameters	**1st solution**	**2nd Solution**	**3rd solution**	**4th Solution**
Encoding	Binary	Binary	Binary	Binary
Selection	Roulette wheel	Roulette wheel	Roulette wheel	Group Selection
Crossover	1-point	2-point	2-point	2-point
Mutation	Bitwise	Swap	Bitwise	Bitwise
Elitism	No	Yes	No	No
Max. profit	2879	3072	3100	3108
Max. weight	999	1000	1000	1000

Fig 6: Results of different parameter settings

Comparison of results with different parameter settings is analyzed. It has a significant impact on the maximum profit obtained in the solution of knapsack problem.

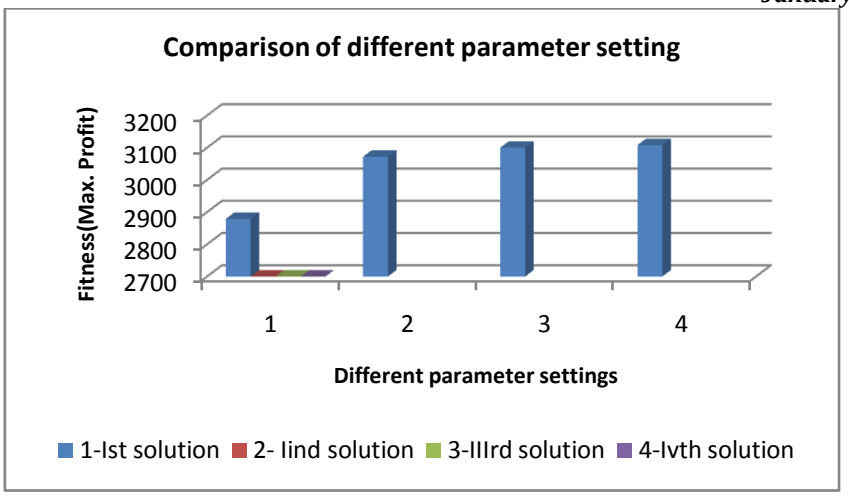

Fig 7: Effect of different GA parameters and operators settings on the Maximum Profit.

Here, ***Ist*** solution is the result of knapsack problem using a simple genetic algorithm. Total weight of knapsack fills with 999, whereas the maximum profit results into **2879**. This result is far less than other solutions. In the ***IInd*** solution, some improved parameter setting are performed, which results into maximum profit of **3072** and fills the knapsack with full capacity i.e. 1000. Parameters used in this algorithm are: *Roulette-wheel selection method*, *2-point crossover*, and *bitwise mutation*. The **IIIrd solution** is an improvement over IInd solution. It results in maximum profit of **3100**. The ***IVth*** solution results into maximum profit of **3108,** which is very good in comparison with previous solutions. In this algorithm, the *Roulette-wheel selection method* is replaced with *Group selection method*, which improves the performance a lot.
The selected items are: 1,2,4,6,8,9,10,11,13,15,16,17,19,20,22,23,24,25,26,27,29,35,37,40,41,47,49,50.

Complexity of the program

Since the number of chromosomes in each generation and the number of generations are fixed, the complexity of the program depends only on the number of items that may be placed in the knapsack. The following abbreviations are used: *N* for the number of items, *S* for the population size, and *G* for the number of possible generations.
The function that initializes the array of chromosomes has a complexity of O(N). The fitness, crossover function, and mutation function that checks for the terminating condition do not depend on N, but on the size of population have constant times of running O(1).
The selection, crossover, and mutation operations are performed in a *for* loop, which runs S times. Since S is a constant, the complexity of the whole loop is O(N). All these genetic operations are performed in a *do-while* loop which runs at most G times. Since G is a constant, it will not affect the overall complexity of the program, Thus, the total complexity of the program is O(N).

VI. CONCLUSION

The selection of suitable parameters and operators has a significant impact on the performance of a genetic algorithm. The nature of genetic operators is problem-specific. The suitable selection of crossover and mutation operators is very important. The different crossover and mutation operators are compared in this paper. Two-point crossover operator and bitwise mutation proves to be successful for "knapsack problem". The performance of a GA also depends on the number of crossover points used while performing crossover operation. Results show that using multiple number of crossover points enhances the performance rather than using a single crossover point. A suitable parameter selection implemented on a knapsack problem improves the performance by 7.2%.
The genetic algorithm used for the knapsack problem reduces the complexity from exponential to linear, which makes it possible to find approximately optimal solution for a NP problem.

VII. FUTURE SCOPE

The results can be useful for the analysis of other NP-hard problems for future research work. The performance of genetic algorithms can be compared for knapsack problem by using the other mutation and crossover operators. More suitable parameter setting can be found for knapsack problem. Also, the parameter and operator selection criteria for a genetic algorithm can be analyzed using some other NP- hard problem like Bin packing, Transportation problem, and Hamilton problem.

REFERENCES

[1] Anabela Simoes, Ernesto Costa, "*An Evolutionary Approach to the Zero/one Knapsack Problem: Testing Ideas from Biology*", Proceeding of fifth International Conference on Artificial Neural Networks and Genetic Algorithms Prague, Czech Republic, pp.22-25 April 2001.

[2] Christine L.Mumford, "*Comparing Representations and Recombination operators for the multi-objective 0/1 Knapsack Problem*", 2003, pp.854-861, IEEE.
[3] Colin Reeves, "*Genetic Algorithms*", chapter 3, pp. 55-82.
[4] David E. Goldberg, "*Genetic Algorithms in Search, Optimization and Machine learning",* Addition Wesley, Reading, MA,1989.
[5] David E. Goldberg and Kalyanmoy Deb, "*A comparative Analysis of selection schemes used in Genetic Algorithms*", Foundations of Genetic Algorithms Edited by Gregory J.E. Rawlins, 1991, pp.69-92.
[6] Hristaheva, Maya and Shrestha Dipti, "*Solving the 0-1 Knapsack Problem with Genetic Algorithms"*.
[7] Khoa Duc Tran, "*A survey of genetic algorithm parameters and their setting",*
[8] M. Srinivas, L.M. Patnaik, "*Adaptive probabilities of Crossover and Mutation in genetic algorithms*", IEEE transactions on systems, man and cybernetics, vol.24, no. 4,April 1994, pp. 656-667.
[9] Onur Boyabatli, Ihsan Sabuncuoglu, "parameter selection in genetic algorithms", Systematics, Cybernetics and Informatics, Vol. 2, No.4 , pp.78-83.
[10] Scott H. Doughlas," Variations on a simple genetic algorithm", Rochester Institute of technology, Rochester, NY.
[11] Siamak Sarmady," An investigation on genetic algorithm parameters".
[12] Stanley Gotshall, Bart Rylander,"Optimal population size and the genetic algorithm", IEEE.
[13] Yu-an Zhang, Makoto Sakamoto, Hiroshi Furutani," Effects of population size and Mutation rate on results of genetic algorithm", Fourth international conference on Natural Computation, IEEE,2008.
[14] Yi Yang, Qian-sheng Fang, "The improved genetic algorithm for solving Knapsack Problem Based on Handel-C", IEEE.
[15] Grefenstette, John J., "Optimization of Control parameters for Genetic Algorithms", IEEE Transactions on Systems, Man and Cybernetics, Vol. smc, No.1 pp.222-228, IEEE Lop number 5406073, 1986.

