8th International Conference from "Scientific Computing to Computational Engineering"
8th IC-SCCE
Athens, 4-7 July, 2018
© LFME

IMPROVED GENETIC ALGORITHM WITH ADAPTIVE OPERATORS

Anurak Choeichum[1], Narongdech Keeratipranon[1], and Chaiyaporn Khemapatapan[1]

[1] College of Innovative Technology and Engineering (CITE)
Dhurakij Pundit University (DPU), Bangkok, Thailand
anurak.choeichum@gmail.com, chaiyaporn@dpu.ac.th

Keywords: Genetic Algorithm, Local Optimum, Adaptive Operators

Abstract.*This paper shows how to improve genetic algorithm (GA) using adaptive operators for solving local optimum of optimization problem. The local optimum caused lack of variety of processing methods. Normally, GA uses only one mutation and one crossover causing trapped into local optimum in highly complex solutions. The objective is to improve genetic algorithm to be processed with lower iteration and more efficient result by adapting and selecting new genetic operators: crossover and mutation. There are seven crossover and seven mutation operators for using in adaptation and selection process. The proposed method will select new genetic operator for different ways of processing. This approach requires various patterns of crossover and mutation operators. The system is able to select appropriate operator to process each time. There are two scenarios, randomized and improved, to adapt and select new genetic operators. The latter scenario will compare performance between last and previous results. This method can find best solution faster than traditional GA with lower iteration because use of new genetic operators at each iteration provides more chance to overcome local optimum problem. The results indicate that adaptive operators can hit best solution faster than traditional genetic algorithm 28%. Moreover, the proposed method can achieve performance higher than traditional one 22%.*

1 INTRODUCTION

Evolutionary algorithms are the ones that follow the Darwin concept of Survival of fittest mainly used for optimization problems for more than four decades.[1] The Genetic Algorithm (GA) is a parallel and global search technique that emulates natural genetic operators. GA operates on a population of candidate solutions encoded in a finite set of strings, called chromosome. In order to obtain optimality, each chromosome exchanges data by borrowing operators from natural genetics to produce a better solution.

In global optimization problems, the particular challenge is that an algorithm may be trapped in the local optimum of the objective function. The GA has received considerable attention regarding its potential as a novel optimization technique for complex problems and has been successfully applied in various areas [2], such as bio-energy allocation [3], designed the sizing of device [4], transmission expansion planning [5], and GA improvements [6].

Improvements in GA have been sought in the optimal proportion and modification of the main parameters, namely probability of crossover, probability of mutation, crossover rated, mutation rated, population size, crossover operator, and mutation operator. The crossover and mutation are very important and commonly used operators to maintain diversity in population. They can find a result that closes to the answer of given problem. New population of solutions is created from previous solution, and basic strategy used in GA to create the target solution is to crossover and mutate the parent.

Various crossover and mutation techniques are built to get the optimal solution as minimum generation as possible. The selection of crossover and mutation operators is very important on the performance of GA. Using an unsuitable genetic operator will not make the best or target solution. From the part, using GA to find the best solution has problems and weaknesses, such as cannot get the target solution for complex problem, need a large population for finding the best solution, need lots of iteration, and trapped in the local optimum.

From the problems mentioned above, we have modified GA by increasing the diversity of genetic operators with creating a variety of genetic operators for different ways of finding the best solution. This modification will increase the chances of getting the best solution and overcome the local optimum problem. This method will improve the traditional GA to be more effective. Figure 1. shows the flow chart of traditional GA that uses single crossover and mutation technique.

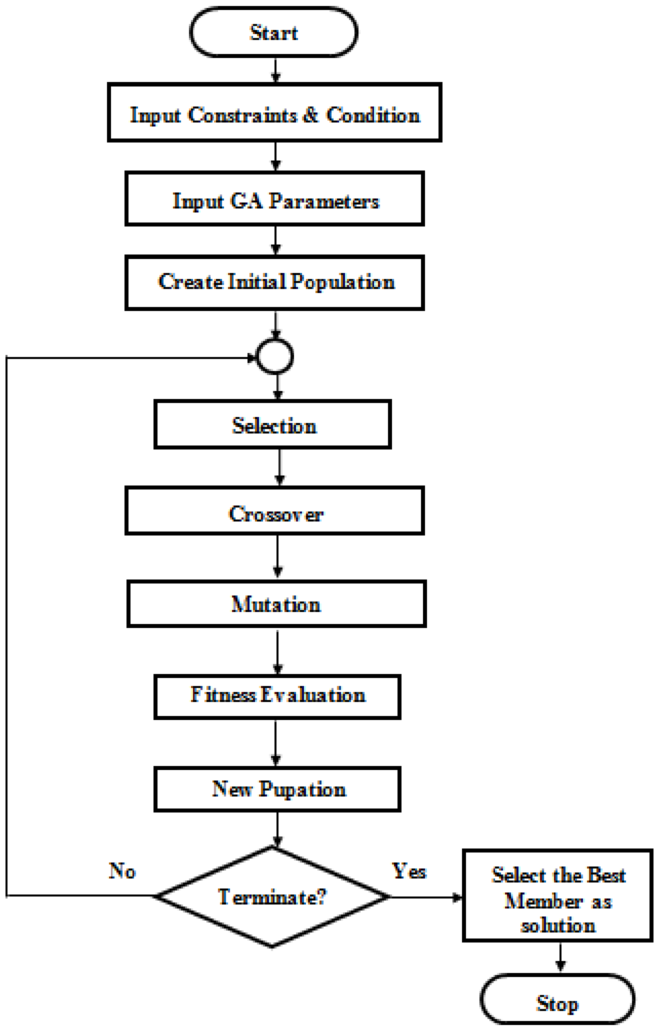

Figure 1. Flow Chart of traditional genetic algorithm

2 ADAPTIVE OPERATORS

The genetic operators, crossover and mutation operators, can be adaptively changed in each processing iteration. This paper has created five crossover operators and created five mutation operators which have different characteristics for each process. This algorithm selects the various genetic operators adaptively, and applied efficient operators to the individuals of GA populations.

2.1 Crossover Operator

There are seven crossover operators for this implementation, as shown in Table 1. The various general recombination operators from a set of available operators for GA made it necessary to fine tune the settings in order to achieve good results. The crossover operators are created as follows.

1) **Single-point Crossover It** is the most popular crossover [7]. A crossover site is aimlessly selected along the length of the mated strings and bits next to the crosssites are exchanged. The head and tail of one chromosome break up. If both head and tail have good genetic materials, then none of the offspring will get the both good features directly.

2) **K-Point Crossover** It uses the random crossover point to combine the parents same as per 1-Point crossover. To provide the great combination of parents it selects more than one crossover points to create the offspring or child [8]. It firstly selects the two parents used for crossover and then randomly selects *K* crossover points. Two offspring are created by combining parents at crossover point.

3) **Shuffle Crossover It** selects the two parents for crossover. It firstly randomly shuffles the genes in the

both parents but in the same way. Then it applies the 1-Point crossover technique by randomly selecting a point as crossover point and then combines both parents to create two offspring. After performing 1-point crossover the genes in offspring are then unshuffled in same way as they have been shuffled.[9]

4) **Uniform Crossover** Each gene in offspring is created by copying it from the parent chosen according to the corresponding bit in the binary crossover mask of same length as the length of the parent [10]. A new crossover mask is generated arbitrarily for each pair of parent chromosomes. The quantity of crossover point is not fixed initially. The offspring have a mixture of genes from both the parents [7].

5) **Average Crossover It** is the value based crossover technique. It uses two parents to perform crossover and creates only one offspring [8]. Average Crossover creates one offspring from taking average of the two parents. It selects two parents as X and Y and generate the child Z as follows: each gene in a child is taken by averaging genes from both parents.

6) **Discrete Crossover It** uses real random number to create one child from two parents. Unlike the uniform crossover only one child is generated in Discrete Crossover. It selects two parents as X and Y and generate the child Z such that it select genes of both the parents uniformly. The random real number decides from which parent to take the genes for child [8].

7) **Flat Crossover It** uses real random number to create one child from two parents. Similar to Discrete Crossover, it selects the genes from parent based on uniform random real number. The selected random real number should be a subset of set having the minimum and maximum of the genes of the both parents. It selects two parents as X and Y and generate the child Z such that it selects a random real number which is either min or max from genes in both parents and then assign this real number in child gene [9].

Order	Kind of Crossover Operators
C1	Single-point Crossover
C2	K-Point Crossover
C3	Shuffle Crossover
C4	Uniform Crossover
C5	Average Crossover
C6	Discrete Crossover
C7	Flat Crossover

Table 1.Kind of cross over operators

2.2 Mutation Operator

There are seven mutation operators for this implementation, as shown in Table 2. The mutation operators perform interchange, shuffling, and swap from a set of available operators for GA can give excellent solutions in various problem domains and overcome the local optimum problem. The mutation operators are created as follows.

1) **Insert Mutation** First of all, it randomly picks two allele values. Then it moves the second allele to follow the first, shifting the rest along to accommodate. It can be noted that it preserves most of the order and the adjacency information [10].
2) **Inversion Mutation** In order to perform inversion, it randomly picks two alleles and then inverts the substring between them [10]. It preserves most adjacency information and only breaks two links but it leads to the disruption of order information [11].
3) **Scramble Mutation** In this, from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.Subset does not have to be contiguous.
4) **Swap Mutation** To perform swap mutation, it selects two alleles at random and swaps their positions. It preserves most of the adjacency information but broken links disrupts order information.
5) **Interchanging Mutation** Two random positions of the string are chosen and the bits corresponding to those positions are interchanged.
6) **Uniform Mutation** The Mutation operator changes the value of chosen gene with uniform random value selected between the user specified upper and lower bound for that gene. It is used in case of real and integer representation.
7) **Creep Mutation** A random gene is selected and its value is changed with a random value between lower and upper bound. It is used in case of real representation.

Order	Kind of Mutation Operators
M1	Insert Mutation
M2	Inversion Mutation
M3	Scramble Mutation
M4	Swap Mutation
M5	Interchanging Mutation
M6	Uniform Mutation
M7	Creep Mutation

Table 2.Kind of mutation operators

3 DESIGNED METHODOLOGY

This paper, we use a population of size 20, crossover probability of 80%, mutation probability of 80%, and reproduction probability of 100%.

3.1 Population Selection

This method is done by using the roulette wheel technique. The concept of roulette wheel selection is that size of each partition depends on its fitness value. The higher fitness value, the bigger partition, and hence the higher chance to be selected for the next generation,

$$Pk_k = \frac{f_k}{\sum_{j=1}^{pop\_size} f^2{}_j} \tag{1}$$

where Pk_k is section probability, $f_k$is a fitness value of member k, and $f^2{}_j$ is a chromosome or each population in a total population.

3.2 Reproduction

The reproduction is used to generate the next generation from those selected through genetic operators (Crossover and Mutation) based on the principle of better fitness.

3.3 Fitness function

The fitness function of this implementation can be described by

$$\text{maximize } P = \sum_{d=1}^{D}(Eg_d * Pr_d) - \sum_{d=1}^{D}\sum_{b\in B}(Cf_b + \sum_{a\in A} Cm_{a,b}) + \sum_{d=1}^{D} I_{a,b,d} * Cmo_{a,b} - \sum_{d=1}^{D} I_{a,d} * Cp_a \tag{2}$$

where d is a member D, b is a member B, a is a member A, Eg is a gained from sold value, Pr is a selling price value, Cf is a fig cost, Cm is a maintanance cost, Cmo is an operation cost, I is product volume, and Cp is power cost.

3.4 Termination

The termination is a criterion which the GA decides whether to continue finding or to stop. This process section is repeated until a termination condition has been reached, as shown in equation 3. The condition is a combination between satisfying the maximum criteria and the maximum number of generation reached.

$$\begin{gathered} It_t \geq It_{max} \\ P_t \geq P_{target} \end{gathered} \tag{3}$$

where It_t is current iteration, and It_{max} is maximum iteration, $P_t$is current solution, and P_{target} is the target solution.

3.5 Operator Selection

This section can divide into two scenarios, randomized and improved, to adapt and select new genetic operators. The first scenario will change the new genetic operators with randomly in all iteration, and the second scenario will change the new genetic operator and compare the performance between last and previous results.

4 IMPLEMENTATION

This implementation has divided the processing method into two different scenarios. By the first scenario, it performs genetic operator changing in all processing cycles or all iterations. In the second scenario, it is to change the genetic operator to the specified conditions.

4.1 First Scenario

The improvement of GA in this scenario will randomly change the genetic operator from all created operators, both crossover operator and mutation operator. The genetic operator modification of this nature implies a natural process, which a diverse and unpredictable at the same time. Figure 2.,shows flow chart of GA with adaptive operators version 1 that using the genetic operator changing in all iteration form seven crossover and seven mutation technique.

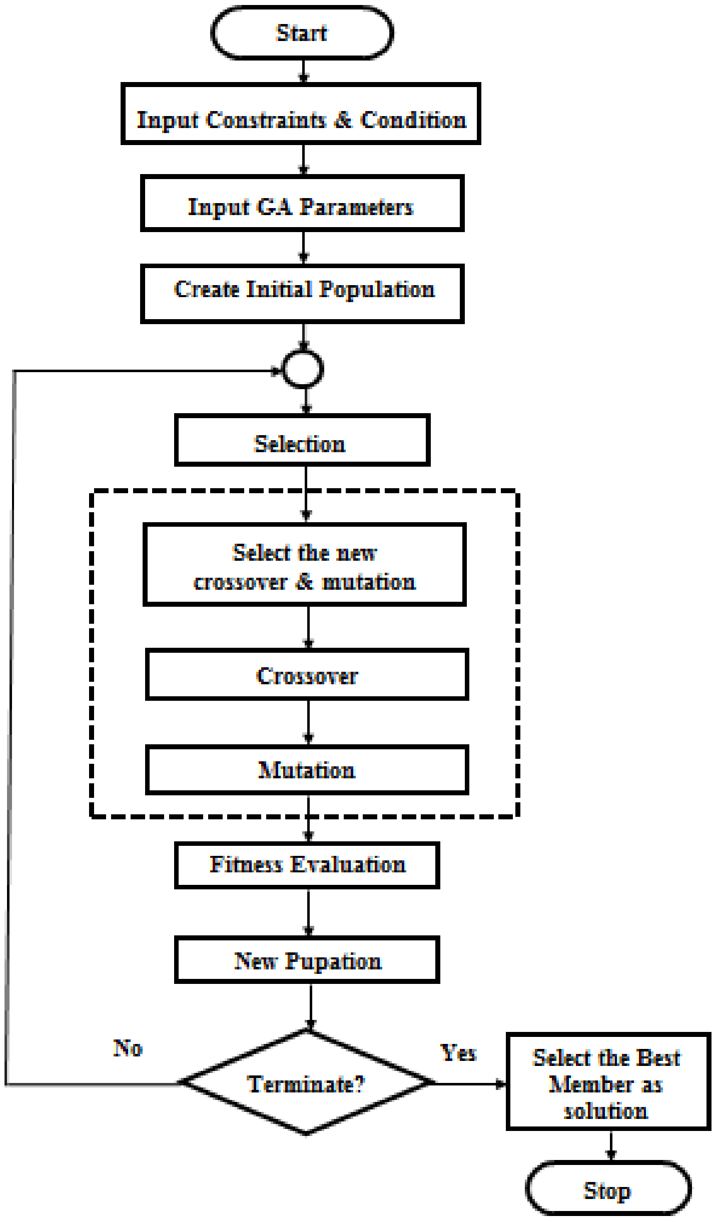
Start
Input Constraints & Condition
Input GA Parameters
Create Initial Population
Selection
Select the new crossover & mutation
Crossover
Mutation
Fitness Evaluation
New Pupation
No
Terminate?
Yes
Select the Best Member as solution
Stop

Figure 2. Flow Chart of Adaptive Operators V.1

4.2 Second Scenario

The improvement of GA in this scenario will randomly change the genetic operator from all created, both crossover operator and mutation operator similar to first scenario. But this scenario will change the genetic operator when the condition is satisfied.

This scenario compares performance between last and previous results, if the performance of last result is not better than previous five results, then the GA will randomly change the genetic operator immediately. But if the last result is better than previous five results, the GA will not make any changes, and use the same genetic operator in the previous iteration. Figure 3. shows flow chart of GA with adaptive operators version 2 that using the genetic operator changing in condition form seven crossover and seven mutation technique.

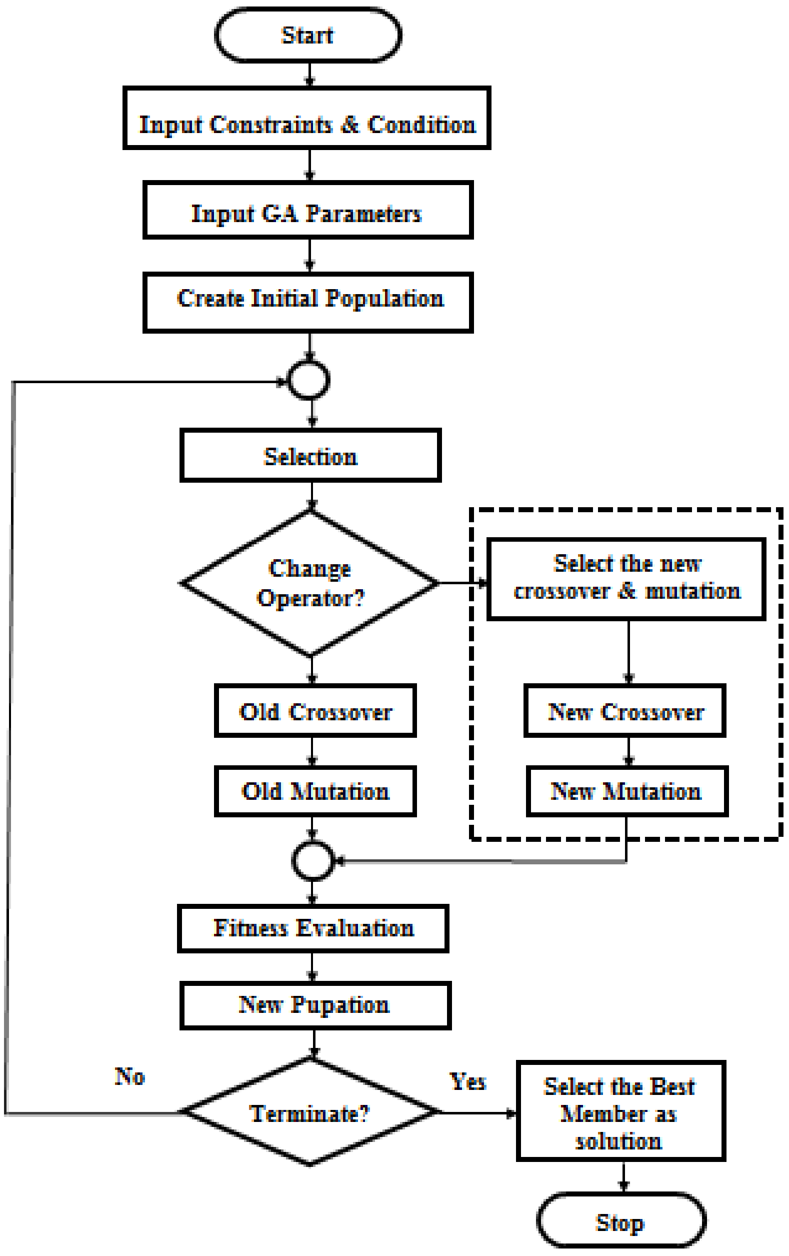

Figure 3. Flow Chart of Adaptive Operators V.2

5 EXPERIMENTAL RESULTS

The results of two scenarios are compared to the traditional GA. It can be noted from the figure 4 and 5 that the the proposed method can find target solution, but the traditional GA cannot do. The second scenario can get the best solution faster with lower iteration than the first scenario, because the second scenario changes the genetic operator with the appropriate condition that affected more result.

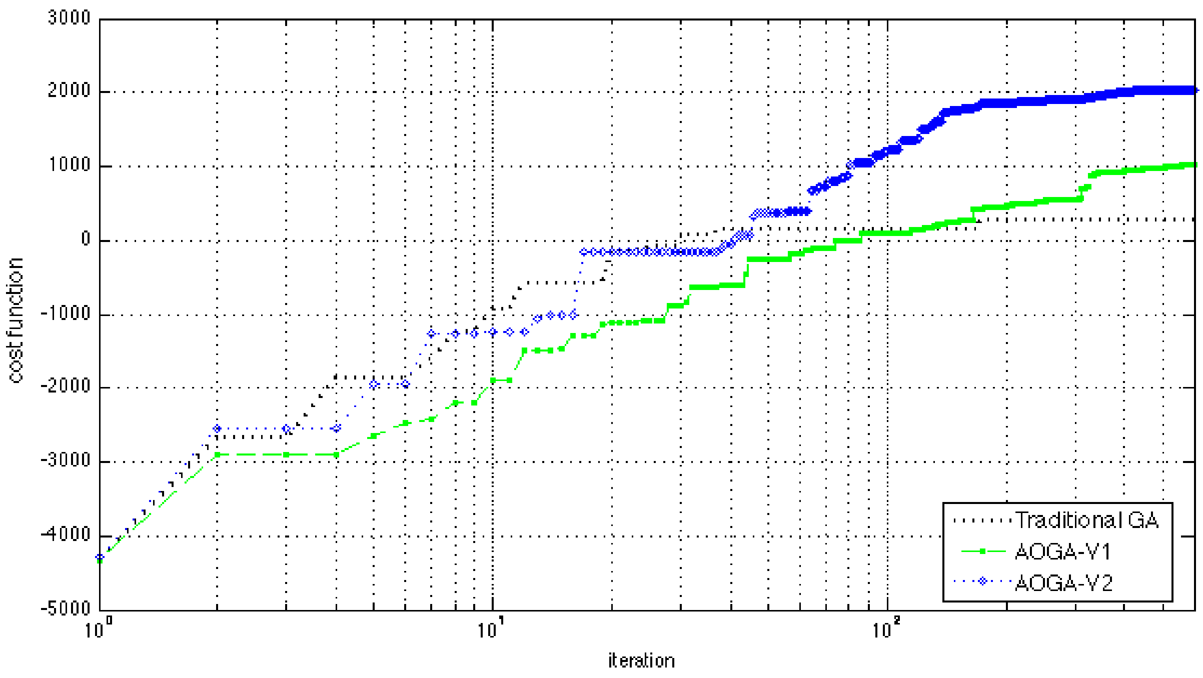

Figure 4. Average results for three methods

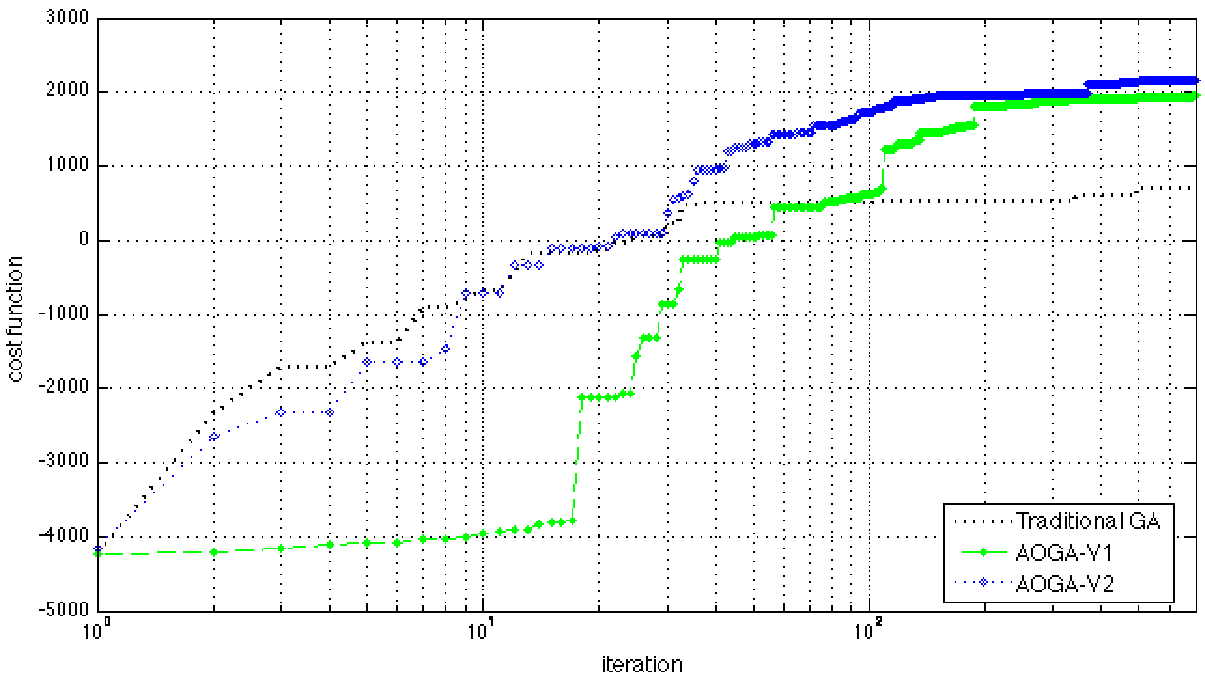

Figure 5. Best results of each method

The figure 5. shows the best results of each method. This result can describe the performance of AOGA-V1 that this genetic operator changing with randomly is less effective as shown in the average result in figure 4. These method has poor results compared to the AOGA-V2 when the number of iteration is more than previous one or it can change the operator in a good population. This will make it possible to reach the best solution, as shown in figure 5.

6 CONCLUSION

This paper improves genetic algorithm (GA) using adaptive operators for solving local optimum and getting the best solution. The adaptation was used in changing among several different crossover and mutation operators during the GA run. The adaptive operator process indicates that the selection of different operators affected in changes during the evolution process. The major advantage of this proposed method is we can get the target solution and overcome the local optimum from complex problem. The minor advantage of this approach is that we do not have to fine tune many necessary parameters of this algorithm beforehand.

It can be noted that the proposed method provided excellent results on test problem. The experimental results show that the proposed adaptive operator method (AOGA-V2) can find optimal solution or close-to-optimal solutions, and it is more efficient than the traditional GA.

REFERENCES

[1] J. Holland, (1975), *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor

[2] D.E. Goldberg, (1989),*Genetic algorithms in search, optimisation, and machine learning*, Addison Wesley Longman, Inc., ISBN 0-201- 15767-5

[3] A. Choeichum, N Keeratipranon, and C. Khemapatapan, (2017), “Multi-fuel allocation for power generation using genetic algorithms”, Journal of reviews on global economics, June, Vol.6

[4] Al-Shamma, A., & Addoweesh, K. E., (2012),“Optimum Sizing of Hybrid PV/Wind/Battery/Diesel System Considering Wind Turbine Parameters Using Genetic Algorithm”,*IEEE International Conference on Power and Energy (PECon)*

[5] Arabali, A., Ghofrani, M., Amoli, M. E., Fadali, M. S., & Aghtaie, M. M. (2014),“A Multi-Objective Transmission Expansion Planning Framework in Deregulated Power Systems With Wind Generation”, IEEE Transactions on Power Systems, Vol. 29

[6] Magyar, G., Johnsson, M., & Nevalainen, O. (2000),“An adaptive hybrid genetic algorithm for the three-matching problem”, IEEE Transactions on Evolutionary Computation, Vol. 4, Issue: 2

[7] Nitasha Soni, Tapas Kumar, (2014), “Study of Various Crossover Operators in Genetic Algorithms”, International Journal of Computer Science and Information Technologies, Vol. 5 (6)

[8] Tomasz Dominik Gwiazda (2006), *Genetic Algorithms Reference*, Volume –I, Poland: Tomasz Gwiazda

[9] A.J. Umbarkar and P.D. Sheth, (2015), “Crossoveroperatorsin genetic algorithms: A review”, ICTACT Journal on soft computing, October, Vol.6

[10] S.N. Sivanandam and S. N. Deepa., (2007), *Introduction to Genetic Algorithms*, Springer, ISBN 9783540731894

[11] Nitasha Soni, Tapas Kumar, (2014), “Study of Various Mutation Operators in Genetic Algorithms”, International Journal of Computer Science and Information Technologies, Vol. 5 (3)