Journal of Computing and Applied Informatics (JoCAI) Vol. 02, No. 2, 2018 | 87 - 100

talenta PUBLISHER

DATA SCIENCE

Journal of Computing and Applied Informatics

Genetic Algorithms Dynamic Population Size with Cloning in Solving Traveling Salesman Problem

Erna Budhiarti Nababan[1*], Opim Salim Sitompul[2], and Yuni Cancer[3]

[1,2,3]Universitas Sumatera Utara, Medan, Indonesia

Abstract. Population size of classical genetic algorithm is determined constantly. Its size remains constant over the run. For more complex problems, larger population sizes need to be avoided from early convergence to produce local optimum. Objective of this research is to evaluate population resizing i.e. dynamic population sizing for Genetic Algorithm (GA) using cloning strategy. We compare performance of proposed method and traditional GA employed to Travelling Salesman Problem (TSP) of A280.tsp taken from TSPLIB. Result shown that GA with dynamic population size exceed computational time of traditional GA.

Keyword: population size, local optimum, early convergence, Travelling Salesman Problem

Abstrak. *Ukuran populasi dari algoritma genetik klasik ditentukan secara konstan. Ukurannya tetap konstan selama proses berlangsung. Untuk menghadapi masalah yang lebih kompleks, ukuran populasi yang lebih besar perlu dihindari dari konvergensi awal untuk menghasilkan optimum lokal. Adapun tujuan dari penelitian ini adalah untuk mengevaluasi perubahan ukuran populasi, yaitu populasi dinamis Algoritma Genetika (GA) menggunakan strategi cloning. Kami membandingkan kinerja metode yang diusulkan dengan GA tradisional yang digunakan untuk Traveling Salesman Problem (TSP) dari A280.tsp yang diambil dari TSPLIB. Hasil menunjukkan bahwa GA dengan ukuran populasi dinamis mempunyai waktu komputasi yang lebih baik dari waktu komputasi yang dihasilakn GA tradisional*

Kata Kunci: *ukuran populasi, optimum local, konvergensi awal,*

Received 13 April 2018 | Revised 29 May 2018 | Accepted 23 June 2018

1. Introduction

Among several parameters of genetic algorithm, population size is an important parameter that can affect the performance of genetic algorithms. In a classical genetic algorithm, the population size is constantly fixed continuously during the evolutionary search until the maximum generation is achieved [1]. Figure 1 shows a population of several individuals, individuals 1 to n, for several generations the size of the population remains the same or constant until the maximum generation is achieved.

*Corresponding author at: Department of Information Technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jalan Alumni No. 9 Kampus USU, Medan 20155, Indonesia
E-mail address: opim@usu.ac.id

ISSN: 2580-6769 | e-ISSN: 2580-829X | DOI: 10.32734/jocai.v2.i2-326
Journal Homepage: https://talenta.usu.ac.id/JoCAI

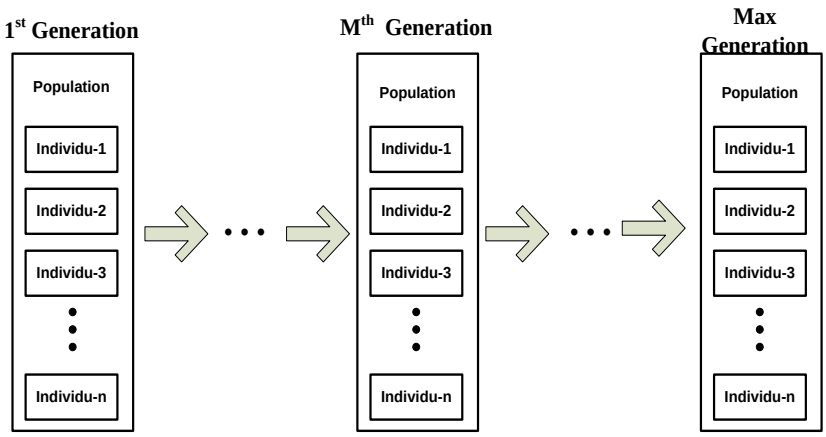

Figure 1. Constant population size on Genetic Algorithm

The dynamic population size is the number of individuals in the population of each generation that can be changed by plus and minus based on the best fitness value during the evolutionary process lasting until the maximum generation is reached [2]. Population dynamic demonstrates the population's capability in exploring and exploiting their potential habitat. Based on the ecological principles of natural population dynamics, dynamic populations should be more appropriate for evolutionary computation than fixed sized population. Fixed size population is strong contrast with population entities in nature. Biological populations are dynamic in both space and time [3]. Dynamic populations should be more appropriate for evolutionary computation. By testing five dynamic population sizing, which is which is random fluctuation population, increasing population, decreasing population, bell-shaped population and inverse bell-shaped population to mimic natural insect, [2] concluded that dynamic population size is more efficiently than fixed sized population in term of the number of fitness function evaluation and memory space requirement. According to [4] population size is one of the important parameters that affect the performance of genetic algorithms. On complex issues, the optimal population size is difficult to determine. Furthermore [5] stated that problem size and complexity of the problem is underlying the arrangement of population size. However if population size is too large or too small, it will trigger general problem of GA which is too large population size will increase computational time and cause convergence time longer. As [6] said the larger the population size, the better the solution. However, their research shown, population size above 100 chromosomes did not make better results while computing time continues to increase. On the other hand too small population size will result to a premature convergence and trapped to a local optimum [7]. Adjusting population size during a run could be more worthwhile than changing the operator parameters [4].

2. Previous Research

Many techniques have been used in setting the size of population to get an optimal solution. Research related to dynamic population size have been done by [8] by introducing methods of addition and subtraction of population size based on changes in the best fitness values in the

population. The population size needs to be improved to explore the search space, while population size reduction is done to improve the quality of solutions in the search space. The addition of new population sizes is done by cloning some individuals with the best fitness value. The results shown that genetic algorithms with population size changes during the evolution process get faster computational time compare to classical genetic algorithms.

In their research [9] reduced the size of the population adaptively. The process of evolution begins with a large population size then population size will continue to decrease depending on its best fitness value. In other words if the fitness value increases then the population size is reduced. Using this method, the genetic algorithm can produce better solutions and faster computational time then classical genetic algorithms.

Population reduction method was employed by [10] where population size is divided into n group. The purpose of this study is to get the best chromosome or individual from the search space. After initialization, population is divided into n groups. Each group was controlled by a complete tournament after which the best individual of each group is selected as the new population. This method produced better performance and better solution than that of classical genetic algorithms.

3. Method

In this research data is taken from TSPLIB in the form of two-dimensional symmetrical TSP: A280.tsp file containing coordinates of each city. For example, data of five cities used with the location coordinates presented in Table 1.

Table 1. Coordinates of five cities on A280.tsp dataset TSPLIB

No	Coordinate X	Coordinate Y
1	*288*	*149*
2	*288*	*129*
3	*270*	*133*
4	*256*	*141*
5	*256*	*157*

3.1. Euclidian Distance

Distance between cities *C* calculated using Euclidean formula (1):

$$C_{i,i+1} = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \quad (1)$$

Using (1), distance of five cities obtained is shown in Table 2.

Table 2. The distance of five cities

C	1	2	3	4	5
1	*0*	*20*	*24,08*	*32,98*	*32,98*
2	*20*	*0*	*18,44*	*34,18*	*42,52*
3	*24,08*	*18,44*	*0*	*16,12*	*27,78*
4	*32,98*	*34,18*	*16,12*	*0*	*16*
5	*32,98*	*42,52*	*27,78*	*16*	*0*

3.2. Chromosome Representation and Initial Population

The chromosome representation or encoding technique used in this research is permutations of the sequence of genes. Each city is represented by integer number and the sequence of genes in the chromosomal represent order of cities/routes to be traversed to get fitness value.

Initialization of the population is an integer number generated randomly in [1, n], where n is the length of the chromosome. For example, the initial population formation process given population size is 8 individuals shown in Table 3.

Table 3. Initial Population

Individual	Chromosome	Distance = Objective Function
1	*1 2 3 4 5*	*103,548*
2	*1 3 2 4 5*	*125,683*
3	*2 5 4 1 3*	*134,028*
4	*4 1 5 3 2*	*146,370*
5	*4 2 5 1 3*	*149,889*
6	*2 5 3 4 1*	*139,415*
7	*5 3 2 1 4*	*115,209*
8	*3 1 5 2 4*	*149,889*

3.3. Evaluation of Fitness value

Fitness value of each individual is calculated using (2).

$$fitness = \frac{1}{total\ distance} \quad (2)$$

Fitness of individuals is shown in Table 4

Table 4. Fitness value of each individu

Individual	Chromosome	Total Distance	Fitness
1	*1 2 3 4 5*	*103,548*	$\frac{1}{103,548} = 0,0097$
2	*1 3 2 4 5*	*125,683*	$\frac{1}{125,683} = 0,0080$
3	*2 5 4 1 3*	*134,028*	$\frac{1}{134,028} = 0,0075$
4	*4 1 5 3 2*	*146,370*	$\frac{1}{146,370} = 0,0068$
5	*4 2 5 1 3*	*149,889*	$\frac{1}{149,889} = 0,0067$
6	*2 5 3 4 1*	*139,415*	$\frac{1}{139,415} = 0,0072$
7	*5 3 2 1 4*	*115,209*	$\frac{1}{115,209} = 0,0087$
8	*3 1 5 2 4*	*149,889*	$\frac{1}{149,889} = 0,0067$

3.4. Selection

The selection process in this research is done by using roulette wheel selection method. We first calculate total fitness Fit_{tot} of all individual,

$$Fit_{tot} = \sum F_k \qquad (3)$$

where F_k is individual fitness

$$Fit_{tot} = 0,0097 + 0,0080 + 0,0075 + 0,0068 + 0,0067 + 0,0072 + 0,0087 + 0,0067 = 0,0611$$

Afterward, relative fitness P_k of each individual is calculated using (4) to select individual for crossover (see Table 5)

$$P_k = \frac{F_k}{Fit_{tot}} \qquad (4)$$

Table 5. Relative fitness of individuals

Individual	Relative Fitness (P_k)	Individual	Relative Fitness (P_k)
1	$\frac{0,0097}{0,0611} = 0,1580$	*5*	$\frac{0,0067}{0,0611} = 0,1092$
2	$\frac{0,0080}{0,0611} = 0,1302$	*6*	$\frac{0,0072}{0,0611} = 0,1174$

Continued on next page

Table 5. Continued from previous page

Individual	Relative Fitness (P_k)	Individual	Relative Fitness (P_k)
3	$\frac{0{,}0075}{0{,}0611} = 0{,}1221$	*7*	$\frac{0{,}0087}{0{,}0611} = 0{,}1421$
4	$\frac{0{,}0068}{0{,}0611} = 0{,}1118$	*8*	$\frac{0{,}0067}{0{,}0611} = 0{,}1092$

3.5. Crossover

The crossover process is done using the partially mapped crossover (PMX) method by taking 2 randomly cut points on each individual.

3.6. Mutation

The method of mutation used is to exchange one or more gene values in chromosomes randomly. The value of a mutated gene in one population is determined by probability mutation (*Pm*)

3.7. Dynamic Population

After the mutation process, the individuals generated are evaluated to see the best fitness value. The size of the population will increase or decrease depending on the change in the best fitness value. If the best fitness value increases then the population size will be increased. If the best fitness value remains the same for T generation, the population size will increase. When the best fitness value decreases, the population size will be reduced.

Addition of population size is done by cloning some individuals with good fitness and by randomly generating a number of new individuals. Meanwhile, reduction of population size is done by eliminating individuals with poor fitness. Overall process of adaptive population is as follow:

a. Calculate individual fitness after mutation process (see Table 6).

Table 6. Individual's fitness value after mutation

Individual	Genes	Total Distance	*Fitness*
1	*1 5 2 4 3*	*149,889*	*0,0067*
2	*1 5 3 4 2*	*131,070*	*0,0076*
3	*2 5 3 4 1*	*139,415*	*0,0072*
4	*2 5 4 1 3*	*134,028*	*0,0075*
5	*1 3 2 4 5*	*125,683*	*0,0080*
6	*2 1 5 3 4*	*131,070*	*0,0076*
7	*3 2 1 4 5*	*115,209*	***0,0087***
8	*5 3 2 1 4*	*115,209*	***0,0087***

The current best fitness value after the mutation process undergoes a change. In the initial population the best fitness value is 0.0097, after the mutation, the best fitness value decreases to 0.0087. Since the best fitness value is decreased then the population size is reduced using the following equation (5).

$$Pop_N = Pop_O * (1 - F_{dec}) \quad (5)$$

$$Pop_N = 8 * (1 - 0{,}4) = 4{,}8 \approx 5$$

where F_{dec} is decreasing factor in [0,1] interval

By eliminating individuals with smaller fitness values the size of the population in new generation is five individuals, as shown in Table 7.

Table 7. Individuals in new generation

Individual	Genes	Total Distance	*Fitness*
1	*1 5 3 4 2*	*131,070*	*0,0076*
2	*1 3 2 4 5*	*125,683*	*0,0080*
3	*2 1 5 3 4*	*131,070*	*0,0076*
4	*3 2 1 4 5*	*115,209*	***0,0087***
5	*5 3 2 1 4*	*115,209*	***0,0087***

After eliminating individual with smaller fitness value (on previous population), recalculate the fitness value of the new individual on new generation (see Table 8).

Table 8. Fitness value of individual on new generation

Individual	Genes	Total Distance	*Fitness*
1	*5 3 4 1 2*	*139,415*	*0,0072*
2	*3 5 2 4 1*	*161,550*	*0,0062*
3	*4 1 2 3 5*	*115,209*	*0,0087*
4	*4 1 5 3 2*	*146,370*	*0,0068*
5	*5 1 2 3 4*	*103,548*	***0,0097***

As shown on Table 7, the best fitness value obtained was 0.0087. After recalculating the fitness value of five individuals, the best fitness value increased to 0.0097. Since the best fitness value is increased, the population size will be cloned [11] using equation (6).

$$Pop_N = Pop_O + \left(Pop_O * \left(F_{inc} * (G_{max} - G_{cur}) * \left(\frac{Fmax_N - Fmax_O}{Fmax_{init}} \right) \right) \right) \quad (6)$$

$$Pop_N = 5 + \left(5 * \left(0{,}1 * (20 - 2) * \left(\frac{0{,}0097 - 0{,}0087}{0{,}0097} \right) \right) \right) = 5{,}9 \approx 6$$

where

Pop_N : new population size

Pop_O : previous population size

F_{inc} : increasing factor in [0,1] interval

G_{max} : maximum number of generation

G_{cur} : current generation

$Fmax_N$: the best fitness value of current generation

$Fmax_O$: the best fitness value of previous generation

$Fmax_{init}$: the best fitness value of initial population

Using (6) the population size of the new generation is 6 individuals. The new generation has the addition of 1 individual from the previous 5 individuals shown in Table 9.

Table 9. Population size to the new generation

Individual	Genes	Total Distance	*Fitness*
1	*5 3 4 1 2*	*139,415*	*0,0072*
2	*3 5 2 4 1*	*161,550*	*0,0062*
3	*4 1 2 3 5*	*115,209*	*0,0087*
4	*4 1 5 3 2*	*146,370*	*0,0068*
5	*5 1 2 3 4*	*103,548*	***0,0097***
6	*1 3 5 2 4*	*161,550*	*0,0062*

b. A whole process (selection-crossover-mutation-update population size) will be repeated until stopping criterion is met.

3.8. Stopping Criterion

The stopping criterion in this research is maximum number of generation which is determined by user.

4. Result and Discussion

In this section we will analyze genetic algorithm with constant population size and dynamic population size to solve Travelling Salesman Problem (TSP). Data set used is a280.tsp taken from TSPLIB.

4.1. TSP solution using Genetic Algorithm with constant population size

Testing done to solve Traveling Salesman Problem (TSP) problem with test data file a280.tsp with parameter setting as follow:

Population size = 100, 200, and 500

Maximum generation = 200, 500, and 1000

Probability of crossover (Pc) = 0.6

Probability of mutation (Pm) = 0.001

Testing performed using the population size of 100, 200 and 500 individuals, with the maximum generation of 200, 500 and 1000 generations, the probability of crossover (Pc) 0.6 and the mutation probability (Pm) 0.001 is used constantly during the iteration. Testing is done 5 times. Result obtained shown in Table 10, 11 and 12 respectively.

Table 10. Testing on 100 individuals and 200, 500 and 1000 generation GA with constant population size

Testing	*The best fitness*			Computational time (*ms*)		
	200 generation	500 generation	1000 generation	200 generation	500 generation	1000 generation
1	*0.3446*	*0.3355*	*0.3376*	*920*	*11263*	*22371*
2	*0.3545*	*0.3340*	*0.3340*	*1202*	*11263*	*22339*
3	*0.3318*	*0.3342*	*0.3342*	*1030*	*11247*	*22418*
4	*0.3346*	*0.3319*	*0.3369*	*936*	*11295*	*22371*
5	*0.3455*	*0.3326*	*0.3471*	*1139*	*11310*	*22371*
Mean	*0.3422*	*0.3336*	*0.3380*	*1045,4*	*11275.6*	*22374*

Table 11. Testing on 200 individuals and 200, 500 and 1000 generation GA with constant population size

Testing	*The best fitness*			Computational time (*ms*)		
	200 generation	500 generation	1000 generation	200 generation	500 generation	1000 generation
1	*0.3370*	*0.3333*	*0.3354*	*9063*	*22449*	*44757*
2	*0.3340*	*0.3337*	*0.3339*	*9064*	*22464*	*44788*
3	*0.3357*	*0.3357*	*0.3381*	*9111*	*22495*	*44819*
4	*0.3355*	*0.3357*	*0.3401*	*9141*	*22480*	*44819*
5	*0.3361*	*0.3369*	*0.3380*	*9157*	*22464*	*44757*
Mean	*0.3357*	*0.3351*	*0.3321*	*9107.2*	*22470.4*	*44788*

Table 12. Testing on 500 individuals and 200, 500 and 1000 generation GA with constant population size

Testing	*The best fitness*			Computational time (*ms*)		
	200 generation	500 generation	1000 generation	200 generation	500 generation	1000 generation
1	*0.3390*	*0.3401*	*0.3383*	*22807*	*56285*	*112259*
2	*0.3339*	*0.3380*	*0.3380*	*22745*	*56316*	*112508*
3	*0.3349*	*0.3413*	*0.3381*	*22838*	*57720*	*112243*
4	*0.3350*	*0.3363*	*0.3414*	*22776*	*56207*	*112367*
5	*0.3338*	*0.3383*	*0.3406*	*24461*	*56207*	*112274*
Mean	*0.3353*	*0.3388*	*0.3393*	*23125.4*	*56547*	*112330.2*

After we test the problem with constant population size, we then test the dynamic population size with the same parameters as we used in constant population.

4.2. TSP solution using Genetic Algorithm with dynamic population size

We then test the algorithm to a280.tsp with dynamic population size. The parameter used is same as parameter value in genetic algorithm with constant population with additional parameter values as follows:

Population Size	= 100, 200, and 500
Maximum Population Size	= 100, 200, and 500
Minimum Population Size	= 10
Maximum Generation	= 200, 500, and1000
Probablity crossover (*Pc*)	= 0.6
Probability mutation (*Pm*)	= 0.001
Increasing Factor	= 0.1
Decreasing Factor	= 0.4

Testing performed by using the population size of 100, 200 and 500 individuals, with the maximum generation of 200, 500 and 1000 generations, the probability of crossover (Pc) 0.6 and the mutation probability (Pm) 0.001 is used constantly during the iteration. Testing is done 5 times. Result obtained shown in Table 13, 14 and 15 respectively.

Table 13. Testing on 100 individuals and 200, 500 and 1000 generation GA with dynamic population size

Testing	*The best fitness*			Computational time (*ms*)		
	200 generation	500 generation	1000 generation	200 generation	500 generation	1000 generation
1	*0.3446*	*0.4082*	*0.5222*	*920*	*5242*	*11326*
2	*0.3545*	*0.4237*	*0.4750*	*1202*	*5616*	*11216*
3	*0.3318*	*0.4172*	*0.4877*	*1030*	*5398*	*11669*
4	*0.3346*	*0.4136*	*0.5146*	*936*	*5803*	*11684*
5	*0.3455*	*0.4132*	*0.4983*	*1139*	*5148*	*11622*
Mean	*0.3422*	*0.4152*	*0.4996*	*1045,4*	*5441.4*	*11503.4*

Table 14. Testing on 200 individuals and 200, 500 and 1000 generation GA with dynamic population size

Testing	*The best fitness*			Computational time (*ms*)		
	200 generation	500 generation	1000 generation	200 generation	500 generation	1000 generation
1	*0.3490*	*0.4366*	*0.5704*	*1997*	*9298*	*21513*
2	*0.3595*	*0.4488*	*0.5410*	*2215*	*9766*	*21403*
3	*0.3470*	*0.4270*	*0.5274*	*2138*	*9875*	*21763*
4	*0.3630*	*0.4489*	*0.5399*	*1903*	*9282*	*21497*
5	*0.3448*	*0.4619*	*0.5509*	*1701*	*9672*	*20498*
Mean	*0.3527*	*0.4446*	*0.5459*	*1990.8*	*9578.6*	*21334.8*

Table 15. Testing on 500 individuals and 200, 500 and 1000 generation GA with dynamic population size

Testing	*The best fitness*			Computational time (*ms*)		
	200 generation	500 generation	1000 generation	200 generation	500 generation	1000 generation
1	*0.3570*	*0.4856*	*0.5508*	*3416*	*21606*	*49811*
2	*0.3618*	*0.4813*	*0.5689*	*5522*	*21107*	*48361*
3	*0.3995*	*0.4722*	*0.5560*	*6568*	*20857*	*47861*
4	*0.3581*	*0.4870*	*0.5701*	*3900*	*15007*	*45662*
5	*0.3576*	*0.4593*	*0.6016*	*4612*	*15990*	*50779*
Mean	*0.3668*	*0.4771*	*0.5695*	*4803.6*	*18913*	*48494.8*

Comparison of average results obtained of the best fitness value and average computational time of the constant population size and dynamic population size of genetic algorithm illustrated in Figure 1 and Figure 2.

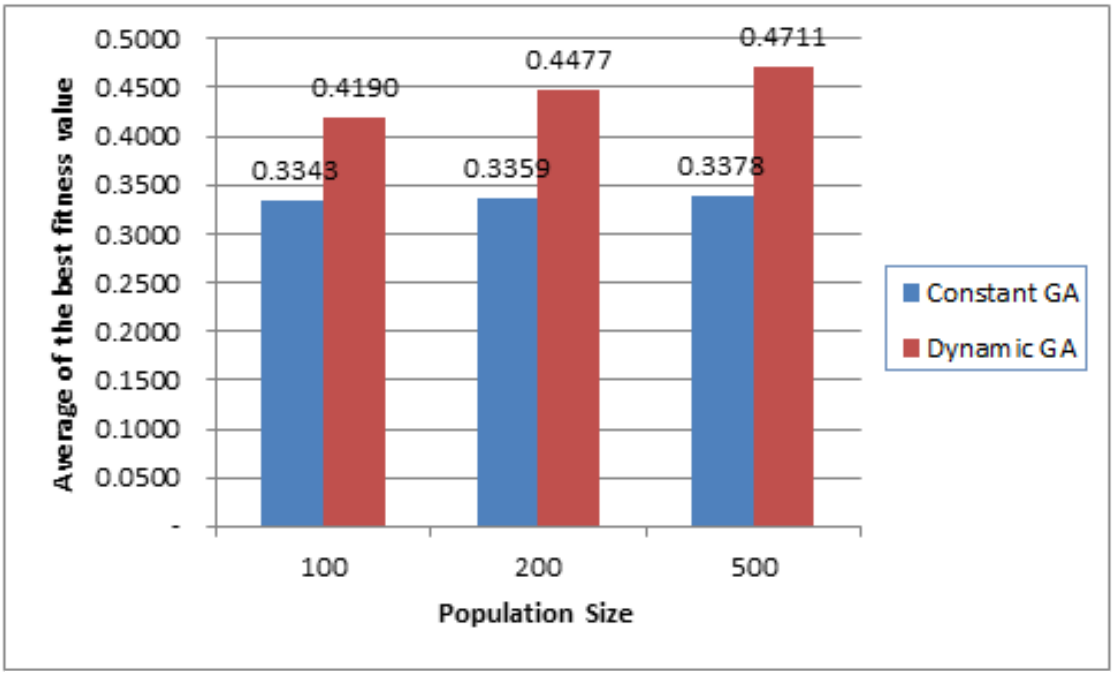

Figure 1 Average of best fitness value by constant population and dynamic population

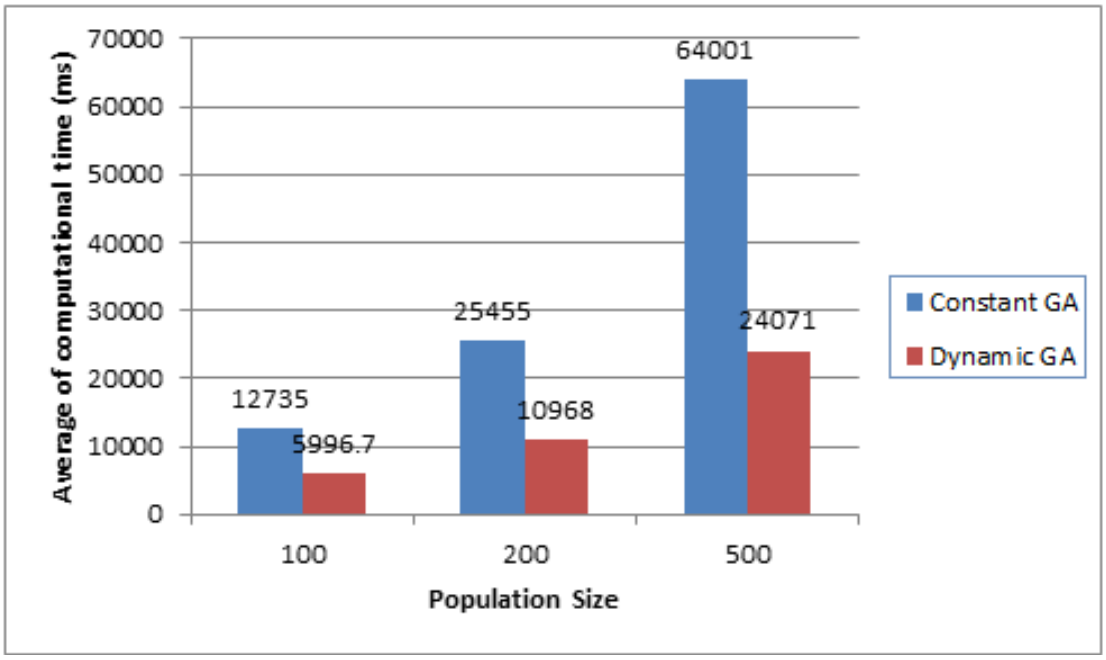

Figure 2 Average of computational time by constant population and dynamic population

Based on Figure 1 and Figure 2, average of the best fitness value and average of computational time obtained by dynamic population exceeds than that of constant population. Method of determining the population size is done by increasing or decreasing the population size dynamically based on the best fitness value. If the best fitness value increases then the population size increases, in the contrary if the best fitness value is reduced then the population size is reduced.

5. Conclusion and Future Research

Population size is one of the important parameters that affect the performance of genetic algorithms. Population size can affect the diversity of the population. Generally in classical genetic algorithms, population size is constantly regulated throughout the evolutionary process. On complex issues, optimal population size is difficult to determine. Population size changes can be made by increasing and decreasing the size of the population during the evolutionary process is underway based on the change in the best fitness value. The addition of population size is done by cloning some individuals with the best fitness. By using dynamic population size is the number of individuals in the population of each generation that can be altered by adding and subtracting the best fitness value during the evolutionary process until the maximum generation is achieved.

Based on the results obtained it is proven that genetic algorithm with dynamic population size can get better results than that of obtained by the population with constant size. Dynamic population size can increase the best average fitness value rather than a constant population size. The larger the population size, the better the solution obtained as evidenced by the increase in the average value of the best fitness.

REFERENCES

[1] Agoston E. Eiben, and Smith, J.E *Introduction to Evolutionary Computing*. Springer-Verlag Berlin Heidelberg, New York. 2003.

[2] Dinabandhu Bhandari, Murthy, C.A. and Pal Sankar.K. Variance as a stopping criterion for genetic algorithms with ellitist model. *Fundamenta Informaticae*, Volume 120, pages: 145-164. 2012

[3] Michalewicz, Zbigniew. *Genetic algorithms + data structures = evolution programs*. Springer-Verlag Berlin Heidelberg: New York. 1996.

[4] Alan Piszcz. and Terence Soule. Genetic programming: optimal population sizes for varying complexity problems. *Genetic and Evolutionary Computation Conference*, pages: 86-91. 2006.

[5] Fernando G.Lobo and Claudio F. Lima. A review of adaptive population sizing schemes in genetic algorithms. *Proceedings of the Genetic and EvolutionaryComputation Conference (GECCO'05),* pages: 228-234.2005

[6] Olympia Roeva., Stefka Fidanova and Paprzycki, M. Influence of the population size on the genetic algorithm performance in case cultivation process modelling. *Proceedings of the 2013 Federated Conference on Computer Science and Information System*, pages: 371-376. 2013

[7] Vlasis K. Koumousis and Cristos P. Katsaras. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Transactions on Evolutionary Computation*, Volume 10, Issue 1, pages: 19-28. 2006.

[8] Agoston E. Eiben, Marchiori, E. and Valkó, V.A. Evolutionary algorithms with on-thefly population size adjustment. *Parallel Problem Solving from Nature PPSN VIII, Lecture Notes in Computer Science*, pages: 41-50.2004.

[9] Joshua W. Hallam, Olcay Akman. and Fusun Akman. Genetic algorithms with shrinking population size. *Computational Statistics*, Volume 25, Issue 4, pages: 691-705.

[10] Raja, V. and Bhaskaran, M. Improving the performance of genetic algorithm by reducing population. *International Journal of Emerging Technology and Advanced Engineering , Volume 3 Issue 8, pages*: 86-91.2013.

[11] Muzid, S. Dinamisasi parameter algoritma genetika menggunakan population resizing on fitness improvement fuzzy evolutionary algorithm (PROFIFEA). *Prosiding SNATIF Ke-1 Tahun 2014,* pages: 471-478. 2014.