A genetic algorithm for solving large scale global optimization problems

M L Shahab, F Azizi, B A Sanjoyo, M I Irawan, N Hidayat, A M Rukmi
Department of Mathematics, Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia
E-mail: luthfishahab@matematika.its.ac.id

Abstract. There are many problems in the real world that can be modeled as large scale global optimization problems. Usually, large scale global optimization problems are global optimization problems where the dimensions are greater than or equal to 1000. In this research, we propose a genetic algorithm that can be used to solve large scale optimization problems with dimensions up to 100000. To measure the capabilities of the proposed genetic algorithm, we use five different test functions. Based on the results obtained, it can be inferred that the proposed genetic algorithm can find a good solution in a fairly short time.

1. Introduction
There are many problems in the real world that can be modeled as large scale global optimization problems. Usually, large scale global optimization problems are global optimization problems where the dimensions are greater than or equal to 1000. Large scale global optimization problems are difficult to solve because they have the following two things. First, the search space of large scale global optimization problems increases exponentially as the dimensions increase. Secondly, large scale global optimization problems cannot be solved properly using old optimization algorithms [1].

Evolutionary algorithms are more widely used to solve these problems. It is because evolutionary algorithms are easier to use and can find fairly good solutions in a relatively short time. There are various types of evolutionary algorithms that are often used, including simulated annealing, genetic algorithms, differential evolution, particle swarm evolution, firefly algorithms, cuckoo search, bat algorithms, flower pollination algorithms, and others [2]. Among the algorithms mentioned, the genetic algorithm is one of the most widely applied algorithms to solve various problems.

There are several studies in the literature that have tried using evolutionary algorithms to solve large scale global optimization problems. For examples, Ali and Tawhid [3] proposed a hybrid particle swarm optimization and genetic algorithm for problems with up to 1000 variables, Sun et al. [4] proposed a modified whale optimization algorithm for problems with up to 1000 variables, Wu et al. [1] proposed a hybrid algorithm for problems with up to 1000 variables, Zhang et al. [5] proposed MAGA-BigOpt for problems with up to 4864 variables, and Zhong et al. [6] proposed MAGA for problems with up to 10000 variables.

In this research, we propose a genetic algorithm that can be used to solve large scale optimization problems with dimensions up to 100000. The proposed algorithm must be able...
to solve the problems in a reasonable amount of time.

2. Genetic algorithms
Genetic algorithms are optimization methods inspired by the biological mechanisms of evolution and heredity which were first introduced by Holland in the 1960s. Genetic algorithms are one of evolutionary algorithms that have been used to solve many problems [2]. For example, genetic algorithms have been used extensively for numerical optimization, combinatorial optimization, the traveling salesman problem [7], the vehicle routing problem [8], classifier systems [6], multiple sequence alignment [9], parameter estimation [10], designing architecture of neural networks [11], and many other problems.

Genetic algorithms begin by defining a chromosome. The function of the chromosome is to represent the solution of the problem. Genetic algorithms work by creating a population consisting of several chromosomes. Next, it is necessary to define a fitness value that will be used to measure each chromosome in the population.

In general, genetic algorithms have three main operators to process chromosomes in a population. These operators are crossover, mutation, and selection [2]. Crossover is an operator to combine several parts of a solution with parts of other solution to get a new solution. The main objective of the crossover is to provide mixing of the solutions. Mutation is an operator for replacing some parts of a solution with random values. The purpose of mutation is to increase the diversity of the population and provide a mechanism for not being trapped in a local optimum. Selection (or elitism) is an operator to filter chromosomes with good fitness value and remove chromosomes with bad fitness value. Good chromosomes will be passed on to the next generation and become a new population to be reprocessed by genetic algorithms.

After genetic algorithms run several times or stop with certain conditions, the chromosome with the best fitness value in the population will be converted into a solution of the problem being worked on. The pseudo code of genetic algorithms is shown in Figure 1.

3. Proposed genetic algorithm
These are the characteristics of the proposed genetic algorithm used in this research:

- Suppose that we want to optimize a function of $d$ variables. The chromosome is a $d$-dimensional array representing a solution $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ of $f(\mathbf{x})$. In this research, we set $x_i \in [-10, 10]$ for $i = 1, \ldots, d$. 

---

Define a chromosome representing the solution
Define fitness function $F(\mathbf{x})$
Generate an initial population
Initialize the probabilities of crossover ($p_c$) and mutation ($p_m$)
Initialize the maximum number of generation ($n$)

while ($t < n$)
    Generate child by crossover with probability $p_c$
    Update obtained child by mutation with probability $p_m$
    Accept the new solutions if their fitness are better
    Select the current best chromosome for the next generation (elitism)
    Update $t = t + 1$
end while

Generate solution from the best chromosome

Figure 1. Pseudo code of genetic algorithms
A population in the proposed genetic algorithm consists of 100 chromosomes. In the first generation, all chromosomes will be generated randomly. For the next generation, a population will consist of some previous generation chromosomes and some new chromosomes obtained from crossover and mutation.

The fitness function is equal to the objective (test) function $f(x)$ that will be explained in the next section.

We use tournament selection for choosing two parent chromosomes before performing a crossover. The size of the tournament selection is five. It is done by first, selecting five chromosomes from the population randomly. Then the chromosome with the best fitness is chosen as the first parent. The second parent is picked using the similar process. With this scheme, the chromosomes which have better fitness will have a higher probability to be chosen.

Suppose that there are two parent chromosomes, $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$. First, we create an empty child chromosome $z = (z_1, \ldots, z_d)$. Then, crossover is done, for $i = 1, \ldots, d$, by choosing

$$z_i \in \left\{ x_i, y_i, \frac{x_i + y_i}{2} \right\}$$

which gives the best fitness for $z$. With this crossover, it is guaranteed that the child chromosome is equal or better than the parents. In one generation, the genetic algorithm generates 100 child chromosomes using this crossover.

Mutation is applied to all child chromosomes obtained from crossover with probability $p_m = 1$. Suppose that the chromosome $z = (z_1, \ldots, z_d)$ is obtained from crossover. First, we create an empty child chromosome $w = (w_1, \ldots, w_d)$. Then, mutation is done, for $i = 1, \ldots, d$, by choosing the best

$$w_i \in \{ z_i, z_i + \epsilon_i, z_i - \epsilon_i \}$$

and $\epsilon_i$ is chosen randomly from

$$\left[ \begin{array}{c} 1 \\ \frac{1}{2u} \end{array} \right]$$

where $u$ is the number of current generation. With this mutation, it is also guaranteed that the new chromosome is equal or better than the old one.

Selection (elitism) which is used in this research is elitism replacement with filtration. It is performed by combining 100 old (previous generation) chromosomes and 100 child chromosomes obtained from crossover and mutation. Then, all 200 chromosomes are sorted based on their fitness values. If there is a chromosome which is equal to other chromosomes, we only keep one chromosome and remove the others. If the number of remaining chromosomes is less than 100, we generate new random chromosomes until we have 100 chromosomes. The best 100 chromosomes will be passed on to the next generation and the other chromosomes will be discarded.

The stopping condition is whether the maximum number of generation has been reached or not. In this research, the maximum number of generation is 10000.

### 3.1. Dealing with many variables

The proposed genetic algorithm can be used to optimize all types of functions. However if a function $f(x)$ satisfies

$$f(x) = \sum_{i=1}^{d} f_i(x_i),$$

then the fitness function is equal to the objective function $f(x)$.
Define a chromosome representing the solution
Define fitness function $F(x)$
Generate an initial population (100 chromosomes)

\[ \textbf{while} (t < 10000) \]
\[ \quad \textbf{while} (i < 100) \]
\[ \quad \text{Choose the first parent with tournament selection} \]
\[ \quad \text{Choose the second parent with tournament selection} \]
\[ \quad \text{Generate child by crossover} \]
\[ \quad \text{Update obtained child by mutation} \]
\[ \quad \text{Update } i = i + 1 \]
\[ \textbf{end while} \]

Elitism replacement with filtration
\[ \textbf{if} (t \equiv 0 \text{ mod } 10) \]
\[ \quad \text{Calculate all fitness from the beginning} \]
\[ \textbf{end if} \]
\[ \textbf{end while} \]

Generate solution from the best chromosome

---

**Figure 2.** Pseudo code of the proposed genetic algorithm

i.e., one variable does not influence other variables in the function and it can be decomposed into many smaller functions of single variable, then there is a simple approach that can be used to optimize it faster. It has many advantages, especially when dealing with many variables. The approach is to use a shortcut for calculating the fitness of the child obtained from crossover or mutation. Suppose that the fitness of chromosome $x = (x_1, \ldots, x_n)$ is $F(x)$. The fitness of $x'$ obtained from $x$ by replacing the element $x_i$ with $a$ is

$$F(x') = F(x) - f_i(x_i) + f_i(a). \quad (5)$$

With equation (5), we do not need to calculate the fitness from the beginning each time the child is produced. However, because doing this kind of operations will accumulate errors for floating numbers, then we calculate the fitness of each chromosome from the beginning for every 10 generations of genetic algorithm.

4. Results

4.1. Test functions

In this research, we use five different test functions obtained from [2] with different domains. The names of the functions are Sphere, Schumer Steiglitz, Quintic, Step, and Alpine. The functions will be used to measure the capabilities of the genetic algorithm proposed in this research. We also use five different dimensions for all test functions. The dimensions are 10, 100, 1000, 10000, and 100000.

- Sphere function is defined by

$$f(x) = \sum_{i=1}^{d} x_i^2 \quad (6)$$

where $x_i \in [-10, 10]$. The global minimum of Sphere function is $f(x^*) = 0$ and it is located at $x^* = (0, \ldots, 0)$. 
Table 1. Results of the proposed genetic algorithm

| Function          | Dimension | \( f(x^*) \)         | Time   |
|-------------------|-----------|-----------------------|--------|
| Sphere            | 10        | \( 2.20 \times 10^{-92} \) | 1      |
| Sphere            | 100       | \( 9.15 \times 10^{-35} \) | 9      |
| Sphere            | 1000      | \( 9.16 \times 10^{-18} \) | 67     |
| Sphere            | 10000     | \( 6.26 \times 10^{-16} \) | 557    |
| Sphere            | 100000    | \( 3.24 \times 10^{-14} \) | 5663   |
| Schumer Steiglitz| 10        | \( 1.06 \times 10^{-147} \) | 2      |
| Schumer Steiglitz| 100       | \( 2.03 \times 10^{-63} \)  | 9      |
| Schumer Steiglitz| 1000      | \( 2.78 \times 10^{-36} \)  | 72     |
| Schumer Steiglitz| 10000     | \( 4.72 \times 10^{-34} \)  | 564    |
| Schumer Steiglitz| 100000    | \( 4.95 \times 10^{-32} \)  | 5889   |
| Quintic           | 10        | 0                     | 2      |
| Quintic           | 100       | \( 3.24 \times 10^{-13} \) | 12     |
| Quintic           | 1000      | \( 6.88 \times 10^{-7} \)  | 97     |
| Quintic           | 10000     | \( 1.89 \times 10^{-5} \)  | 783    |
| Quintic           | 100000    | \( 4.88 \times 10^{-4} \)   | 7586   |
| Step              | 10        | 0                     | 2      |
| Step              | 100       | 0                     | 9      |
| Step              | 1000      | 0                     | 85     |
| Step              | 10000     | 0                     | 761    |
| Step              | 100000    | 0                     | 7625   |
| Alpine            | 10        | \( 1.56 \times 10^{-35} \) | 4      |
| Alpine            | 100       | \( 2.55 \times 10^{-10} \) | 32     |
| Alpine            | 1000      | \( 6.08 \times 10^{-8} \)  | 290    |
| Alpine            | 10000     | \( 1.15 \times 10^{-6} \)  | 2458   |
| Alpine            | 100000    | \( 3.27 \times 10^{-5} \)  | 21855  |

- Schumer Steiglitz function is defined by
  \[
  f(x) = \sum_{i=1}^{d} x_i^4
  \]  
  
  where \( x_i \in [-10, 10] \). The global minimum of Schumer Steiglitz function is \( f(x^*) = 0 \) and it is located at \( x^* = (0, \ldots, 0) \).

- Quintic function is defined by
  \[
  f(x) = \sum_{i=1}^{d} \left| x_i^5 - 3x_i^3 + 4x_i^2 + 2x_i^2 - 10x_i - 4 \right|
  \]  
  
  where \( x_i \in [-10, 10] \). The global minimum of Quintic function is \( f(x^*) = 0 \) and they are located at \( x^* = (a_1, \ldots, a_d) \) where \( a_1, \ldots, a_d \in \{-1, 2, b\} \) and \( b \approx -0.4026 \).

- Step function is defined by
  \[
  f(x) = \sum_{i=1}^{d} \lfloor |x_i| \rfloor
  \]  

where $x_i \in [-10, 10]$. The global minimum of Step function is $f(x^*) = 0$ and they are located at $x^* = (a_1, \ldots, a_d)$ where $a_1, \ldots, a_d \in (-1, 1)$.

- Alpine function is defined by

$$f(x) = \sum_{i=1}^{d} |x_i \sin x_i + 0.1x_i|$$

where $x_i \in [-10, 10]$. The global minimum of Alpine function is $f(x^*) = 0$ and they are located at $x^* = (a_1, \ldots, a_d)$ where $a_1, \ldots, a_d \in A \cup \{0\}$ and $A = \{b \in [-10, 10] \mid \sin(b) = -0.1\}$.

### 4.2 Results

We have created a program to implement the proposed genetic algorithm. The program is written in the Java programming language with the NetBeans IDE. The program can be accessed online at [https://github.com/luthfishahab/gaslsgop](https://github.com/luthfishahab/gaslsgop). When running the program, we use a computer with the Intel Core I5-2100U Processor and 4GB RAM.

The main results of the proposed genetic algorithm can be seen in Table 1 where the time in the fourth column is in seconds ($s$). In this research, we assume that a solution $x^*$ is good enough if $f(x^*) < 10^{-3}$. Based on these results, we can see that our genetic algorithm can find good solutions for all test functions and dimensions. For Step function, the algorithms can find its optimal solutions. It is because Step function has many optimum points and they can be detected easily by the genetic algorithm. The genetic algorithm also finds really good solutions for Sphere function and Schumer Steiglitz function, even when the dimension is 100000.

It also can be seen from Table 1 that the proposed genetic algorithm finds the solutions in appropriate time. For example, the proposed genetic algorithm takes 5663 seconds or about 1.57 hours for solving Sphere function with 100000 variables. On average, if the dimension is increased by 10 times, the time needed to solve the problem increases 8.24 times. The time required for solving each function is different, depending on the complexity of the function. It takes more time for Alpine function which has sine function because the Java language programming must

![Figure 3. The time needed by the genetic algorithm for solving Sphere function](https://example.com/figure3.png)
use special class for calculating it. In Figure 3, we plot the time needed by the proposed genetic
algorithm for finding solutions of Sphere function and the graph seems to be linear.

In Figure 4 on the last page, we present the function/fitness values of the best chromosomes
of the first 10 generations. All functions presented in the figure have 100000 variables. It
appears that the fitness values move down very rapidly for all types of functions. Among the
five functions shown, Schumer Steiglitz function is the fastest descending function and Alpine
function is the slowest one.

5. Conclusion and future works
In this research, we propose a genetic algorithm that can be used to solve large scale optimization
problems with dimensions up to 100000. We also propose a simple approach to make the genetic
algorithm run faster. Based on the results obtained, it can be inferred that the proposed genetic
algorithm can find a good solution in a fairly short time.

For the future, the algorithm that has been proposed in this research can be redeveloped
in order to solve other problems, especially for non-separable problems. Researchers can also
combine the genetic algorithm with other heuristic algorithms so that its capabilities can be
increased. However, in the development, the runtime must be considered carefully so that the
proposed algorithm can be used by many people.

Acknowledgement
This research is supported by the Department of Mathematics, Institut Teknologi Sepuluh
Nopember, Ministry of Education and Culture. The number of the research contract is
1588/PKS/ITS/2020.

References
[1] Wu X, Wang Y, Liu J and Fan N 2019 IEEE Access 7 103354–103364
[2] Yang X S 2014 Nature-inspired optimization algorithms (Elsevier)
[3] Ali A F and Tawhid M A 2017 Ain Shams Engineering Journal 8 191–206
[4] Sun Y, Wang X, Chen Y and Liu Z 2018 Expert Systems with Applications 114 563–577
[5] Zhang Y, Zhou M, Jiang Z and Liu J 2015 2015 IEEE Congress on Evolutionary Computation (CEC) (IEEE)
pp 703–707
[6] Zhong W, Liu J, Xue M and Jiao L 2004 IEEE Transactions on Systems, Man, and Cybernetics, Part B
(Cybernetics) 34 1128–1141
[7] Shahab M L, Ambarwati T J, Soetrinos S and Irawan M I 2019 International Journal of Computing Science
and Applied Mathematics 5 61–65
[8] Shahab M L, Utomo D B and Irawan M I 2016 Journal of Theoretical & Applied Information Technology 87
[9] Chowdhury B and Garai G 2017 Genomics 109 419–431
[10] Priya K, Babu T S, Balasubramanian K, Kumar K S and Rajasekar N 2015 Sustainable Energy technologies
and assessments 12 46–52
[11] Idriess M A J, Ramchoum H, Ghanou Y and Ettaouil M 2016 2016 3rd International Conference on Logistics
Operations Management (GOL) (IEEE) pp 1–4
Figure 4. The function/fitness values of the best chromosomes of the first 10 genetic algorithm generations for (a) Sphere function, (b) Schumer Steiglitz function, (c) Quintic function, (d) Step function, and (e) Alpine function with 100000 variables.