ISSN: 2223-9553
Volume 1, Issue 3, November 2011

EFFECT OF CROSSOVER OPERATOR ON RUNNING TIME FOR SOLVING 10 NUMERIC OPTIMIZATION PROBLEMS WITH BINARY GENETIC ALGORITHM

Syariful Alim
Bhayangkara Surabaya University,
INDONESIA.
syalihbara@gmail.com ,

Retantyo Wardoyo
Gajah Mada University,
INDONESIA.
rw@ugm.ac.id

ABSTRACT

Binary genetics algorithm(BGA) constitutes technical optimization based on species evolution which actually confines natural selection that is crossover and mutation for survival. It draws an immense attention due to its capability of technical optimization or problem solving for numeric problems, especially in its development of crossover method. The research is aimed to design and create software to solve 10 Numeric Optimization Problems (NOP) through 10 crossover operator application of binary genetics algorithm. Crossover probability and mutation were employed randomly and population was limited to 20 populations and 36 chromosome each generation. Application, then, run 1000 generations each 100 times which provided 100 databases.The research comes into result that Crossover Shuffle and Uniform Crossover Operators are minimum value of 6 NOP, Precedence Preservative; Two Point Crossover is minimum value of 5 NOP, Multi Pont Crossover is minimum value of 4 NOP, One Bit Adaptation, Three Parents, One Point and Partially Matched Crossover is minimum value of 3 NOP. One running time of all operators for 1000 generations was around 15-16 minutes.

Keywords: technical optimization, Numeric Optimization Problem (NOP), crossover, binary genetic algorithm, running time

INTRODUCTION

Genetic algorithm is a heuristic search algorithm based on the mechanism of evolution of living things. Genetic algorithm was first introduced by John H. Holland in his book "Adaptation in Natural and Artificial Systems" in 1975. Holland presented the genetic algorithm with evolutionary theory in biology, particularly Darwinian theory of evolution. The basic concept of the genetic algorithm consists of the process of natural selection, crossover and mutation (Michalewicz,1996). Search techniques in genetic algorithm solutions can be conducted simultaneously on a group of solutions known as population. The population consists of several individuals, known as chromosomes, which is symbolized by a series of binary numbers.

With the process of evolution, the chromosomes are iterated to produce the best generation. To determine the best generation we used a measuring tool called the fitness function. The term chromosome in genetic algorithm refers to the pool of candidates / candidate solution of a specific problem which is often represented as a string of bits. The gene is a single bit or a block that consists of contiguous bits that represent a certain element of candidate solutions. Crossover is the exchange of genetic material between chromosomes of the parent. Mutation is the swap / changes a bit in certain positions are done randomly with the other bits, ie 0 to 1 or vice versa. Each generation will produce new chromosomes are formed from the previous generation using reproduction operators (reproduction) that interbreed (crossover) and mutation (mutation). Fitness

values in a chromosome will indicate the quality of the chromosomes in the population. The next generation is known as a child(offspring) are formed from the combination of two previous generations of chromosomes that act as a parent (parent) using crossbreeding operator (crossover).

The process of determining crossover once in getting the right solution or the best chromosome compared the process of mutation. This can be seen from the development of some methods more than the crossover mutation methods. Thus, if the crossover operator in a given population the right then the execution time (running time) to get the solution will be faster and lead to the best solution.

MATERIALS AND METHODS

The concept of Genetic Algorithm was first introduced by John Holland in 1978. The purpose of this concept is to apply what happens in the process of reproduction in living things in the universe into the process of finding the solution of some problems of mathematics and informatics. Genetic algorithms viewed as an abstract form of the evolution of living things.

This algorithm contains a sequential procedure to process a new population to other populations. This algorithm uses a process of natural selection inspired from the theory of genetics, with some operators, namely: cross-mating (crossover) and mutation (Sivanandam & Deepa, 2008). In general, flowchart and the cycle of Genetic Algorithm is shown figure 1 and 2.

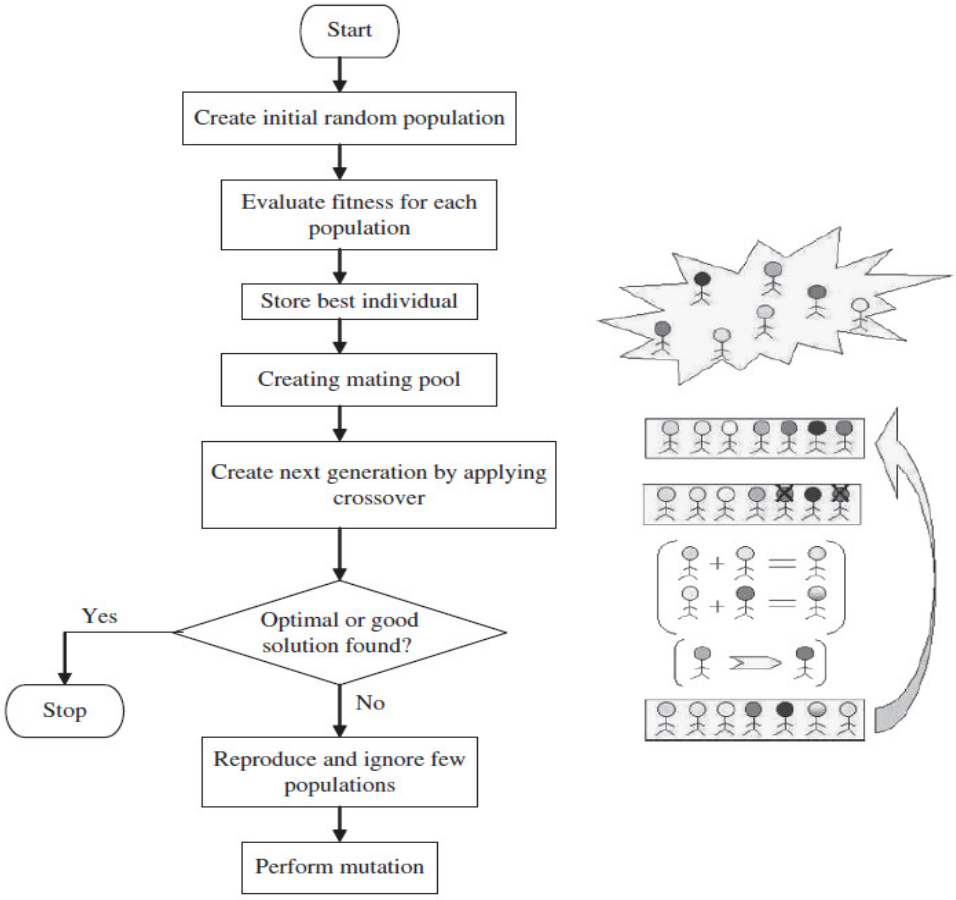

Figure 1. GA Flowchart

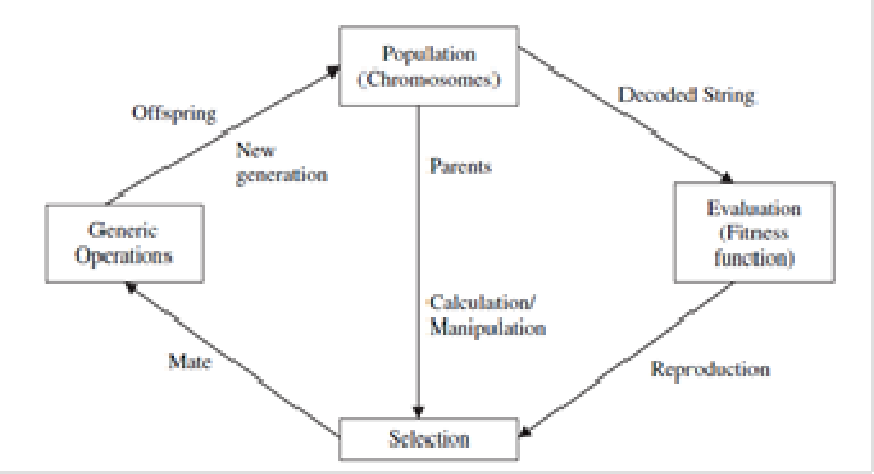

Figure 2. GA Cycle

The main elements of the genetic algorithm are individuals and populations. Individual is a candidate solution while the population is a group of individuals or a few candidate solutions. (Sivanandam & Deepa, 2008).

Individual

Each individual is a candidate / candidate solutions. Each individual group forms two forms of the solution as follows:

1. Chromosome, which is the basis of genetic information (genotype)
2. Phenotype, which is a model of a chromosome.

A chromosome is divided into genes. A gene is a representation of a candidate solution. Each solution corresponds to a gene on the chromosome. One solution is represented by a single chromosome that is different from others so in a chromosome can be sure to have information out the solutions that they represent. Examples of representation chromosome and phenotype is shown in Figure 3.

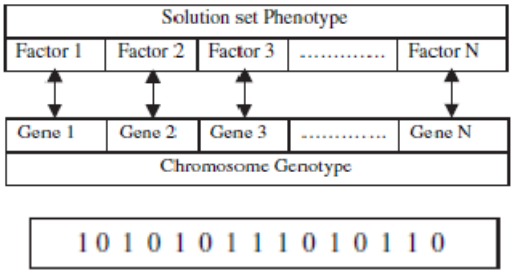

Figure 3. Representation Of Chromosome and Phenotype

Genes are the basic instructions for building a genetic algorithm. A chromosome is a series of genes. Genes can describe the solution of problems but not the solution itself. A gene consists of bits of binary string represented by the interval of the lower limit to upper limit. Coverage limits can be divided into a number of intervals indicated by the bit string in the genes. A bit string of length n can represent the value 0 to (2n - 1). Gene binary representation can be viewed as in figure 4:

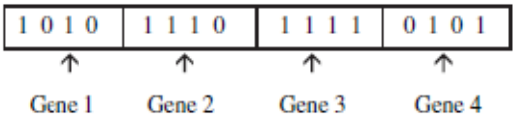

Figure 4. Representation of Binary Gen

ISSN: 2223-9553
Volume 1, Issue 3, November 2011

The structure of each gene is shown in the record that maps a genotype to phenotype. Mapping is important to change the set of solutions of the model into a form that can be used by several operatorsof Genetic Algorithms.

Fitness

Individual fitness in genetic algorithm is the value of the objective function. Chromosomes must be encoded and the objective function must be evaluated before a head count of fitness,. Fitness is not only indicate the best solution, but also shows the proximity of the chromosome with the optimal point.

Fitness function in a multi-criteria optimization will become more difficult to determine. What to do if a better solution for a specific criterion but bad for the other criteria? Actually the main problem of definition is the best solution, not how to implement a genetic algorithm to solve it. If a fitness function can be obtained from a combination of various criteria can give good results, then these criteria can be combined with a consistent way.

Population

A population is a collection of individuals who tested and determined by the phenotype parameters plus other information contained in the search process. Two important aspects of the populations used in genetic algorithm are:

1. Population of early generation
2. Population size

Population size will depend on the complexity of the problem and population size in the initialization process randomly or established beforehand. Sometimes it happens that a random population initialization gives better results before the process of genetic algorithm. Problems using chromosome in the form of binary code, each bit is initialized to 1 or 0 randomly.

Ideally, the population must first have a set of genes as large as possible so as to explore the entire candidate solution or lead to a solution. All possible solutions must be raised within the population to obtain this, the initial population, in most cases, chosen at random. Not rule out the possibility, sometimes can be used heuristic approaches to see the condition of the initial population. So that the fitness of the population will have high value and help the genetic algorithm to find solutions more quickly. The drawback is it takes a collection of genes that is large enough to be able to ensure this. If the population does not have considerable differences, the algorithm will only explore the data space is small and will not find a solution as expected.

Population size also causes other problems. The larger the population, it is increasingly easy to explore the search environment. Computing can be said to be complete when each individual has a slightly different and further development is not possible by the mutation. So it can be concluded, that the large population is very useful, but it takes time, costs and more memory to perform computing on large populations. Example representation of a population can be seen in Figure 5.

Population	Chromosome 1	1 1 1 0 0 0 1 0
	Chromosome 2	0 1 1 1 1 0 1 1
	Chromosome 3	1 0 1 0 1 0 1 0
	Chromosome 4	1 1 0 0 1 1 0 0

Figure 5. Representation of Population

Genetic Algorithms Data Structures

The main data structure of the algorithm is genetic chromosome, phenotype, the value of the objective function and fitness values. The entire population of chromosomes can be stored in an array of individuals and the length of genotype. Design variable phenotype obtained from the mapping chromosome are also stored in an array. Mapping depends on the encoding scheme used. Objective function value can be either a scalar or vector value, and as important as fitness values. Fitness value is taken of the objective function by using scaling or ranking functions and can be stored as vectors.

Search Strategy

The search process consists of initialization population and continued with the generation of new individuals to obtain the desired end state. There are several purposes for the search process, including the value of fitness is to find minimum / maximum. It can not be ascertained because it depends on the type of model genetic algorithms that can be used. Always there is the possibility of the next iteration in the search to produce better solutions. In some cases, the search process will run for years and still not produce a better solution than the first iteration.

Another goal is a faster process of convergence. When the objective function becomes very expensive to run, you may prefer a faster convergence, but there is the possibility of local convergence and the growing number optimum point.Another aim is to produce a diverse set of best solutions. When the solution space consists of some optimum point, which is equal in value fitness, it is better to choose between the optimum points.

Breeding

The process of breeding (breeding) is the core of genetic algorithm. In this process, the query produces a new individual with hopes, the new individual is a better individual. Breeding stage consists of three steps:

a) Selection of the parent / parents (Selection)
b) Crossover to generate new individuals (the Offspring or children)
c) Substitution of the old individuals in the population with new individuals (Replacement)

HOW TO RESEARCH

Representation of cases that will be conducted in this study was to test the software that is made by using 10 and 10 function test to determine the crossover operator running time or execution time required for 1000 generations.

Function Test

Function or numerical equations used in the implementation of the software is divided into two types, namely Unimodal and Multimodal. The unimodal equation is a numerical equation has only one global solution. While a function is said to multimodal if it has two solutions or more local optimum value. All functions are implemented using two numerical variables.

Unimodal function

Unimodal functions used in the implementation of genetic algorithm systems are:
1. Bohachevsky 01
This function has a global minimum value = 0 at (x1, x2) = 0
$y = x_1^2 + 2\ X_2^2\text{-}0.3\cos(3 * pi * x_1)\text{-}0.4\cos(4 * pi * x2) + 0.7$ (1)
2. Bohachevsky 02
This function has a global minimum value = 0 at (x1, x2) = 0
$y = x_1^2 + 2\ X_2^2 2\text{-}0.3 * \cos(3 * pi * x_1) * \cos(4 * pi * x_2) + 0.3$ (2)
3. Bohachevsky 03

This function has a global minimum value = 0 at (x1, x2) = 0
$y = x_1\ x_2\ 2 + 2 * 2 - 0.3 * \cos(3 * pi * x_1 + 4 * pi * x_2) + 0.3$ (3)
4. Michalewicz
This function has a global minimum value = 5.21 at (x1, x2) = 0
$y = 5.21 + x_1 \sin(4Pi * x1) + x_2 \sin(20pi * x2)$ (4)
5. Booth
This function has a global minimum value = 0 at (x1, x2) = (1,3)
$y = (x_1 + 2 * x_2 - 7)^2 + (2 * x_1 + -5)^2$ (5)

Multimodal Function
Multimodal functions used in the implementation of genetic algorithm systems are:
1. Salomon
This function has a global minimum value = 0 at (x1, x2) = 0
$y = -\cos(2pi\ (x_1^2 + x_2^2)^{0.5}) + 0.1 * (x_1^2 + X_2^2)^{0.5} + 1$ (6)
2. Michalewicz
This function has a global minimum value = 0 at (x1, x2) = 0
$y = -x_1 * \sin(\sin x_1^2/pi)^2 + x^2 * \sin(\sin x_2^2/pi)^2)$ (7)
3. Griewangk
This function has a global minimum value = 1 at (x1, x2) = 0
$y = ((x_1^2 + x_2^2) / 4000) - (\cos(x_1 * pi) * \cos(x_2 * pi / 1{,}414)) + 1$ (8)
4. Schaffer
This function has a global minimum value = 0 at (x1, x2) = 0
$y = ((x_1^2 + x_2^2)^{0.25}) * [(\sin 2\ (x_1^2 + x_2^2)^{0.1)} + 1]$ (9)
5. Rastrigin
This function has a global minimum value = 0 at (x1, x2) = 0
$y = 20 + (x_1^2 - 10 * \cos(2 * pi * x_1)) + (x_2^2 - 10 * \cos(2 * pi * x_2))$ (10)

All the functions of both unimodal and multimodal above will look for the minimum value in the boundary region: $-5 \leq 5$ and $-5 \leq x1 \leq x2 \leq 5$. The length of the chromosome is used as much as 36 bits. The first 18 bits to 18 bits x1 and x2 for the second.

Design of Data Flow Diagrams (DFD)

A program designed is a software application that implements a binary genetic algorithm to solve optimization problems numerically. The program receives input from the user of the required parameters through the user interface has been provided. And the program periodically displays the required solution to the user. Figure 6is a diagram of the context of the intended application.

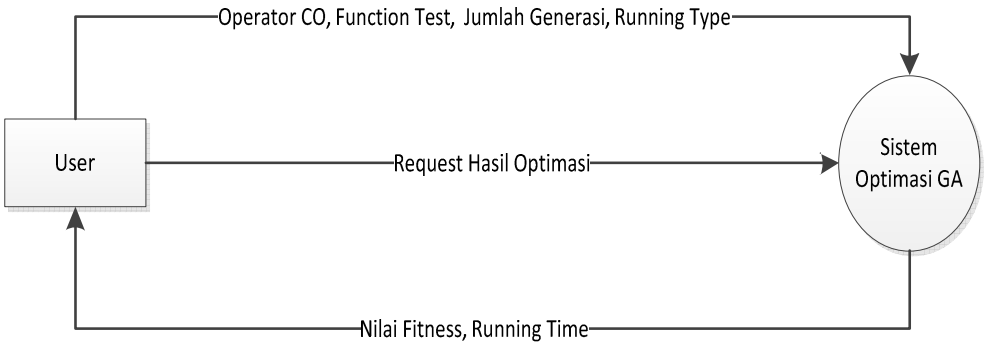

Figure 6. Context Diagram

Main Software Functions

ISSN: 2223-9553
Volume 1, Issue 3, November 2011

Referring to the results of the study on genetic algorithm and its application in several previous numerical equation, the program needs to implement the main functions as follows:

1. The program is made capable of receiving input parameters from the user's genetic algorithm manually or automatically
2. Program capable of implementing the genetic operators (crossover and mutation).
3. Especially for the crossover operator, the program can perform the crossover process execution according to user choice.
4. The program is able to implement the selection operator by generating a random number between 0-1 and then compare with the value cumulative fitness (QF)
5. Program capable of displaying the results of execution of each 1000 generation and select the best solution in each generation.

The main functions of the program can be viewed through the DFD in Figure 7 below:

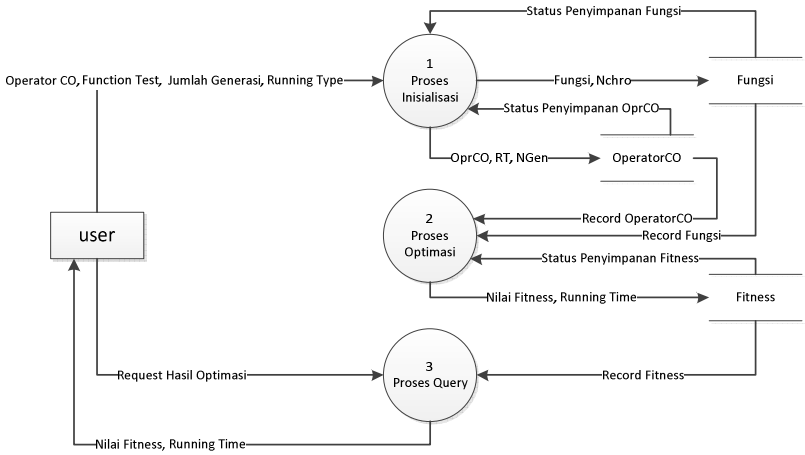

Figure 7. DFD Level 1 – GA Application

Generation Population / Initialization

Generation process starts from generating a population of chromosomes based on the NOP are selected by the user. This election form has been provided by the system. Which contains a function operator crossover, running the type and number of generations. Each gene on the chromosome represents the variables x1 and x2. Generation of chromosomes will stop until the population size of 20 chromosomes are met.

Validation is required for chromosome chromosomes that are members of the population is a valid chromosome. Population generation process can be seen through the DFD in Figure 8. and 9:

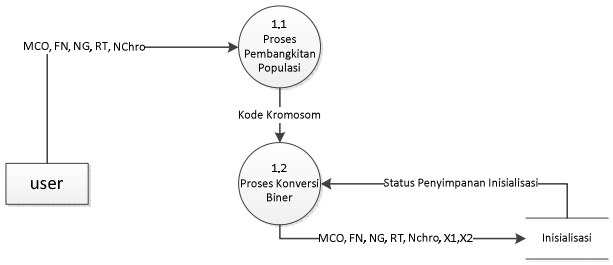

Figure 8. DFD Level 2 – Initialization

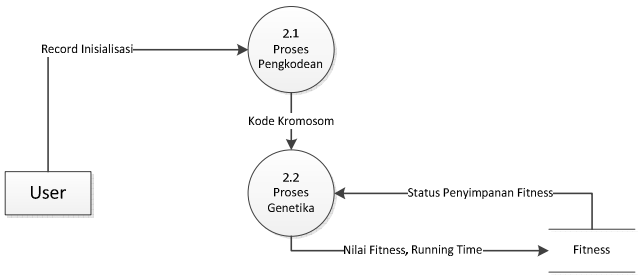

Figure 9. DFD Level 2 – Conversion Code

Selection

The selection process aimed at selecting the chromosomes are entitled to live and become a member of the next population. Chromosome with the best fitness value is expected to pass into the next member of the chromosomes while the poor will die. Selection mechanism chosen is Roulette Wheel Selection. Step-by-step selection with roulette wheel are as follows:

1. Generating random number between 0-1
2. Compare it with the q_k
3. Choosing the chromosome if the random number generated $<q_k$

Fitness evaluation

The evaluation process carried out on the fitness or an objective function for each chromosome. In this study, the objective function is to calculate the value of a NOP. The evaluation process will result in the values of fitness of each chromosome with best fitness values. The evaluation process can be seen on the DFD in Figure 10:

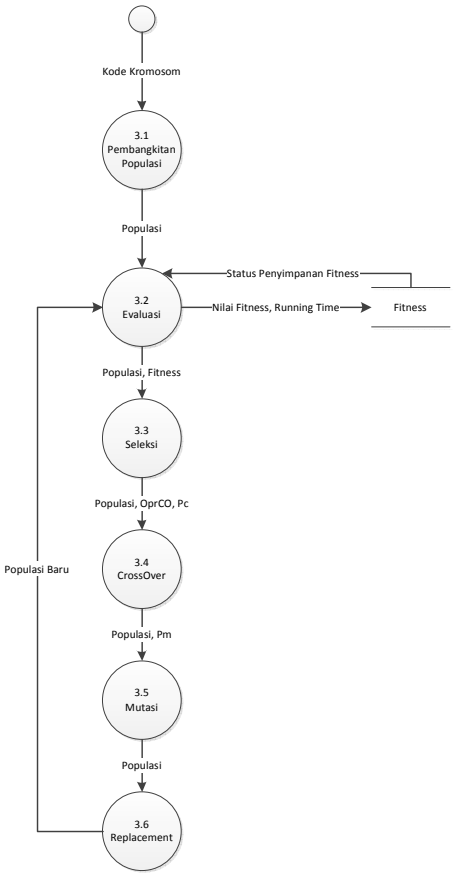

Figure 10. DFD Level 2 - Evaluation

Crossover

The process aims to crossover between chromosomes selected with a particular chromosome. The selection of chromosomes based on a random number between 0-1. As explained in the Roulette Wheel. The process of crossover on binary genetic algorithm more streamlined than other genetic algorithms. Algorithms some crossover of this study refer to the book Genetic Algorithms Reference (Gwiazda, 2006). The algorithm of 10 crossover in this study are : One Point Crossover (OPC), Two Point Crossover (TPC), Multi-Point Crossover (MPC), Uniform Crossover (UFC), Three Parent Crossover (3pc), precedence Preservative Crossover (PPC), Shuffle Crossover (SHC), One Bit Adaptation, Crossover (1BAC), Partially Matched Crossover (PMC), Masked Crossover.

RESULTS AND DISCUSSION

Effect of Total Generation

The number of generations used is 1000 generations. The resulting data is data plus 20,000 20x10x10x1000 chromosome initialization data. So that will produce 2.02 million the total amount of generation data. Although some of the results of previous studies the number of generations is not so influential on the results of running that due to the randomization (generation of random numbers by the program) to the chromosomes that will crossover and selected.

Effect of Crossover Operator on Fitness Value

From some crossover used in this study are that managed to find the minimum value of a certain numerical equations and some that have not been obtained. It is possible for several reasons:

1. Much less number of generations.
2. Selection of random chromosomes, chromosomes could be better than the other chromosomes was selected to do crossover.
3. The process produces chromosome crossover performed worse than the previous chromosomes.

Effect of Crossover Operator on Running Time

Some of the crossover can be found with a fast time, although the whole process (if it is run in a 1000 generation) runs a bit long. The length of time for 1000 generations because there are some steps that are not there / done by the other crossover operators. The other thing is because randomized selection of chromosomes. The selection of chromosomes is done by random selection of chromosomes that are at risk well worth it for crossover and mutated. The selected chromosomes are then contributed to a worse value than ever before.

The results of the execution time was found that all the resulting database has a time between 15-17 minutes for one run. This means that the total time required to generate a database with 1000 generations, 10 and 10 crossover operator function tests were within the 15 * 10 * 100-17 * 10 * 100 = 15000-17000 minute or about 25-28 hours. The time difference between the crossover operators are not so significant to be taken into account because only about 1-2 minutes.

MPC crossover operator is the operator of crossover fastest to complete 10 NOP and take as much as 155.56 minutes for 1000 generations. PPC and crossover operators are crossover operator longest to complete 10 NOP is 163.2 minutes. Shuffle and Uniform Crossover operator is able to find six of the tested function test. While other operators range from 3-5 NOP.

CONCLUSION

Based on the results of running the program from 10operatorscrossover, 10 equations and 1000generation that was tested can be concluded that:

1. Uniform Crossover shuffle can get a minimum value of 6NOP

2. Precedence Preservative and Two Point Crossover obtain the minimum value of 5NOP
3. Masked and Multi-Point Crossover obtain the minimum value of 4equationsNOP
4. One Bit Adaption, Three Parent, One Point and Partially Matched Crossover obtain the minimum value of 3NOP
5. Multi-Point Crossover operator is the crossover operator the fastest to complete 10NOPand takes155.56minutes.
6. Precedence Preservative Crossover operator is the operator of crossover longest to complete 10NOPand takes156.62minutes.

Authors suggest for the development of similar research for the better is as follows:

a. The number of chromosomes over generations and propagated to the fitness value is more precise than previously possible. Crossover and mutation probability is set before hand.
b. Test and develop the use of Multi-Point and Uniform Crossover Shuffle to some other equations.
c. Using parallel computing and computational software combined with better database.

REFERENCES

Michalewizs, Zbigniew, (1996). *Genetic Algorithm + Data Structures = Evolution Programs* – 3rd rev. and extended edition, Springer-Verlag Berlin Heidelberg New York.

Djunaedi Kosasih & Rinaldo (1997). Analisis Aplikasi Algoritma Genetika Untuk Pencarian Nilai Fungsi Maksimum, *Jurnal FTSP ITB.*

Fogel, David B, (1998). *Using Fitness Distributions to Design More Efficient Evolutionary Computations*, Natural Selection, Inc.

HuawenXu, (1999). *Comparison of Genetic Operators on a General Genetic Algorithm Package*, Shanghai Jiao Tong University, Shanghai, China.

Ishibuci, Hisao,(2005). *Effect of Crossover Operations on the Performance of EMO Algorithms*, Dagstuhl Seminar Proceedings 04461, Osaka Prefecture University, Japan.

Dominic Gwiazda, Thomas, (2006*). Genetic Algorithm Reference* – Volume I Cross-Over for Single Objective Numerical Optimization Problems, Published Thoma sGwiazda Ebook – Lomiank – Poland.

Sanjoyo, (2006). *Penaksiran Model Fungsi Cobb-Douglas* dan CES denganmetode Algoritm a Genetika, Jakarta.

Kosasih, (2007). Analisis Algoritma Genetika Dalam Proses Desain Struktur Perkerasan, *Jurnal FTSP ITB, Bandung.*

Sivanandam& Deepa, S.N., (2008). *Introduction to Genetic Algorithms*, Springer-Verlaag Berlin Heidelberg

Al Hajri, Mohammad Tamimi, (2009), Assessment of Genetic Algorithm Selection, Crossover and Mutation Techniques in Reactive Power Optimization, *IEEE Journal.*

Garg, Poonam, 2009, A Comparison between Memetic algorithm and Genetic algorithm for the cryptanalysis of Simplified Data Encryption Standard algorithm, IJNSA Journal