Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 7, July 2016, pg.425 – 429

The Role of Crossover on Optimization of a Function Problem Using Genetic Algorithms

Wafaa Mustafa Hameed

Computer Science, Cihan University \ Campus \ Sulaimaniya, Iraq
afafa_ju223@yahoo.com

Abstract— This work aims at studying the behaviour of different types of crossover operators in the performance of genetic algorithm. This paper accumulated most of the types of crossover operators these types are implemented on Optimization of a function problem. From the comparison study of One-Point crossover and other kinds depending on the same parameters which Michalewicz [1] used, Pc=0.25, Pm=0.01, Pop_size=20 NG=1000 the experimental results Show that the Shuffle crossover was the best to be applied For the optimization of a function followed by Simplex and One -Point crossover operator.

Keywords— Crossover, Genetic algorithm, probability of crossover, probability of mutation, population size, number of generation

I. INTRODUCTION

In this paper I will try to study the effect of different types of crossover operators on certain known problem, optimization of a function. This problem is chosen since it own a high complexity (the size and the shape of the search space), which, cannot be solved using traditional known searches, like exhaustive search method.

II. PREVIOUS STUDIES

Huang C. [2] investigates the role of crossover in the immune system model based GA systems with respect to discovering multiple peaks and maintaining sub populations. This work has been concentrating on the study of the role one-point crossover.

III. USE GENETIC ALGORITHMS IN OPTIMIZATION OF A FUNCTION

A. Problem Definition

There is a large class of interesting problems for which no reasonably fast algorithms have been developed. Given such a hard optimization problem it is often possible to find an efficient algorithm whose solution is approximately optimal. We discuss the basic features of a GA for optimization of a simple function [3]

$f(x_1,x_2) = 21.5 + x_1 . \sin(4\pi x_1) + x_2 . \sin(20\pi x_2)$

where $-3.0 \le x_1 \le 12.1$ and $4.1 \le x_2 \le 5.8$.

Since x_1, x_2 are real numbers, this implies that the search space can be huge and that traditional methods can fail to find the optimal solution

B. Problem Representation

To apply the GA for maximizing $f(x)$, a genetic representation of solution to the problem must be appropriately chosen first. The simple GA uses the binary representation in which each point is described by a chromosome vector coded as a ***binary string***. We use a binary vector as a chromosome to represent real values of the variable x, the length of the vector depends on the required precision, which in this example, is six places after the decimal point.

The domain of the variable x_1 has length 15.1; the precision requirement implies that the range [-3.0,12.1] should be divided into, at least 15.1×10000 equal size ranges. This means that 18 bits are required as the first part of the chromosome:

$2^{17} < 151000 < 2^{18}$

The domain of the variable x_2 has length 1.7; the precision requirement implies that the range [4.1,5.8] should divided into, at least ,1.7×10000 equal size ranges. This means that 15 bits are required as the second part of the chromosome:

$2^{14} < 17000 < 2^{15}$

The total length of a chromosome (solution vector) is then m=18+15=33 bits; the first 18 bits code x_1 and remaining 15 bits (19–33) code x_2.

Let us consider an example chromosome:

(010001001011010000111110010100010)

corresponds to (x_1,x_2) = (1.052426,5.755330). The fitness value for this chromosome is:

f (1.052426,5.755330) = 20.252640

C. Initial Population

To optimize the function f using genetic algorithm, I create a population of pop_size chromosomes. All 33 bits in all chromosomes are initialized randomly.

1-4 Evaluation function

Evaluation function for binary vector v is equivalent, the function f:

$eval(v) = f(x)$,

where the chromosome v represents the real value x.

During the evaluation phase, we decode each chromosome and calculate the fitness function from(x_1,x_2) values just decode.

For example, three chromosomes:

v_1 = (10011010000000111111110100110111111),

v_2 = (11100010010011011100101010011010),

v_3 = (00001000011001000000101011101101),

corresponding to values x_1 and x_2 respectively. Consequently, the evaluation function would rate them as follows:

$eval(v_1) = f$(6.084492,5.652242) = 26.0196001

$eval(v_2) = f$(10.348434,4.380264) = 7.580015

$eval(v_3) = f$(-2.516603,4.390381) = 19.526329

Clearly, the chromosome v_1 is the best of the three chromosomes, since its evaluation returns the highest value.

D. Genetic Operators

1) Selection Operator

In this selection part of the GA, the survivor set is chosen so as to keep not only the fittest points but also some unvourable points. Roulette wheel sums up the fitness' of all individuals and calculates each individual percentage of the total fitness.

The percentage of the total fitness is then used as the probabilities to select N individuals from the set population and copy them into the set selected-parents.

2) The Crossover Operator

Individuals from the set selected-parents are mated to generate offsprings for the next generation. The two parents generate two offsprings using a crossover operation. For example, to illustrate the crossover operator on chromosome with a crossover probability Pc, I generate random integer number pos from the range 1...32. The number *pos* indicate the position of the crossing point. The pair of chromosomes are:

v_1 = (10011010000000111111110100110111111),

v_3 = (00001000011001000000101011101101),

and the generated number *pos*=9. These chromosomes are cut after the 9th bit and replaced by a pair of the offspring:

the two resulting offspring are:

v_1'= (10011010001100100000101011101101),

v_3'= (00001000000000111111110100110111111),

these offspring evaluate to
$f(v_1')$=29.741

3) Mutation Operator

Mutation is a random change of one or more genes (positions in a chromosome) with a probability equal to the mutation rate Pm a gene is changed/swapped, i.e 0→1 and 1→0. The probability for a mutation is usually kept small. Assume that the fifth gene from the v_3 chromosome was selected for a mutation. Since the fifth gene in this chromosome is 1, it would be flipped into 0. So the chromosome v_3 after this mutation would be:
$_3$'= (00000000000000011111110100110111111).

E. Genetic Parameters

For this particular problem, Michalewicz [1] used the following parameters: population size pop_size=20, probability of crossover Pc=0.25, probability of mutation Pm =0.01.

Experimental Results

In table I we provide the generation number for which we noted improvement in the evaluation function, together with the value of the function. The best chromosome after (200) generations was: v_{max} = (11111010100111111111110101001111)
Which corresponds to a value (x_1,x_2)=(12.002307,5.712009), and $f(v_{max})$ is slightly larger than 38.629.

TABLE I
Results Of 200 Generations For Optimization Of A Function Problem.

Generation number	Evolution function
0	36.290817
1	37.338592
6	37.758549
15	37.759151
37	37.973587
54	38.048171
73	38.208092
91	38.279362
114	38.607517
192	38.629047

For this problem, a simulation has been constructed in order to apply the GA, using the crossover parameters mentioned above, the following results are be obtained:
v_{max} = (01101111111111111111011011111111111)
Which corresponds to a value (x_1,x_2)=(12.0995,5.7995), and $f(v_{max})$=38.761.

F. Optimization of a Function Problem

To see the effect of using different types of crossover operators on this problem, Michalewicz [1] used the One-Point crossover depending on the following parameters: Pc=0.25, Pm=0.01, Pop_size=20 NG=1000. Table II describes the comparison study of the iterations results between the above crossover and the other kinds which are implemented on this problem. In addition, the table shows the average of iterations and finesses results of (10) runs.

Table II
Comparison Study Of One-Point Crossover And Other Kinds.

Crossover	NG	Fitness
1-P	361	38.724
2-P	455	38.663
M-P	395	38.716
Un	455	38.649
SIMP	351	38.750
SHUF	310	38.761

From table II, the average iterations results show that the Shuffle[4], (followed by Simplex and One-Point)[5],[3] crossover operator is the best, because it makes a randomly shuffle in both parents before they are exchange. The Uniform [3] and Two-Point crossover[6] operators are the worst, because there is a destruction in the building blocks (good genes).

IV. CONCLUSION & FUTURE SCOPE

This research finds that: For the optimization of a function, the Shuffle crossover was the best to be applied. Our future work will extend to the study for other kinds of crossover operators.

REFERENCES

[1]. Michalewicz, Z., "Genetic Algorithm + Data Structure = Evolution Programs", 3rd Revised and Extended Edition, Springer- Verlag Berlin Heidelberg, New York, 1996.
[2]. Huang C., "The Role of crossover in an immunity based Genetic Algorithm for Multi Model Function Optimization", proceeding of 2004, Genetic and Evolutionary Computation Conference,2004.
[3]. Mühlenbein, H. and Schlierkamp D., "Predictive Models for the Breeder Genetic Algorithms, Evolutionary Computation", Vol.1, No.1, 1993.
[4]. Rudolf Kruse, Christian Borgelt, Frank Klawonn, Christian Moewes, Matthias Steinbrecher "Computational Intelligence: A Methodological Introduction" , Springer Science & Business Media, Mar 27, 2013
[5]. Carlos A. Coello-Coello, Julie Greensmith, Natalio Krasnogor, Pietro Liò, Giuseppe Nicosia, Mario Pavone , "Artificial Immune Systems: 11th International Conference", ICARIS 2012 Taormina, Italy, August 28-31,2012 proceeding.
[6].S.N. Sivanandam, S. N. Deepa ,Introduction to Genetic Algorithms , Springer- Verlag Berlin Heidelberg, New York, 2008.

