Available online at www.worldscientificnews.com

World Scientific News

WSN 16 (2015) 53-72 EISSN 2392-219

# Find Optimal Solution for Eggcrate Function Based on Objective Function

**Lubna Zaghlul Bashir**

University of Technology, Baghdad, Iraq

E-mail address: lubna_zaghlul@yahoo.com

**ABSTRACT**

In modern compilers and optimizers, the set of possible optimizations is usually very large. In general, for a given application and workload, optimization options do not accrue toward ultimate performance. To avoid selection complexity, users tend to use standard combination options, in general, however, these standard options are not the optimal set for a specific application executing its representative workload. Genetic algorithm developed by Goldberg was inspired by Darwin's theory of evolution which states that the survival of an organism is affected by rule "the strongest species that survives". Darwin also stated that the survival of an organism can be maintained through the process of reproduction, crossover and mutation. Darwin's concept of evolution is then adapted to computational algorithm to find solution to a problem called objective function in natural fashion. In this work test the function known as the “Eggcrate Function”; is described mathematically as: $F(X) = X1^2 + X2^2 + 25(\mathrm{Sin}^2 X1 + \mathrm{Sin}^2 X2)$ In this problem, there are two design variables with lower and upper limits of $[-2\pi , 2\pi]$. We use genetic algorithm to find optimal solution for solving this problem, Basic philosophy of genetic algorithm and its flowchart are described. Step by step numerical computation of genetic algorithm for solving the eggcrate function will be briefly explained. The results shows that the eggcrate function has a known global minimum at [0, 0] with an optimal function value of zero.

***Keywords*:** Eggcrate function; Evolutionary algorithm; Optimization; Objective function

## 1. INTRODUCTION

The Genetic Algorithm (GA) is a relatively simple heuristic algorithm that can be implemented in a straightforward manner. It can be applied to a wide variety of problems including unconstrained and constrained optimization problems, nonlinear programming, stochastic programming, and combinatorial optimization problems. It is widely used in several fields such as management decision making, data processing... Information and Financial Engineering. Because of their population approach, they have also been extended to solve other search and optimization problems efficiently, including multimodal, multi objective [1-3].

Genetic Algorithms (GAs) are a part of the evolutionary algorithms, which is a rapidly growing areas of artificial intelligence. GAs are inspired by Darwin's theory of biological evolution. By mimicking this process, genetic algorithm are able to "evolve" solutions to real world problems. Holland was the first person to put computational evolution on a firm theoretical footing. GAs are optimization algorithms based on the concepts of biological evolution and genetics. In this algorithm, the design variables are represented as genes on a chromosome. GAs feature a group of candidate individuals (which is called population) on the response surface. Through environmental selection and the genetic operators, mutation and recombination, chromosomes with better fitness are found. Natural selection guarantees that best chromosomes with better fitness will survive in the future populations. Using the recombination operator the GA combines genes from two parent chromosomes to form children (new chromosomes) that have a high probability of having better fitness than their parents. On the other hand, mutation allows new areas of the response surface to be explored. GAs offer a generation improvement in the fitness of the chromosomes and after many generations will create chromosomes containing the optimized variable settings [1-3].

## 2. OPTIMIZATION TECHNIQUES

### 2. 1. Analytical

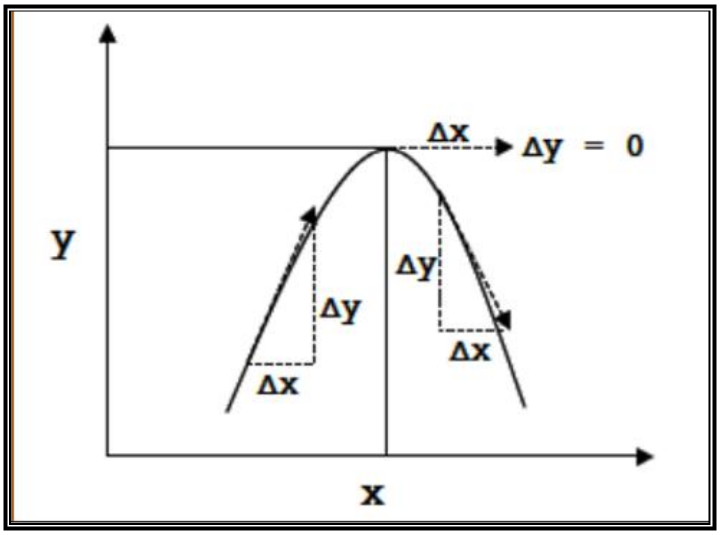


**Figure 1.** Analytical Technique.

Given y = f(x), take the derivative of w.r.t. x, set the result to zero, solve for x, Works perfectly, but only for simple, analytical Functions.as illustrated in Figure 1.

## 2. 2. Gradient-based, or hill-climbing

Given y = f(x)
- pick a point x0
- compute the gradient ∇f(x0)
- step along the gradient to obtain x1 = x0 + α ∇f(x0).
- repeat until extreme is obtained

Requires existence of derivatives, and easily gets stuck on local extreme. Figure 2 illustrate hill-climping technique.

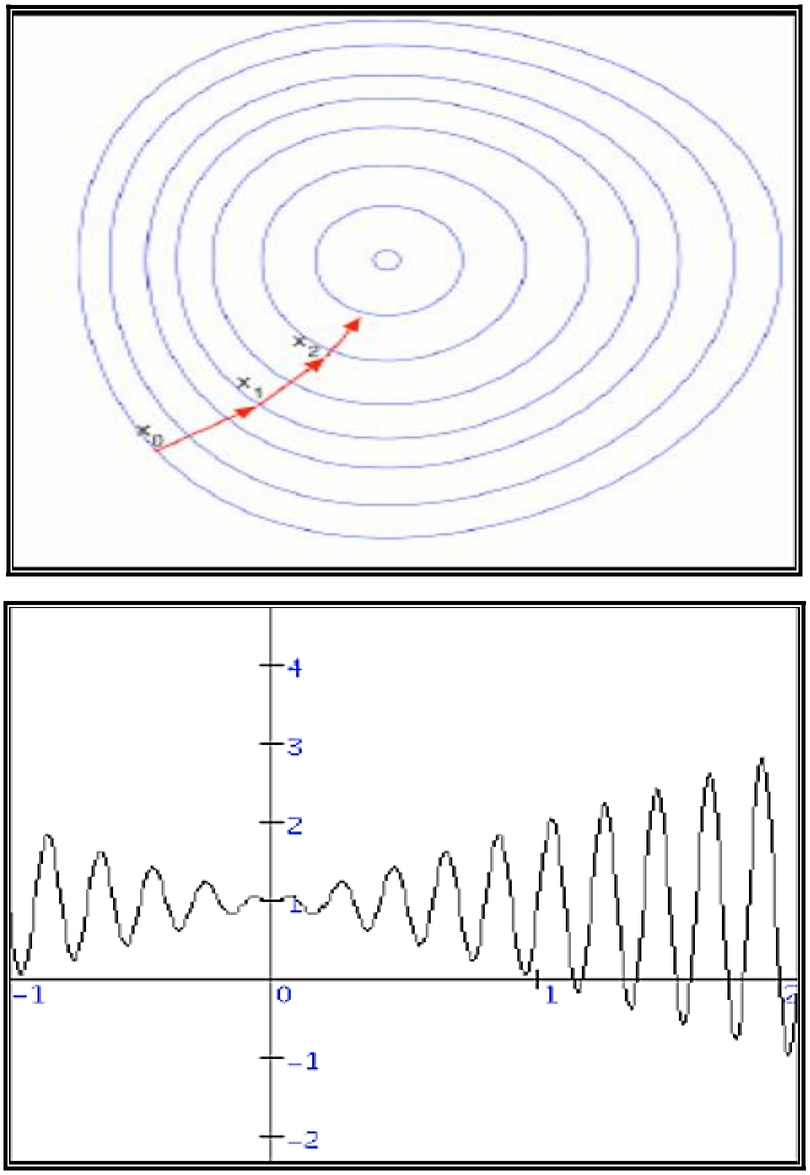


**Figure 2.** Hill- Climbing Technique.

### 2. 3. Enumerative

Test every point in the space in order.

### 2. 4. Random

Test every point in the space randomly.

### 2. 5. Genetic Algorithm (Evolutionary Computation)

- Does not require derivatives, just an evaluation function (a fitness function)
- Samples the space widely, like an enumerative or random algorithm, but more efficiently
- Can search multiple peaks in parallel, so is less hampered by local extreme than gradient-based methods
- Crossover allows the combination of useful building blocks, or schemata (mutation avoids evolutionary Dead-ends)
- Robust [4].

## 3. BASIC PHILOSOPHY

Genetic algorithm developed by Goldberg was inspired by Darwin's theory of evolution which states that the survival of an organism is affected by rule "the strongest species that survives". Darwin also stated that the survival of an organism can be maintained through the process of reproduction, crossover and mutation. Darwin's concept of evolution is then adapted to computational algorithm to find solution to a problem called objective function in natural fashion. A solution generated by genetic algorithm is called a chromosome, while collection of chromosome is referred as a population. A chromosome is composed from genes and its value can be either numerical, binary, symbols or characters depending on the problem want to be solved. These chromosomes will undergo a process called fitness function to measure the suitability of solution generated by GA with problem. Some chromosomes in population will mate through process called crossover thus producing new chromosomes named offspring which its genes composition are the combination of their parent. In a generation, a few chromosomes will also mutation in their gene. The number of chromosomes which will undergo crossover and mutation is controlled by crossover rate and mutation rate value. Chromosome in the population that will maintain for the next generation will be selected based on Darwinian evolution rule, the chromosome which has higher fitness value will have greater probability of being selected again in the next generation. After several generations, the chromosome value will converges to a certain value which is the best solution for the problem [5-7]. Figure 3. Shows chromosome representation as string of bits. And Figure 4 shows other chromosome representation [4].

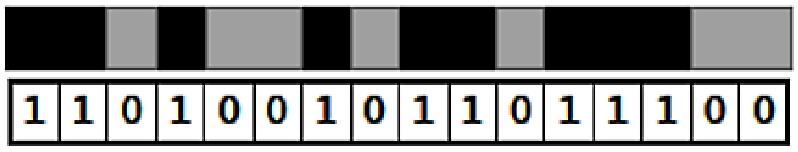


**Figure 3.** Chromosome Representation (String of Bit).

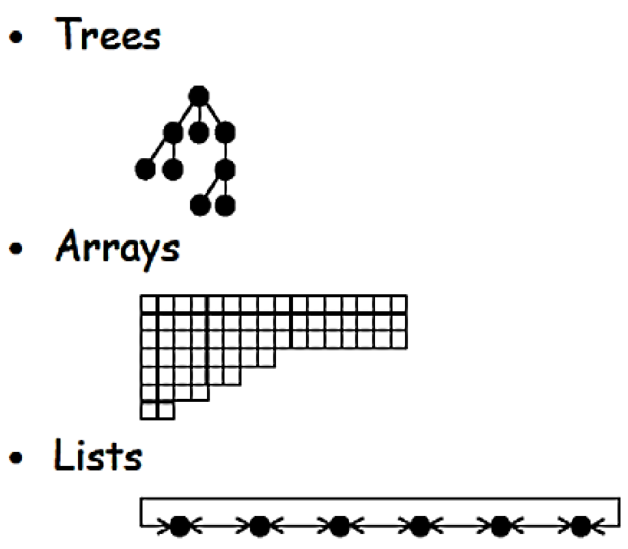


**Figure 4.** Other Bit Representation (Trees, Arrays, Lists).

## 4. GENETIC ALGORITHM OPERATORS

The primary work of the Simple Genetic Algorithm (SGA) is performed in three routines:

1. Select.
2. Crossover.
3. Mutation.

Their action is coordinate by a procedure called generation that generates a new population at each successive generation.

### 4. 1. Selection

The purpose of selection is, of course, to emphasize the fitter individuals in the population in hopes that their offspring will in turn have even higher fitness [8].

Methods of selection

- Roulette wheel selection.
- Sigma scaling techniques.
- Tournament selection.
- Ranking methods.
- Elitism.
- Boltzmann selection.
- Steady state selection [8,9].

Holland’s original GA used fitness- proportionate selection, in which the expected value of an individual (i.e., the expected number of times an individual will be selected to reproduce) is that individual’s fitness is divided by the average fitness of population. The most common method to implement this is ‘roulette wheel” sampling [8]:

## Roulette Wheel Selection

Each individual is assigned a slice of a circular "roulette wheel" the size of the slice being proportional to the individual's fitness. The wheel is spun N times, where N is the number of individuals in the population. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the next generation [11]. Figure 5 illustrate Roulette Wheel Selection [12].

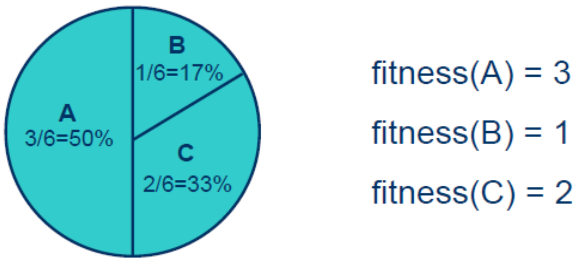


**Figure 5.** Roulette Wheel Selection

## Other section methods

- *Elitist selection*:
  - Chose only the most fit members of each generation.
- *Cut off selection*:
  - Select only those that are above a certain cut off for the target function [9,10].

## 4. 2. Crossover

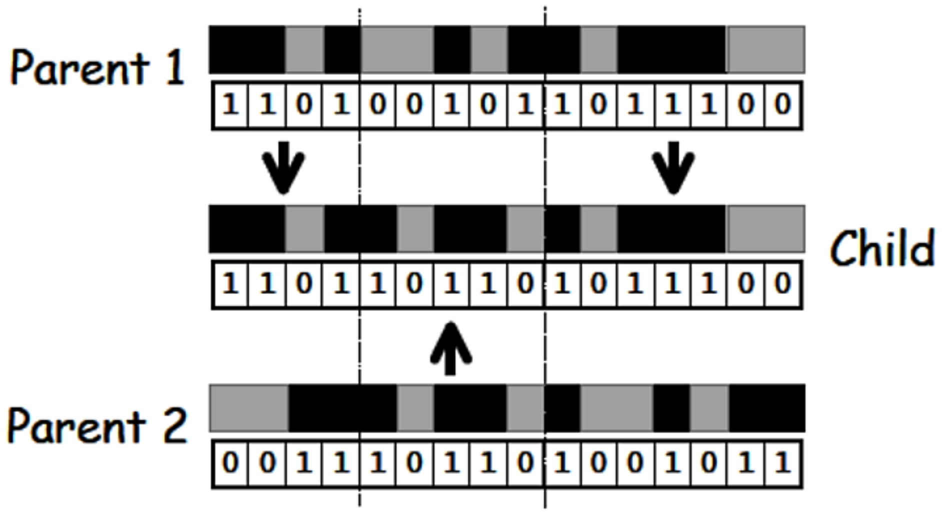


**Figure 6.** Crossover Operator.

In contrast is applied with high probability. It is a randomized yet structured operator that allows information exchange between points. Simple crossover is implemented by choosing a random point in the selected pair of strings and exchanging the sub strings defined by that point. Figure 6 shows how Crossover mixes information from two parent strings, producing offspring made up of parts from both parents. This operator which does no table lookups or backtracking, is very efficient because of its simplicity [4,5,10].

**Several possible crossover strategies**

1. Randomly select a single point for a crossover.
2. Multi point crossover. Avoids cases where genes at the beginning and end of a chromosome are always split
3. Uniform crossover. A random subset is chosen The subset is taken from parent 1 and the other bits from parent 2. Figure 7 illustrate crossover methods [9,13].

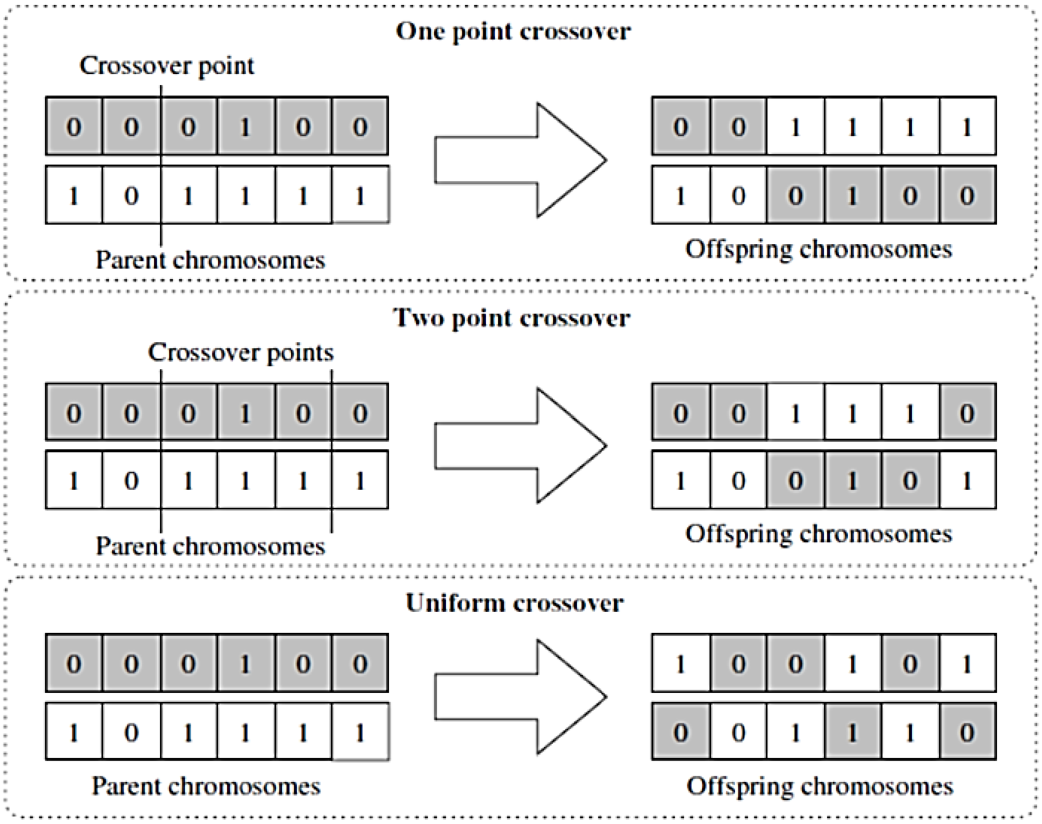


**Figure 7.** One-point, two-point, and uniform crossover methods.

### 4. 3. Mutation

As in natural systems, is a very low probability operator and just flips a specific bit. Where the bits at one or more randomly selected positions of the chromosomes are altered. The mutation process helps to overcome trapping at local maxima. Figure 8 shows the effect of the one-bit mutation operator in the binary case [4,5,10].

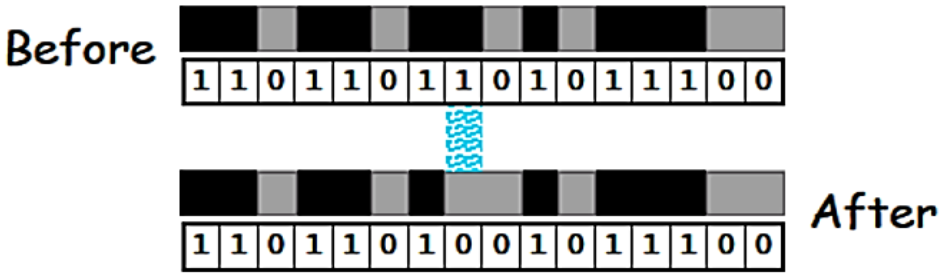


**Figure 8.** Mutation operator.

## 5. A SIMPLE GENETIC ALGORITHM

A simple GA works as follows:

1. Determine the initial population of creatures.
2. Determine the fitness of the population.
3. Reproduce the population using the fittest parents of the last generation,
4. Determine the crossover point, this can also be random,
5. Determine if mutation occurs and if so on which creature(s).
6. Repeat from step 2 with the new population until condition (X) is true. [14-17].

Can be implemented as three modules;

1. The evaluation module.
2. The population module.
3. The reproduction module.

- Initial population
- Evaluations on individuals
- Reproduction
- Choose suitable parents (by evaluation rating)
- Produce two offspring (by probability)
- Mutation
- Domain knowledge – evaluation function. [14-17].

Figure 9. Illustrates genetic algorithm diagram (for 20 generation) [14].

The simple genetic algorithm (SGA) is described by Goldberg [5] and is used here to illustrate the basic components of the GA. A pseudo-code outline of the SGA is shown in Figure. 10. The population at time t is represented by the time-dependent variable P, with the initial population of random estimates being P(0). Using this outline of a GA. [16,18,19].

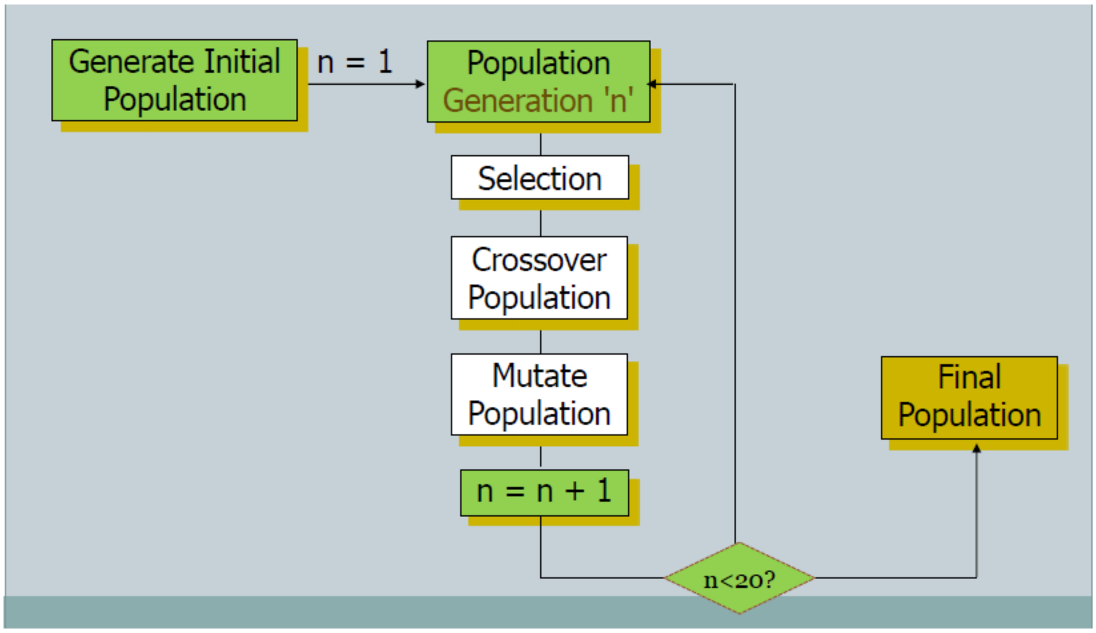
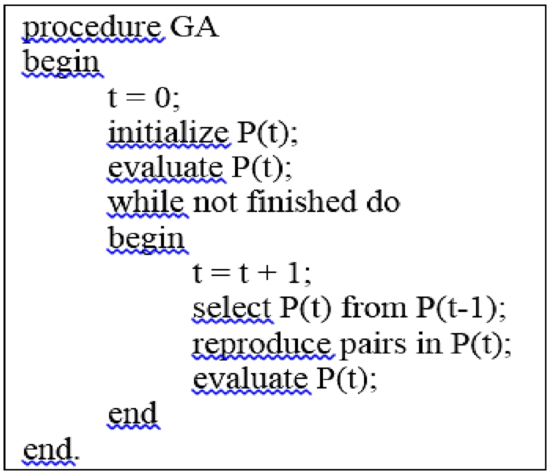


**Figure 9.** Genetic algorithm diagram (for 20 generation).



```
procedure GA
begin
          t = 0;
          initialize P(t);
          evaluate P(t);
          while not finished do
          begin
                    t = t + 1;
                    select P(t) from P(t-1);
                    reproduce pairs in P(t);
                    evaluate P(t);
          end
end.
```

**Figure 10.** Simple Genetic Algorithm.

## 6. THE EGGCRATE FUNCTION(CASE STUDY)

In this work we need to find optimal solution for solve function is known as the "Eggcrate Function"; it is described mathematically in Equation 1. In this problem, there are two design variables with lower and upper limits of [-2π, 2π].

$$\text{MINIMIZE} \quad F(X) = X1^2 + X2^2 + 25 \cdot (\text{Sin}^2 \cdot X1 + \text{Sin}^2 \cdot X2) \qquad (1)$$

### 6. 1. Initialization

We define the number of chromosomes in population are (6), then generate random value of gene (X1, X2) for 6 chromosomes. Table 1 illustrate the (6) chromosomes.

**Table 1.** The (6) Chromosomes.

| | | |
|---|---|---|
| Chromosome[1] | [X1,X2] | [30;60] |
| Chromosome[2] | [X1,X2] | [45;90] |
| Chromosome[3] | [X1,X2] | [15;180] |
| Chromosome[4] | [X1,X2] | [90;0] |
| Chromosome[5] | [X1,X2] | [135;30] |
| Chromosome[6] | [X1,X2] | [60;45] |

### 6. 2. Evaluation

To compute the objective function value for each chromosome produced in initialization step. Table 2. Illustrate the 6 objective functions.

**Table 2.** The (6) Objective Functions.

| O. F [i] | $F(X) = X1^2 + X2^2 + 25(Sin^2X1 + Sin^2X2)$ | Result | Result |
|---|---|---|---|
| O. F [1] | $= 30\text{^}2+60\text{^}2+25*(\sin^2(30)+Sin^2(60))$ | = 4524.999 | = 4525 |
| O. F [2] | $= 45\text{^}2+90\text{^}2+25*(\sin^2(45)+Sin^2(90))$ | = 10162.500 | = 10163 |
| O. F [3] | $= 15\text{^}2+180\text{^}2+25*(\sin^2(15)+Sin^2(180))$ | = 32626.674 | = 3.2627 |
| O. F [4] | $= 90\text{^}2+0\text{^}2+25*(\sin^2(90)+Sin^2(0))$ | = 8125.000 | = 8125 |
| O. F [5] | $= 135\text{^}2+30\text{^}2+25*(\sin^2(135)+Sin^2(30))$ | = 19143.750 | = 19144 |
| O. F [6] | $= 60\text{^}2+45\text{^}2+25*(\sin^2(60)+Sin^2(45))$ | =5 656.249 | = 5656 |

### 6. 3. Selection

The fittest chromosomes have higher probability to be selected for the next generation. To compute fitness probability we must compute the fitness of each chromosome. Table 3 illustrate the 6 finesses.

**Table 3.** The (6) Finesses.

| Fitness[i] | = 1 / o. f [i] | Result | Result |
|---|---|---|---|
| Fitness[1] | = 1 / o. f [1] | = 1 / 4525 | = 0.000220994 |

| | | | |
|---|---|---|---|
| Fitness[2] | = 1 / o. f [2] | =1 / 10163 | = 0.0000983961 |
| Fitness[3] | = 1 / o. f [3] | = 1 / 32627 | = 0.00000306497 |
| Fitness[4] | = 1 / o. f [4] | = 1 / 8125 | = 0.000123077 |
| Fitness[5] | = 1 / o. f [5] | = 1 / 19144 | = 0.0000522357 |
| Fitness[6] | = 1 / o. f [6] | = 1 / 5656 | = 0.000176803 |
| | | Total | = 0.000674572 |

The probability for each chromosomes is formulated by: P[i] = Fitness[i] / Total Table 4 illustrate the (6) probabilities.

**Table 4.** The (6) probabilities.

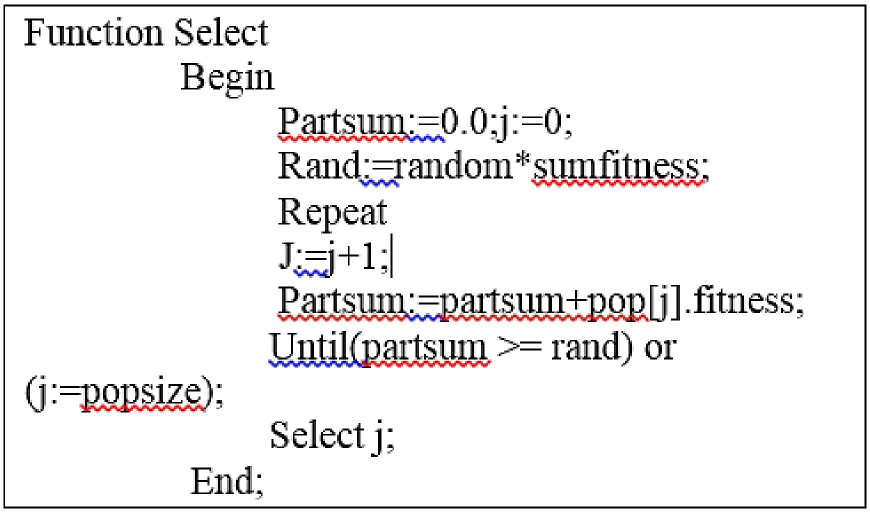
| P[i] | Fitness[i]/ Total | Result |
|---|---|---|
| P[1] | = 0.000220994 / 0.000674572 | = 0.32760695 |
| P[2] | = 0.0000983961 / 0.000674572 | = 0.145864552 |
| P[3] | = 0.00000306497 / 0.000674572 | = 0.004543584 |
| P[4] | = 0.000123077 / 0.000674572 | = 0.18245187 |
| P[5] | = 0.0000522357 / 0.000674572 | = 0.077435303 |
| P[6] | = 0.000176803 / 0.000674572 | = 0.262097144 |

From the probabilities above we can see that Chromosome [1] that has the highest fitness, this chromosome has highest probability to be selected for next generation chromosomes. For the selection process we use roulette wheel, for that we should compute the cumulative probability values. Table 5 illustrate the (6) cumulative probability values and Figure 11 illustrate function select procedure [20].

**Table 5.** The (6) Cumulative Probability Values.

| Chromosome | | Cumulative Probability Values |
|---|---|---|
| C[1] | = 0.32760695 | = 0.32760695 |
| C[2] | = 0.32760695 + 0.145864552 | = 0.473471502 |
| C[3] | = 0.32760695 + 0.145864552 + 0.004543584 | = 0.478015086 |
| C[4] | = 0.32760695 + 0.145864552 + 0.004543584 + 0.18245187 | = 0.660466956 |

| | | |
|---|---|---|
| C[5] | = 0.32760695 + 0.145864552 + 0.004543584 + 0.18245187 + 0.077435303 | = 0.737902259 |
| C[6] | = 0.32760695 + 0.145864552 + 0.004543584 + 0.18245187 + 0.077435303 + 0.262097144 | = 0.999999403 = 1 |



```
Function Select
            Begin
                    Partsum:=0.0;j:=0;
                    Rand:=random*sumfitness;
                    Repeat
                    J:=j+1;
                    Partsum:=partsum+pop[j].fitness;
                   Until(partsum >= rand) or
(j:=popsize);
                   Select j;
             End;
```

**Figure 11.** Function Select Procedure.

## 6. 4. Crossover

In this work, use one-cut point, i.e. randomly select a position in the parent chromosome then exchanging sub-chromosome. Parent chromosome which will mate is randomly selected and the number of mate Chromosomes is controlled using crossover rate (ρc) parameters. Pseudo-code for the crossover process is as follows:

```
begin
k← 0;
while(k<population) do
R[k] ← random(0-1);
if (R[k] < ρc ) then
select Chromosome[k] as parent;
end;
k = k + 1;
end;
end;
```

Chromosome k will be selected as a parent if R [k] <ρc. Suppose we set that the crossover rate is 25%, then Chromosome number k will be selected for crossover if random generated value for Chromosome k below 0.25. The process is as follows: First we generate a random number R as the number of population. Table 6 illustrate the (6) random values.

**Table 6.** The (6) Random Values.

| Random Values | The Number |
|---|---|
| R[1] | = 0.113 |
| R[2] | = 0.471 |
| R[3] | = 0.982 |
| R[4] | = 0.228 |
| R[5] | = 0.511 |
| R[6] | = 0.162 |

For random number R above, parents are Chromosome [1], Chromosome [4] and Chromosome [6] will be selected for crossover.

Chromosome[1] >< Chromosome[4]
Chromosome[1] >< Chromosome[6]
Chromosome[6] >< Chromosome[4]

Chromosome[1] >< Chromosome[4]
= [30,60] >< [90,0]
offspring1 = [30,90] , offspring2 = [60,0]

Chromosome[1] >< Chromosome[6]
= [30,60] >< [60,45]
offspring3 = [30,60] , offspring4 = [60,45]

Chromosome[6] >< Chromosome[4]
= [60,45] >< [90,0]
offspring5 = [60,90] , offspring6 = [45,0]

Thus Chromosome population after experiencing a crossover process illustrated in Table 7 and Figure 12. Illustrate crossover procedure [20].

**Table 7.** The (6) offspring's.

| Offspring's | Chromosome |
|---|---|
| Offspring [1] | = [30,90] |
| Offspring [2] | = [60,0] |
| Offspring [3] | = [30,60] |
| Offspring [4] | = [60,45] |

| | |
|---|---|
| Offspring [5] | = [60,90] |
| Offspring [6] | = [45,0] |

```
Procedure crossover
Begin
If flip (pcross) then begin
          Jcross:=rnd(1,lchrom-1);
          Ncross:=ncross+1;
          End else
          Jcross:=lchrom;
          For j:=1 to jcross do begin

Child1[j]:=mutation(parent1[j],pmutation,nmutation);

Child2[j]:=mutation(parent2[j],pmutation,nmutation);
          end;
          If jcross<> lchrom then
         For j:=jcross+1 to lcross do begin

Child1[j]:=mutation(parent2[j],pmutation,nmutation);

Child2[j]:=mutation(parent1[j],pmutation,nmutation);
          end;
end;
```

**Figure 12.** Procedure Crossover.

## 6. 5. Mutation

Number of chromosomes that have mutations in a population is determined by the mutation rate parameter. Mutation process is done by replacing the gen at random position with a new value. The process is as follows. First we must calculate the total length of gen in the population. In this case the total length of gen is total gen = number of gen in Chromosome * number of population
= 2 * 6
= 12

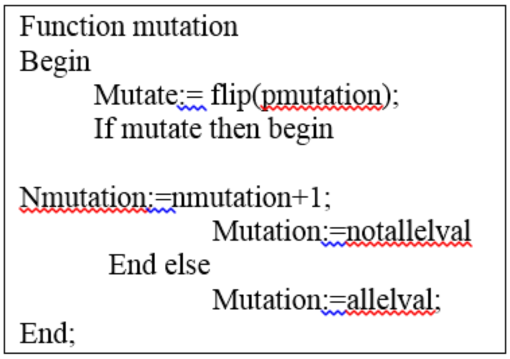
Mutation process is done by generating a random integer between 1 and total gen (1 to 12). If generated random number is smaller than mutation rate(ρm) variable then marked the position of gen in chromosomes. Suppose we define ρm 10%, it is expected that 10% (0.1) of total gen in the population that will be mutated:
number of mutations = 0.1 * 12
= 1.2
≈ 1

Suppose generation of random number yield 6 and 11 then the chromosome which have mutation are Chromosome number 3 gen number 2 and Chromosome 6 gen number 1. The value of mutated gens at mutation point is replaced by random number between (-2∏, 2∏).

Suppose generated random number are 30 and 0 then Chromosome composition after mutation are illustrated in Table 8 and Figure 13. Illustrate mutation procedure [20].

**Table 8.** The (6) Chromosomes After Mutation.

| Offspring's | Chromosomes |
|---|---|
| Offspring [1] | = [30,90] |
| Offspring [2] | = [60,0] |
| Offspring [3] | = [30,30] |
| Offspring [4] | = [60,45] |
| Offspring [5] | = [60,90] |
| Offspring [6] | = [0,0] |



```
Function mutation
Begin
        Mutate:= flip(pmutation);
        If mutate then begin

Nmutation:=nmutation+1;
                    Mutation:=notallelval
          End else
                    Mutation:=allelval;
End;
```

**Figure 13.** Function Mutation.

Finishing mutation process then we have one iteration or one generation of the genetic algorithm. We can now evaluate the objective function after one generation as illustrated in Table 9.

**Table 9.** The Objective Function after One Generation.

| O.F [i] | $F(X) = X1^2 + X2^2 + 25(Sin^2X1+Sin^2X2)$ | Result |
|---|---|---|
| O.F [1] | $= 30^\wedge2+90^\wedge2+25*(\sin^2(30)+Sin^2(90))$ | = 9031 |
| O.F [2] | $= 60^\wedge2+0^\wedge2+25*(\sin^2(60)+Sin^2(0))$ | = 3619 |
| O.F [3] | $=30^\wedge2+30^\wedge2+25*(\sin^2(30)+Sin^2(30))$ | = 1812 |
| O.F [4] | $= 60^\wedge2+45^\wedge2+25*(\sin^2(60)+Sin^2(45))$ | = 5656 |
| O.F [5] | $= 60^\wedge2+90^\wedge2+25*(\sin^2(60)+Sin^2(90))$ | = 11743 |
| O.F [6] | $= 0^\wedge2+0^\wedge2+25*(\sin^2(0)+Sin^2(0))$ | = 0 |

From the evaluation of new offspring's we can see that the objective function is decreasing, this means that we have better Chromosome or solution compared with previous Chromosome generation.

Chromosome [3] must change by offspring [6].
Chromosome [5] must change by offspring [3].
Chromosome [2] must change by offspring [2].

New Chromosomes for next iteration are illustrated in Table 10.

**Table 10.** New (6) Chromosomes for Next Iteration.

| | | |
|---|---|---|
| Chromosome[1] | [X1,X2] | [30;60] |
| Chromosome[2] | [X1,X2] | [60; 0] |
| Chromosome[3] | [X1,X2] | [0;0] |
| Chromosome[4] | [X1,X2] | [90;0] |
| Chromosome[5] | [X1,X2] | [30;30] |
| Chromosome[6] | [X1,X2] | [60;45] |

The Objective Function Of Each Chromosome Is Illustrated In Table 11.

**Table 11.** The Objective Function of Each Chromosome.

| O.F[i] | $F(X) = X1^2 + X2^2 + 25(Sin^2X1+Sin^2X2)$ | Result |
|---|---|---|
| O.F[1] | $= 30\hat{}2+60\hat{}2+25*(sin^2(30)+Sin^2(60))$ | =4525 |
| O.F [2] | $= 60\hat{}2+0\hat{}2+25*(sin^2(60)+Sin^2(0))$ | =3619 |
| O.F [3] | $=0\hat{}2+0\hat{}2+25*(sin^2(0)+Sin^2(0))$ | =0 |
| O.F [4] | $= 90\hat{}2+0\hat{}2+25*(sin^2(90)+Sin^2(0))$ | =8125 |
| O.F [5] | $= 30\hat{}2+30\hat{}2+25*(sin^2(30)+Sin^2(30))$ | =1812 |
| O.F [6] | $= 60\hat{}2+45\hat{}2+25*(sin^2(60)+Sin^2(45))$ | =5656 |

From the evaluation of new chromosomes we can see that the objective function is decreasing, These new Chromosomes will undergo the same process as the previous generation of Chromosomes such as evaluation, selection, crossover and mutation and at the end it produce new generation of Chromosome for the next iteration. This process will be repeated until a predetermined number of generations. For this function, after running 50 generations, best chromosome is obtained:
Chromosome = [0,0]

This means that:

X1 = 0, X2 = 0,

If we use the number in the eggcrate problem

F(X) = 0^2+0^2+25*(sin$^2$(0)+Sin$^2$(0)) = 0

## 7. EXPERIMENTAL RESULT

Executing the simple genetic algorithm code (SGA) written in Pascal programming language, the primary data structure is the population of 26 string and the following parameters, and Table.12.illistrate eggicrate function population

SGA Parameters

Population size = 26
Chromosome length = 2
Maximum of generation = 20
Crossover probability = 0.25
Mutation probability = 0.01

**Table 12.** Eggicrat Function Population.

| x1 | x2 |
|---|---|
| 0 | 0 |
| 15 | 15 |
| 30 | 30 |
| 45 | 45 |
| 60 | 60 |
| 75 | 75 |
| 90 | 90 |
| 105 | 105 |
| 120 | 120 |
| 135 | 135 |
| 150 | 150 |
| 165 | 165 |
| 180 | 180 |
| 0 | 180 |
| 15 | 165 |

| | |
|---|---|
| 30 | 150 |
| 45 | 135 |
| 60 | 120 |
| 75 | 105 |
| 90 | 90 |
| 105 | 75 |
| 120 | 60 |
| 135 | 45 |
| 150 | 30 |
| 165 | 15 |
| 180 | 0 |

At the end of the run repeating the process hundreds of times until the best solution is determined. The Eggcrate function has a known global minimum at [0, 0] with an optimal function value of zero. Figure 14 illustrate optimal solution for eggicrate function.

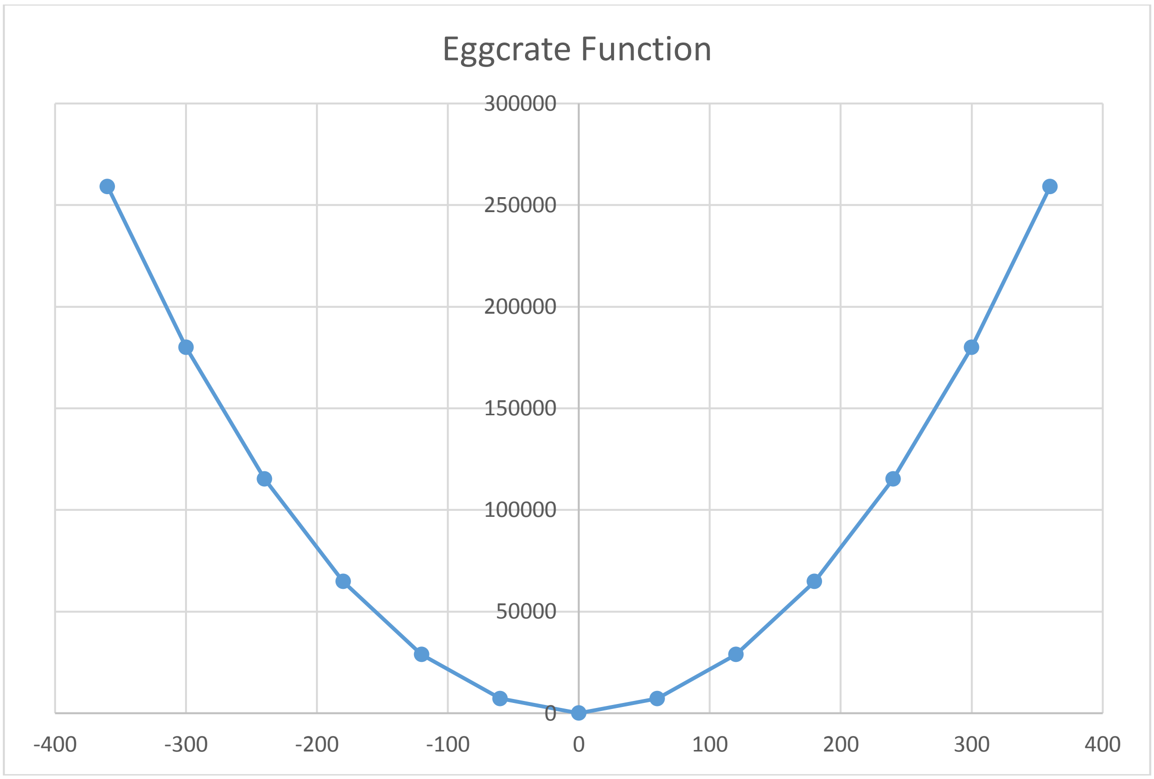


**Figure 14.** Optimal Solution for Eggcrate Function.

## 8. CONCLUSIONS

1. Objective scaling has been suggested for maintaining more careful control over the allocation of trails to the best string.
2. Many practical optimization problems require the specification of a control function or functions over a continuum.
3. GA must have objective functions that reflect information about both the quality and feasibility of solutions.
4. Survival of the fittest, the most fit of a particular generation (the nearest to a correct answer) are used as parents for the next generation.
5. The most fit of this new generation are parents for the next, and so on. This eradicates worse solutions and eliminates bad family lines.
6. The GA does find near optimal results quickly after searching a small portion of the search space.
7. Results shows that the genetic algorithm have the ability to find optimal solution for solving eggcrate function.
8. GA resistant to trapped in local optima.
9. GA perform very well for large scale optimization problems.
10. GA Can be employed for a wide variety of optimization problems.

## Reference

[1] Ahmed A. EL- Sawy, Mohamed A. Hussein, EL-Sayed M. Zaki, A. A. Mousa, ”An Introduction to Genetic Algorithms: A survey A practical Issues*”, International Journal of Scientific & Engineering Research*, Volume 5, Issue 1, January 2014.

[2] Guy Bashkansky and Yaakov Yaari,” Black Box Approach for Selecting Optimization Options Using Budget-Limited Genetic Algorithms, 2007.

[3] Ms. Dharmistha, D. Vishwakarma,” Genetic Algorithm based Weights Optimization of Artificial Neural Network”, *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, Vol. 1, Issue 3, August 2012.

[4] Larry Yaeger, ”Intro to Genetic Algorithms” Artificial Life as an approach to Artificial Intelligence” Professor of Informatics Indiana University, 2008.

[5] Goldberg David E., “Genetic Algorithm in Search, Optimization, and Machine Learning”, Addison Wesley Longmont, International Student Edition 1989.

[6] Andrey Popov, ”Genetic algorithms for optimization”, Hamburg, 2005.

[7] Nanhao Zhu and Ian O’Connor,” iMASKO: A Genetic Algorithm Based Optimization Framework for Wireless Sensor Networks”, *Journal of Sensor and Actuator Networks* ISSN 2224-2708, www.mdpi.com/journal/jsan/, 2013.

[8] Zhou Qing Qing and Purvis Martin, “A Market-Based Rule Learning System” aGuangDong Data Communication Bureau China Telecom 1 Dongyuanheng Rd., Yuexiunan, Guangzhou 510110, China, Department of Information Science, University

of Otago, PO Box 56, Dunedin, New Zealand and/or improving the comprehensibility of the rules, 2004.

[9] Abo Rajy , ”Genetic algorithms”, 2013.

[10] Lubna Zaghlul Bashir, Rajaa Salih Mohammed, “Solving Banana (Rosenbrock) Function based on fitness function”, *World Scientific News*, 6 (2015) 41-56.

[11] Bull Larry, “Learning Classifier Systems: A Brief Introduction”, Faculty of Computing, Engineering & Mathematical Sciences University of the West of England, Bristol BS16 1QY, U.K. Larry, 2004.

[12] Olesya Peshko,”Global Optimization Genetic Algorithms”, 2007.

[13] Kumara Sastry, David Goldberg, Graham Kendall, ”GENETIC ALGORITHMS”, University of Illinois, USA, University of Nottingham, UK, 2011.

[14] Damian Schofield, Rong Qu,”Genetic algorithms”, 2012.

[15] Mitchell Melanie,” An Introduction to Genetic Algorithms”, A Bradford Book the MIT Press Cambridge, Massachusetts - London, England, Fifth printing, 1999.

[16] Saif Hasan, Sagar Chordia, Rahul Varshneya, ”Genetic algorithm”, February 6, 2012.

[17] R.C. Chakraborty “Fandemantaels of genetic algorithms”: AI course lecture 39-40, notes, slides, 2010.

[18] Andrew Chipperfield, Peter Fleming, Hartmut Pohlheim, Carlos Fonseca,” Genetic Algorithm TOOLBOX For Use with MATLAB”, Department of Automatic Control and System Engineering, 2001.

[19] Lubna Zaghlul Bashir, ”Using Evolutionary Algorithm to Solve Amazing Problem by two ways: Coordinates and Locations”, *World Scientific News* 7 (2015) 66-87.

[20] Lubna Zaghlul Bashir, ”Development of Adaptive Control Agent Simulation Using Genetic Algorithm”, A Thesis Submitted to the Department of Computer Science, University of Technology, 2005.

( Received 19 June 2015; accepted 03 July 2015 )