Available online at www.worldscientificnews.com

World Scientific News

WSN 13 (2015) 66-87 EISSN 2392-2192

Using Evolutionary Algorithm to Solve Amazing Problem by Two Ways: Coordinates and Locations

Lubna Zaghlul Bashir

Building and Construction Department, University of Technology, Baghdad, Iraq

E-mail address: lubna_zaghlul@yahoo.com

ABSTRACT

In the computer science field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a meta heuristic) is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. This paper studies the following problem: A square construction site is divided into 9 grid units. We need to use GAs to determine the best location of two temporary facilities A and B, so that: 1. Facility A is as close as possible to facility B; 2. Facility A is as close as possible to the fixed facility F.; 3. Facility B is as far as possible to the fixed facility F. We use two different ways to solve the problem, first by coordinates and second by grid locations. Experimentally the two way results shows that the genetic algorithm have the ability to find optimal solution or find solutions nearby optimal solutions.

***Keywords*:** genetic algorithm; evolutionary algorithms; optimization; fitness; objective function

1. INTRODUCTION

Optimization is a process that finds a best, or optimal solutions for a problem. The optimization problems are catered around three factors:

1. An objective function: which is to be minimize or maximize.
2. A set of unknowns or variables: that effect the objective function.
3. A set of constraints: that allow the unknowns to take on certain values.

An optimization problem is defined as finding values of the variables that minimize or maximize the objective function while satisfying the constraints but exclude others [1]. The Genetic Algorithms are direct, stochastic method for optimization. Since they use populations with allowed solutions (individuals), they count in the group of parallel algorithms. Due to the stochastic was of searching, in most cases, it is necessary to set limits at least for the values of the optimized parameters [2].

2. SCHEME OF THE EVOLUTIONARY ALGORITHMS

The EA holds a population of individuals (chromosomes), which evolve means of selection and other operators like crossover and mutation. Every individual in the population gets an evaluation of its adaptation (fitness) to the environment. In the terms of optimization this means, that the function that is maximized or minimized is evaluated for every individual. The selection chooses the best gene combinations (individuals), which through crossover and mutation should drive to better solutions in the next population [2].

3. OPTIMIZATION PROBLEMS

Genetic algorithms evaluate the target function to be optimized at some randomly selected points of the definition domain. Taking this information into account, a new set of points (a new population) is generated. Gradually the points in the population approach local maxima and minima of the function.

Genetic algorithms can be used when no information is available about the gradient of the function at the evaluated points. The function itself does not need to be continuous or differentiable. Genetic algorithms can still achieve good results even in cases in which the function has several local minima or maxima. These properties of genetic algorithms have their price: unlike traditional random search, the function is not examined at a single place, constructing a possible path to the local maximum or minimum, but many different places are considered simultaneously. The function must be calculated for all elements of the population. The creation of new populations also requires additional calculations. In this way the optimum of the function is sought in several directions simultaneously and many paths to the optimum are processed in parallel. The calculations required for this feat are obviously much more extensive than for a simple random search [3].

4. GENETIC ALGORITHM

A genetic algorithm developed by J.H. Holland 1975 [4]. Genetic algorithms are stochastic search techniques that guide a population of solutions towards an optimum using the principles of evolution and natural genetics. In recent years, genetic algorithms have become a popular optimization tool for many areas of research, including the field of system

control, control design, science and engineering. Significant research exists concerning genetic algorithms for control design and off-line controller analysis.

Genetic algorithms are inspired by the evolution of populations. In a particular environment, individuals which better fit the environment will be able to survive and hand down their chromosomes to their descendants, while less fit individuals will become extinct. The aim of genetic algorithms is to use simple representations to encode complex structures and simple operations to improve these structures. Genetic algorithms therefore are characterized by their representation and operators. In the original genetic algorithm an individual chromosome is represented by a binary string. The bits of each string are called genes and their varying values alleles. A group of individual chromosomes are called a population. Basic genetic operators include reproduction, crossover and mutation.

Genetic algorithms are especially capable of handling problems in which the objective function is discontinuous or non-differentiable, non-convex, multimodal or noisy. Since the algorithms operate on a population instead of a single point in the search space, they climb many peaks in parallel and therefore reduce the probability of finding local minima [5,6]

5. BIOLOGICAL TERMINOLOGY

A) Gene – a single encoding of part of the solution space, i.e. either single bits or short blocks of adjacent bits that encode an element of the candidate solution. Gene illustrates in Figure 1.

B)

Figure 1. Gene Representation.

C) Chromosome – a string of genes that represents a solution. Chromosome illustrates in Figure 2.

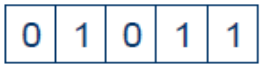


Figure 2. Chromosome Representation.

Chromosomes can be:

- Bit strings (0110, 0011, 1101, …)
- Real numbers (33.2, -12.11, 5.32, …)
- Permutations of element (1234, 3241, 4312, …)
- Lists of rules (R1, R2, R3, …Rn…)
- Program elements (genetic programming)
- Any other data structure

D) Population – the number of chromosomes available to test. Population is illustrated in Figure 3.

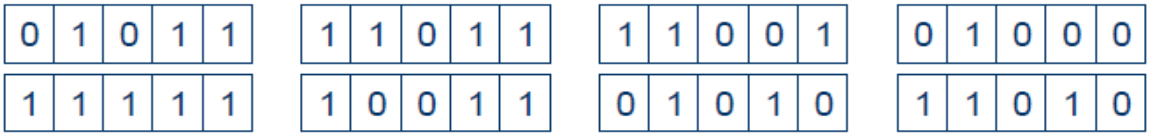


Figure 3. Population Representation.

E) Crossover

1. Choose a random point
2. Split parents at this crossover point
3. Create children by exchanging tails
4. Probability of crossover is typically in range (0.6, 0.9)

Figure 4 shows crossover operator.

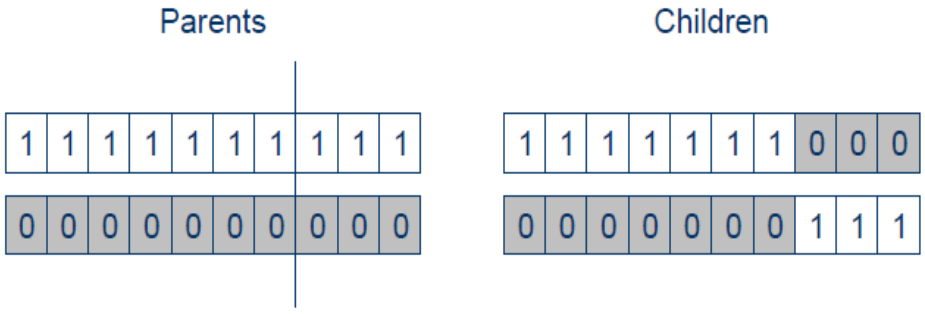


Figure 4. Crossover Operator.

F) Other Crossover Types

- Two-point crossover Figure 5 shows two point crossover operator.

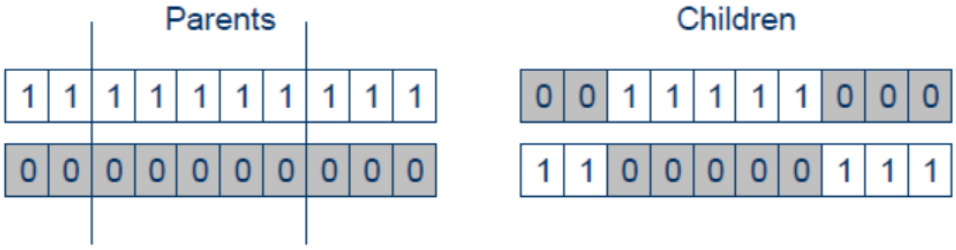


Figure 5. Two Point Crossover Operator.

- Uniform crossover: randomly generated mask Figure 6 shows uniform crossover operator.

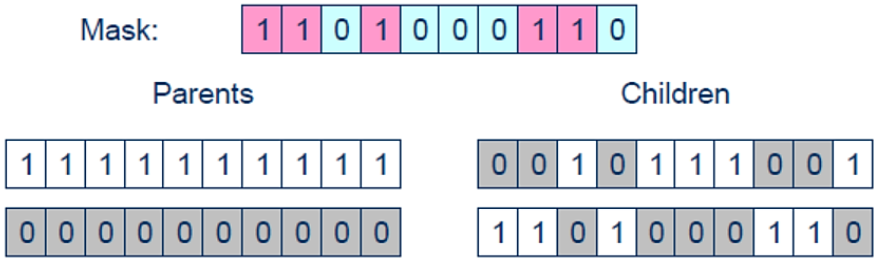


Figure 6. Uniform Crossover Operator

G) Mutation

1. Alter each gene independently
2. Mutation probability is typically in range (1/population size /chromosome length)

Figure 7 shows mutation operator.

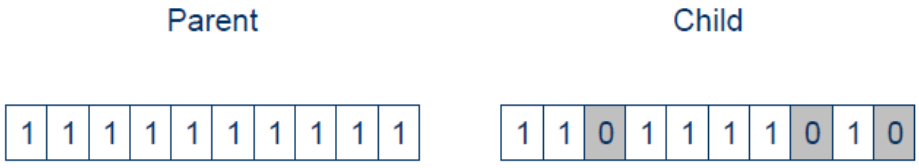


Figure 7. Mutation Operator.

Mutation can be performed in a way that maximizes fitness function [6].

H) Selection

Parents with better fitness have better chances to produce offspring. Fitness function – measure of goodness of the candidate solution. Figure 8 illustrate selection operator.

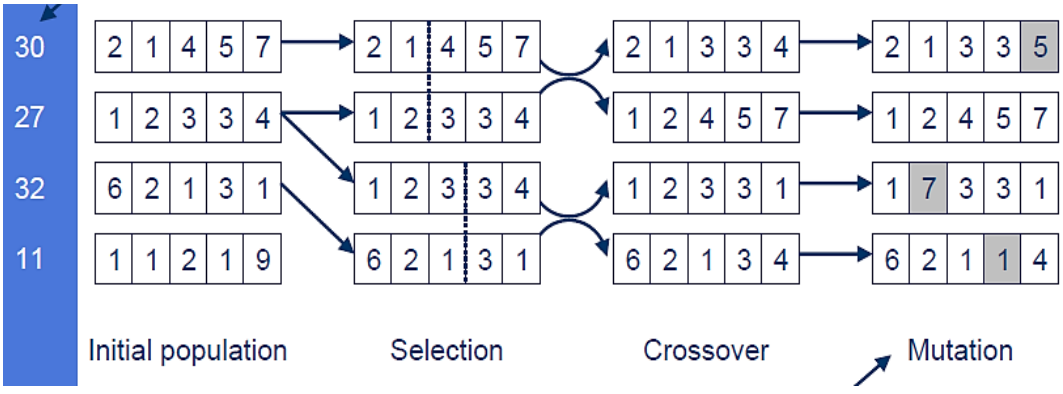


Figure 8. Selection Operator.

I) Selection: Roulette Wheel

1. Better solutions get higher chance to become parents for next generation solutions.
2. Roulette wheel technique:
   - Assign each individual part of the wheel.
   - Spin wheel N times to select N individuals [6].

Figure 9 shows roulette wheel selection.

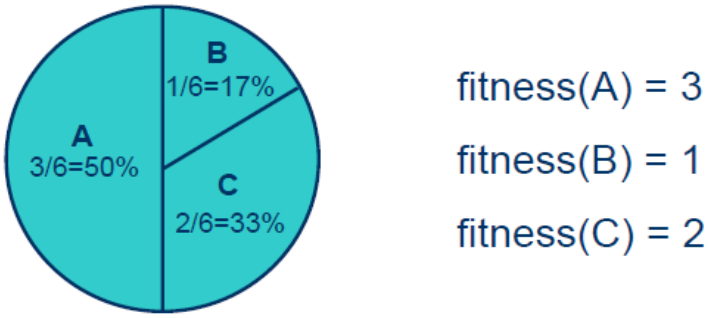


Figure 9. Roulette Wheel Selection.

J) Fitness

1. Fitness is computed for each individual.
2. To help maintain diversity or differentiate between similar individuals, raw objective scores are sometimes scaled to produce the final fitness score
   - Rank - no scaling
   - Linear scaling (fitness proportionate) – normalizes based on min and max fitness in population
   - Sigma truncation scaling - normalizes using population mean and std. dev., truncating low-fitness individuals
   - Sharing (similarity scaling) - reduces fitness for individuals that are similar to other individuals in the population [7].

K) Replacement

- Simple or generational GAs replace entire population per the dictates of the selection scheme
- Steady state or online GAs use different replacement Scheme
- Replace worst
- Replace best
- Replace parent
- Replace random
- Replace most similar (crowding)

• Only replace worst and replace most similar are Generally very effective (when replace parent works, it is because parents are similar) [7].

6. GENETIC ALGORITHM CODE

Given a clearly defined problem to be solved and a bit string representation for candidate solutions, a simple GA works as follows:

1. Start with a randomly generated population of n l−bit chromosomes (candidate solutions to a problem).
2. Calculate the fitness $f(x)$ of each chromosome x in the population.
3. Repeat the following steps until n offspring have been created:
a. Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. Selection is done "with replacement," meaning that the same chromosome can be selected more than once to become a parent.
b. With probability pc (the "crossover probability" or "crossover rate"), cross over the pair at a randomly chosen point (chosen with uniform probability) to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents. (Note that here the crossover rate is defined to be the probability that two parents will cross over in a single point. There are also "multi0point crossover" versions of the GA in which the crossover rate for a pair of parents is the number of points at which a crossover takes place.)
c. Mutate the two offspring at each locus with probability pm (the mutation probability or mutation rate), and place the resulting chromosomes in the new population. If n is odd, one new population member can be discarded at random.
4. Replace the current population with the new population.
5. Go to step 2 [7-10].

Genetic algorithm code illustrated in Figure 10 and basic flow diagram of a genetic algorithm illustrated in Figure 11 [11-13].

```
begin
        t=0
        initialize Pt
        evaluate Pt
        while(termination condition not satisfied) do
        begin
                t=t+1
                select Pt from Pt-1
                mutate Pt
                evaluate Pt
        end
end
```

Figure 10. Genetic Algorithm Code.

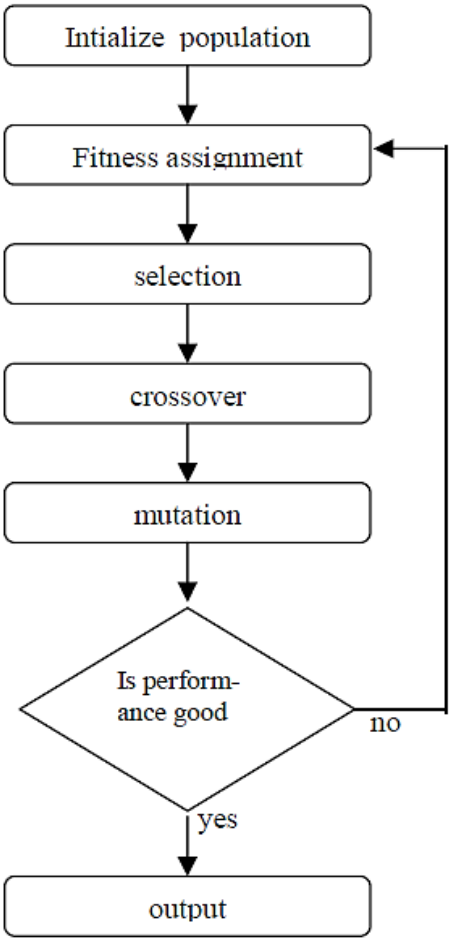


Figure 11. Genetic Algorithm Diagram.

7. THE PROBLEM

A square construction site is divided into 9 grid units, as illustrated in Figure 11,we need to use GAs to determine the best location of two temporary facilities A and B, so that:

- Facility A is as close as possible to facility B.
- Facility A is as close as possible to the fixed facility F.
- Facility B is as far as possible to the fixed facility F.

A square problem illustrated in Figure 12.

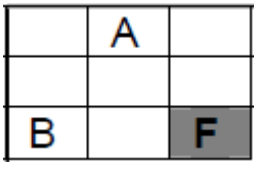


Figure 12. Square Problem.

7. 1. Problem Representation (Using Coordinates)

Figure 13 illustrated A,B locations from X and Y axis coordinates.

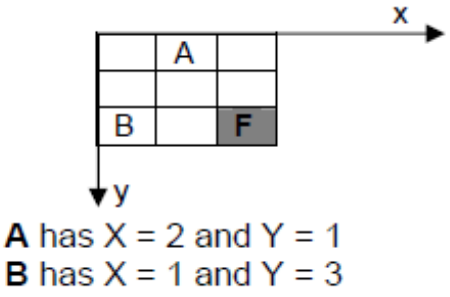


Figure 13. X and Y coordinates.

7. 2. Chromosome Structure

The variables in the problem are the locations of facilities A & B. Then, the chromosome structure are as follows. Note that the genes of a chromosome are the problem variables.Figure.14 shows chromosome structure.

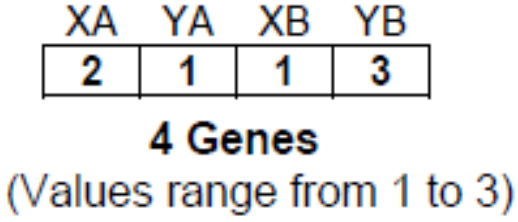


Figure 14. Chromosome Structure.

7. 3. Generate Population (50 to 100 is reasonable diversity & processing time).

Figure 15 shows sample of population.

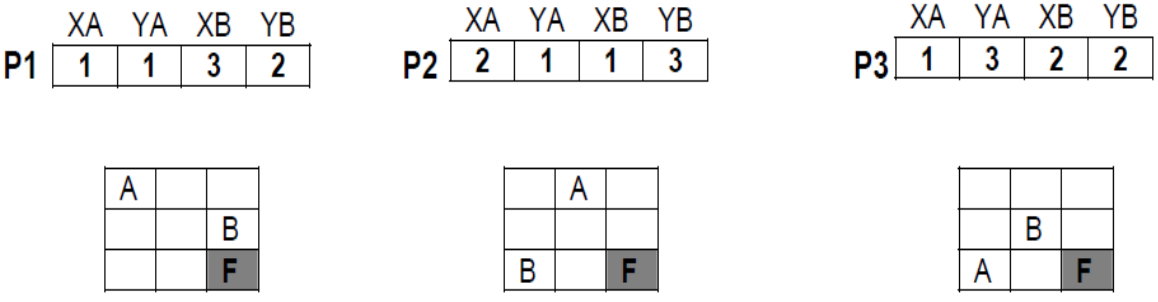
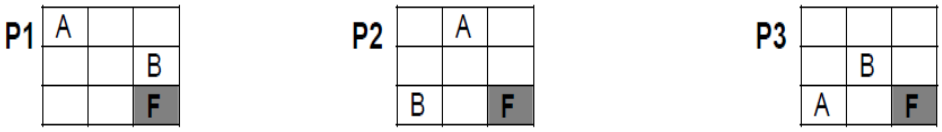


Figure 15. Sample of Population.

7. 4. Evaluate the Population



Objective function = Minimize site score = Minimum of Σ d . W

$$\textbf{Score} = \textbf{dAB} \cdot \textbf{WAB} + \textbf{dAF} \cdot \textbf{WAF} + \textbf{dBF} \cdot \textbf{WBF} \qquad (1)$$

Let's consider the closeness weights (W) as follows (from past notes):
WAB = **100** (positive means A & B close to each other)
WAF = **100** (A & F close to each other)
WBF = **-100** (negative means B & F far from each other)

Let's also consider the distance (d) between two facilities as the number of horizontal and vertical blocks between them.
P1 Score = **dAB** · **WAB** + **dAF** · **WAF** + **dBF** · **WBF** = 3 · 100 + 4 · 100 + 1 · (-100) = **600**
P2 Score = **dAB** · **WAB** + **dAF** · **WAF** + **dBF** · **WBF** = 3 · 100 + 3 · 100 + 2 · (-100) = **400**
P3 Score = **dAB** · **WAB** + **dAF** · **WAF** + **dBF** · **WBF** = 2 · 100 + 2 · 100 + 2 · (-100) = **200**

7. 5. Calculate the Merits of Population Members

Merit of **P1** = (600 + 400 + 200) / 600 = **2**
Merit of **P2** = (600 + 400 + 200) / 400 = **3**
Merit of **P1** = (600 + 400 + 200) / 200 = **6**
the sum of merits = 11
the smaller score gives higher merit because we are interested in minimization. In case of maximization, we use the inverse of the merit calculation.

7. 6. Calculate the Relative Merits of Population Members

RM of **P1** = merit * 100 / Sum of merits
= 2 · 100 / 11 = **18**
RM of **P2** = merit * 100 / Sum of merits
= 3 · 100 / 11 = **27**
RM of **P3** = merit * 100 / Sum of merits
= 6 · 100 / 11 = **55**

7. 7. Randomly Select Operator (Crossover or Mutation)

Crossover rate = 96% (marriage is the main avenue for evolution)
Mutation rate = 4% (genius people are very rare)

To select which operator to use in current cycle, we generate a random number (from 0 to 100). If the value is between 0 to 96, then crossover, otherwise, mutation

7. 8. Use the Selected Operator (Assume Crossover)

Randomly select two parents according to their relative merits of Step 7. 6.as shows in figure 16.

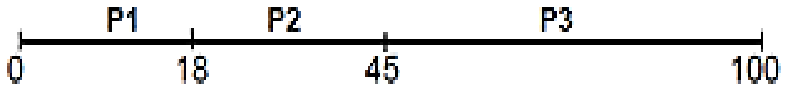


Figure 16. Relative Merits on a Cumulative Horizontal Scale

For first parent, we generate a random number (0 to 100). According to its value, we pick the parent. For example, assume value is 76, then **P3** is selected.

For the 2nd parent, get a random number (0 to 100). Assume 39, then **P2** is picked. Let's apply crossover to generate an offspring.Figure.17 shows the two offspring's results.

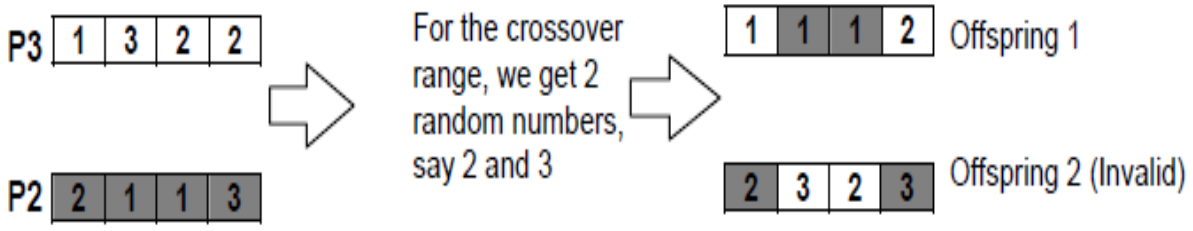


Figure 17. Two Offspring's Results

7. 9. Evaluate the Offspring

Figure 18 show offsprng1

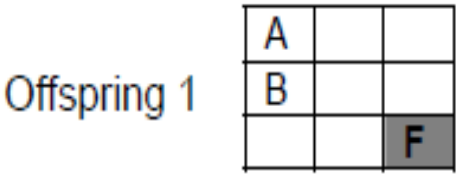


Figure 18. Offsprng1

The Offspring 2 is invalid because both facilities A & B are at same coordinates (x = 2 and Y = 3) and this is not allowed

Offspring1 Score = **dAB · WAB + dAF · WAF + dBF · WBF**
= 1 · 100 + 4 · 100 + 3 · (-100) = **200**

7. 10. Compare the Offspring with the Population (Evolve the Population)

Since the offspring score = 200 is better than the worst population member (P1 has a score of 600), then the offspring survives and P1 dies (will be replaced by the offspring).

Accordingly, P1 becomes:

1	1	1	2

7. 11. Experimental Result

Executing the simple genetic algorithm code (SGA) written in Pascal programming language, the program call three routines, selection, crossover, mutation as illustrated in Figures 19, 20, 21 respectively.

```
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 19. Function Select

```
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 20. Function Crossover

```
Function mutation
Begin
        Mutate:= flip(pmutation);
        If mutate then begin
                Nmutation:=nmutation+1;
                Mutation:=notallelval
         End else
                Mutation:=allelval;
End;
```

Figure 21. Function Mutation

The primary data structure is the population of 56 strings illustrated in table.1 and the following parameters:

SGA Parameters

Population size = **56**
Chromosome length = **4**
Maximum of generation = **10**
Crossover probability = **0.6**
Mutation probability = **0.03**

At the end of the run repeating the process thousands of times until the best solution is determined. Figure 22 shows one of the top solutions.

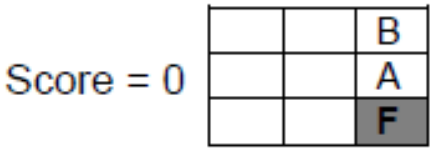


Figure 22. The Top Solution.

Table 1. Initial Population (Using Coordinates)

	XA	YA	XB	YB
1.	**1**	**1**	**3**	**1**
2.	**1**	**1**	**1**	**2**
3.	**1**	**1**	**2**	**2**
4.	**1**	**1**	**3**	**2**
5.	**1**	**1**	**1**	**3**

6.	**1**	**1**	**2**	**3**
7.	**2**	**1**	**1**	**1**
8.	**2**	**1**	**3**	**1**
9.	**2**	**1**	**1**	**2**
10.	**2**	**1**	**2**	**2**
11.	**2**	**1**	**3**	**2**
12.	**2**	**1**	**1**	**3**
13.	**2**	**1**	**2**	**3**
14.	**3**	**1**	**1**	**1**
15.	**3**	**1**	**2**	**1**
16.	**3**	**1**	**1**	**2**
17.	**3**	**1**	**2**	**2**
18.	**3**	**1**	**3**	**2**
19.	**3**	**1**	**1**	**3**
20.	**3**	**1**	**2**	**3**
21.	**1**	**2**	**1**	**1**
22.	**1**	**2**	**2**	**1**
23.	**1**	**2**	**3**	**1**
24.	**1**	**2**	**2**	**2**
25.	**1**	**2**	**3**	**2**
26.	**1**	**2**	**1**	**3**
27.	**1**	**2**	**2**	**3**
28.	**2**	**2**	**1**	**1**
29.	**2**	**2**	**2**	**1**
30.	**2**	**2**	**3**	**1**
31.	**2**	**2**	**1**	**2**
32.	**2**	**2**	**3**	**2**
33.	**2**	**2**	**1**	**3**
34.	**2**	**2**	**2**	**3**
35.	**3**	**2**	**1**	**1**
36.	**3**	**2**	**2**	**1**
37.	**3**	**2**	**3**	**1**
38.	**3**	**2**	**1**	**2**
39.	**3**	**2**	**2**	**2**
40.	**3**	**2**	**1**	**3**
41.	**3**	**2**	**2**	**3**

42.	**3**	**2**	**1**	**1**
43.	**3**	**2**	**2**	**1**
44.	**3**	**2**	**3**	**1**
45.	**3**	**2**	**1**	**2**
46.	**3**	**2**	**2**	**2**
47.	**3**	**2**	**1**	**3**
48.	**3**	**2**	**2**	**3**
49.	**1**	**3**	**1**	**1**
50.	**1**	**3**	**2**	**1**
51.	**1**	**3**	**3**	**1**
52.	**1**	**3**	**1**	**2**
53.	**1**	**3**	**2**	**2**
54.	**1**	**3**	**3**	**2**
55.	**1**	**3**	**1**	**3**
56.	**1**	**3**	**2**	**3**

8. LOCATIONS VS COORDINATES

Solve the same problem by using locations instead of coordinates and compare between the results in the two cases.

8. 1. Problem Representation (Using Locations)

Figure 23 shows location of A and B and F on the square divided into 9 grid units.

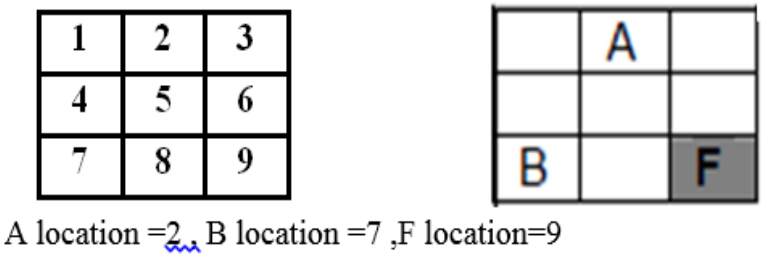


Figure 23. Location of A & B & F on the square

8. 2. Chromosome Structure

The variables in the problem are the locations of facilities A & B. Then, the chromosome structure are as follows. Note that the genes of a chromosome are the problem variables. Figure 24 illustrates chromosome structure.

A location	B location
2	7

2 genes (values range from 1 to 8).

Figure 24. Chromosome Structure.

8. 3. Generate Population (50 to 100 is reasonable diversity & processing time).

Figure 25 illustrates sample of population.

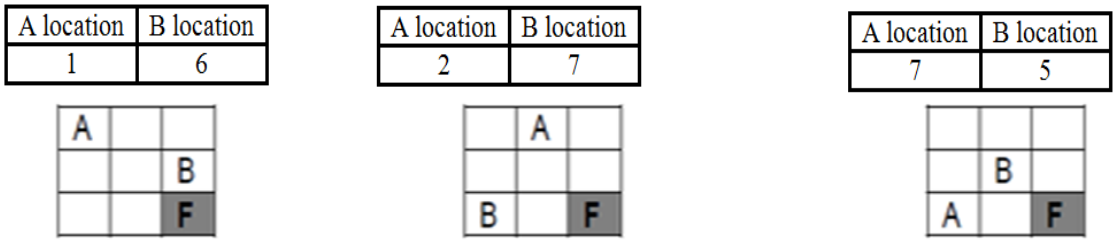
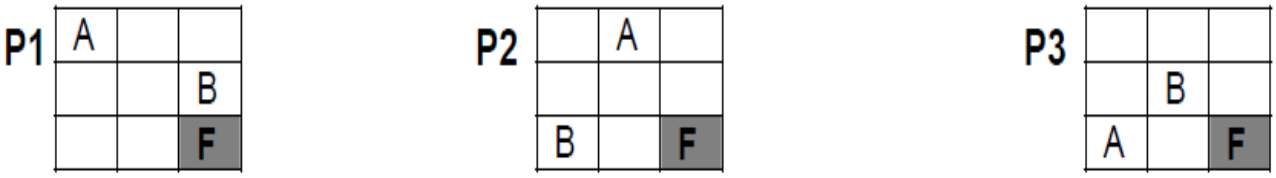


Figure 25. Sample of Population

8. 4. Evaluate the Population



Objective function = Minimize site score = Minimum of Σ d . W

Score = dAB · WAB + dAF · WAF + dBF · WBF (1)

Let's consider the closeness weights (W) as follows (from past notes):
WAB = **100** (positive means A & B close to each other)

WAF = **100** (A & F close to each other)
WBF = **-100** (negative means B & F far from each other)

Let's also consider the distance (d) between two facilities as the number of horizontal and vertical blocks between them.

P1 Score = **dAB · WAB + dAF · WAF + dBF · WBF**
= 3 · 100 + 4 · 100 + 1 · (-100) = **600**
P2 Score = **dAB · WAB + dAF · WAF + dBF · WBF**
= 3 · 100 + 3 · 100 + 2 · (-100) = **400**

P3 Score = **dAB** · **WAB** + **dAF** · **WAF** + **dBF** · **WBF**
= 2 · 100 + 2 · 100 + 2 · (-100) = **200**

8. 5. Calculate the Merits of Population Members

Merit of **P1** = (600 + 400 + 200) / 600 = **2**
Merit of **P2** = (600 + 400 + 200) / 400 = **3**
Merit of **P1** = (600 + 400 + 200) / 200 = **6**
Notice the sum of merits = 11

Notice that smaller score gives higher merit because we are interested in minimization. In case of maximization, we use the inverse of the merit calculation.

8. 6. Calculate the Relative Merits of Population Members

RM of **P1** = merit * 100 / Sum of merits
= 2 . 100 / 11 = **18**
RM of **P2** = merit * 100 / Sum of merits
= 3 . 100 / 11 = **27**
RM of **P3** = merit * 100 / Sum of merits
= 6 . 100 / 11 = **55**

8. 7. Randomly Select Operator (Crossover or Mutation)

Crossover rate = 96% (marriage is the main avenue for evolution)
Mutation rate = 4% (genius people are very rare)

To select which operator to use in current cycle, we generate a random number (From 0 to 100). If the value is between 0 to 96, then crossover, otherwise, mutation

8. 8. Use the Selected Operator (Assume Crossover)

Randomly select two parents according to their relative merits of Step 8. 6. as shown in Figure 26.

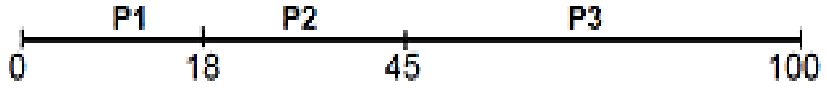


Figure 26. Relative Merits on a Cumulative Horizontal Scale.

For first parent, we generate a random number (0 to 100). According to its value, we pick the parent. For example, assume value is 10, then **P1** is selected.

For the 2nd parent, get a random number (0 to 100). Assume 39, then **P2** is picked. Let's apply crossover P1 with P2 to generate an offspring's. Figure 27 shows the two offspring's results.

P1

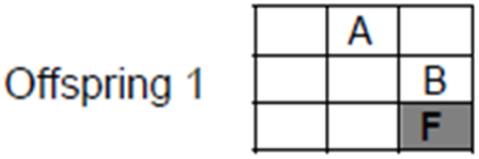
A location	B location
1	6

P2

A location	B location
2	7

Offspring1

A location	B location
2	6



Offsspring2

A location	B location
1	7

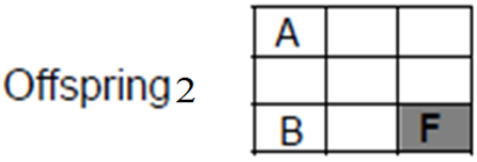


Figure 27. The Two Offspring's.

8. 9. Evaluate the Offspring

Offspring 1 Score = **dAB · WAB + dAF · WAF + dBF · WBF**
= 2 · 100 + 3 · 100 + 1 · (-100) = **400**
Offspring 2 Score = **dAB · WAB + dAF · WAF + dBF · WBF**
= 2 · 100 + 4 · 100 + 2 · (-100) = **400**

8. 10. Compare the Offspring with the Population (Evolve the Population)

Since the offspring1 score and offspring2 score = 400 is better than the worst population member (P1 has a score of 600), then the offspring survives and P1 dies (will be replaced by the offspring1 or offspring2).

8. 11. Experimental Results

The primary data structure is the population of 56 string illustrated in Table 2 and the following parameters:

SGA Parameters

Population size = **56**
Chromosome length = **2**
Maximum of generation = **10**
Crossover probability = **0.6**

Mutation probability = **0.03**

At the end of the run repeating the process thousands of times until the best solution is determined. One of the top solutions when the score = 0

Table 2. Initial Population (Using Locations).

	A location	**B location**
1.	1	2
2.	1	3
3.	1	4
4.	1	5
5.	1	6
6.	1	7
7.	1	8
8.	2	1
9.	2	3
10.	2	4
11.	2	5
12.	2	6
13.	2	7
14.	2	8
15.	3	2
16.	3	1
17.	3	4
18.	3	5
19.	3	6
20.	3	7
21.	3	8
22.	4	1
23.	4	3
24.	4	1
25.	4	5
26.	4	6
27.	4	7
28.	4	8
29.	5	2
30.	5	3
31.	5	4

32.	5	1
33.	5	6
34.	5	7
35.	5	8
36.	6	1
37.	6	3
38.	6	4
39.	6	5
40.	6	2
41.	6	7
42.	6	8
43.	7	2
44.	7	1
45.	7	4
46.	7	5
47.	7	6
48.	7	3
49.	7	8
50.	8	1
51.	8	2
52.	8	3
53.	8	4
54.	8	5
55.	8	6
56.	8	7

9. CONCLUSIONS

- Experimentally results shows that the genetic algorithm have the ability to find optimal solution or find solutions nearby optimal solutions.
- Experimental shows the same results using location or coordinates.
- It is encouraging that the heuristic GA is very effective in identifying optimal solutions.
- Because of the stochastically behaviour of the GA it is also concluded that there are no guarantees that GA will reach the optimal solution in every run. However it is shown that all results obtained with GA are close to optimal in single-objective problem.

- It is shown that the algorithm is capable to generate a diverse set of non-dominated solutions and is able to capture extreme solutions to all objective functions.
- GA represents an intelligent exploitation of a random search used to solve optimization problem.
- The GA is well suited to and has been extensively applied to solve complex design optimization problems because it can handle both discrete and continuous variables, and nonlinear objective and constrain functions without requiring gradient information.
- Fitness scaling has been suggested for maintaining more careful control over the allocation of trails to the best string.
- Many practical optimization problems require the specification of a control function or functions over a continuum.
- GA must have objective functions that reflect information about both the quality and feasibility of solutions.
- 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.
- The most fit of this new generation are parents for the next, and so on. This eradicates worse solutions and eliminates bad family lines.

References

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

[2] Andrey Popov," Genetic Algorithms For Optimization" Hamburg, 2 005.

[3] Lubna Zaghlul Bashir, Nada Mahdi, *World Scientific News* 5 (2015) 138-150.

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

[5] 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.

[6] Olesya Peshko,"Global OptimizationGenetic Algorithms", 2007.

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

[8] Colin Reeves, "Genetic algorithms", School of Mathematical and Information Sciences Coventry University Priory St Coventry CV1 5FBE-mail: C. Reeves@conventry. ac.uk http://www.mis.coventry.ac.uk/~colinr, 2005.

[9] Mitchell Melanie," An Introduction to Genetic Algorithms", A Bradford Book The MIT Press Cambridge, Massachusetts London, England, Fifth printing, 1999.

[10] Saif Hasan, Sagar Chordia, Rahul Varshneya, "Genetic algorithm", February 6, 2012.

[11] Ashish Ghosh, Satchidananda Dehuri, ”Evolutionary algorithms for multi criterian optimization: a servay”, *International journal of computing and information system* Vol. 2, No 1, April 2004.

[12] From Wikipedia, the free encyclopedia.

[13] Lubna Zaghlul Bashir, Rajaa Salih Mohammed Hasan, *World Scientific News* 6 (2015) 41-56.

(Received 08 June 2015; accepted 22 June 2015)