SOLVING MIN-MAX PROBLEMS USING GENETIC ALGORITHM

1Ibrahiem M. M. EL EMARY, 2Al Dahoud ALI, 3Mona M. Abd EL-KAREEM
1King Abdulaziz University, P.O. Box 18388, Jeddah, King Saudi Arabia
2Faculty of Science and IT, Al Zaytoonah University, Amman, Jordan
3Faculty of Science, Suez Canal University, Ismailia, Egypt.
omary57@hotmail.com, aldahoud@alzaytoonah.edu.jo, monaia_21@yahoo.com

Keywords: Min-Max Problem, Genetic Algorithms, Ant Colony Optimization and Fitness Function

Abstract: A min-max genetic algorithm was originally designed for simultaneous minimization and maximization of the same object functions during the same optimization run. In this paper, we apply (G.A) technique to solve the min-max problems. So, a great attention was focused on the illustration of how to use genetic algorithms (G.A) techniques in finding the solution of the min-max problems. According the obtained results, the proposed methodology of solution reaches the feasible area reasonably fast and consistently and produces relatively good results. Also shown from the results that depend on using G.A for solving min-max problems, we see that the optimum solution can be achieved quickly and notice that the solution is almost equal to the analytical one. Nevertheless, on using the (G.A) to solve these min-max problems, we obtain the result very swiftly.

1. INTRODUCTION

Genetic algorithms are computer based optimization methods that uses the Darwinian evolution of nature as a model [12]. The solution base of the problem is encoded as individuals that are chromosomes consisting of several genes. The GAs are simplified models derived from the natural genetics, e.g. in GAs the individual (phenotype) is usually exactly derived from the chromosome (genotype), whereas in the nature phenotype is not directly derived genotype, but the age, living conditions, diseases, accidents etc. of an individual have effect to the phenotype.

In GAs the virtual individuals are tested against the problem represented as a fitness function. The better the fitness value individual gets the better chance it has to be selected to be a parent for new individuals. The worst individuals are removed from the population in order to make room for the new generation. Using crossover and mutation operations GA creates new individuals. In crossover, we select the genes for a new chromosome from each parent using some reselected practice, e.g. one-point, two-point or uniform crossover. In mutation we change random genes of the chromosome either randomly or using some predefined strategy. The GA strategy is usually elitist and follows the survival of the fittest principles of Darwinian evolution.

Also, Genetic algorithm is defined as a computational model simulating the process of genetic selection and natural elimination in biologic evolution. Pioneering work in this field was conducted by Holland in the 1960s [2, 3]. As a high efficient search strategy for global optimization, genetic algorithm demonstrates favorable performance on solving the min-max problems. With comparing to traditional search algorithms, genetic algorithm is able to automatically acquire and accumulate the necessary knowledge about the search space during its search process, and self-adaptively control the entire search process through random optimization technique. The computation of usual genetic algorithm is an iterative process.
that simulates the process of genetic selection and natural elimination in biologic evolution.

For each iteration, candidate solutions are retained and ranked according to their quality. A fitness value is used to screen out unqualified solutions. Genetic operators, such as crossover, mutation, translocation and inversion, are then performed on those qualified solutions to estimate new candidate solutions of the next generation. The above process is carried out repeatedly until certain convergent condition is met. And finally, GA is a general search method [4] that uses analogs of genetic operators on a population of states in a search space to find those states that have high fitness values. In min-max problems, genetic operators and coding methods are designed in advance so that the individuals may satisfy the constraints.

From a search strategic point of view this means that variables are randomly selected and then values which will be assigned to them are also randomly selected from genes contained within a population. Therefore search by crossover in GA can not be considered efficient. Furthermore, mutation is a random search method itself. When we think about efficiency of global search that is the most important characteristic of GA; we must admit that the rate of search is low, because GA stresses random search rather than directional search.

This paper is organized in six sections. Related works are discussed in details in section two; section three talks about Min. Max problem; in section four, we propose our methodology of solution and section five deals with the simulated results. The paper was terminated in section six with conclusion and future works.

2. RELEVANT WORKS

P.E. Amiolemhen [1] used genetic algorithms for determination of the optimal machining parameters in the conversion of a cylindrical bar stock into a continuous finished profile. That work proposed an optimization technique based on genetic algorithms for the determination of the cutting parameters in multi-pass machining operations by simultaneously considering multi-pass roughing and single-pass finishing operations. The optimum machining parameters are determined by minimizing the unit production cost of converting a cylindrical bar stock into a continuous finished profile involving seven machining operations; with each operation subject to many practical constraints. The cutting model developed for each machining operation is a non-linear, constrained problem. Experimental results show that the proposed technique is both effective and efficient.

Ant colony optimization [6] (ACO) uses many ants (or agents) to traverse the solution space and find locally productive areas. While usually inferior to genetic algorithms and other forms of local search, it is able to produce results in problems where no global or up-to-date perspective can be obtained, and thus the other methods cannot be applied.

R.K. Sahoo [11] uses genetic algorithms for multi-component aromatic extraction. That work applied (GA) which leads to globally optimal binary interaction parameters from multi-component liquid-liquid equilibrium data, has been recently demonstrated for some ternary, Quaternary and quinary systems. The binary interaction parameters are related to each other through the closure equations. In that work, the binary interaction parameters based on non-random two liquid (NRTL) activity coefficient model have been estimated using GA, without and with closure equations for 65 multi-component aromatic extraction systems: 53 ternary, 9 quaternary and 3 quinary systems. Parameters that satisfy the closure equations exhibit better root mean square deviations than those that do not satisfy the closure equations. Root mean square deviation value without implementation of closure equations is 0–80% better than literature as compared to 0–90% better with implementation of closure equations.

Aromatics, such as benzene, toluene and xylene are considered essential in the chemical industry because they are the source of many organic chemicals. These aromatics are presented in naphtha. High purity aromatics are difficult to be separated using ordinary distillation operation, since they form several binary azeotropes with non-aromatics. Extraction is, therefore, a better choice to separate the aromatics from naphtha, as they are preferentially soluble in a variety of solvents. To predict the separation, it is necessary to know the liquid–liquid equilibrium (LLE) data for a particular system. Various activity coefficient
models, such as the universal quasi chemical (UNIQUAC) and the non-random two liquid (NRTL) can be used to predict the LLE. Each of these models requires proper binary interaction parameters that can represent LLE for highly non-ideal liquid mixtures usually encountered in aromatic extraction. These parameters are usually estimated from the known experimental LLE data via optimization of a suitable objective function.

B.M.Kariuki [10] use genetic algorithms for solving crystal structures from powder diffraction data, that work uses GA to tackle crystal structure solution from powder diffraction data in the case of a previously unknowns structure-ortho-thymotic acid. In that structure solution calculation, the structural fragment was subjected to combine translation and rotation within the unit cell, together with variation of selected intermolecular degree of freedom under the control of a genetic algorithm, in which a population of trail structures is allowed to evolve subject to well defined procedures for mating, mutation, and natural selection.

Importantly, the genetic algorithm approach adopts the ‘direct-space’ philosophy for structure solution, and implicitly avoids the problematic step of extracting the intensities of individual reflections from the powder diffraction data. The structure solution was found efficiently in the genetic algorithm calculation, and was then used as the initial structural model in Riveted refinement calculations.

3. MIN - MAX PROBLEMS

In this section, we will extract certain steps from the section on graphing in order to determine the points at which the given function has a maximum or minimum value. First we need some terminology. This is best summarizing in the Fig.1 below.

![Fig.1 Determining the Points at Which the Given Function Has a Maximum or Minimum Value](image)

Since we are talking about points on the graph of a function there are two numbers for each point, the first and the second coordinate. In the case of these extreme points (i.e. either maximum or minimum) we call the first coordinate the local max (min) point and the second coordinate the local max (min) value. Local refers to the fact that a local maximum value is only greater than the function values for first coordinates nearby the local max point. Similarly for local minimum value: The local maximum value is not necessarily greater than all function values. On the graph above you should see that the value at x=5 is greater than the local maximum value because the point (5,f(5)) is higher than the point (local max pt, local max value).

We could talk about global maximum function values -i.e. function values that are greater than all other function values. Similarly for global minimum function values will not be discussed here. Also, the problem of determining all of the extreme values (local or global) is more complicated than what we will do here. This will serve as an introduction to max-min problems and our technique will work for all of the functions that we will consider in the homework and tests.

Now let's figure out which computations have to be done in order to find the extreme points. Let's draw tangent lines to the graph at the extreme points shown in Fig.2.
We can make the following observations. The tangent lines at the extreme points have slope equal to zero (horizontal tangent lines). But that's not enough to distinguish the points. We need something else. Look at the concavity at these points. It should be clear that the graph has to be concave downward at the local maximum and concave upward at the local minimum. So the second derivative's value must be negative at the local maximum and positive at the local minimum. We summarize this in the following fig.3.

Fig.3 Describing Concave Downward at the Local Maximum and Concave Upward at the Local Minimum.

Procedure for computing local maximum and minimum:

In order to locate the local extreme points and classify which points are local maximum and which are local minimum requires the following steps.

1. Compute the derivative $f'(x)$.
2. Solve the equation $f'(x) = 0$. There might be several solutions.
3. Compute the second derivative $f''(x)$.
4. Evaluate $f''(x)$ for each solution obtained in step 2.
5. Classify each point as local maximum or local minimum.
6. Compute the function value for each point obtained in step 2.

4. THE PROPOSED METHODOLOGY OF SOLUTION

GENETIC ALGORITHMS

Problems which appear to be particularly appropriate for solution by genetic algorithms include timetabling and scheduling problems, and many scheduling software packages are based on GAs. Genetic algorithms are often applied as an approach to solve global optimization problems. As a general rule of thumb genetic algorithms might be useful in problem domains that have a complex fitness landscape as recombination is designed to move the population away from local optima that a traditional hill climbing algorithm might get stuck in.

A genetic algorithm is a search technique used in finding true or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms form a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination). Genetic algorithms are implemented as a computer simulation in which a population of abstract representations (called chromosomes or the genotype or the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions.

Traditionally, solutions are represented in binary strings of 0s and 1s, but other encodings are also possible [7]. The evolution usually starts from a population of randomly generated individuals. Genetic algorithms form one of the best ways to solve a problem for which a little is known. They are very general algorithms that work well in any search space. A genetic algorithm is able to create a high quality solution.
Genetic algorithms (GAs) are search procedures that use the mechanics of natural selection and natural genetics. The genetic algorithm, first developed by John H. Holland in the 1960’s, allows computers to solve difficult problems. It uses evolutionary techniques, based on function optimization and artificial intelligence, to develop a solution.

The basic operation of a genetic algorithm is simple. First a population of possible solutions to a problem is developed. Next, the better solutions are recombined with each other to form some new solutions. Finally the new solutions are used to replace the poorer of the original solutions and the process is repeated.

**GA PROCEDURE**

A typical genetic algorithm requires two things to be defined:
- A genetic representation of the solution domain.
- A fitness function to evaluate the solution domains, which are of course given in our algorithm.

A standard representation of the solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size that facilitates simple crossover operation. Variable length representations were also used, but crossover implementation is more complex in this case.

**FITNESS FUNCTION**

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. The fitness function is a computation that evaluates the quality of the chromosome as a solution to a particular problem, [5, 9].

The fitness function is a black box for the GA. Internally, this may be achieved by a mathematical function, a simulation model, or a human expert that decides the quality of a chromosome [8].

Once we have the genetic representation and the fitness function defined, GA proceeds to initialize a population of solutions randomly, and then improve it through repetitive application of mutation, crossover, inversion, and selection operators.

**5. SIMULATED OUTPUT RESULTS**

In this section, we solve some numerical examples applied by genetic algorithm using different parameters shown in the three case studies below:

**Case Study 1:**

Find max \( f = x^2 + 2x + 1 \), where \( 0 \leq x \leq 1 \) subject to the following assumptions:
- The maximum number of generation = 10
- Population size = 8
- Mutation probability = 0.06
- Crossover probability = 0.6

<table>
<thead>
<tr>
<th>Generation N°</th>
<th>Fitness function</th>
<th>Value of x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.263267</td>
<td>0.806452</td>
</tr>
<tr>
<td>3</td>
<td>3.500520</td>
<td>0.870968</td>
</tr>
<tr>
<td>5</td>
<td>3.500520</td>
<td>0.870968</td>
</tr>
<tr>
<td>7</td>
<td>4.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>10</td>
<td>4.00000</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

Table 1 Simulated Output Results of Case Study 1

From this example, we see that we can achieve the optimum solution after the sixth generation. Then, max \( f = 4 \) at \( x = 1 \)

**Case Study 2:**

Find max \( f = y^2 - 2x^2 + 4x - xy - 4 \), where \( -2 \leq x \leq 2 \) and \( -3 \leq y \leq 3 \) subject to the following assumptions:
- The maximum number of generation = 20
- Population size = 8
- The total string length = 8
### Table 2 Simulated Output Results of Case Study 2

<table>
<thead>
<tr>
<th>Generation N°</th>
<th>Fitness function Value of x</th>
<th>Value of y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.528889 0.66667</td>
<td>-2.20000</td>
</tr>
<tr>
<td>3</td>
<td>5.755556 0.93333</td>
<td>-2.20000</td>
</tr>
<tr>
<td>5</td>
<td>6.840000 2.00000</td>
<td>-1.80000</td>
</tr>
<tr>
<td>7</td>
<td>9.240000 2.00000</td>
<td>-2.20000</td>
</tr>
<tr>
<td>9</td>
<td>9.240000 2.00000</td>
<td>-2.20000</td>
</tr>
<tr>
<td>11</td>
<td>15.00000 2.00000</td>
<td>-3.00000</td>
</tr>
<tr>
<td>15</td>
<td>15.00000 2.00000</td>
<td>-3.00000</td>
</tr>
<tr>
<td>20</td>
<td>15.00000 2.00000</td>
<td>-3.00000</td>
</tr>
</tbody>
</table>

From this example, we see that we can obtain the optimum solution after 11 generation. Then, max f = 15 at x = 2, y = -3

### Case Study 3:

Find max f= 9x+2y+5z, where -2 ≤ x ≤ 2, -3≤ y ≤ 3, and -1≤ z ≤ 1 subject to the following assumptions:
- The maximum number of generation = 20
- Population size = 8
- The total string length = 12
- The sub string length = 4
- Mutation probability = 0.06
- Crossover probability = 0.6

<table>
<thead>
<tr>
<th>Generation N°</th>
<th>Fitness function Value of x</th>
<th>Value of y</th>
<th>Value of z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.13333 2.00000</td>
<td>-2.60000</td>
<td>-0.73333</td>
</tr>
<tr>
<td>4</td>
<td>13.3333 2.00000</td>
<td>-2.60000</td>
<td>0.06666</td>
</tr>
<tr>
<td>6</td>
<td>16.3333 2.00000</td>
<td>-1.00000</td>
<td>0.06666</td>
</tr>
<tr>
<td>8</td>
<td>22.60000 2.00000</td>
<td>-0.20000</td>
<td>1.00000</td>
</tr>
<tr>
<td>10</td>
<td>22.60000 2.00000</td>
<td>-0.20000</td>
<td>1.00000</td>
</tr>
<tr>
<td>12</td>
<td>29.00000 2.00000</td>
<td>3.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>16</td>
<td>29.00000 2.00000</td>
<td>3.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>20</td>
<td>29.00000 2.00000</td>
<td>3.00000</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

From this example, we notice that we can obtain the optimum solution after 12 generation. Then, max f = 29 at x = 2, y = 3, and z = 1

We notice that after being duplicated for more than one iteration and applied on the same parameters, we reach to the optimum solution and take the final solution to compare it within the exact solutions.

From the results of the tables (1, 2 and 3) which indicate G.A. in numerical examples, we see that the solution is almost equal to the analytical one. Nevertheless on using the G.A. to solve these numerical examples, we obtain the result very swiftly.

### 6. CONCLUSION

As a high efficient search strategy for global optimization, genetic algorithm demonstrates favorable performance for solving the optimization problems.

Optimization problems can be solved with genetic algorithm through efficient encoding, selection of fitness function and various genetic operations.

Crossover is identified as the most significant operation to the final solution. From the result which indicates G.A. for solving in optimization problems we see that we can reach to the optimum solution and the solution is almost equal to the analytical one. Nevertheless on using the G.A. to solve these optimization problems, we obtain the result very swiftly.

As a future work, we plan to apply genetic algorithms for training and designing intelligence technique systems such as artificial neural networks to predict a three dimensional protein structure.

### REFERENCES

