Pop, J Comput Sci Syst Biol 2016, 9:4
DOI: 10.4172/jcsb.1000227

Journal of
Computer Science & Systems Biology

Research Article Open Access

A Novel Improved Genetic Algorithm Based on the Fixed Point Theorem and Triangulation Method

Iulian Pop*

North University Center at Baia Mare, Technical University of Cluj-Napoca, Romania

Abstract

Genetic algorithms are widely used for solving a multitude of complex problems. Methods for increasing the genetic algorithms accuracy are of great importance. One such method consist of using the fixed point theory combined with an improved genetic algorithm. This paper aims to provide such an algorithm with an increased convergence accuracy. In order to increase the algorithm operates on a simplicial triangulation over the searching space. We use the crossover, mutation and increase dimension genetic operators to obtain a convergent population that contains only fully labelled simplexes. In order to obtain better results, we use a custom increase dimension operator that significantly boosts the overall fitness. The increase dimension operator converts non-labelled individuals into nearly labelled/fully labelled ones. The solution, the global optimum point, is obtained after applying the Hessian matrix onto the final population. Our obtained results clearly show an improvement over existing work due to implementation particularities of the algorithm.

Keywords: Improved genetic algorithm; Fixed point theorem; Triangulation method; Dual multimodal functions

Introduction

Standard genetic algorithms (GA) have revolutionized computing technology representing solid and adaptive models that are based on the idea of natural selection [1,2]. GA are designed to emulate the processes observed in natural evolution. GA are used as search and random optimization techniques but are structured algorithms that demonstrate their efficiency by operating on stored information to create new offspring with superior performance.

GA propose a gradual approach towards an optimal solution starting from an initial population. The main problem with standard GA is that they greatly depend on the chosen initial population and can be easily trapped in a local extreme point.

In order to overcome this problem Zhang et al. [2] has used the triangulation method to improve the research behavior of the GA. This is done by combining two search methods, a local and a random one, in order to obtain the global optimal solution without stalling. The triangulation method basically consist of populating the three vertices of a simplex by starting from a random point and computing the three vertices coordinates by using a constant parameter, h. Zhang et al. [1], multiple values were tested for the h parameter in order to compute the global optimum point. The h parameter represents the main constant value used by the triangulation method.

An attempt to improve the genetic algorithm's results uses the fixed-point theory in conjunction with a genetic algorithm [3]. For more information on fixed-point we refer to Ref. [4-7].

Zhang et al. [8,9] combined the fixed-point theory and the triangulation method with a genetic algorithm in order to increase the accuracy of the results. The necessary and sufficient condition of the extreme point is $\nabla f(x^*)=0$ meaning that the point gradient equals to 0. Since g: $R_n \to R_n, (x \in R_n)$, the optimization problem is converted to a fixed point problem using $g(x)=x-\nabla f(x)$. After the conversion the genetic operators are applied to the result.

The aim of our paper is to describe a novel improved genetic algorithm for solving dual multimodal functions. As we will see in Section 3, our proposed method delivers better results in comparison with existing work [1,2].

Our paper is organized in four main sections. It starts with this introduction and continues with section two that highlights the GA and its operators. In addition we analyze the representation, parameters, triangulation and fixed point implementation details. Section three contains the computational results compared with the Zhang et al. [1,2]. Finally, the conclusions section summarizes the obtained results and presents future research directions.

The Developed Improved Genetic Algorithm

In this section we present the implementation particularities of the novel improved genetic algorithm. This novel genetic algorithm features all the major components of a standard GA and a new one for the increase dimension operator. All the components of the developed algorithm are presented in Figure 1.

The first step in our genetic algorithm is to set the genetic parameters. After that the initial population is generated based on these genetic parameters. The initial population consists of a list of simplexes. We then use the genetic operators to evolve the population until we have a convergent population, meaning a population of completely labelled simplexes. These genetic operators used are:

1. Crossover
2. Mutation
3. Increase Dimension
4. Selection Operator

Since we use a dual multimodal function we have multiple optimal

***Corresponding author:** Iulian Pop, North University Center at Baia Mare, Technical University of Cluj-Napoca, Romania, Tel: +40720031123; E-mail: pop.yulyan@gmail.com

Received May 26, 2016; **Accepted** June 09, 2016; **Published** June 14, 2016

Citation: Pop I (2016) A Novel Improved Genetic Algorithm Based on the Fixed Point Theorem and Triangulation Method. J Comput Sci Syst Biol 9: 105-111. doi:10.4172/jcsb.1000227

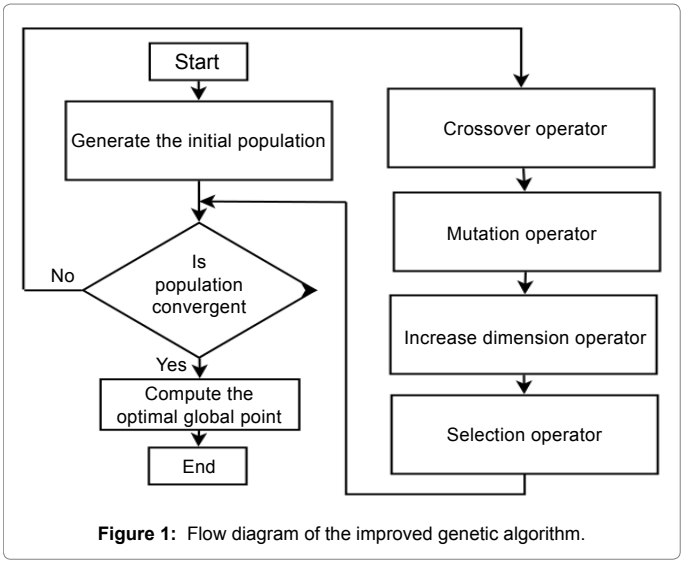

Figure 1: Flow diagram of the improved genetic algorithm.

local points. In order to obtain the optimal global point, the Hessian matrix is applied on all local optimal points resulting the solution. Our algorithm implementation was built in C#, using the Maple API in order to solve mathematical equations. A friendly user interface was created for demonstrating the algorithm's efficiency and stability.

Genetic representation

The goal of our algorithm is to find completely labeled simplexes that represent optimal local points. Since each simplex has three vertices we use the following data structure:

1. A point data structure has only two float properties, X and Y. This is a basic structure used to build up more complex ones on top.

2. A vertex data structure is derived from the point basic data structure that will be used to map the three vertices of each simplex. Besides the usual point properties a vertex also has the following:

a) Label is a value used to store the computed integer label value (0, 1 or 2). The label of each vertex of the simplex is calculated applying the formula (1);

$$f(x)=\begin{cases}0, if\ g_1(x)-x_1\geq 0, g_2(x)-x_2\geq 0\\ 1, if\ g_1(x)-x_1<0, g_2(x)-x_2\geq 0\\ 2, if\ g_2(x)-x_2<0\end{cases} \quad (1)$$

where g_1 and g_2 represent the the conversion of the function f to a fixed point problem [8,10] with respect to x_1 and x_2 variables.

b) Hessian determinant is used to store the value of the computed Hessian determinant;

c) The computed function value is used to store the calculated function value for this vertex(point);

3. A simplex is constructed from three vertexes labelled Vertex0, Vertex1 and Vertex2. Based on these vertices the simplex type is computed. Also the original point of generation is kept as a reference.

4. The SimplexType is an enum that has the following possible values: None, NearlyCompletedLabled and CompletlyLabled. We compute the type following the rules:

a. Completely labelled - if its labels have three distinct values 0,1 and 2

b. Nearly-completely - if its labels have two distinct values 0,1 or 0,2 or 1,2

c. Non-labelled – if its labels have only one distinct value 0 or 1 or 2

5. The population is composed of a list for simplexes. An IsConvergent flag is used to quickly and conveniently check if the population is convergent.

Initial population

The generation of the initial population plays an important role in a standard genetic algorithm. The initial population generation technique influences the probability and the required time to obtain the desired solution. There are two possible generation methods for the initial population:

- Random: This method uses a pseudo random generator to obtain random individuals.
- Constructive: This method of generating the initial population does not rely on creating the individuals randomly and implies the use of a schema in order to have a better control over the resulting population.

Due to the fact that this method can generate sub-optimal solutions or even optimal ones the genetic algorithm's efficiency and speed can be greatly improved [11].

In our algorithm, we generate the initial population by choosing random points (x, y) from the function's definition domain. The algorithm then generates a simplex from each point, each with its own three vertices and label. This list of simplexes represents the initial population.

Fitness

It's calculated based on simplex type. The evaluation order is from the fittest to the weakest: completely labelled, nearly-completely labelled and non-labelled. Because we don't have more degrees of fitness it is very difficult to choose between individuals when selecting for the next generation. That is why we chose to use one current population that increases in size during crossover and mutation and gets trimmed to the initial size during selection.

Genetic operators

In order to increase the algorithm's efficiency we modified the genetic operators to obtain the best results.

The crossover operator: Based on the crossover probability input parameter a group of individuals are selected for the crossover operation. Since this operation is applied on two simplexes, the selected number of individuals must be multiple of two. We then select the ordered parents (groups of two simplexes), and run them through the crossover operation, resulting in a new child that is added to the population. This operation increases the number of individuals from the population based on the crossover probability, for example choosing a crossover probability of 0.6 will result in a population increase of approximately 60% (more simplexes) than the current population.

The crossover operates on two input parents and generates a new child [11]. All vertexes from the two parents are extracted and from them the required three vertexes for the new child are selected. The vertex selection process takes vertexes with different label first, if any, and then selects the remaining ones from the vertexes that have not been chosen. The purpose is to select from the two simplexes whose vertices together have the highest fitness. The higher fitness is closer to a completely labeled simplex. A vertex cannot be chosen two times. Except the label selection priority (different label vertexes are selected first), the vertexes are chosen in the following order: firstParent. Vertex0, firstParent.Vertex1, firstParent.Vertex2, secondParent.Vertex0, secondParent.Vertex1, secondParent.Vertex2.

Mutation operator: Based on the mutation probability input parameter a group of individuals are selected for the mutation operation. The mutation probability is inversely related to the fitness as this operation is not applied to completely labeled individuals. The resulting mutated individual is added to the population thus increasing the population by keeping the original individual. This ensures that the population overall fitness is not decreased by generating an individual with lower fitness. The operation consists of:

- Only the non-labelled and nearly-completely labelled individuals are processed because mutating completely labelled individuals can only result in an individual with a lower fitness. The simplexes are chosen based on the mutation probability parameter.
- The first vertex from the selected simplex is mutated by subtracting the pre-set mutation value from its base coordinates (x, y). From this processed value a new point will be generated with the following coordinates (x-mutationValue, y-mutationValue). This new mutated point will be used to generate a new mutated simplex based on the H parameter and triangulation method. The mutationValue should be smaller than h in order for the newly created vertex to stay in the definition domain of the function.

Increase dimension operator: The increase dimension operator updates the population by trying to convert the weakest individuals (non-labelled) into better ones (nearly-completely labelled or even completely labeled). It operates on one individual at a time. This operator does not handle nearly-completely labelled or completely labeled individuals as it tries to prevent the genetic algorithm getting stuck in a local area. Since this operator only works with non-labelled individuals it updates the individual and does not generate a new changed individual as it cannot decrease the population overall fitness. Starting from the increase dimension operator presented by Zhang [12] we implemented our version consisting of the following steps:

1. The first step is to generate a vector from the first and second vertex. Based on it a normalized vector is generated.
2. According to a predefined stepSize [2.6] we go through the normalized vector and for each step we generate a new point.
3. For each new point we generate a new simplex based on that point, h and the triangulation method.
4. We check if the new simplex is non-labelled.
5. If the simplex is non-labelled the algorithm continues. If a solution is not found the individual is not updated.
6. If the simplex is nearly-completely labelled or completely labelled the algorithm stops as we have found the value that we were looking for. The individual is replaced in the population.

Selection operator: The selection operator chooses the fittest individuals first. It selects the first PopulationSize number of simplexes from the current population ordered by simplex type: completely labeled, nearly-completely-labeled or non-labeled. The result represents the new generation that has PopulationSize number of simplex. This way we keep the generations at a constant length. Also a domain check is done on the individuals in order to make sure that if an invalid simplex has been generated, it will not be selected in the next generation.

Stop condition

Like all standard genetic algorithms we need a stop condition, a way to determine that the algorithm has found the required solution and that it needs to stop. Usually the stop condition is quite simple, and it involves a simple check. Since our algorithm handles simplexes, the stop condition is linked to this entity and consists of checking that all simplexes are completely labelled. Since each simplex knows its state, the stop condition implies checking the state of all simplexes.

Genetic parameters

The novel improved genetic algorithm like any other genetic algorithm requires a few input parameters. This input parameters affect the algorithm's speed and accuracy. From the standard genetic parameters list, we supply the population size and the well-known crossover/mutation probabilities. In our particular case, we also supply the algorithm with the function that will use for calculations and the definition domain. Also we provide a constant value that will be used for the triangulation process.

- Function - is the field where you can enter a new function that is intended to be processed;
- Domain (Min and Max) - fields that contain the definition domain of the function;
- H - Is the value that is added to vertex0 to obtain vertex1. Based on vertex1's values a new vertex, vertex2 is obtained by adding the value of h. This is the way the triangulation is done. We ensure that the value of the constant h, required for triangulation is sufficiently small in relation to the definition domain. In the used test cases, the definition domain is set to [-3, 3] and we choose h to be one of the following values 1, 0.5, 0.1.
- PopulationSize - is the size of the population, representing the number of the simplexes in one generation;
- Crossover Probability=represents the crossover probability;
- Mutation Rate - probability that the mutation occurs of an individual;
- Mutation Value - value that is added to a vertex.
- StepSize - is a numeric value representing the step size used by the increase dimension operator [2.4.3]; it is used to generate new points from a vector stating from the start of the vector to the end.

Triangulation and fixed point implementation

As a computation engine for our novel improved genetic algorithm we use Maple, via the OpenMaple interface.

This interface allows us to embed calls to the Maple engine directly in our application.

This is accomplished by referencing its dynamic-link library (.dll) file, maplec.dll, which is located in the Maple binary directory.

We use the C# interface, as there are many programing languages supported, to access the OpenMaple API.

The first step is to encode the input function in a maple compatible format,forexample: "$f := 2\times x_1^2 - 1.05\times 1^4 + (1\div 6)\times x_1^6 - x_1\times x_2 + x_2^2$" . After that we compute the partial derivative of the function using the Maple DIFF command for both x_1 and x_2 (2)(3).

$$fDiffx_1:=diff(f,x_1) \quad (2)$$

$$fDiffx_2:=diff(f,x_2) \quad (3)$$

Based on (2)(3) the problem to be optimized is converted into a fixed-point problem using the following formula $g(x)=x-\nabla f(x)$. The maple commands used for converting to a fixed-point problem are shown in (4)(5).

$$fGx_1:=x_1-diff(f,x_1) \quad (4)$$

$$fGx_2:=x_2-diff(f,x_2) \quad (5)$$

In order to generate each vertex's label we use the (6)(7) maple commands that uses the fixed-point problem formulas previously defined (4)(5).

$$fLx_1:=fGx_1-x_1 \quad (6)$$

$$fLx_2:=fGx_2-x_2 \quad (7)$$

After the genetic algorithms runs, we obtain many minimal local points, which are completely labeled simplexes.

To find the fixed point, we search across the local minimal points for the one who has the Hessian matrix positively defined. This means that it's determinant is greater than 0 thus resulting in the global minimum point.

Hessian matrix is calculated for all 3 simplex vertices, at the time of the vertex's creation thus it's stored at vertex level.

The (8)(9) maple commands are used to compute the Hessian determinant.

$$whith(VectorCalculus):fHessian(f[x_1,x_2]) \quad (8)$$

$$whith(VectorCalculus):fHessianDet:=(Hessian(f[x_1,x_2]),determinant)) \quad (9)$$ [2].

For computing the function value for a certain vertex we use the (10) maple command since the vertex is basically a point.

$$eval(f,[x_1=\{0\},x_2=\{1\}]) \quad (10)$$

Computational Results

In order to asses our algorithm we performed comparison on benchmarks instances from literature [1,2] where available.

An application was created that has an intuitive interface Figure 2 through which you can easily change the fields needed for testing new sets of data.

We have tested the algorithm using the following function (11). The graphical representation of this function is presented in Figure 3.

$$f_1 : \min f_1(x) = 2x_1^2 - 1,05x_1^4 + \frac{x_1^6}{6} - x_1x_2 + x_2^2, -3 \le x_1 \le 3, -3 \le x_2 \le 3 \quad (11)$$

To test the efficiency of the new algorithm we have used two values for h (1, 0.5).

For each value of h (1, 0.5) we used three sets of dates as presented in Table 1.

Since Zhang et al. [1,2] has not published the algorithm he used nor the complete datasets used some fields in Table 1 don't have a value (N/A).

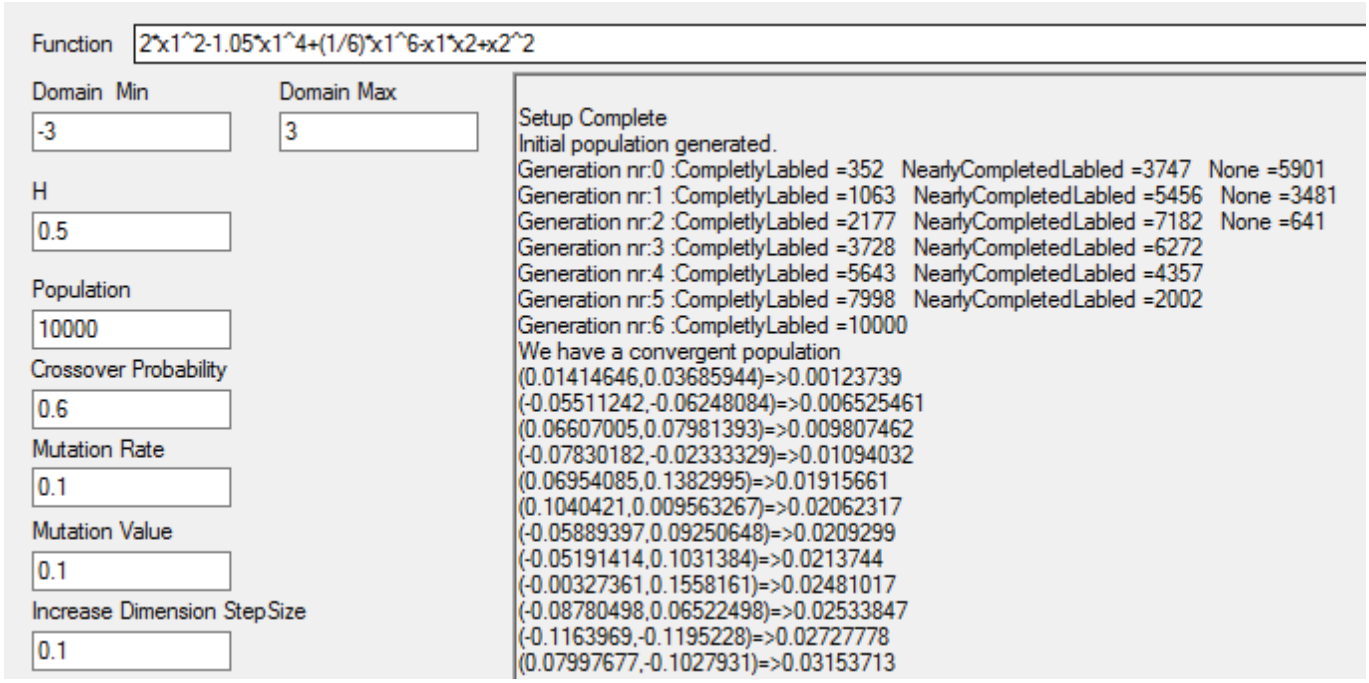

Figure 2: Application interface.

The best results from Table 1 are highlighted and were obtained using an initial population with 10000 individuals.

As the initial population is increased, the algorithm accuracy increases as well. These results are better than the ones obtained by Zhang et al. [1,2].

Choosing a Crossover Probability of 0.6 will result in a population increase of approximately 60% (more simplexes) compared with the current population. As clearly shown in Figure 4 the obtained results for subsequent generations are converging.

The results generated by the implemented algorithm are shown in Figures 5, 6 and 7. The red point represents the global minimal point of the function, having the coordinates (0.01485147, 0.01237264) and the function value *f* of 0.000410412.

We can observe in Table 2 that for lower value of the h parameter we

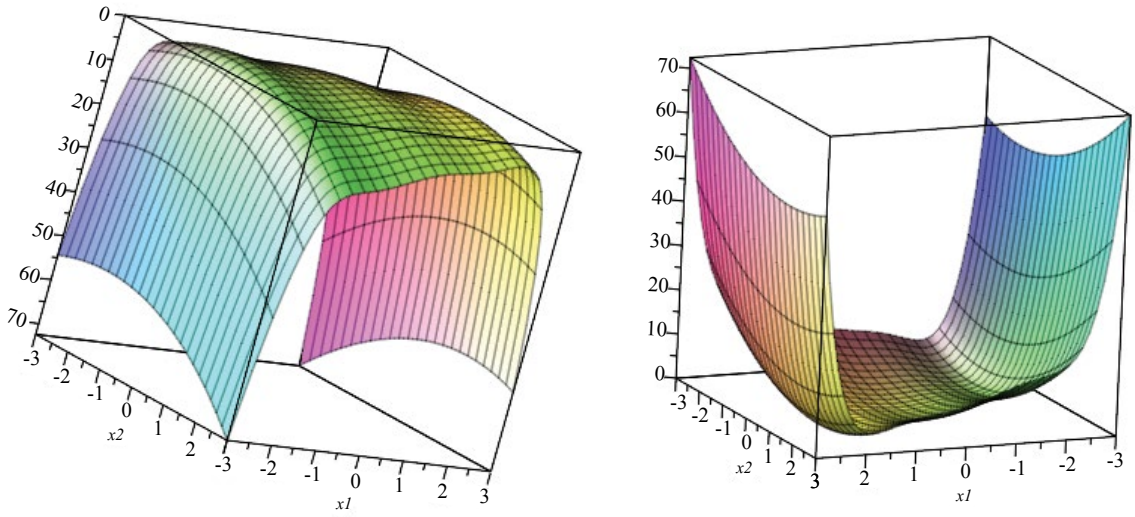

Figure 3: Graphic model for f_1.

Sets of values for GA [Table 3]	Point coordinates (x,y)	Our results (function value)	Zhang results (function value) [1,2]
Set of values 1	(-0.07025898, -0.05255371)	0.008916605	N/A
Set of values 2	(0.01575422, 0.03153596)	0.000994019	N/A
Set of values 3	**(0.01485147, 0.01237264)**	**0.000410412**	N/A
N/A	(0.036168, 0.002933)	N/A	0.002517

Table 1: Results for *h*=1.

```
Initial population generated.
Generation nr:0 :CompletlyLabled =383 NearlyCompletedLabled =4698 None =4919
Generation nr:1 :CompletlyLabled =1185 NearlyCompletedLabled =6574 None =2241
Generation nr:2 :CompletlyLabled =2356 NearlyCompletedLabled =7644
Generation nr:3 :CompletlyLabled =3863 NearlyCompletedLabled =6137
Generation nr:4 :CompletlyLabled =5738 NearlyCompletedLabled =4262
Generation nr:5 :CompletlyLabled =8111 NearlyCompletedLabled =1889
Generation nr:6 :CompletlyLabled =10000
We have a convergent population
```

Figure 4: Convergence of the algorithm.

Sets of values for GA [Table 3]	Point coordinates (x,y)	Our results (function value)	Zhang results (function value) [1,2]
Set of values 4	(0.01283163, -.03332525)	0.001867463	N/A
Set of values 5	(-0.0009697764, 0.004777217)	0.00002933557	N/A
Set of values 6	**(-0.001642168, -0.001397997)**	**0.000005052073**	N/A
N/A	(0.014164, 0.000000)	N/A	0.000412

Table 2: Results for *h*=0.5.

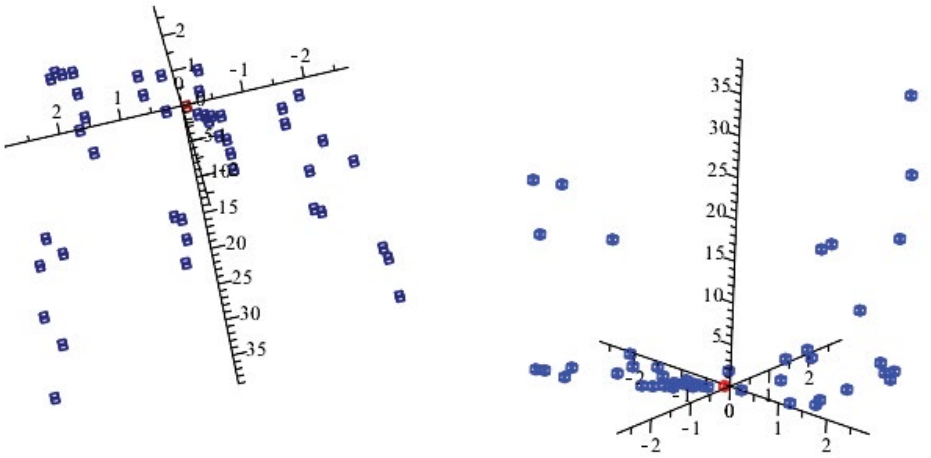

Figure 5: Population size=300.

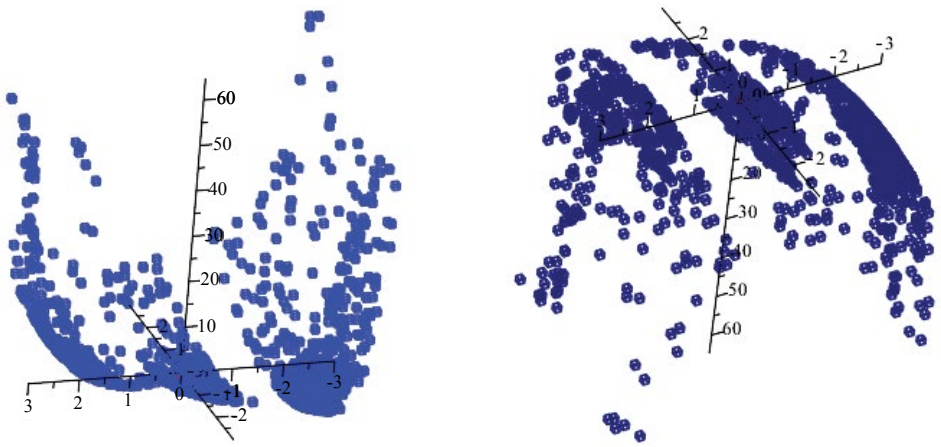

Figure 6: Graphic for 5000 individuals.

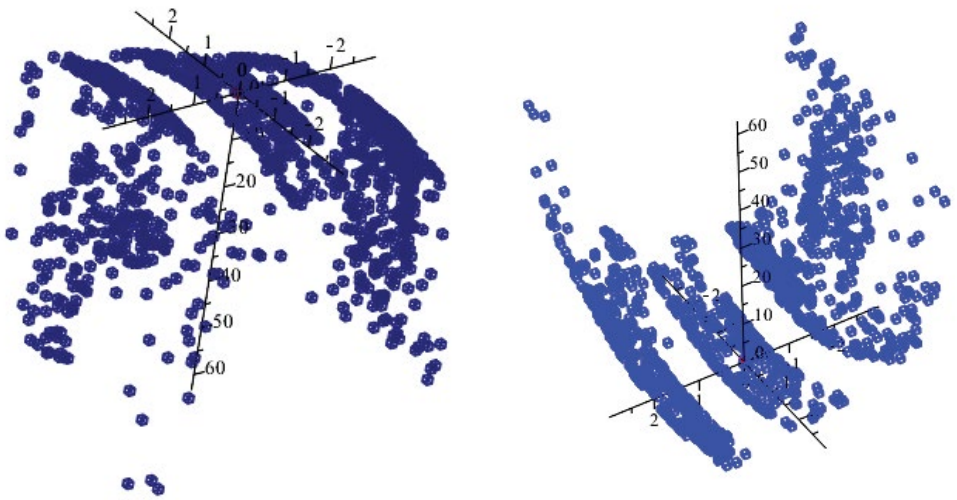

Figure 7: Population size=10000.

GA Parameter \ Set of values	Set of values 1	Set of values 2	Set of values 3	Set of values 4	Set of values 5	Set of values 6
Population	300	10000	10000	1000	5000	10000
Crossover Probability	0.1	0.1	0.6	0.1	0.6	0.6
Mutation Rate	0.9	0.1	0.1	0.6	0.1	0.1
Mutation Value	0.1	0.1	0.1	0.1	0.1	0.1
Increase Dimension Step size	0.1	0.1	0.1	0.01	0.1	0.1

Table 3: Sets of values for GA.

obtained better results. For these parameters the minimal global point has the following coordinates: (-0.001642168,-0.001397997) and the function value is 0.000005052073.

Since Zhang et al. [1,2] has not published the algorithm he used nor the complete datasets used some fields in Table 1 don't have a value (N/A). The best results are highlighted and are better than the ones obtained by Zhang et al. [1,2].

Conclusion

In this paper we described a novel improved genetic algorithm for solving dual multimodal functions. The main features of our approach are:

- We operate on a simplicial triangulation over the searching space;
- We used the fixed-point theory in conjunction with the GA;
- We obtain a convergent population by using custom GA operators.

In this way, we managed to significantly increase the accuracy of the improved genetic algorithm by implementing and tweaking the genetic operators and specially the increase dimension operator that manages to generate individuals with higher fitness.

Further on the focus is to adapt and optimize the algorithm for solving combinatorial optimization problems, as they represent a significant percentage of total optimization problems, for example Committed Travelers problem, Knapsack problem, etc. (Table 3).

References

1. Jingjun Z, Hongxia W, Ruizhen G (2011) Study of an improved genetic algorithm based on fixed point theory and hK_1 triangulation in euclidean space. Jr Comp 6: 2173-2179.
2. Jingjun Z, Yanmin S, Ruizhen G, Yuzhen D (2008) An improved genetic algorithm based on hK_1 triangulation. International seminar on futer information technology and management engineering, Leicestershire, United Kingdom.
3. Gregory JER (1991) Foundations of Genetic Algorithms 1991 (FOGA 1). Morgan Kaufmann Publisher Inc., USA.
4. Wang Z (1993) Simplicial Fixed Points Algorithm. Press of National University of Defense Technology.
5. Hiroshi N, Chikahiro T, Hideki A (2005) An efficient learning algorithm for finding multiple solutions based on fixed-point homotopy method. Proceedings of International Joint Conference on Neural Networks, Montreal, Canada.
6. Bugajewski D (2000) Fixed Point Theorems in Hyperconvex Spaces Revisited. Math Comput Modell 32: 1457-1461.
7. Andrei B, Vasile B (2013) Applications of the PL homotopy algorithm for the computation of fixed points to unconstrained optimization problems. Creat Math Inform 22: 41-46.
8. Yuzhen D, Jingjun Z, Ruizhen G, Yanmin S (2009) An improved genetic algorithm based on hK_1 subdivision and fixed point. International conference on business intelligence and financial engineering, Beijing, China.
9. Yuzhen D, Jingjun Z, Ruizhen G, Yanmin S (2009) An improved genetic algorithm based on J1 subdivision and fixed point theory. International conference on intelligent human-machine systems and cybernetics, Hangzhou, Zhejiang, China.
10. Jingjun Z, Yuzhen D, Ruizhen G, Yanmin S (2009) An improved genetic algorithm based on fixed point theory for function optimization. International conference on computer engineering and technology, Singapore.
11. Michalewicz Z (2013) Genetic algorithms + data structures=evolution programs. Springer Science & Business Media, Berlin, Germany.
12. Jingjun Z, Yanmin S (2009) An improved genetic algorithm based on K1 triangulation with variable coefficient for optimization of multimodal functions. 4th IEEE Conference on Industrial Electronics and Applications, Xi'an, China.