Using Genetic Algorithm to Solve Travelling Salesman Optimization Problem Based on Google Map Coordinates for Duhok City Areas

Ziyad Hazim Abid Al-Jabbar

College of Management and Economics, Duhok University, Duhok, Kurdistan Region – Iraq

ABSTRACT

This research aims to solve one of the Non-Deterministic Polynomial (NDP) Problems by using one of the artificial intelligence techniques, which is genetic algorithm. Travelling salesman problem (TSP) is one of the difficult optimization problems, the aim of this problem is to get the optimal solution which is represented by the shortest path for (n) visited areas of the city. The number of possible solutions that will be generated, searched and compared when solving this problem for (n) areas is equal to (n!). This number exponentially increased with the increasing of the number of areas. With the large number of areas, which produces a huge number of possible solutions, the traditional search algorithms will be collapsed, and the problem will become a hard (NDP) Problem. In this case it becomes necessary to rely on artificial intelligence techniques, which are based on the biologically-inspired principle. During this research the travelling salesman problem was formulated and programmed in proportion to the concept of genetic algorithm (GA) to produce Travelling Salesman Genetic Algorithm (TSGA). One of the cities of Kurdistan Region of Iraq (Duhok) was selected as a case study to implement the TSGA algorithm. Initially, the study depends on Google earth program to determine the coordinates for number of Duhok's areas. These coordinates were saved as a (.kml) file format, then the required cleaning and normalization operations were accomplished on this file to produce the pure coordinates, that were stored as an excel file format (.xls) . TSGA algorithm depends on these excel coordinates as an input file to create the initial generation of paths, then the objective function for each path of the this generation was calculated, and then the parent selection, crossover and mutation functions were applied to get the group of the best paths. TSGA algorithm, then, continues to regenerate a number of successive generations, and afterwards recalculate and create the new group of the best paths for each generation to enhance the result. Finally, and depending on the criterion of stopping, this algorithm will cease to create new generations and suggest the final result that represents the shortest path for visited areas. Matlab program was used to implement the TSGA algorithm and to analyze the result. The results of this algorithm were optimum and near optimum for most of the problems at a reasonable time.

KEYWORDS : Genetic Algorithm, Optimization, Non Deterministic Problem, Travelling Salesman Problem, Google coordinate.

Academic Journal of Nawroz University
(AJNU) Volume 7, No 3 (2018).
Received 3 March 2018;
Regular research paper : Published 20 June 2018
Corresponding author's e-mail : Ziyad.alJabbar @gmail.com

1. Theoretical Part

1.1. Introduction:

The optimum (or near to optimum) solution is necessary to get successful decisions. Travelling Salesman Problem (TSP) is considered one of the linear optimization problems. The aim of this problem is to find the shortest path for number of visited areas. This problem becomes more difficult as the size of the problem matrix increases (i.e. increasing the number of areas to be visited), because of the need for a long time to accomplish a lot of search and comparison operations before reaching the final optimum solution that will

doi : 10.25007/ajnu.v7n3a207

finally represent the shortest path connecting these areas. As the problem becomes more difficult it is necessary to use artificial intelligence techniques. The genetic algorithm is one of these techniques. A survey study was carried out on the use of genetic algorithm in solving optimization problems and especially the research studies related with travelling salesman problem, however, unfortunately, all of these studies have not clearly articulated the model formulation of this problem in proportion to the concept of genetic algorithm [1, 2, 3]. Therefore, this research includes firstly, a presentation of the mathematical model of the travelling salesman problem and when the difficulty arises in this problem, and then an introduction to the genetic algorithm was presented with a comprehensive study of how to formulate and program the travelling salesman problem in proportion to the concept of the genetic algorithm. As a result to this study, a new algorithm named Travelling Salesman Genetic Algorithm (TSGA) was produced. In order to use realistic values, the applied case study was to select a path for a number of areas in the city of Dohuk. "Google Earth" program was used to determine the coordinates (latitude and longitude) of these areas. These coordinates were imported as a (.kml) file from google earth to excel program, after that, the required cleaning and normalization operations were accomplished to get the pure coordinate file (.xls) that will become a coordinate input file to the genetic algorithm. Genetic algorithm then will start with a number of steps that are represented with creating the initial generation, calculating the objective function for each path of the initial generation. Depending on random sampling, parent selection will be applied on the best parents of the current generation. Then, crossover and mutation functions will be applied to enhance the objective function for those selected parents. Genetic algorithm, then, continues to regenerate a number of successive generations. After that, recalculate the best path for each generation in order to suggest the final shortest path. Matlab program was used to program the TSGA algorithm and analyze the results. Three criteria were depended on to measure the efficiency of the results: distance of the path, number of generations needed to improve the solution and the required time to get the final solution. The results obtained through this research supported and confirmed the possibility of solving difficult travelling salesman problems using the genetic algorithm in a very short and reasonable period of time. The tables and shapes that are included in the research in both theoretical and practical parts are prepared by the researcher.

1.2. Motivation and Research Problem:

The inability of traditional search algorithms to solve difficult optimization problems like large dimensional matrix of travelling salesman problem which is impossible to solve using traditional search algorithm.

1.3. Research Hypothesis:

The ability to solve large dimensional matrix of travelling salesman problem by using one of the artificial intelligence techniques, Genetic Algorithm, to get the optimum or near optimum solution at a reasonable time.

1.4. Introduction to Travelling Salesman Problem (TSP):

All optimization problems have an objective function that may be either maximization or minimization. Travelling Salesman Problem is considered one of the optimization problems, the aim of this problem is to visit all the related cities with the shortest distance. This problem can be represented as an integer linear programming problem depending on the mathematical model formulation shown in table (1). [1].

Table (1) : Mathematical model formulation of the TSP problem

		Areas (1..n)			
	i				
j		1	2	…….	n
Areas (1..n)	1	d_{11}	d_{12}	…….	d_{1n}
	2	d_{21}	d_{22}	…….	d_{2n}
	.	.	.		.
	.	.	.		.
	.	.	.		.
	.	.	.		.
	n	d_{n1}	d_{n2}	…….	d_{nn}

doi : 10.25007/ajnu.v7n3a207

$$\text{Minimize } (z) = \sum_{i=1}^{n} \sum_{j=1}^{n} d(i,j) \cdot x(i,j) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$\mathbf{d(i,j)}$ = Direct distance between city i and city j.

$$\mathbf{x(i,j)} = \begin{cases} 1, & \text{if there is trammision from city i to city j} \\ 0, & \text{otherwise} \end{cases}$$

Subject to:

$$\sum_{i=1}^{n} x(i,j) \quad i=1,2,\ldots,n \quad n \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$\sum_{j=1}^{n} x(i,j) \quad j=1,2,\ldots,n \quad ١ \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The number of possible solutions that are generated when solving TSP problem for (n) cities is equal to (n!). It is possible to solve this problem by using traditional search algorithms if the number of cities is limited. For example, if the number of cities is (5), the number of possible solutions (alternative paths) is equal to (120). If the number of cities is large, for example (50), then the number of possible solutions equals (≈3E+64). This means the number of possible solutions exponentially increased with the increasing of the number of cities. With this large number of possible solutions, the traditional search algorithms will collapse, and the problem becomes a Non Deterministic Polynomial hard problem. In this case it is necessary to rely on artificial intelligence techniques. [4]. Figure (1) prepared by the researcher)

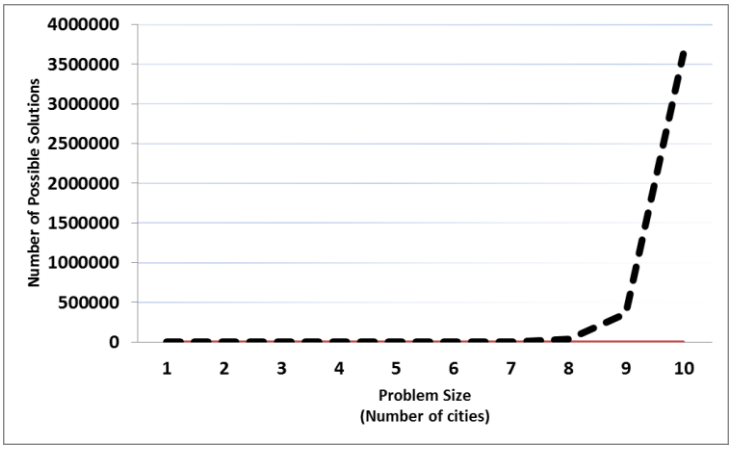


Figure (1) : Number of possible solutions associating with determined problem size

Since the sequence of the visited cities is important to suggest the shortest path, i.e. for (50) visited cities, the number of possible different sequences (alternative paths) for these cities is equal to (50!). Therefore, the optimal solution (path) is represented with the optimal sequence of visited cities. Figure (2, prepared by the researcher) illustrates all possible sequences of visited cities for a problem with size (4) [4].

doi : 10.25007/ajnu.v7n3a207

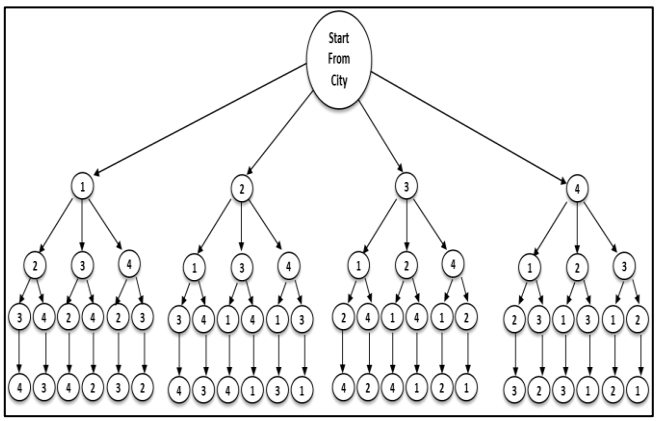


Figure (2) : All possible sequences of visited cities for the problem with size (4)

1.5. Introduction to Genetic Algorithm (GA):

GA is like other artificial intelligence techniques, such as; neural networks and fuzzy logic which are all based on the biologically-inspired principle. Artificial intelligence techniques are used to solve the problems that cannot be solved by using conventional methods. The idea of computing evolution was originated in the 1960s by I. Rechenberg through his research work "Evolution Strategies". In the year 1975, Genetic Algorithm was developed as a useful technique for optimization problems by John Holland in his book "Adaptation in Natural and Artificial Systems". GA consists of terms derived from natural evolution like Gene, Chromosome and Generation. [5, 6]

1.6. Travelling Salesman Genetic Algorithm (TSGA) Steps:

Genetic algorithm consists of a number of variables and functions. The general concept of these variables and functions is the same, but they differ in their formulation and representation in proportion to the specificity of the problem with its objective function and constraints. The following is an illustration of the representation of these variables and functions in proportion to the travelling salesman problem.

1.6.1. Representation for GA variables:

- **Gene:** Which represents the feature or character for the basic unit of the genetic optimization problem. One gene was allocated for each area. The value of each gene can be randomly assigned by the sequence number of each visited area.
- **Chromosome:** Which consists of a number of genes that, in this research, was equal to the number of visited areas. Each chromosome represented an individual table for each path of visited areas. That means one chromosome was assigned for each corresponding path, i.e. chromosome no.1 corresponding to path no.1, chromosome no.2 corresponding to path no. 2 and so on.
- **Generation:** Which is a collection of chromosomes (paths). It was represented by the determined number of paths. A large generation takes a long time to get the result, but it gives the best chance to get the optimal solution. A smaller generation takes a less calculation time, but the opportunity to get the optimal solution is less. When the algorithm starts, the numbers of visited areas of each path of the initial generation are created randomly. Then, the algorithm regenerates a number of successive generations depending on the selection of the best paths from the previous generation. Table (2) represents a generation consisting of "n" paths and "m" visited cities. The general steps that were required to create the initial generation can be represented as follows: [5,6,7]
- **Population:** Which represents the number of candidate chromosomes (paths) in contributing to the creation of the new generation. Therefore, it represents a part of the current generation and has a great potential to contribute to the creation of the new generation by being selected by the parents selection function, which will be explored later.

```
for (i = 1 ; i<=  No_Of_ Generated_path ; i++)
  {
     for (j = 1 ; j<=  No_of_Visited_Cities ; j++)
        {
Get the Random_No_for (City Number)
Put the Random_City_Number(j)  into the Generated_path_No (i)
         }
         Put the Generated_path_No (i) into the Generation
  }
```

doi : 10.25007/ajnu.v7n3a207

		Gene 1	Gene 2	Gene 3	…….	Gene m
	Path For visited cities	city 1	city 2	city 3	…….	city m
	Distance Between 2 cities	Distance between (m-1)	Distance between (1-2)	Distance between (2-3)	…….	Distance between (3-m)
Chromosomes 1- n	Path no. 1	city 2	city 3	city 1	…….	city 2
	Path no. 2	city m	city 3	city 2	…….	city m
	Path no. 3	city 1	city 2	city m	…….	city 1
	…….	…….	…….	…….	…….	…….
	Path no. n	city 2	city 1	city m	…….	city 2

1.6.2. Implementation of GA functions:

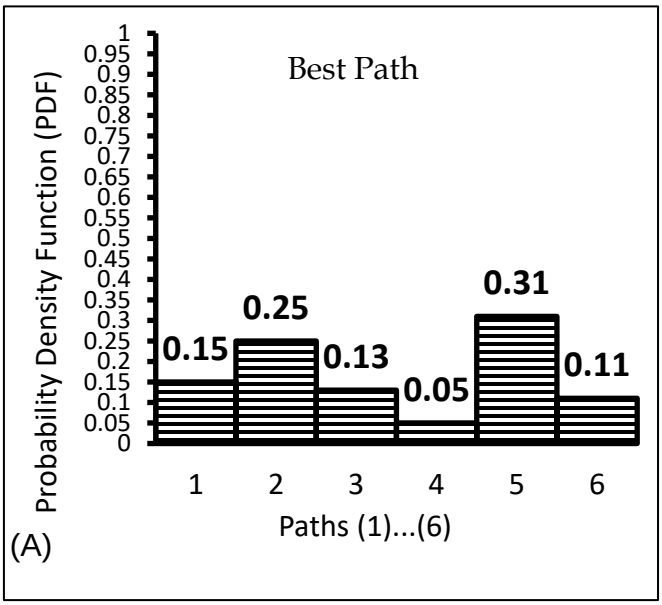
Fitness Function is one of the most important steps of genetic algorithm. It is used to select the parents that will contribute in the next generation construction. Fitness function depends on the objective function of optimization problem. It may be equal to the objective function, or it may need a little change. In travelling salesman problem, the objective function is to get the minimization, i.e. the best path is the shortest Path Distance (PD), and this path will produce the best values for Path Fitness (PF), Probability Density Function (PDF) and Cumulative Distribution Function (CDF). As a result, this path will have the highest probability of contributing to the creation of the new generation. Table (3) illustrates the calculation of PF, PDF and CDF values for a generation consisting of 6 paths for 4 areas. The following pseudo code shows the required steps to calculate the PF, PDF and CDF values. [5,6,7]. Figure (3).

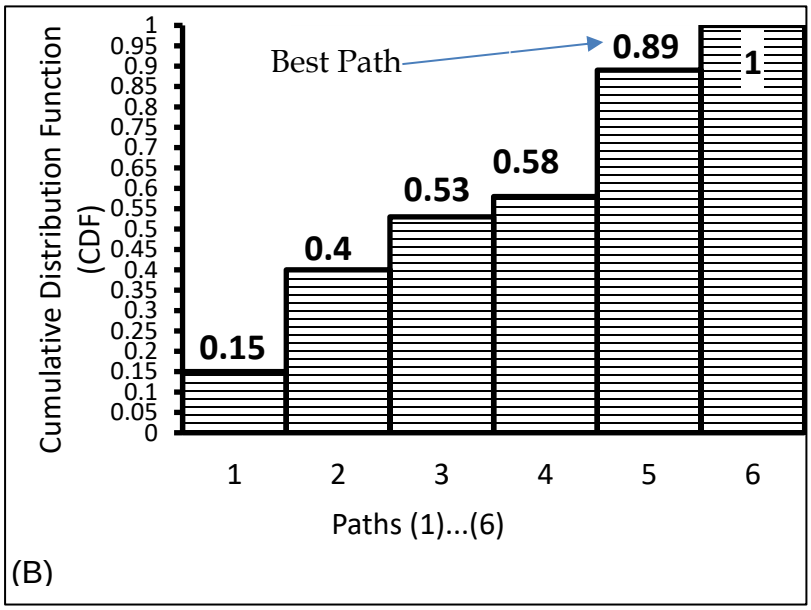
```
TF=0;
For ( i:= 1 ;i<=6;i++)
{ PF_i = ABS (PD_i − MaxTD) + MinTD ;
TF+ =  PF_i ; }
For ( i:= 1 ;i<=6;i++)
{ PDFi = PF_i/ TF  ; }
For ( i:= 1 ;i<=6;i++)
                {
                   CDF_i = 0.0 ;
                         For (j:= 1 ; j=i; j++)
                        CDF_i +=PDF_j  ;
                 }
```

Table (3) : Calculation of PF, PDF and CDF values for 6 different paths

Paths	Path Distance (PD_i)	Path Fitness (PF_i)	Probability Density Function (PDF_i)	Distribution Cumulative Function (CDF_i)
Path no. 1	11	7.8	0.15	0.15
Path no. 2	6	12.8	0.25	0.4
Path no. 3	12	6.8	0.13	0.53
Path no. 4	16	2.8	0.05	0.58
Path no. 5	2.8	16	0.31	0.89
Path no. 6	13	5.8	0.11	1
Maximum of Total Distance MaxTD=16 Minimum of Total Distance MinTD=2.8		TF = 52 Total Fitness	TP=1 Total Probability	

doi : 10.25007/ajnu.v7n3a207





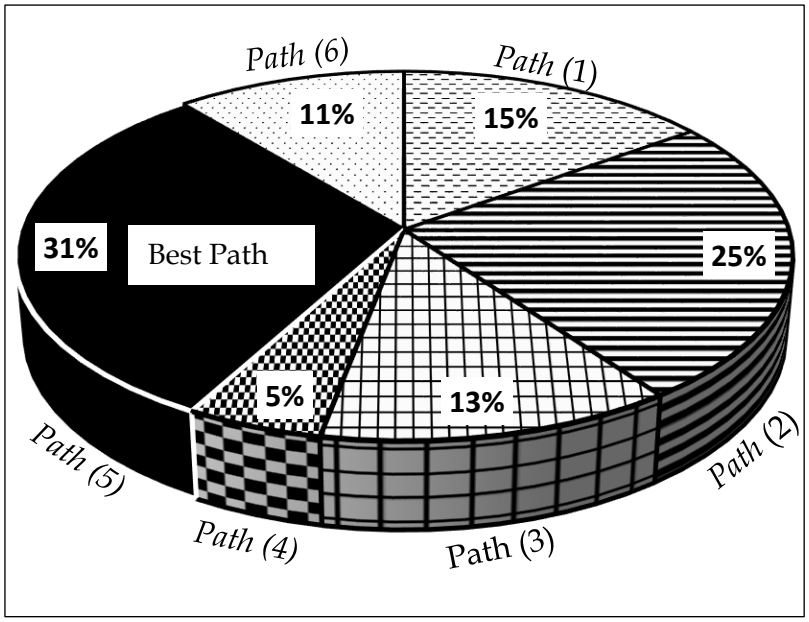


Figure (3): Random Sampling implementation for Parent Selection function
(A) Probability Density Function (PDF)
(B) Cumulative Distribution Function (CDF)
(C) Roulette Wheel

• **Parents Selection Function** works depending on the result of the fitness function for each parent, i.e. if the parent has a higher fitness value, it will have a greater chance to be chosen and to become a member of the next generation. Depending on the values of (PDF) and (CDF) for each path, random sampling operation can be applied to select a number of best paths from the current generation in order to create the new generation. Random sampling operation can be represented as a roulette wheel. As shown in table (3) and figures (3.A, 3.B and 3.C), path number (5) has the shortest Path Distance (PD=2.8), and thus, it will have the highest Path Fitness value (PF=16). As a result, path (5) will have a higher probability (Probability Density Function (PDF=0.31)) to be chosen as one of the best paths that contribute to create the next generation. [5,6,7]

• **Crossover Function:** The operation of this function is to make random exchanging between the multiple (points) values of the analogous city areas for two selected paths in order to configure a new path which has better fitness value than the original two paths. [5,6,7].

doi : 10.25007/ajnu.v7n3a207

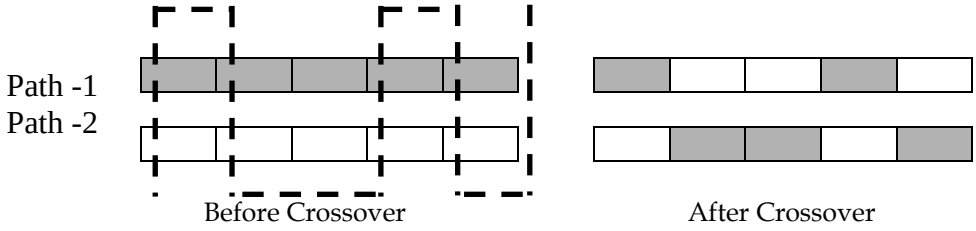


Figure (4) : Crossover Function

• **Mutation Function:** The output path from the crossover function will become an input to the mutation function. The operation of this function is to make a random switching between two specific areas within the path in order to improve fitness value for this path as much as possible and that path will be represented as a member in the next generation. Thus, this function will expand the generation with a number of suitable different good paths.

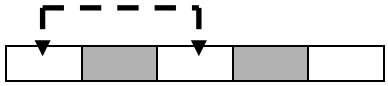


Figure (5) : Mutation Function

The following pseudo code illustrate the general steps that are required to implement the GA functions: [5,6,7].

```
For ( i = 1 ; i <= No_Of_ Generated_Paths_in_The_Current_Generation ;i++)
            {
Calculate  the Path_Fitness_Function (PF) (Objective_Function_for_Generated_Path i)
Calculate  the Probability_Density_Function (PDF) (Fitness_Function i)
Calculate Cumulative_Distribution_Function  (CDF) (Probability_Density_Function i)
    }
For  (  i = 1 ; i<=  No_Of_Paths_in_New_Generation; i++)
  {
Generate  Random_No_for (Path_1)
Parent_1=Drop the Random_No_for(Path_1) on CDF scheme.
Generate  Random_No_for (Path_2)
Parent_2=Drop the Random_No_for(Path_2) on CDF scheme.
New_Parent_C=Crossover (Parent_1, Parent_2)
New_Parent_M=Mutation(New_Parent_C)
Put the (New_Parent_M) into The_New_Generation.
      }
```

• **Stopping Criterion**: There are several methods for measuring the stopping criterion of the genetic algorithm, which vary with the type of the applied problem. This research is based on the pre-definition of a specific number of generations that does not give an improvement to the solution. With the creation of this number of generations, the implementation of the genetic algorithm stops.

2. Practical Part:

2.1. Software Required:

• **Google Earth** program provides the possibility of determining the coordinates of a group of areas. When these coordinates are connected they form a shape, i.e. the vertexes of the shape will represent the coordinates of the areas. Therefore, it is suitable to use Google Earth program for this purpose. Google coordinates are represented as longitude and latitude format. The average of differences between google coordinates and actual places is approximately equal to 21.08 meters. Therefore, the accuracy of google coordinates is considered suitable for this study. The coordinates of areas are stored as (.kml) file format. KML is one of the data file formats that refer to the Keyhole Markup Language. It was initially developed by Keyhole Inc. in 2004. KML is an XML notation that is used to express the geographic annotation and visualization within the Internet-based, two-dimensional maps and three dimensional earth browsers. KML became an international standard of the Open Geospatial Consortium in 2008.

• **Excel program** was used to implement the required cleaning and normalization operations on the imported

doi : 10.25007/ajnu.v7n3a207

KML coordinates file in order to produce the pure coordinates that were stored as an excel file format (.xls). This file will then be fed to the TSGA algorithm.

- **MATLAB** is a scientific software package that was used to design and program TSGA algorithm depending on the input excel coordinates file. MATLAB provides a variety of useful functions to implement the genetic algorithm. Given the ability and flexibility of MATLAB's high-level language, problems can be programmed in m-files with MATLAB's advanced data analysis, visualization tools and special purpose application domain toolboxes. The user is presented with a uniform environment to explore the potential of genetic algorithms. Furthermore, the Genetic Algorithm Toolbox uses the matrix functions to build a set of variety of tools for implementing a wide range of genetic algorithm methods. [10][11].

2.2. Practical steps to implement the (TSGA) algorithm:

TSGA algorithm was applied using realistic values of coordinates for 20 areas of the city of Dohuk, i.e. we have to apply travelling salesman problem on a matrix with the size of (20×20). This size of matrix needs (≈2.4E+18) of searches and comparison operations which are impossible to accomplish using traditional search algorithms.

The practical part of the research was divided into three main steps:

- **Step one:** Preparing Coordinates:

This step was assigned to prepare the matrix of coordinates, which included the following sequencing tasks: Figure (6.A).

1. Identifying the area's points of the path on the Google earth program. Figure (8)
2. Extracting the area's coordinates from Google earth program as a (.kml) file. Figure (7)
3. Making the required cleaning and normalization operations to get the pure coordinates which were saved as an excel file (.xls). This will become the input file to TSGA algorithm. Table (4)

Table (4) : Excel file for the coordinates of determined areas

Seq.	City	Google earth coordinates	
		Longitude	Latitude
1	Someel	42.85004188	36.85855113
2	AvroCity	42.91390475	36.85347364
3	Malta	42.93993068	36.85715058
4	Shandokha	42.96433464	36.8498984
5	Police neighborhood	42.99584323	36.84762183
6	Military neighborhood	43.02555993	36.84745414
7	Nazarki	43.04410478	36.84954492
8	Eitita	43.05035777	36.85130169
9	Industrial Area	43.04330454	36.85246871
10	Baroshki	43.0079746	36.85927203
11	Jalli Market	42.99882876	36.86260201
12	Krebassi	42.98889517	36.86826888

doi : 10.25007/ajnu.v7n3a207

13	Kayar City	42.9823449	36.86670561
14	Dabin1	42.97801055	36.86782658
15	Kro	42.96292464	36.86822905
16	Zariland	42.96559325	36.87137374
17	Shahki	42.95854773	36.87141811
18	Masika	42.94366327	36.87246105
19	Dabin2	42.94559599	36.87731415
20	Zarka	42.93120548	36.86891228

• **Step two:** Programming and Executing the TSGA algorithm: Figure (6.B)

This step depends on the coordinate file that was created in the previous step. This step is the actual implementation of TSGA algorithm which included the following steps:

1. Creating the initial generation, which consists of a number of paths.
2. Calculating the objective function, fitness value, probability density function and cumulative distribution for each path of the generation.
3. Forming the new generation based on the contribution of the best paths for the current generation, where paths with a high degree of fitness will have a higher opportunity to contribute to the formation of the new generation. Thus, the parent selection function will depend on random selection function, which is based on the fitness value of each path.
4. Applying the crossover function on each pair of selected paths, and the path resulted will pass to the mutation function.
5. The enhanced path, which is produced by the mutation function, will be added as a new member of the new generation.
6. Objective function and fitness value will usually be calculated after the completion of the creation of each new generation.
7. The regeneration process will continue to regenerate new generation(s) in order to enhance the last enhanced path of the last generation.
8. The execution of TSGA algorithm stops when a predefined value - the number of generations that are created without enhancing the last solution - is reached and suggests the final enhanced path as an optimum or near to optimum solution.

3. Results and Discussions: Figures (9-14)

1. In the beginning, the (tsga.m) program showed the coordinates of the determined (20) areas of Duhok city. (Figure 9).
2. After re-executing the (tsga.m) program five times (Table 5), we concluded the following:

a. There is a difference in the number of generations, e.g. the execution number (1) requires regenerating (71) generations, and execution number (5) requires regenerating (23) generations.

b. The required time for each one of the five executions is very short compared to the size of the problem.

c. All five executions gave the same result (shortest path distance) that was equal to (41.4466 KM). This supports the optimality of the result.

Table (5) : Results of executing the TSGA program

Execution Number	Figure Number	Number of Regenerations	Time required (Second)	Shortest Path (KM)
1	10	71	7.43	41.4466
2	11	69	2.49	41.4466
3	12	58	2.58	41.4466
4	13	38	2.56	41.4466
5	14	23	2.51	41.4466

4. Conclusions:

TSGA algorithm was designed and programmed to solve large dimensional matrices for TSP problem in order to get the optimum or near optimum solution at a reasonable time. According to the results, the problem with size (20X20), which needs (22.4E+18) of search and comparison operations, was difficult to solve in traditional methods. But, in this research, this problem was solved in a short time ranging between (2.51-7.43 Sec.), and gave an optimum solution that was represented by the shortest path distance equal to (41.4466 Km). Although different sequences were adopted for the same areas, the TSGA algorithm suggested the same shortest path with a different number of generations. That means, in the case of difficult TSP, or any Non Deterministic Polynomial

doi : 10.25007/ajnu.v7n3a207

problems, it is necessary to depend on artificial intelligence techniques and especially genetic algorithm, because it gave high efficiency in terms of optimality of the results and the time required. This research recommends developing the TSGA algorithm to solve the large-scale optimization problems using the idea of parallelism and FPGA (field-programmable gate array) technologies.

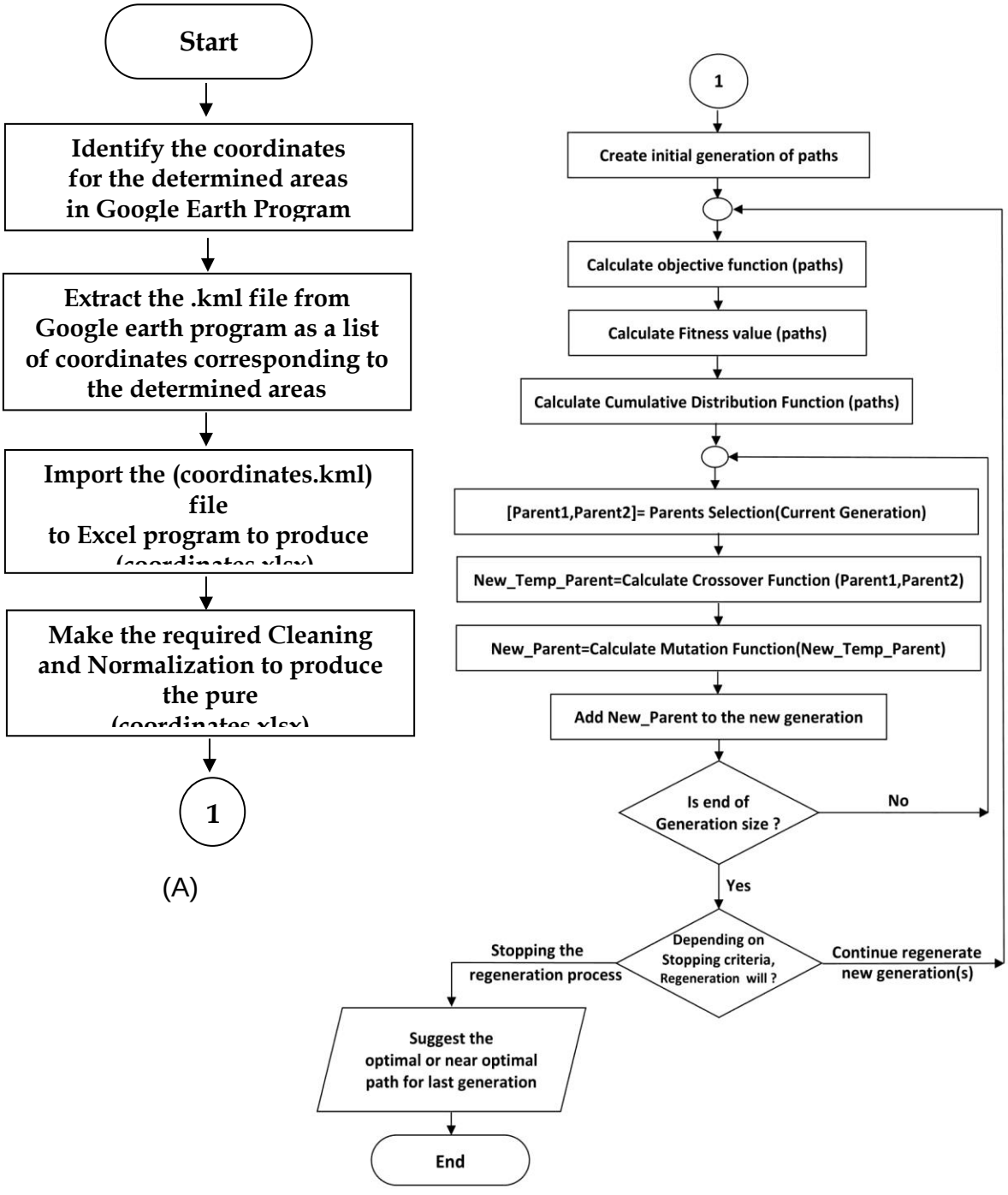


Figure (6): General steps for the research work
(A) Preparation of the matrix of coordinates
(B) General steps for the TSGA

doi : 10.25007/ajnu.v7n3a207

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:gx="http://www.google.com/kml/ext/2.2"
xmlns:kml="http://www.opengis.net/kml/2.2" xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
	<name>KmlFile</name>
	<Style id="inline2">
		<LineStyle>
			<color>ff0000ff</color>
			<width>2</width>
		</LineStyle>
		<PolyStyle>
			<fill>0</fill>
		</PolyStyle>
	</Style>
	<StyleMap id="inline4">
		<Pair>
			<key>normal</key>
			<styleUrl>#inline2</styleUrl>
		</Pair>
		<Pair>
			<key>highlight</key>
			<styleUrl>#inline3</styleUrl>
		</Pair>
	</StyleMap>
	<Style id="inline3">
		<LineStyle>
			<color>ff0000ff</color>
			<width>2</width>
		</LineStyle>
		<PolyStyle>
			<fill>0</fill>
		</PolyStyle>
	</Style>
	<Placemark>
		<name>Path for some Duhok areas</name>
		<styleUrl>#inline4</styleUrl>
		<LineString>
			<tessellate>1</tessellate>
			<coordinates>
42.8500418844087 3,36.85855112616122,0 42.91390475050901,36.85347364047421,0
42.93993068470527,36.85715057538469,0 42.96433464271895,36.84989839798274,0
42.99584323384502,36.84762183215475,0 43.0255599324226,36.84745413638164,0
43.04410477516732,36.84954492417069,0 43.05035776683565,36.85130168875801,0
43.04330454267287,36.85246871300858,0 43.00797460250042,36.85927203108634,0
42.99882875665496,36.86260200642651,0 42.98889516556352,36.86826887793087,0
42.98234490300371,36.86670561105295,0 42.97801054966512,36.86782657877689,0
42.96292463877568,36.86822905308405,0 42.96559324727517,36.87137374481948,0
42.9585477273833,36.87141811427765,0 42.94366327407882,36.8724610490341,0
42.9455959861743,36.87731414526456,0 42.93120547573708,36.86891228429226,0
			</coordinates>
		</LineString>
	</Placemark>
</Document>
</kml>
```

Extracted coordinates (Longitude and latitude) for 20 areas

doi : 10.25007/ajnu.v7n3a207

Figure (7) : KML coordinate file extracted from Google earth program

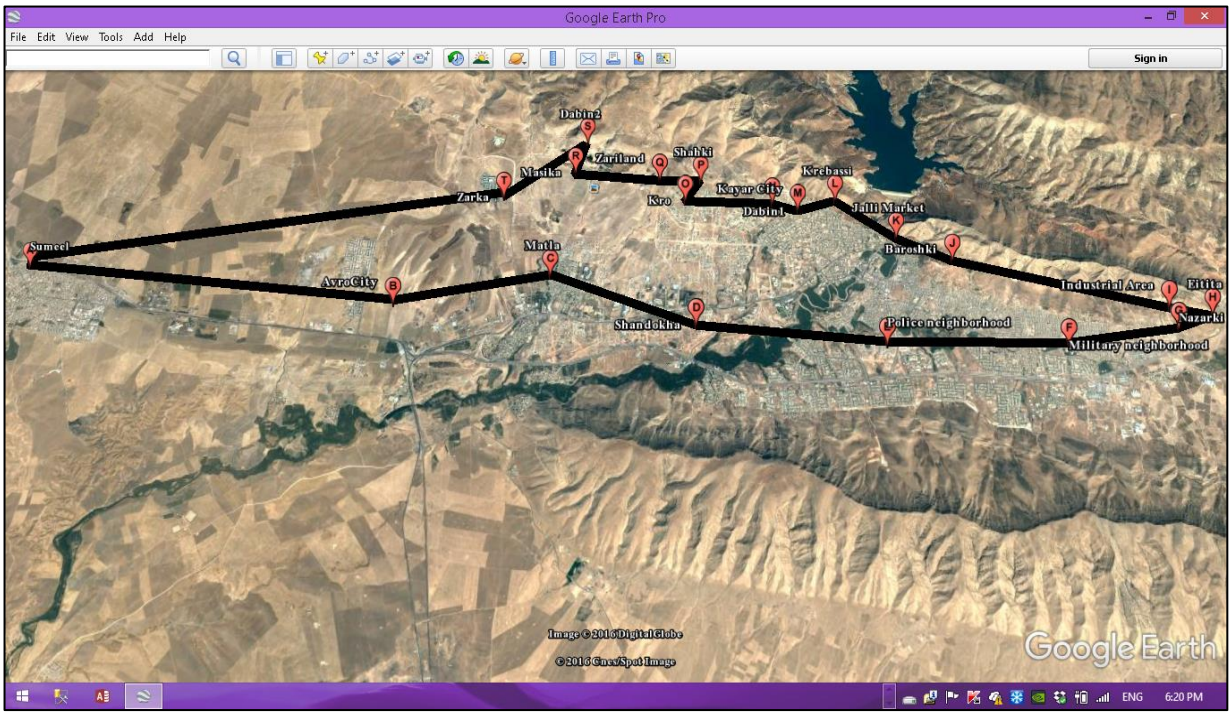
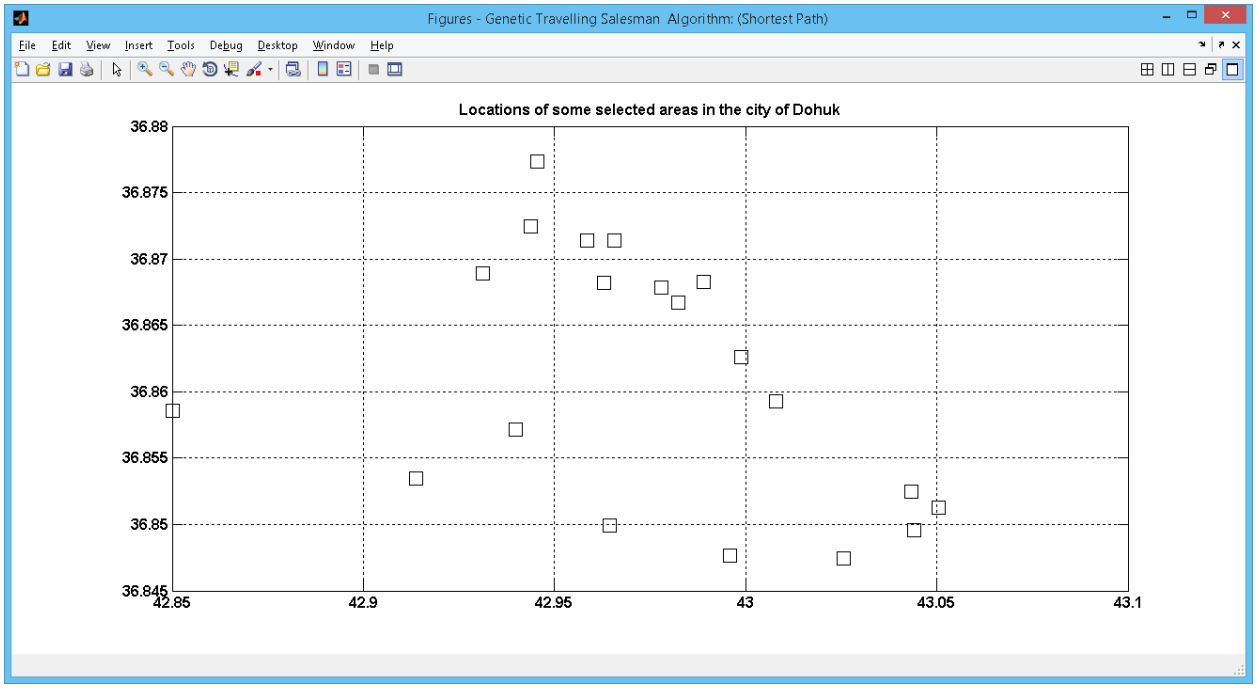
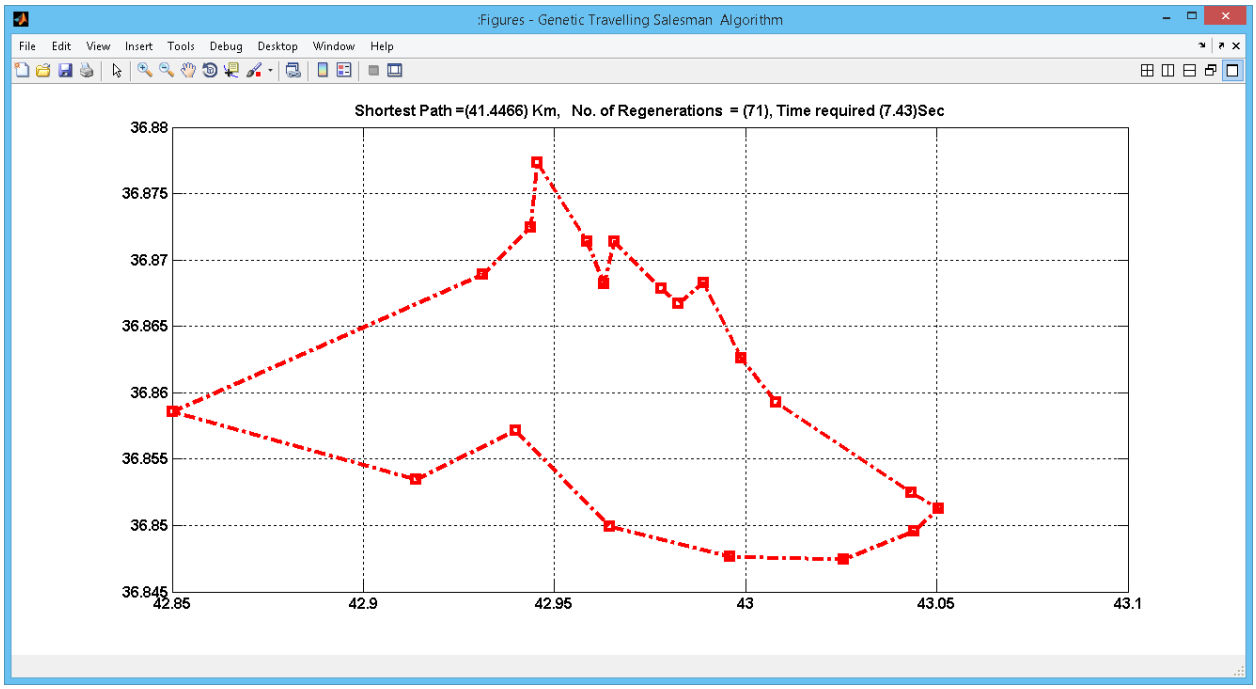


Figure (8) : Determined areas for the city of Dohuk by Google earth program



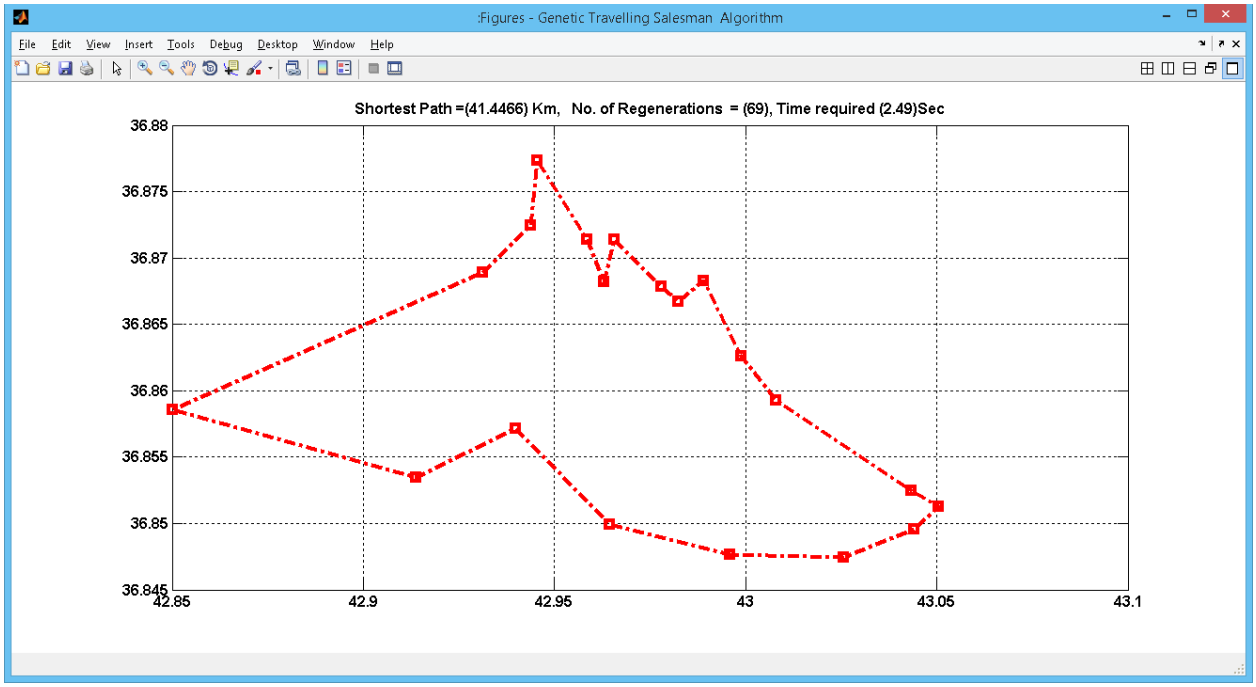
doi : 10.25007/ajnu.v7n3a207

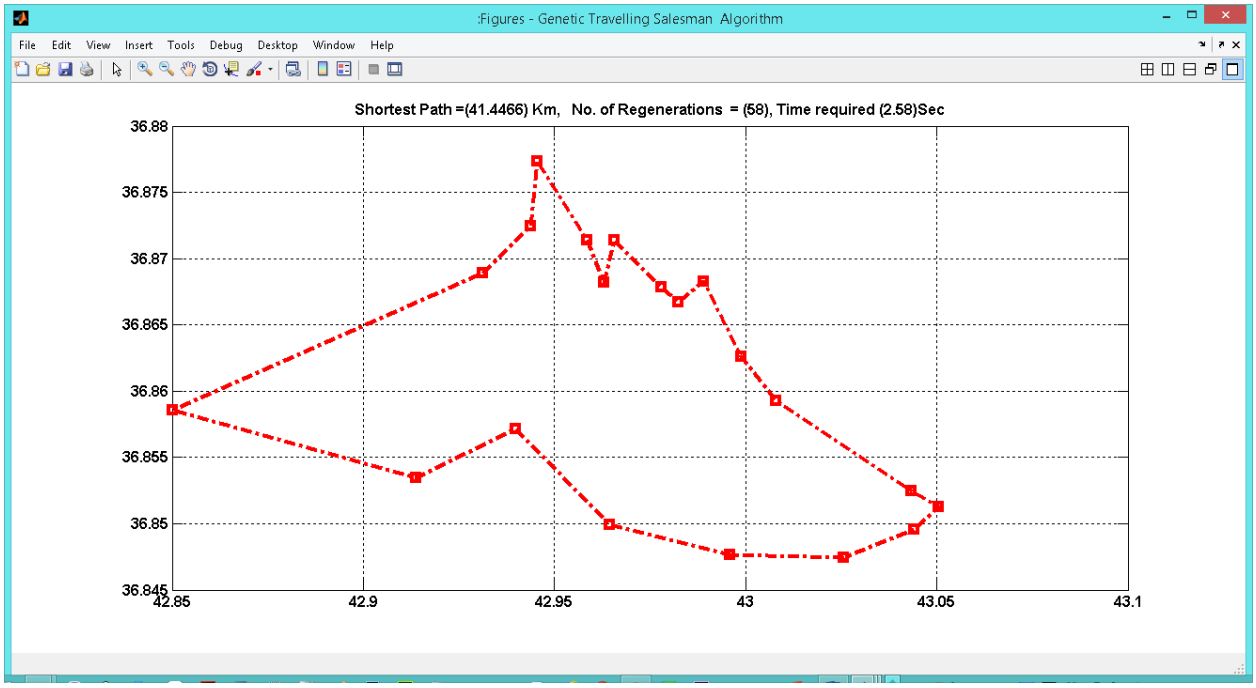
Figure (9) : Initial representation for Duhok city areas (Longitude, Latitude)



doi : 10.25007/ajnu.v7n3a207

Figure (10) : Execution (1) Shortest Path = (41.4466) Km, No. of Regenerations = (71), Time required (7.43) Sec





doi : 10.25007/ajnu.v7n3a207

Figure (12) : Execution (3) Shortest Path = (41.4466) Km, No. of Regenerations = (58), Time required (2.58) Sec

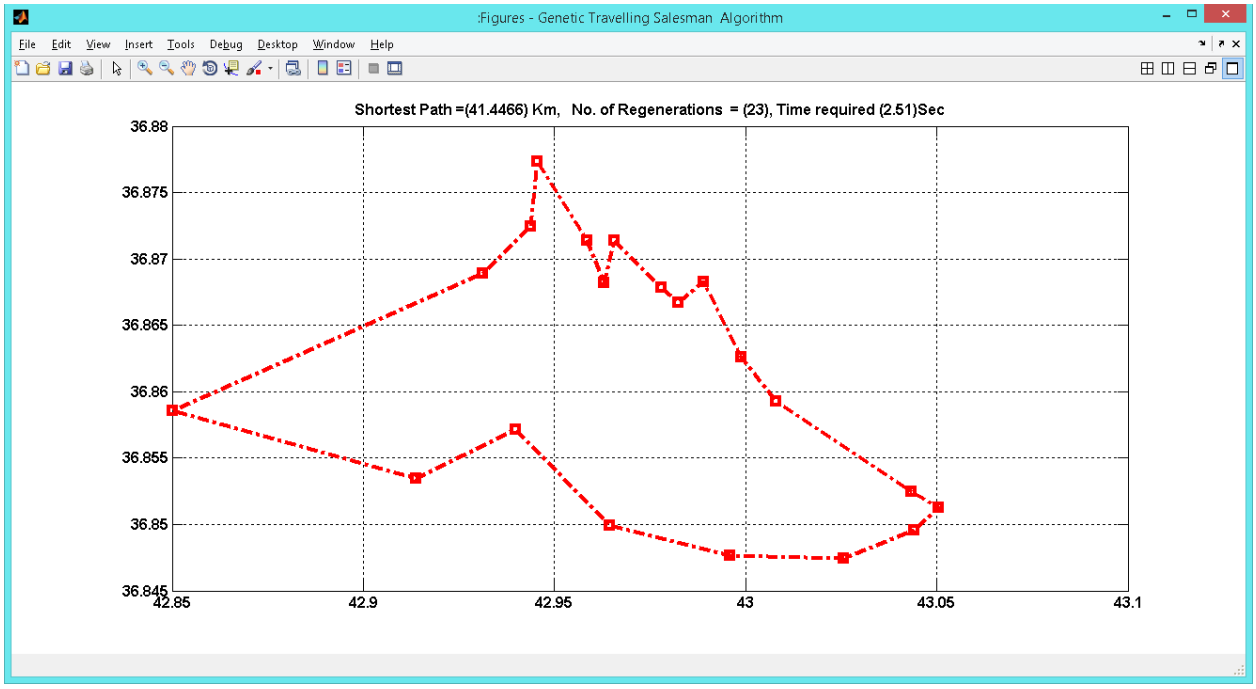
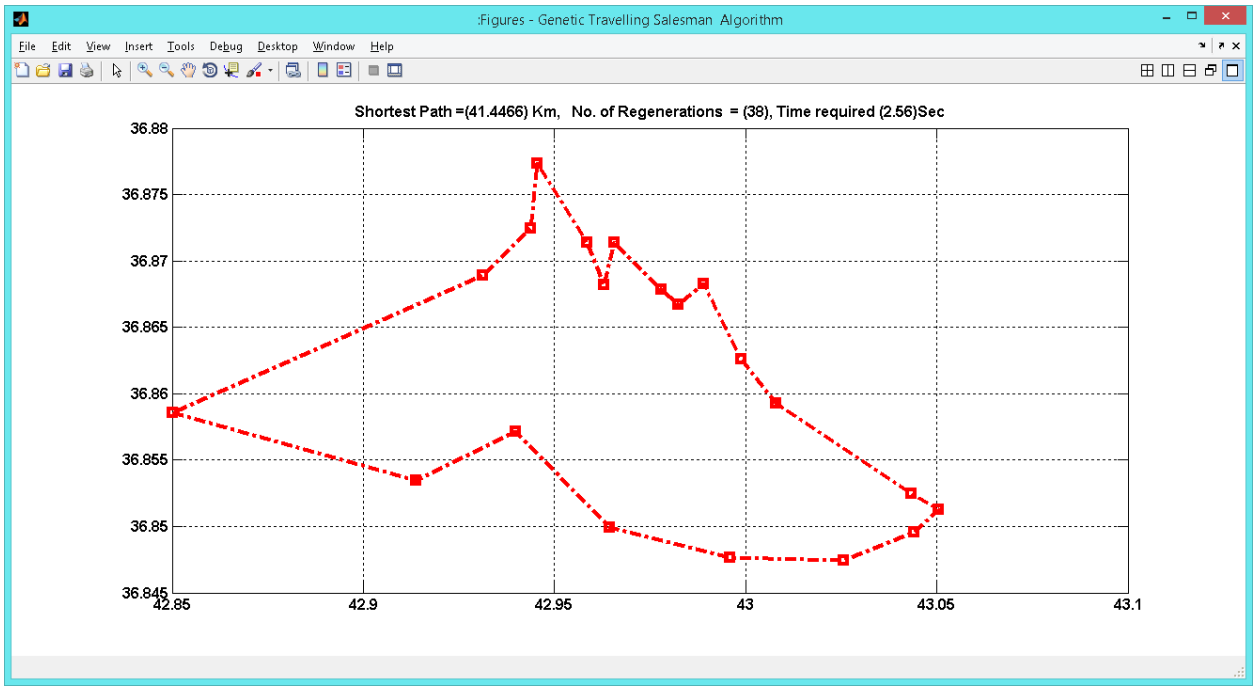


Figure (13) : Execution (4) Shortest Path = (41.4466) Km, No. of Regenerations = (38), Time required (2.56) Sec



Time required (2.51) Sec

doi : 10.25007/ajnu.v7n3a207

References:

Diaby, Moustapha. (2006) "The traveling salesman problem: a linear programming formulation". University of Connecticut, USA.

Mukherjee, Swahum; Ganguly, Srinjoy; Das, Swagatam. (2012) "A Strategy Adaptive Genetic Algorithm for Solving the Travelling Salesman Problem", SEMCCO, India. PP: 778-784.

Al-Sabawi Ahmed Mahmoud Mohammed, (2012) "Using the Branch and bound algorithm and genetic algorithm to solve the Travelling Salesman Problem". Iraqi Journal of Statistical Sciences, Iraq, Volume 12, Issue 21, PP:69-96.(Arabic reference).

Grosan, Crina; ABRAHAM, Ajith. (2011) "Intelligent systems: A modern approach". Springer Science & Business Media.Sivanandam, S. N.; DEEPA, S. N. (2007) "Introduction to genetic algorithms". Springer Science & Business Media.

Hussein, Rana B., Thabet, Hamsa M. (2014)"The Genetic Coefficient with Some Applications". Tanmiat Al-Rafidain Journal, Iraq, Volume 36, Issue 116. (Arabic reference).

Hazim, Ziyad.(1999), "Comparing Different Programming Techniques for Solving. Assignment Problem". College of computer sciences and Mathematics, University of Mosul, Iraq, M.Sc. thesis.

Al-Jawahiri, Zuhair Abdul Wahab Mohammed Hassan. (2011),"Evaluate the coordinate's accuracy for the Google earth program". UOBabylon Journal of Applied and Pure Sciences, Iraq, Issue No 2, Volume No 19. (Arabic reference).

(2016), "KML Data format". Retrieved, from http://geojournalism.org/tracks/type /data-format/ .

Abramson, M. A. (2004), "Genetic algorithm and direct search toolbox". Natick, MA: The Math Work Inc.

Chipperfield, A. J.; Fleming, P. J. "The MATLAB genetic algorithm toolbox In Applied control techniques using MATLAB", IEE Colloquium on (pp. 10-1). IET, 1995.

doi : 10.25007/ajnu.v7n3a207