Solving TSP using Genetic Algorithms - Case of Kosovo

AVNI REXHEPI, AGNI DIKA, ADNAN MAXHUNI
Department of computerics
Faculty of Electrical and Computer Engineering
University of Prishtina
Kodra e Diellit, p.n. 10000 Prishtina
KOSOVO
avni.rexhepi@uni-pr.edu, agni.dika@uni-pr.edu, adnan.maxhuni@uni-pr.edu
http://www.uni-pr.edu/fiek

Abstract: In this paper we describe the use of a Genetic Algorithm to solve the TSP problem for Kosovo Cities/Municipalities, where one has to minimize the travelling distance between the cities (locations). Since the TSP problem is a NP problem, approaches such as dynamic programming, backtracking, branch and bound, etc. are not very useful for solving it. For problems traditionally thought of as computationally infeasible such as the TSP, Genetic Algorithms prove to be the best approach in obtaining solutions.
John Holland's book "Adaptation in Natural and Artificial Systems" (1975, 1992) showed how the evolutionary process can be applied to solve a wide variety of problems using a highly parallel technique that is called the "Genetic Algorithm" (GA). The genetic algorithm transforms a population (set) of individual objects, each with an associated fitness value, into a new generation of the population using the Darwinian principle of reproduction and survival of the fittest and analogs of naturally occurring genetic operations such as crossover (sexual recombination) and mutation. Each individual in the population represents a possible solution to a given problem. The genetic algorithm attempts to find a very good (or best) solution to the problem by genetically breeding the population of individuals over a series of generations.
Kosova is a relatively small country, in aspect of territory size, which is 10887 Km2. Besides capital Prishtina, there are six other bigger cities and about 23 smaller cities, while the rest of locations are villages. Basiclly, there are 30 municipalities, with 1467 localities, in total, with about two milion residents. Recently, seven localities are proposed to be recognised as municipalities and are in the process of transformation.
For solving the problem, we designed an application in C#, which will use a genetic algorithm to find the best possible solution for TSP Kosova problem.
The first part of this paper is a brief presentation Kosovo and Kosova municipalities.

Key-Words: - Genetic Agorithms, TSP, C#, Kosovo.

1 Introduction

Kosovo is the newest country in Europe and as per size of the territory, belongs to the small countries (169 in a world rank for territory size). It has 10887 Sq Km. Kosovo is located in Southeast Europe, in Balkan Peninsula, in geographic coordinates 42 35 N and 21 00 E, between Albania, Montenegro, Serbia and Macedonia,. The total border line is 702 km, with border countries: Albania 112 km, Macedonia 159 km, Montenegro 79 km and Serbia 352km.

There are 7 big cities in Kosovo, with capital Prishtina and 30 municipalities in total. Most of the roads are paved. There are around 2000 km of road-ways and 94% of them are paved. Only 2% are international highways, while 66% are national and 32% are regional ways.

Fig.1 – Kosovo position in Balkan Peninsula

In this paper we descripe the Traveling Salesman Problem (TSP) for Kosovo municipalities, using the genetic algorithm.

2 Traveling Salesman Problem

TSP is a problem, where traveling salesman wants to visit each of a set of cities exactly once, starting from hometown and returning to his hometown. His problem is to find the shortest route for such a trip. TSP has a model character in many branches of Mathematics, Computer Science, Operations Research, etc. Linear programing, heuristics and branch and bound which are main components for the most succesful approaches to hard combinatorial optimization problems, were first formulated for the TSP and used to solve practical problem instances in 1954 by Dantzig, Fulkerson and Johnson.

When the theory of NP-completeness developed, the TSP was one of the first problems to be proven NP-hard by Karp in 1972. New algorithmic techniques have first been developed for or at least have been applied to the TSP to show their effectiveness. Such examples are branch and bound, Lagrangean relaxation, Lin-Karneghan type methods, simulated annealing, etc [3].

Representation model is: Let Kn=(V_n, E_n) be the complete undirected graph with n=|V_n| nodes and m=|En|=$\binom{n}{2}$ edges. An edge e with endpoints *i* and *j* is also denoted by *ij*, or by *(i,j)*. We denote by R^{En} the space of real vectors whose components are indexed by the elements of E_n. The component of any vector $z \in R^{En}$ indexed by the edge *e=ij* is denoted by z_e, z_{ij}, or *z(i,j)*.

Given an objective function $c \in R^{En}$, that associates a “length” c_e with every edge *e* of K_n, the symmetric traveling salesman problem consists of finding a Hamiltonian cycle such that its *c*-length (the sum of the lengths of its edges) is as small as possible.

Of special interest are the Euclidean instances of the traveling salesman problem. In these instances the nodes defining the problem correspond to points in the two-dimensional plane and the distance between two nodes is the Euclidean distance between their corresponding points. More generally, instances that satisfy the triangle inequality, i.e., $c_{ij} + c_{jk} \geq c_{ik}$ for all the three distinct *i,j* and *k*, are of particular interest.

For our case, we consider the locations of the cities/municipalities in Kosovo map as nodes of the graph. For to do this, we take their geographic coordinates and than based on that, we calculate their position in our map scaled to a smaller size, for to calculate the real positions and distances, by using the real life values for distances in kilometres between the cities.

3 Genetic Algorithms

Genetic Algorithms (GA) [1,2] are computer algorithms that search for good solutions to a problem within a large number of possible solutions. They were proposed and developed in the 1960s by John Holland, his students, and his colleagues at the University of Michigan. These computational paradigms were inspired by the mechanics of natural evolution, including survival of the fittest, reproduction, and mutation. These mechanics are well suited to resolve a variety of practical problems, including computational problems, in many fields. Some applications of GAs are optimization, automatic programming, machine learning, economics, immune systems, population genetic, and social system.

GAs have been successfully applied to many problems of business, engineering, and science. Because of their operational simplicity and wide applicability, GAs play an important role in computational optimization and operations research [6].

The genetic algorithm transforms a population (set) of individual objects, each with an associated fitness value, into a new generation of the population using the Darwinian principle of reproduction and survival of the fittest and analogs of naturally occurring genetic operations such as crossover (sexual recombination) and mutation. Each individual in the population represents a possible solution to a given problem. The genetic algorithm attempts to find a very good (or best) solution to the problem by genetically breeding the population of individuals over a series of generations.

3.1 Basic elements of GAs

Most GAs methods are based on the following elements: populations of chromosomes, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring. The chromosomes in GAs represent the space of candidate solutions. Possible chromosomes encodings are binary, permutation, value, and tree encodings. GAs require a fitness function which allocates a score to each chromosome in the current population. Thus, it can calculate how well the solutions are coded and how well they solve the problem [2].

The selection process is based on fitness. Chromosomes that are evaluated with higher values (fitter) will most likely be selected to reproduce, whereas, those with low values will be discarded.

The fittest chromosomes may be selected several times, however, the number of chromosomes selected to reproduce is equal to the population size, therefore, keeping the size constant for every generation. This phase has an element of randomness just like the survival of organisms in nature. The most used selection methods, are roulette-wheel, rank selection, steady-state selection, and some others. Moreover, to increase the performance of GAs, the selection methods are enhanced by eiltism. Elitism is a method, which first copies a few of the top scored chromosomes to the new population and then continues generating the rest of the population. Thus, it prevents loosing the few best found solutions.

Crossover is the process of combining the bits of one chromosome with those of another to create an offspring for the next generation that inherits traits of both parents. For example, consider the following parents and a crossover point at position 3:

Parent 1	1 0 0 | 0 1 1 1
Parent 2	1 1 1 | 1 0 0 0
Offspring 1	1 0 0 1 0 0 0
Offspring 2	1 1 1 0 1 1 1

In this example, Offspring 1 inherits bits in position 1, 2, and 3 from the left side of the crossover point from Parent 1 and the rest from the right side of the crossover point from Parent 2. Similarly, Offspring 2 inherits bits in position 1, 2, and 3 from Parent 2 and the rest from Parent 1.

Mutation is performed after crossover to prevent falling all solutions in the population into a local optimum of solved problem. Mutation changes the new offspring by flipping bits from 1 to 0 or from 0 to 1. Mutation can occur at each bit position in the string with some probability, usually very small (e.g. 0.001). For example, consider the following chromosome with mutation point at position 3:

Not mutated chromosome:	1 0 0 0 1 1 1
Mutated chromosome:	1 0 1 0 1 1 1

The 0 at position 3 flips to 1 after mutation.

So, general outline of basic GA:

1. Start: Randomly generate a population of N chromosomes.
2. Fitness: Calculate the fitness of all chromosomes.
3. Create a new population:
   a.Selection: According to the selection method select 2 chromosomes from the population.
   b.Crossover: Perform crossover on the 2 chromosomes selected.
   c.Mutation: Perform mutation on the chromosomes obtained.
4. Replace: Replace the current population with the new population.
5. Test: Test whether the end condition is satisfied. If so, stop. If not, return the best solution in current population and go to Step 2.

Each iteration of this process is called generation.The genetic algorithm object determines which individuals should survive, which should reproduce, and which should die. It also records statistics and decides how long the evolution should continue. A typical genetic algorithm will run forever, so we must build functions for specifying when the algorithm should terminate. These include terminate-upon generation, in which you specify a certain number of generations for which the algorithm should run, and terminate-upon-convergence, in which you specify a value to which the best-of-generation score should converge. One can customize the termination function to use own stopping criterion and must tell the algorithm when to stop. Often the number-of generations is used as a stopping measure, but you can use goodness-of-best-solution, convergence-of-population, or any problem-specific criterion if you prefer.

There are some flavors of genetic algorithms. For example, the first is the standard 'simple genetic algorithm' described by Goldberg in his book [2]. This algorithm uses nonoverlapping populations and optional elitism. Each generation the algorithm creates an entirely new population of individuals. The second is a 'steady-state genetic algorithm' that uses overlapping populations. In this variation, you can specify how much of the population should be replaced in each generation. The third variation is the 'incremental genetic algorithm', in which each generation consists of only one or two children. The incremental genetic algorithms allow custom replacement methods to define how the new generation should be integrated into the population. So, for example, a newly generated child could replace its parent, replace a random individual in the population, or replace an individual that is most like it. The fourth type is the 'deme' genetic algorithm. This algorithm evolves multiple populations in parallel using a steady-state algorithm. Each generation the algorithm migrates some of the individuals from each population to one of the other populations.

The base genetic algorithm class contains operators and data common to most genetic algorithms. The genetic algorithm contains the statistics, replacement strategy, and parameters for running the algorithm. the population object, a container for genomes, also contains some statistics as well as selection and scaling operators.

The number of function evaluations is a good way to compare different genetic algorithms with various other search methods[3]. The basic algorithm is as follows:

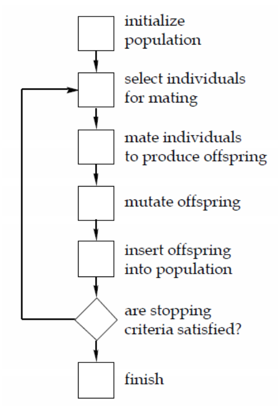

initialize population
select individuals for mating
mate individuals to produce offspring
mutate offspring
insert offspring into population
are stopping criteria satisfied?
finish

Fig.2 – Genetic Algorithm

4 TSP Kosovo

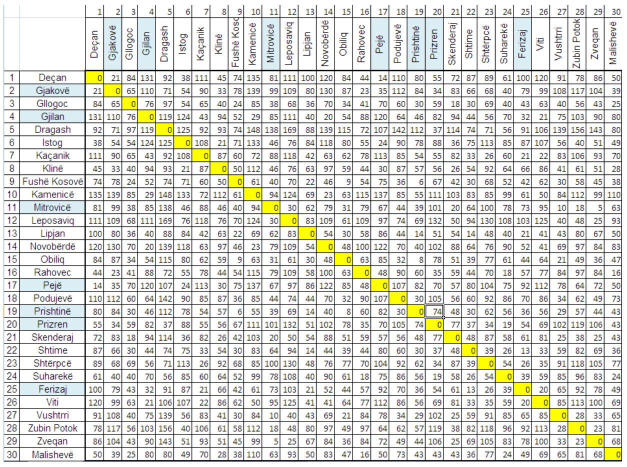
In order to calculate the shortest traveling distance from an initial city, by visiting each one only once and returning to the initial one, we consider the locations as nodes of the graph, in the graph model. In the meantime, the distances between the cities are edges of the graph.

For length of edges we take inter-city distances in kilometres. We have created a matrix of distances between all the municipalities, where the matrix elements c_{ij} are elements of the square symetrical matrix, since distance ij is equal to the distance in the other side ji. The diagonal of the matrix will contain zero values, since diagonal elements of the matrix c_{ij}, where $i=j$, will represent the distance of the city to itself, so in fact it will be the traveling distance of zero kilometres, therefore these elements will be equal to $c_{ij}=0$.

By using complete graph in the definition of the TSP, the existence of a feasible solution is guaranteed, while for general graphs deciding the existence of a Hamiltonian cycle is an NP-complete problem.The number of Hamiltonian cycles in K_n, i.e. the size of the set of feasible solutions of the TSP is (n-1)!/2.

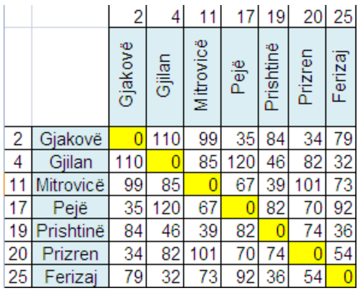
The algorithmic treatment of the TSP ensures an approximation algorithm that cannot guarantee to find the optimum, but wich is the only available technique to find a good solution to a large problem instances. To assess the quality of a solution, one has to be able to compute a lower bound on the value of the shortest Hamiltonian cycle.

We collected the data for inter-city distances from official site of the "Ministry of transport and telecommunications", from municipality web-pages and some of them we calculated using Google-Earth path-calculation of routes in the map.



		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
		Deçan	Gjakovë	Gllogoc	Gjilan	Dragash	Istog	Kaçanik	Klinë	Fushë Koso	Kamenicë	Mitrovicë	Leposaviq	Lipjan	Novobërdë	Obiliq	Rahovec	Pejë	Podujevë	Prishtinë	Prizren	Skenderaj	Shtime	Shtërpcë	Suharekë	Ferizaj	Viti	Vushtrri	Zubin Potok	Zveqan	Malishevë
1	Deçan	0	21	84	131	92	38	111	45	74	135	81	111	100	120	84	44	14	110	80	55	72	87	89	61	100	120	91	78	86	50
2	Gjakovë	21	0	65	110	71	54	90	33	78	139	99	109	80	130	87	23	35	112	84	34	83	66	68	40	79	99	108	117	104	39
3	Gllogoc	84	65	0	76	97	54	65	40	24	85	38	68	36	70	34	41	70	60	30	59	18	30	69	40	43	63	40	56	43	25
4	Gjilan	131	110	76	0	119	124	43	94	52	29	85	111	40	20	54	88	120	64	46	82	94	44	56	70	32	21	75	103	90	80
5	Dragash	92	71	97	119	0	125	92	93	74	148	138	169	88	139	115	72	107	142	112	37	114	74	71	56	91	106	139	156	143	80
6	Istog	38	54	54	124	125	0	108	21	71	133	46	76	84	118	80	55	24	90	78	88	36	75	113	85	87	107	56	40	51	49
7	Kaçanik	111	90	65	43	92	108	0	87	60	72	88	118	42	63	62	78	113	85	54	55	82	33	26	60	21	22	83	106	93	70
8	Klinë	45	33	40	94	93	21	87	0	50	112	46	76	63	97	59	44	30	87	57	56	26	54	92	64	66	86	41	61	51	28
9	Fushë Kosovë	74	78	24	52	74	71	60	50	0	61	40	70	22	46	9	54	75	36	6	67	42	30	68	52	42	62	30	58	45	38
10	Kamenicë	135	139	85	29	148	133	72	112	61	0	94	124	69	23	63	115	137	85	55	111	103	83	85	99	61	50	84	112	99	110
11	Mitrovicë	81	99	38	85	138	46	88	46	40	94	0	30	62	79	31	79	67	44	39	101	20	64	100	78	73	95	10	18	5	63
12	Leposaviq	111	109	68	111	169	76	118	76	70	124	30	0	83	109	61	109	97	74	69	132	50	94	130	108	103	125	40	48	25	93
13	Lipjan	100	80	36	40	88	84	42	63	22	69	62	83	0	54	30	58	86	44	14	51	54	14	48	40	21	41	43	80	67	50
14	Novobërdë	120	130	70	20	139	118	63	97	46	23	79	109	54	0	48	100	122	70	40	102	88	64	76	90	52	41	69	97	84	83
15	Obiliq	84	87	34	54	115	80	62	59	9	63	31	61	30	48	0	63	85	32	8	78	51	39	77	61	44	64	21	49	36	47
16	Rahovec	44	23	41	88	72	55	78	44	54	115	79	109	58	100	63	0	48	90	60	35	59	44	70	18	57	77	84	97	84	16
17	Pejë	14	35	70	120	107	24	113	30	75	137	67	97	86	122	85	48	0	107	82	70	57	80	104	75	92	112	78	64	72	50
18	Podujevë	110	112	60	64	142	90	85	87	36	85	44	74	44	70	32	90	107	0	30	105	56	60	92	86	70	86	34	62	49	73
19	Prishtinë	80	84	30	46	112	78	54	57	6	55	39	69	14	40	8	60	82	30	0	74	48	30	62	56	36	56	29	57	44	43
20	Prizren	55	34	59	82	37	88	55	56	67	111	101	132	51	102	78	35	70	105	74	0	77	37	34	19	54	69	102	119	106	43
21	Skenderaj	72	83	18	94	114	36	82	26	42	103	20	50	54	88	51	59	57	56	48	77	0	48	87	58	61	81	25	38	25	43
22	Shtime	87	66	30	44	74	75	33	54	30	83	64	94	14	44	39	44	80	60	30	37	48	0	39	26	13	33	59	82	69	36
23	Shtërpcë	89	68	69	56	71	113	26	92	68	85	100	130	48	76	77	70	104	92	62	34	87	39	0	54	26	35	91	118	105	77
24	Suharekë	61	40	40	70	56	85	60	64	52	99	78	108	40	90	61	18	75	86	56	19	58	26	54	0	39	59	85	96	83	24
25	Ferizaj	100	79	43	32	91	87	21	66	42	61	73	103	21	52	44	57	92	70	36	54	61	13	26	39	0	20	65	92	78	49
26	Viti	120	99	63	21	106	107	22	86	62	50	95	125	41	41	64	77	112	86	56	69	81	33	35	59	20	0	85	113	100	69
27	Vushtrri	91	108	40	75	139	56	83	41	30	84	10	40	43	69	21	84	78	34	29	102	25	59	91	85	65	85	0	28	33	65
28	Zubin Potok	78	117	56	103	156	40	106	61	58	112	18	48	80	97	49	97	64	62	57	119	38	82	118	96	92	113	28	0	23	81
29	Zveqan	86	104	43	90	143	51	93	51	45	99	5	25	67	84	36	84	72	49	44	106	25	69	105	83	78	100	33	23	0	68
30	Malishevë	50	39	25	80	80	49	70	28	38	110	63	93	50	83	47	16	50	73	43	43	43	36	77	24	49	69	65	81	68	0

Table 1 – Distances between 30 municipalities



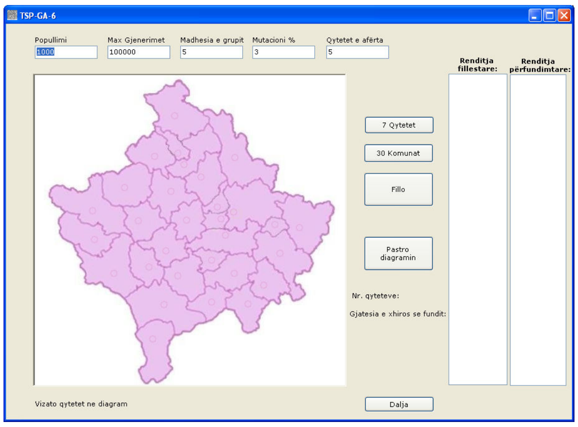
		2	4	11	17	19	20	25
		Gjakovë	Gjilan	Mitrovicë	Pejë	Prishtinë	Prizren	Ferizaj
2	Gjakovë	0	110	99	35	84	34	79
4	Gjilan	110	0	85	120	46	82	32
11	Mitrovicë	99	85	0	67	39	101	73
17	Pejë	35	120	67	0	82	70	92
19	Prishtinë	84	46	39	82	0	74	36
20	Prizren	34	82	101	70	74	0	54
25	Ferizaj	79	32	73	92	36	54	0

Table 2 – Distances between the 7 biggest cities

5 Application

We have built an application in C#, with an image with the small-scaled size of the Kosov map, with depicted municipality boundaries and locations.

It is possible to select a particular city by clicking on the map and we have also added to buttons that make it possible to create locations for seven biggest cities and locations for all thirty municipalities.



TSP-GA-6

Popullimi	Max Gjenerimet	Madhesia e grupit	Mutacioni %	Qytetet e afërta
1000	100000	5	3	5

Nr. qyteteve:

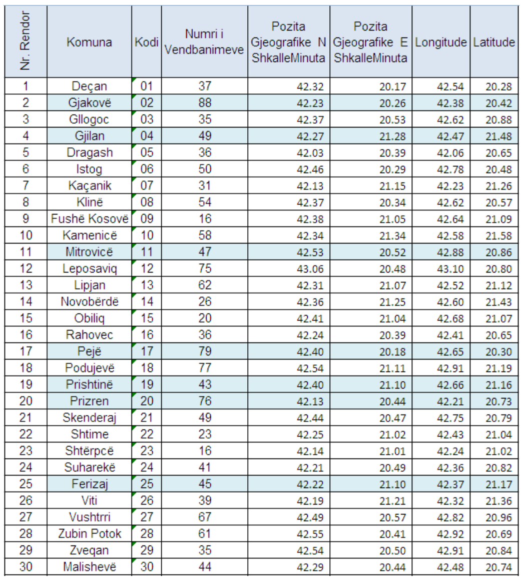
Gjatesia e xhiros se fundit:

Renditja fillestare:

Renditja përfundimtare:

Fig.3 – Kosovo map, in the application

We used a table with geographical coordinates of the cities, to calculate their positions.



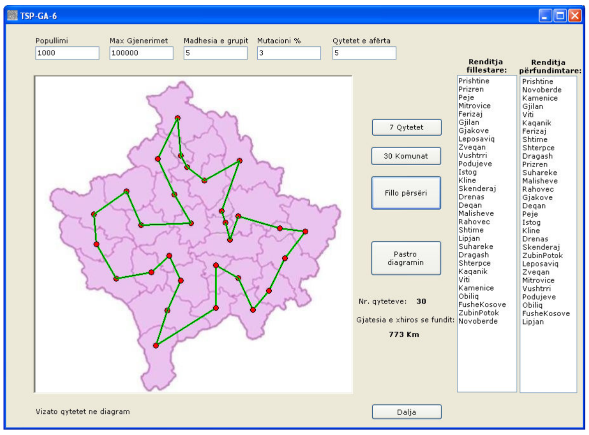
Nr. Rendor	Komuna	Kodi	Numri i Vendbanimeve	Pozita Gjeografike N ShkalleMinuta	Pozita Gjeografike E ShkalleMinuta	Longitude	Latitude
1	Deçan	01	37	42.32	20.17	42.54	20.28
2	Gjakovë	02	88	42.23	20.26	42.38	20.42
3	Gllogoc	03	35	42.37	20.53	42.62	20.88
4	Gjilan	04	49	42.27	21.28	42.47	21.48
5	Dragash	05	36	42.03	20.39	42.06	20.65
6	Istog	06	50	42.46	20.29	42.78	20.48
7	Kaçanik	07	31	42.13	21.15	42.23	21.26
8	Klinë	08	54	42.37	20.34	42.62	20.57
9	Fushë Kosovë	09	16	42.38	21.05	42.64	21.09
10	Kamenicë	10	58	42.34	21.34	42.58	21.58
11	Mitrovicë	11	47	42.53	20.52	42.88	20.86
12	Leposaviq	12	75	43.06	20.48	43.10	20.80
13	Lipjan	13	62	42.31	21.07	42.52	21.12
14	Novobërdë	14	26	42.36	21.25	42.60	21.43
15	Obiliq	15	20	42.41	21.04	42.68	21.07
16	Rahovec	16	36	42.24	20.39	42.41	20.65
17	Pejë	17	79	42.40	20.18	42.65	20.30
18	Podujevë	18	77	42.54	21.11	42.91	21.19
19	Prishtinë	19	43	42.40	21.10	42.66	21.16
20	Prizren	20	76	42.13	20.44	42.21	20.73
21	Skenderaj	21	49	42.44	20.47	42.75	20.79
22	Shtime	22	23	42.25	21.02	42.43	21.04
23	Shtërpcë	23	16	42.14	21.01	42.24	21.02
24	Suharekë	24	41	42.21	20.49	42.36	20.82
25	Ferizaj	25	45	42.22	21.10	42.37	21.17
26	Viti	26	39	42.19	21.21	42.32	21.36
27	Vushtrri	27	67	42.49	20.57	42.82	20.96
28	Zubin Potok	28	61	42.55	20.41	42.92	20.69
29	Zveqan	29	35	42.54	20.50	42.91	20.84
30	Malishevë	30	44	42.29	20.44	42.48	20.74

Table 3 – Geographical coordinates, positions

By running the application, we can calculate the shortest traveling distance between the selected cities, by finding a solution of the TSP using a genetic algorithm.

We also use two lists, where the first one will show the selected cities, while the second one will show the sequnce of the cities in the found solution. Applicacion will calculate and show the total traveling distance in kilometres.

User can set up the numbers for initial population, maximal number of generations, size of the group, percentage of mutations and number of close cities/locations (used by the algorithm, while finding closest locations).



TSP-GA-6

Popullimi: 1000
Max Gjenerimet: 100000
Madhesia e grupit: 5
Mutacioni %: 3
Qytetet e afërta: 5

Nr. qyteteve: 30
Gjatesia e xhiros se fundit: 773 Km

Renditja fillestare:	Renditja përfundimtare:
Prishtine	Prishtine
Prizren	Novoberde
Peje	Kamenice
Mitrovice	Gjilan
Ferizaj	Viti
Gjilan	Kaqanik
Gjakove	Ferizaj
Leposaviq	Shtime
Zveqan	Shterpce
Vushtrri	Dragash
Podujeve	Prizren
Istog	Suhareke
Kline	Malisheve
Skenderaj	Rahovec
Drenas	Gjakove
Deqan	Deqan
Malisheve	Peje
Rahovec	Istog
Shtime	Kline
Lipjan	Drenas
Suhareke	Skenderaj
Dragash	ZubinPotok
Shterpce	Leposaviq
Kaqanik	Zveqan
Viti	Mitrovice
Kamenice	Vushtrri
Obiliq	Podujeve
FusheKosove	Obiliq
ZubinPotok	FusheKosove
Novoberde	Lipjan

Fig.4 – TSP Solution for 30 municipalities

6 Conclusion

Finding a best solution for NP-hard problems is not possible by using standard mathematical approach. Genetic Algorithms are computer algorithms that search for good solutions to a problem within a large number of possible solutions.

Traveling Salesman Problem is a NP-Hard problem. Using a genetic algorithm, we can find a feasible solution, which will not have guaranty to be the best possible solution, but will be the good one, found in reasonable time.

•In order to find the shortest traveling distance between Kosovo cities/municipalities we use a genetic algorithm, in an application built in C#.

•We use geographical coordinates to calculate locations of the nodes/cities in the graph/image.

•We use a matrix of inter-city distances to calculate the length of the traveling distance.

•Application will find the shortest traveling distance between the selected cities.

References:

[1] J. H. Holland, *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[2] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley, 1989.

[3] Michael Junger, Gerhard Reinelt, Giofanni Rinaldi, *The Traveling Salesman Problem*, M.O. Ball et all, Eds. Handbooks in OR & Ms, Vol. 7, Elsevier Science, B.V. 1997

[4] T.-L. Yu, D. E. Goldberg, and Y.-P. Chen, "A genetic algorithm design inspired by organizationaltheory: A pilot study of a dependency structure matrix driven genetic algorithm," IlliGAL Report No. 2003007, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2003.

[5] K. Sastry, D. E. Goldberg, and G. Kendall, "Genetic algorithms: A tutorial," in Introductory Tutorials in Optimization, Search and Decision Support Methodologies, ch. 4, pp. 97–125, Springer, 2005.

[6] Martin Pelikan, *Genetic Algorithms,* MEDAL Report No. 2010007, 2010