An Idea for Finding the Shortest Driving Time Using Genetic Algorithm Based Routing Approach on Mobile Devices

Umit Atila, Ismail Rakip Karas, Cevdet Gologlu, Beyza Yaman, and Ilhami Muharrem Orak

Abstract— People's orientation to the mobile devices all over the world have made the using of route guidance systems that assist drivers on the traffic widespread in daily life. For an effective routing, these systems should take into account the effectual factors of traffic flow such as allowable velocity limits of the roads and density. The computational cost of the system is up to the amount of nodes in road network and effectual factors. When we consider the road networks with excessive number of nodes, finding the exact routes in real time using some well known deterministic methods such as Dijkstra's algorithm on such routing systems may not be accurate using mobile devices with limited memory capacity and processing speed.

In this paper, a Genetic Algorithm (GA) approach applied on a route guidance system for finding the shortest driving time is proposed. A different gene search approach on crossover operation named "first-match-genes" had been introduced. A mobile application for the traffic network of Ankara and the performance of the genetic algorithm tested on networks with 10, 50, 250, 1000 nodes was presented.

Keywords— Genetic algorithm; navigation; route guidance; shortest path; shortest driving time; optimization.

I. Introduction

THE logic behind navigation systems provides the users with the shortest path between departure and destination points. The downside of the current navigation systems is ignoring important decision variables including the traffic density and allowable velocity limits of the roads.

Shortest path problem can be defined as finding the shortest path between two vertices of a directed graph where each arc has been weighted. The shortest path is considered as one of the most fundamental network optimization problems. This problem comes up in practice and arises as a sub problem in many network optimization algorithms [1]. One of the most popular algorithm is the Dijkstra's algorithm conceived, which solves the shortest path problem in O(n2) time on a graph with n number of nodes and positive edge weights [2].

The search process can be carried out in two methods: deterministic search and stochastic search which are effectuated like random base algorithms. Considering the quandary's conditions, the utilization of the stochastic methodology instigates the curtailment of the search space and the simplification of the relationships affecting optimization [3].

Deterministic methods used by researchers from all over the world may not reach the solution for the nonlinear problems and they are subject to excessive solution time as the number of parameters increase. These disadvantages direct the researchers to use other methods such as heuristic techniques. Unlike deterministic methods, heuristic techniques do not guarantee optimal solutions, but they can find good/near optimal solutions within a reasonable time [4].

GA is a heuristic technique developed by John Holland in 1975 based on genetic and natural selection principles [5]. Goldberg proved that GA is one of the powerful search methods in both theory and practice [6]. Genetic algorithm is all-round optimizing method in the simulation of evolution process of species in nature, so GA can adopt a variety of construction methods [7]. GA starts with generating an initial population by random selection of the individuals named chromosomes that each encodes the solution of the problem. Each chromosome that encodes a candidate solution of the problem is made with a combination of significant genes [8]. GA founded based on two fundamental evolutionary concepts:

- A Darwinian notion of fitness, which describes an individual's ability to survive
- Genetic operators, which determine the next generation's genetic makeup based upon the current generation [9].

Conventionally, genetic operations are achieved through crossover and mutation operators. The crossover exchanges partial genes of two chosen individuals to create the new offspring that inherit some characters of their parents. A crossover operator manipulates a pair of individuals (called parents) to produce two new individuals (called offspring) by exchanging segments from the parents' coding. By exchanging information between two parents, the crossover operator provides a powerful exploration capability. A commonly used method for crossover is called one-point [10].

Manuscript received April 10, 2011: Revised version received

U. Atila is with the Directorate of Computer Center, Gazi University, 06500 Besevler, Ankara , TURKEY (e-mail: umitatila@gmail.com).

I.R. Karas is with the Department of Computer Engineering, Karabuk University, 78050 Karabuk, TURKEY (corresponding author to provide phone: 90-370-433-2021; fax: 90-370-433-3290; e-mail: ismail.karas@karabuk.edu.tr).

C. Gologlu is with the Department of Computer Engineering, Karabuk University, 78050 Karabuk, TURKEY (e-mail: cgologlu@karabuk.edu.tr).

B. Yaman is with the Department of Computer Engineering, Karabuk University, 78050 Karabuk, TURKEY (e-mail:beyzayaman@karabuk.edu.tr).

I.M. Orak is with the Department of Computer Engineering, Karabuk University, 78050 Karabuk, TURKEY (e-mail: imorak@karabuk.edu.tr).

The mutation operator randomly alters one or more genes in an individual. Mutations add genetic diversity to the population. Through mutation, GAs can search previously unexplored sections of the solution space. Mutations consequently assure that the entire search space is connected [11].

The first step starts with obtaining the values that fitness function returns for each chromosome and selecting the best chromosomes of initial population, which will form the individuals of the next generation. The parents selected for regeneration are replaced by crossover operation and changed by mutation operation to produce child chromosomes. The chromosomes that are not passed through crossover or mutation and newly generated child chromosomes form a new population [8]. The generation of new populations repeats till defined number of times in advance or is being continued until not having better chromosomes (Fig.1).

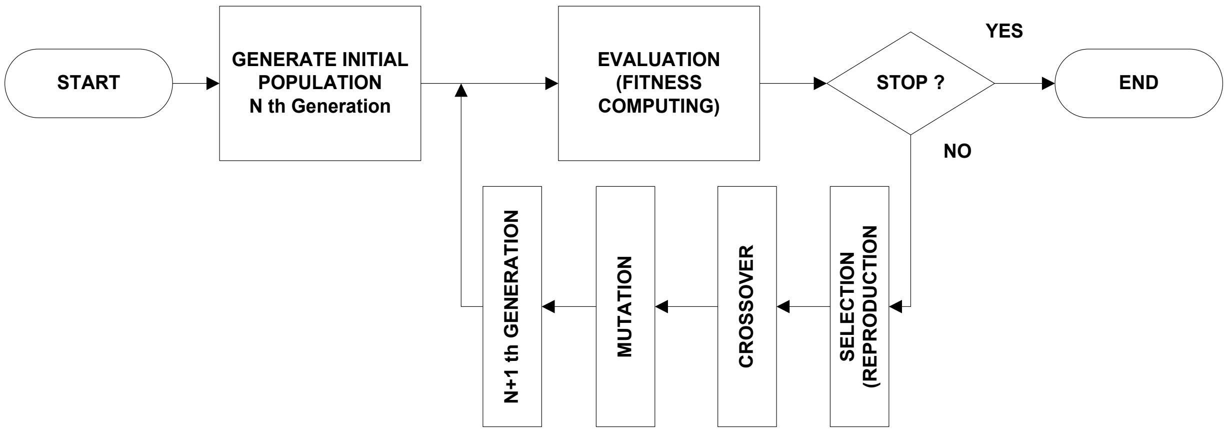

Fig.1 Steps of the Genetic Algorithm

In last decade, there have been number of approaches used Genetic algorithm (GA) in the solution of the shortest path problems. Munemoto et al. implemented a GA, which is practically feasible on wired or wireless network environment [12]. Inagaki et al. proposed an algorithm with fixed length chromosomes [13].

Ahn and Ramakrishna reported that the algorithm proposed by Munemoto required a relatively large population for an optimal solution due to the constraints on the crossover mechanism. Furthermore, it was reported that this was not suitable for large networks or real-time communications. On the other hand, they suggested that the algorithm proposed by Inagaki et al. required a large population to attain an optimal or high quality of solution due to its inconsistent crossover mechanism. They proposed a GA for solving the shortest path problem that uses chromosomes with variable lengths. They showed that as the number of the nodes becomes more than 20, the computing time by adopting the GA was less than that by adopting the Djikstra algorithm [14].

Hasan et al. produced a different solution for the shortest path problem using GA [15]. They employed a chromosome-coding scheme using node indices and distance weights. The complete chromosome of a candidate was divided into node fields, which were equal to the number of nodes in the network. They proposed different crossover and mutation methods, which are appropriate their chromosome encoding. They tested their proposed algorithm on networks with 10, 20, 50 and 100 routers and results demonstrated consistent and speedy convergence for the tested scenarios.

This study presents a route guidance system and a GA approach applied on this routing system to find the shortest driving time. Proposed guidance system provides driving advice for the drivers considering not only the distances but also traffic density and the allowable velocity limits of the roads. Thus, it computes the shortest driving time instead of the shortest path.

The rest of the paper is organized in four sections: describing the shortest driving time problem and the genetic algorithm proposed to solve this problem; the basic design of the proposed route guidance system; experimental results of the genetic algorithm obtained from the networks with different sizes; and the general conclusions of the study.

II. Proposed Genetic Algorithm for Route Guidance System

A. Shortest Driving Time

It is possible to describe underlying topology of a network with directed graph G=(N,A) where N is the set of the nodes, of cardinality n, and A is the set of the arcs, of cardinality m. There is a cost Tij for each (i,j) $\in$ A.

These costs are defined in a cost matrix C=[Tij].Source and destination nodes are respectively shown as B and V. The connection information of the nodes with each other is described in an adjacency matrix Iij shown below:

$$I_{ij} = \begin{cases} 1, \text{if the link from node i to node j exist in} \\ \text{adjacency list} \\ 0, \text{otherwise.} \end{cases}$$

In shortest driving time problem, the cost is Tij which defines the driving time from node i to node j. Using the above definitions, the shortest driving time problem can be formulated as a combinatorial optimization problem minimizing the objective function below as follows [14]:

Minimize

$$\sum_{i=B}^{V} \sum_{\substack{i=B \\ i\neq V}}^{V} T_{ij} I_{ij} \tag{1}$$

subject to

$$\sum_{\substack{i=B \\ i\neq V}}^{V} I_{ij} - \sum_{\substack{i=B \\ i\neq V}}^{V} I_{ji} \quad \begin{cases} 1, \text{ if } i=B \\ -1, \text{ if } i=V \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

and

$$\sum_{\substack{i=B \\ i\neq V}}^{V} I_{ij} \quad \begin{cases} = 0, \text{ if } i=V \\ <=1, \text{ if } i\neq V \end{cases} \tag{3}$$

In shortest driving time problem, cost Tij is calculated as follows:

dij = distance from node i to node j
vij = allowable velocity limit from node i to node j
yij = traffic density from node i to node j

On calculation of the driving time from node i to node j, allowable velocity limits and traffic densities from node i to node j are considered. In this case, driving time can be formulated as showing below:

$$T_{ij} = \frac{d_{ij}}{v_{ij} x\,(1 - y_{ij})} \tag{4}$$

B. Genetic Encoding

In the chromosome structure of proposed GA, node numbers of the route from source to destination are stored as positive integer numbers. Each locus of the chromosome represents an order of a node in a routing path. The chromosome length is static. The total number of nodes N is the length of each chromosome in the network. The node numbers that represent the routing path from source (B) to destination (V) are encoded in the chromosome. If the node number of the solution is smaller than the total node number N, unused genes of the chromosome is assigned by zero value. The chromosome encoding of the proposed GA is shown in Fig.2.

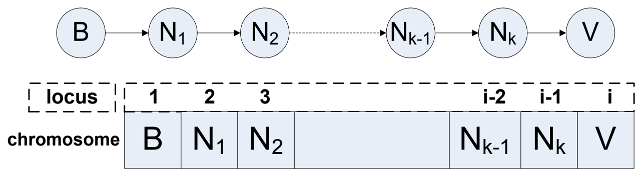

Fig.2 Chromosome encoding of a routing path

C. Creating Initial Population

To produce the initial population DFS (Depth First Search), algorithm is reorganized to produce random paths from source to destination.

D. Fitness Function

The fitness function is the object to be optimized. The fitness function must accurately measure the quality of the chromosome in the population and must have computational efficiency; therefore, the fitness function has a critical importance. The cost of the fitness function described in [14] rearranged according to the formula given in Section 2.1 to compute the shortest driving time as follows:

fi	: fitness function of the i th chromosome
g$_{i(j)}$	: j th gene of the i th chromosome
l	: length of the chromosome
D	: distance between two nodes
V	: allowable velocity limit between two nodes
Y	: traffic density between two nodes

$$f(i) = \frac{1}{\sum_{j=1}^{l-1} \frac{D(g_{i(j)}, g_{i(j+1)})}{V(g_{i(j)}, g_{i(j+1)}) x (1 - Y(g_{i(j)}, g_{i(j+1)})}} \tag{5}$$

E. Selection (Reproduction) of a New Generation

The selection (reproduction) operator is intended to improve the average quality of the population by giving the high-quality chromosomes a better chance to be copied into the next generation. In this study, roulette wheel selection method, which is a proportionate selection method that picks out chromosomes based on their fitness values relative to the fitness of the other chromosomes in the population, was performed.

In roulette wheel selection method, probability (p) of n number of chromosomes with fitness function f is calculated. Probability of the k th chromosome is calculated as follows:

$$pk = \frac{fk}{\sum_{i=1}^{n} fi} \tag{6}$$

Then, the cumulative sum of the probabilities of each chromosome in the population is calculated. The cumulative sum for the k th chromosome is calculated as follows:

$$ck = \sum_{i=1}^{k} pi \tag{7}$$

Then, a random number between 0 and 1 is generated and it is searched which cumulative sums the number is located. If the generated random number is equal or less than the first cumulative value, the first chromosome is passed to new generation directly. Otherwise, the chromosome with greater cumulative sum is passed to new generation. This process is continued as population size.

F. Crossover

The crossover operator generates new individuals called offspring, by recombining the genetic material of two individuals, deemed the parents. Individuals with higher fitness scores are selected with greater probability to be parents and ‘‘pass on’’ their genes to the next generation. In this study a new gene search method named "first-matched gene" is presented in which the first genes matched on two chromosomes as crossover points were selected. Difference of the crossover phase used in this study from classical crossover is that crossover points do not have to be in the same locus of chromosomes [11].

Crossover is done in a loop that repeats as number of chromosomes. At each cycle of the loop, a random number between 0 and 1 is generated and checked if it is smaller than crossover rate. If so, two chromosomes from the population are randomly selected for crossover, otherwise, the loop is continued.

In the end of the crossover operation, two child chromosomes is obtained. If these child chromosomes are infeasible, a repair function for dealing with the infeasible chromosomes is performed. Steps of the crossover phase are shown in Fig.3.

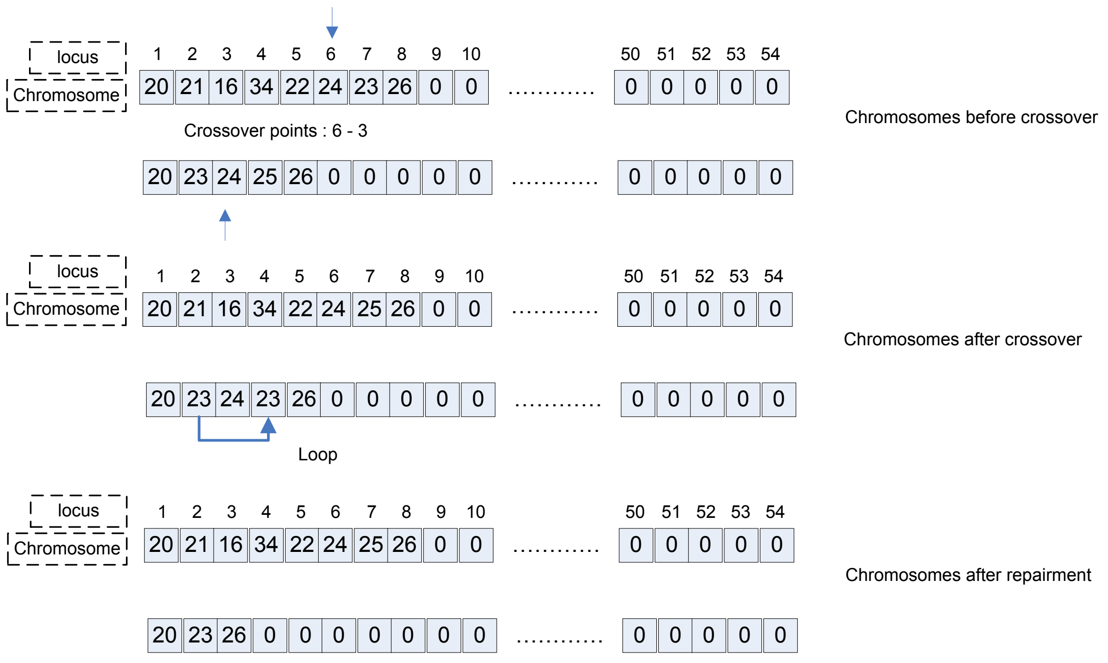

Fig.3 Crossover phase

G. Mutation

The mutation operator preserves diversification in the search. This operator is applied to each offspring in the population with a predetermined probability [16]. Through mutation, GAs can search previously unexplored sections of the solution space. Mutations consequently assure that the entire search space is connected [11].

In this study, mutation is done in a loop that repeats as number of chromosomes. At each cycle of the loop, a random number between 0 and 1 is generated and checked if it is smaller than mutation rate. If so, a chromosome from the population is randomly selected. The mutation point of the chromosome is randomly selected among the genes excluding the source and destination points. A random path is generated using reorganized DFS algorithm from the mutation node to the destination node which is also used for creating initial population. This random generated path is exchanged with the genes starting from mutation point. The mutated chromosome may be feasible. In this case, a repair function for dealing with these infeasible chromosomes is performed. Steps of the mutation phase are shown in Fig.4.

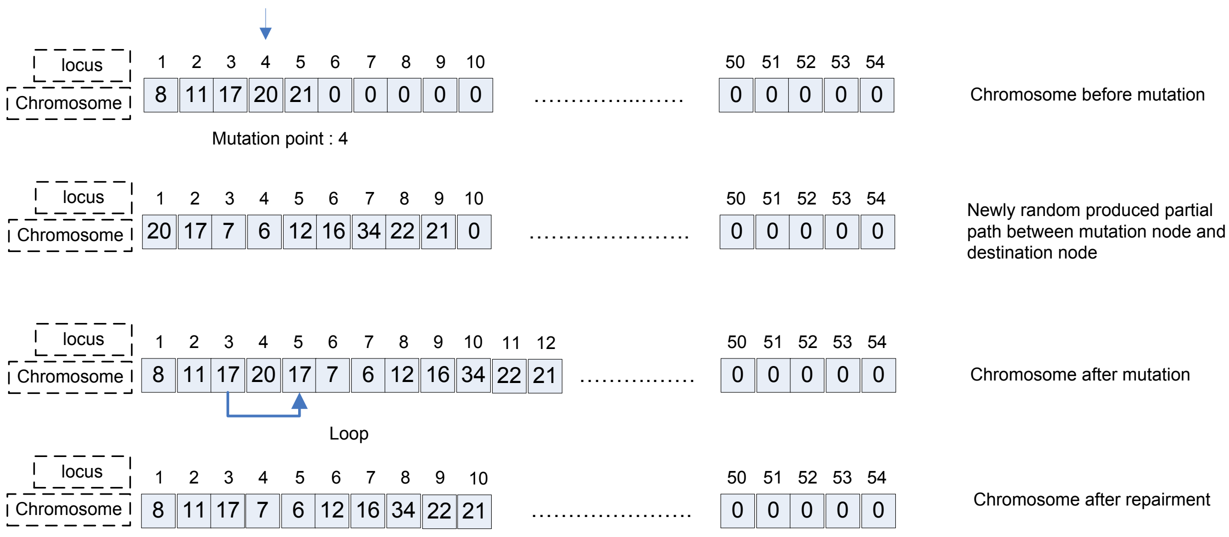

Fig.4 Mutation phase

H. *Termination of the Genetic Algorithm*

To terminate the genetic algorithm, the fitness value of the best chromosome on each generation is checked. If the fitness value of the best chromosome obtained does not change for 10 generations, the algorithm is stopped. The pseudo code of the algorithm is as follows:

```
Step 1: Initialize crossover rate, mutation rate and
        population size.
Step 2: Read the graph.
Step 3: Create initial population.
Step 4: Calculate the fitness values of the chromosomes.
Step 5: Counter=0, Generation = 1
Step 6: Repeat in infinite loop
Step 7: Roulette wheel selection.
Step 8: Crossover on selected chromosomes.
Step 9: Mutation on selected chromosomes.
Step 10:  Calculate the fitness values of the new chromosomes.
Step 11:   If (Generation>1&&
           minimum_fitness value[Generation-1]
           =minimum_fitness_value[Generation-2])
   Counter++; If (Counter>10) Stop loop.
Step 12:   Else Counter=0;
Step 13:  Generation ++; Go to Step-7.
```

III. Structure of the Designed Route Guidance System

The system had been designed for navigation devices and pocket computers that use Microsoft Mobile Operating Systems. By using genetic algorithm, it is intended to avoid the loss on the performance of these devices with restricted memory and processor capacities even on large networks with thousands of nodes. Real-time traffic conditions are obtained from a XML service. This service produces XML based data about the traffic condition periodically. Client devices also take the data periodically. The maps that supposed to be used in the route guidance system should be loaded on the device in advance and therefore only the traffic density is obtained from the XML service over Internet connection in real-time. The route guidance system is shown in Figure 5.

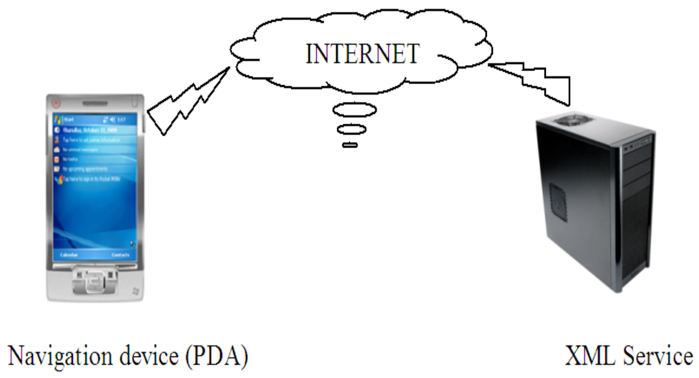

Fig.5 Route guidance system

The traffic density is simulated in the map loaded on the navigation device. Colors of the roads on the map are based on final velocity limits of the roads calculated by considering the traffic densities and allowable velocity limits of the roads. The roads are divided into four groups according to final velocity limits. The color codes of these groups are described in Table I.

Table. I Velocity ranges for coloring the map in route guidance system

Colors	Velocity Limit Ranges (km/h)
Red	0-30
Orange	30-50
Yellow	50-70
White	70-90

A sample map loaded on a pocket pc is shown in Fig.6. The obtained road after calculation is shown to users in blue color.

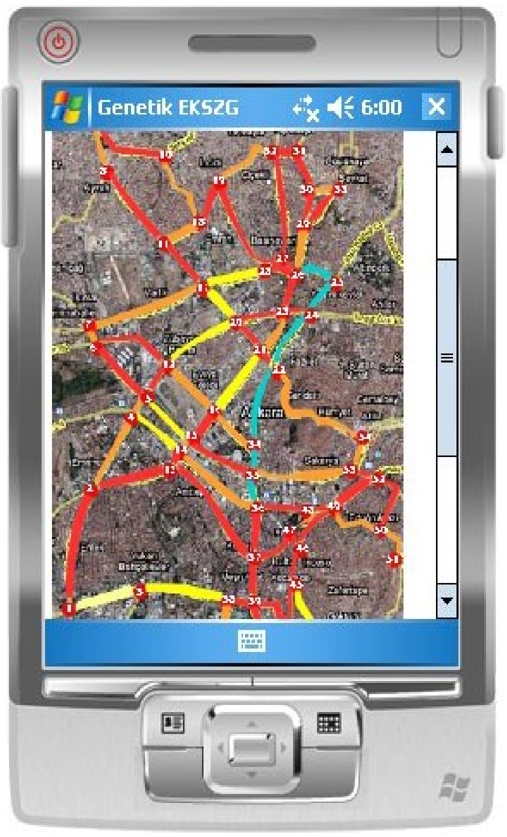

Fig.6 Map screen for the route guidance system

IV. Results

The route guidance system had been developed using C# programming language. Windows Mobile 5.0 Pocket PC R2 emulator installed on Microsoft Visual Studio 2008 is used on experiments. Beside the traffic network of Ankara city with 54 nodes, random generated graphs with 10, 50, 250 and 1000 nodes used. Distances, velocity limits and updated traffic density obtained from XML service periodically generated randomly. Experiments done 100 times for each case are given on result tables.

Our first results are taken from the city map of Ankara with 54 common nodes shown in Fig.7.

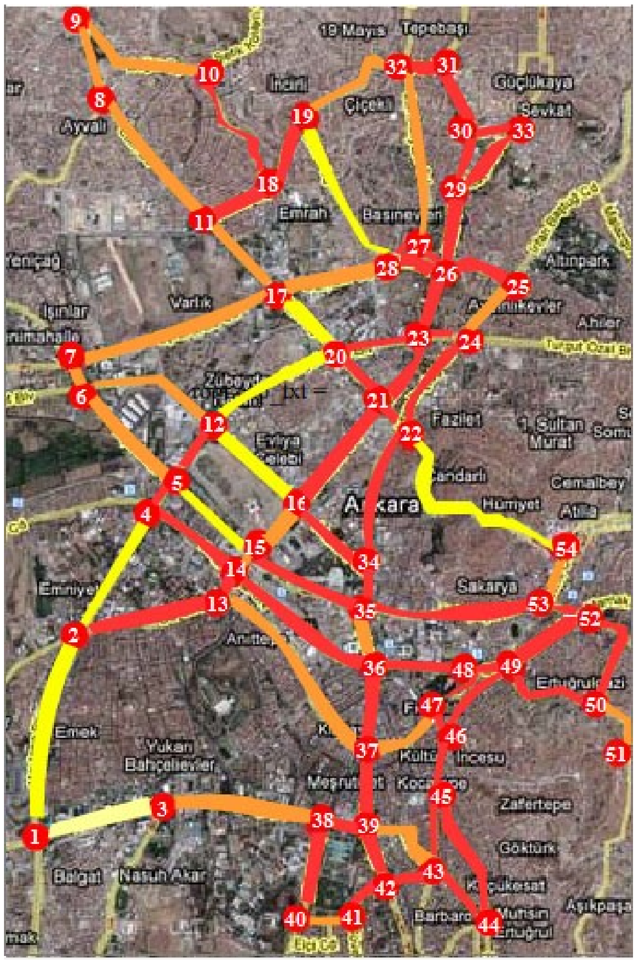

Fig.7 The map of experiments and its traffic density simulation.

A sample path found by the proposed genetic algorithm and Dijkstra algorithm from node 5 to node 8. The found path is 5-12-20-17-11-8 and shown on the map in Fig.8.

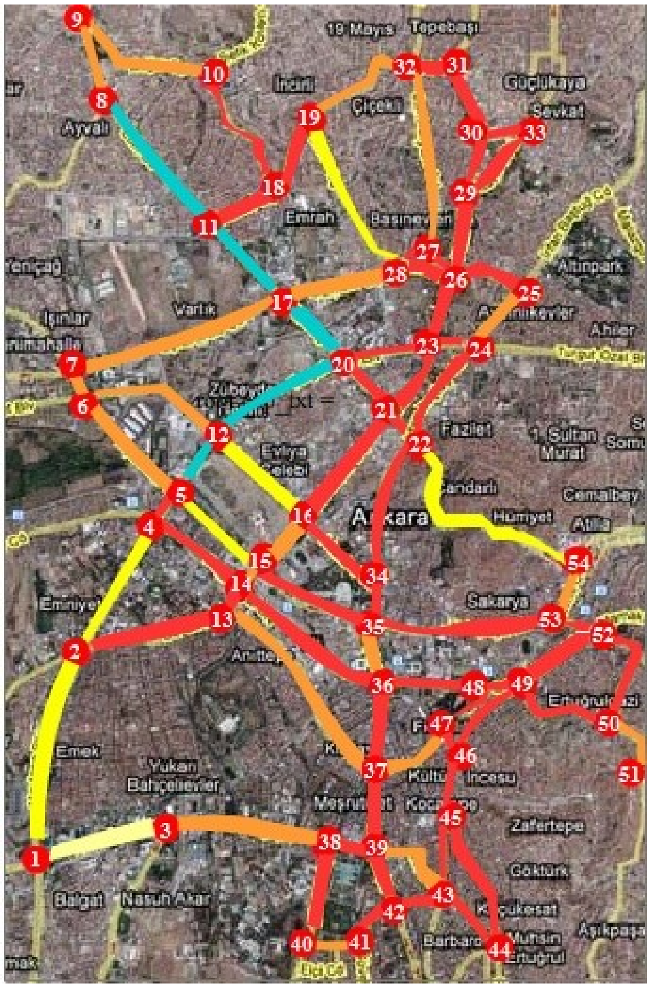

Fig.8 The path both found by Dijkstra algorithm and the proposed genetic algorithm between nodes 5 – 8.

The results of the proposed genetic algorithm found for different routes compared with the exact results found by Dijkstra algorithm and listed in Table II.

Table. II Comparison of the results obtained from proposed genetic algorithm and Dijkstra algorithm

Source Destination	Dijkstra(hour)	GA(hour)	Found Route
5-8	0,187	0,187	5-12-20-17-11-8
1-13	0,092	0,092	1-2-13
4-43	0,198	0,198	4-5-15-14-36-37-39-42-43
12-10	0,168	0,168	12-16-21-20-17-11-18-10
2-24	0,136	0,136	2-13-14-15-16-21-22-24

Performance measurement values like failure ratio and average generations of the results obtained by proposed genetic algorithm is listed in Table III.

Table. III Results obtained from 100 performs of proposed genetic algorithm on 10 different routes.

Source-Destination	Average Generation	Failure Ratio (%)
5-8	12	0
1-13	12	0
4-43	12.1	0
10-14	12.7	5
7-24	16.7	0
40-10	14	0
2-24	27.9	10
14-18	16	4
12-10	12	0
20-54	12	0

Test results performed on random generated graphs with 10, 50, 250 and 1000 nodes are presented below. Table IV shows the average generations of the genetic algorithm to find the shortest driving time.

Table. IV Average generations to find the optimum path

		Number of Nodes			
		10	50	250	1000
Population Size	30	17.05	19.43	33.09	36.4
	50	14.16	19.41	30.57	33.2
	100	12.45	16.13	27.53	28.2
	200	12.01	13.22	25.24	27.6
	400	12	12.21	21.37	23.43
	800	12	12.01	18.14	20.12

As given in Table IV, even the number of nodes grew to 1000, the average generation to find the optimum path was not much than 36.4 in worst case and by increasing the population size it was seen that the algorithm found solutions in less generations. When the number of nodes grew by 10 times from 10 to 1000, the time needed for finding the solution grew only by nearly 2 times.

Table V shows the average difference of the exact routes and approximate routes found by the proposed genetic algorithm.

Table. V Average difference of the exact routes and approximate routes in %

		Number of Nodes			
		10	50	250	1000
Population Size	30	0.6	20.7	140.1	159.3
	50	0	9.1	79.8	87.3
	100	0	3.6	57.1	71.2
	200	0	0.3	34.7	52.8
	400	0	0	20.2	33.1
	800	0	0	4	12.4

According to the results given in Table V, approximate solutions found by the proposed genetic algorithm were very close to exact solutions with the node number 10 and 50. On a graph with 10 nodes, exact solutions obtained when the population size was 50 or greater. When the node number grew to 50, the exact solutions obtained with population starting from 400. By increasing the population size, it was seen that the algorithm got closer to exact solutions. When the node number was 1000, average difference of the exact and the approximate routes was cut from 159.3% to 12.4% with the increase of population size from 30 to 800.

V. Conclusion

When the amount of data being processed is too large on handheld devices, it is not proper to find exact solutions. The GA presented in this paper finds the acceptable approximate solutions effectively even on large networks while considering real-time information.

Results obtained from experiments shows that a route guidance system that computes the shortest driving time considering the real-time traffic information can be designed using GA for handheld devices produced with limited processor and memory capacities.

References

[1] M.H. Xu, Y.Q. Liu, Q.L. Huang, Y.X. Zhang, G.F. Luan, "An improved Dijkstra's shortest path algorithm for sparse network", *Applied Mathematics and Computation*, 2007, Vol.185, No.1 pp.247-254.

[2] E.W. Dijkstra, "A Note on Two Problems in Connection with Graphs " *Numerische Mathematik* , 1959,Vol.1, No.1,pp.269-271.

[3] B. Fahimnia, R. Molaei, M. Ebrahimi, "Genetic algorithm optimization of fuel consumption in compressor stations", *In Proceedings of the 7th WSEAS International Conference on Applied Computer and Applied Computational Science*, 2008, pp. 60-68.

[4] K. Aruga, J. Sessions, A. Akay, W. Chung (2005). "Simultaneous optimization of horizontal and vertical alignments of forest roads using Tabu Search". *Int. J. Forest Eng*. 16 (2): 137-151.

[5] J.H. Holland, *Adaptation in natural and artificial system*, The University of Michigan Press, 1975.

[6] D.E. Goldberg, *Genetic Algorithms in Search Optimizations and Machine Learning*, Boston, Addison-Wesley Longman Publishing Co, 1989.

[7] X.J. Bai, L. Zhou, "A New Genetic Algorithm for solving Traveling Salesman Problem", *In Proceedings of the 8th WSEAS International Conference on Applied Computer and Applied Computational Science*, 2009, pp. 451-455.

[8] D. Whitley, "A genetic algorithm tutorial", *Statistics and Computing*, 1994, Vol.4, No.2,pp.65-85.

[9] K. De Jong, "Learning with Genetic Algorithms: An Overview", *Machine Learning*, 1988, Vol.3, pp.121–138

[10] H. Bao-Juan, Z. Jian, Y. De-Hong, "A novel and accelerated genetic algorithm", *In Proceedings of the 7th WSEAS International Conference on Applied Computer and Applied Computational Science*, 2008, pp. 245-253.

[11] D. Cedric, D. Pawan, "Genetic algorithms for rerouting shortest paths in dynamic and stochastic networks", *European Journal of Operational Research*, 2003, Vol.144,pp.27-38.

[12] M. Munemoto, Y. Takai, Y.A. Sato, "A migration scheme for the genetic adaptive routing algorithm", *In Proceedings of the Systems, Man, and Cybernetics IEEE International Conference*, 1998, pp.2774–2779.

[13] J. Inagaki, M. Haseyama, H. Kitajima, "A genetic algorithm for determining multiple routes and its applications", *In Circuits and Systems, Proceedings of the 1999 IEEE International Symposium*, 1999, pp. 137-140.

[14] C.W. Ahn, R.S. Ramakrishna, "A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations", *IEEE Transactions on Evolutionary Computation*, 2002, Vol.6, No.6,pp.566-579.

[15] B.S. Hasan, M.A. Khamees, A.S.H. Mahmoud, "A Heuristic Genetic Algorithm for the Single Source Shortest Path Problem", *5th IEEE/ACS International Conference on Computer Systems and Applications*, 2007, pp.187-194.

[16] J. Magalhães-Mendes, "Project scheduling under multiple resources constraints using a genetic algorithm", *WSEAS TRANSACTIONS on BUSINESS and ECONOMICS*, Issue 11, Volume 5, November 2008, pp. 487-496.