Scientific Research and Essays Vol. 6(2), pp. 394-405, 18 January, 2011
Available online at http://www.academicjournals.org/SRE
ISSN 1992-2248 ©2011 Academic Journals

Full Length Research Paper

A genetic algorithm approach for finding the shortest driving time on mobile devices

Ismail Rakip Karas[1]* and Umit Atila[2]

[1]Computer Engineering Department, Faculty of Engineering, Karabük University, Baliklarkayasi Mevkii, 78050, Karabuk, Turkey.
[2]Directorate of Computer Center, Gazi University, Teknikokullar, Besevler, 06500, Ankara, Turkey.

Accepted 24 November, 2010

Recently, with the increasing interest in using handheld devices, the application of navigation systems that provide driving information to the drivers has become widespread in daily life. An efficient route guidance system should consider the influential factors of traffic flow such as traffic density and allowable velocity limits of the roads. As the number of influential factors and amount of nodes in road network increase, the computational cost increases. On navigation systems, using handheld devices with limited processing speed and memory capacity, it is not feasible to find the exact optimal solution in real-time for the road networks with excessive number of nodes using deterministic methods such as Dijkstra algorithm. This paper proposes a Genetic Algorithm approach applied to a route guidance system to find the shortest driving time. Constant length chromosomes have been used for encoding the problem. It was found that the mutation operator proposed in this algorithm provided great contribution to achieve optimum solution by maintaining the genetic diversity. The efficiency of the genetic algorithm was tested by applying it on the networks with different sizes.

Key words: Genetic algorithm, navigation, route guidance, optimization, shortest path, shortest driving time.

INTRODUCTION

The use of navigation systems that provide driving information to the drivers has become widespread in daily life. The logic behind this kind of systems provides the users with the shortest path between beginning and end points. The downside of the current navigation systems ignores the important decision variables including the traffic density and allowable velocity limits of the roads. However, the shortest path suggested by a system cannot be the optimal route in every case, if these variables are not being considered. Therefore, an ideal navigation system should not only consider the distances, but also the traffic density and allowable velocity limits between intersection points in real-time.

The shortest path problem

The shortest path problem is to find 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 (Xu et al., 2007). As a brief explanation, let G = (N, A) be a simple directed graph, where N is the set of the nodes, of cardinality n, and A is the set of the arcs, of cardinality m.

Let c: A $\rightarrow$ R be a function which assigns a cost cij to each (i, j) $\in$ A. Given a root r $\in$ N, the problem of the shortest path tree (SPT) is seen in finding a directed tree T such that the (only) path from r to i in T is one of the shortest paths from r to i in G, and for each i $\in$ N which is connected to r, a directed path from r to i exists (Pallottino and Grazia, 1996). The computation of the shortest paths is an important task in network analysis and transportation related analysis. One of the most popular algorithm is the conceived Dijkstra's algorithm, which solves the shortest path problem in O (n^2) time on a graph with n number of nodes and positive edge weights (Dijkstra, 1959).

*Corresponding author. E-mail: ismail.karas@karabuk.edu.tr. Tel: +90370 4332021 (Ext. 188). Fax: +90370 4333290.

Genetic algorithm

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 the deterministic methods, heuristic techniques do not guarantee optimal solutions, but they can find good/near optimal solutions within a reasonable time (Aruga et al., 2005). Genetic algorithm (GA) is a heuristic technique developed by John Holland in 1975 based on genetic and natural selection principles (Holland, 1975). Goldberg (1989) proved that genetic algorithm is one of the powerful search methods in both theory and practice. Genetic algorithm 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 (Whitley, 1994).

The genetic algorithm founded is based on two fundamental evolutionary concepts:

1. A Darwinian notion of fitness, which describes an individual's ability to survive.
2. Genetic operators, which determine the next generation's genetic makeup based on the current generation (De Jong, 1988).

Conventionally, genetic operations are achieved through crossover and mutation operators. The crossover operator generates new individuals called offspring, by recombining the genetic material of two individuals, deemed as the parents. Individuals with higher fitness scores are selected with greater probability to be parents and ''pass on'' their genes to the next generation. This is known as the fitness proportional selection method. Crossovers allow exploitation of successful subspaces of the solution space. The mutation operator randomly alters one or more genes in an individual. Mutations add genetic diversity to the population, in that through mutation, GAs can search previously unexplored sections of the solution space. Consequently, mutations ensure that the entire search space is connected (Cedric and Pawan, 2003).

In the genetic algorithm, the initial population is evaluated on the optimal solution by crossover and mutation operations. The first step starts with obtaining the values that the fitness function returns for each chromosome and selects the best chromosomes of the 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 the newly generated child chromosomes form a new population (Holland, 1975; Whitley, 1994). The generation of new populations is repeated for a defined number of times in advance or is being continued until there are no better chromosomes (Figure 1).

In the last decade, there have been a number of approaches used by GA in the solution of the shortest path problems. Munemoto et al. (1998) implemented a GA, which is practically feasible on the wired or wireless network environment. For encoding the problem, they used chromosomes with variable lengths. They defined crossing points as the loci (positions of nodes in a route), where identical genes (nodes) in both chromosomes (routes) are found at the same location and at the selected location of the crossing point, randomly.

Inagaki et al. (1999) proposed an algorithm with fixed length chromosomes. The chromosomes in the algorithm are sequences of integers and each gene represents a node ID that is selected randomly from the set of nodes connected with the node corresponding to its locus number. In the crossover phase, one of the genes (from two parent chromosomes) is selected at the locus of the starting node ID and put in the same locus of an offspring. One of the genes is then selected randomly at the locus of the previously chosen gene's number. This process is continued until the destination node is reached.

Ahn and Ramakrishna (2002) 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, since Dijkstra's algorithm had a prohibitive computational cost. On the other hand, they suggested that the algorithm proposed by Inagaki et al. (1999) required a large population to attain an optimal or high quality of solution due to its inconsistent crossover mechanism. Some offspring might generate new chromosomes that resemble the initial chromosomes in fitness, thereby retarding the process of evolution. Ahn and Ramakrishna (2002) proposed a GA for solving the shortest path problem that uses chromosomes with variable lengths. A chromosome (routing path) encodes the problem by listing node IDs from its source node to its destination node based on the topological information database (routing table) of the network. The gene of the first locus is always reserved for the source node and the gene of the second locus is randomly selected from the nodes connected with the source node. A chosen node is removed from the topological information database to prevent the node from being selected twice, thereby avoiding loops in the path. It is possible that the algorithm encounters a node for which all of the neighboring nodes have already been visited.

In this case, the defective chromosome is refreshed and reinitialized. In the crossover phase, a set of node pairs, which is commonly included in the two (chosen) chromosomes (but without positional consistency) is

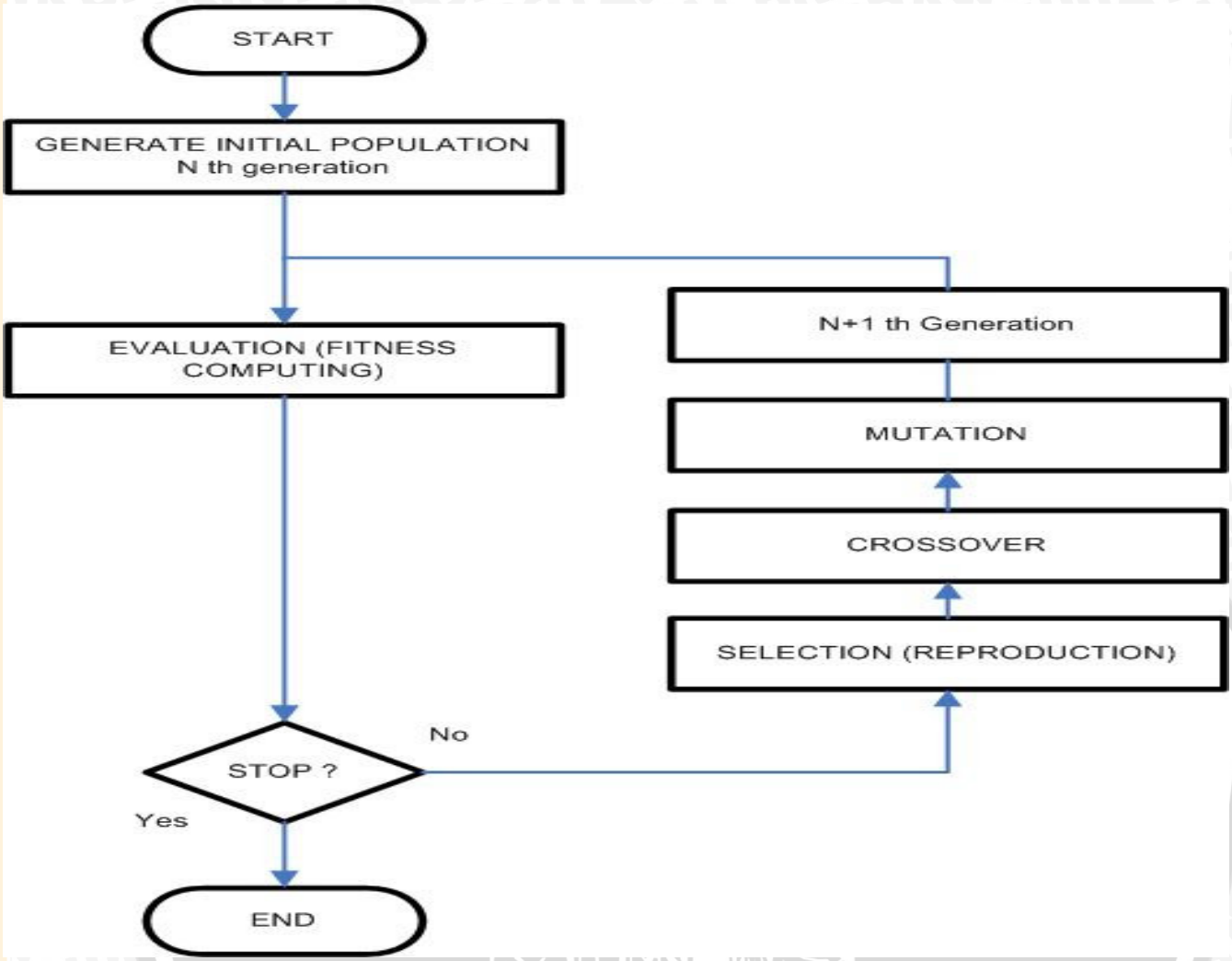

Figure 1. Steps of genetic algorithm.

formed in the first phase. Then, one pair is randomly chosen and the locus of each node becomes a crossing site of each chromosome. Each partial route is exchanged and assembled and thus, two new routes are produced eventually. In the mutation phase, the proposed genetic algorithm generates an alternative partial-route from the mutation node to the destination node using topological database. After crossover and mutation phases, some repair functions are taken to avoid infeasible chromosomes. Ahn and Ramakrishna (2002) worked on networks with 15 to 50 nodes through randomly assigned link costs. The quality results of the proposed algorithm were compared with the algorithms of Munemoto et al. (1998) and Inagaki et al. (1999), which indicated that the solution quality of the proposed algorithm was much higher than the others. Ahn and Ramakrishna (2002) showed that as the number of the nodes becomes more than 20, the computing time by adopting the GA becomes less than that when the Djikstra algorithm is adopted.

Hasan et al. (2007) produced a different solution for the shortest path problem using GA. 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. Each node was represented by three genes. The first value represents the previous array in the classical Dijkstra's algorithm, while the second value is the node number itself and the third value is used to store the cost of the path from the source to the target node. The first node of every candidate path is the source node. Other entries are random nodes, covering all other nodes in the graph, with random predecessors, preventing self-edge nodes. They proposed different crossover and mutation methods, which are appropriate in encoding their chromosomes. They tested their proposed algorithm on networks with 10, 20, 50 and 100 routers, and the results demonstrated

consistent and speedy convergence for the tested scenarios.

Lin et al. (2009) designed a route guidance system based on the genetic algorithm of Ahn and Ramakrishna (2002) for finding the shortest driving time which is their application on virtual maps of square matrix with appropriate to be used on handheld devices. They tested sizes of 4 x 4, 8 x 8, 16 x 16 and 32 x 32, and on a real map with 8039 nodes. They adopted both Dijkstra and GA for the shortest driving time, but they only reported the results of GA as the memory required by the Dijkstra algorithm went beyond the limited memory of the portable device. As a result, they reported that the shortest driving time approach, which can be computed by the GA on handheld device, was feasible to be used in the route guidance system.

Objectives and organization

Increasing the number of nodes in a network, and the parameters considered in calculating the shortest driving time, increases the resource consumption and computational cost of the handheld device with limited processing speed and memory capacity. To overtake these problems, heuristic algorithms with approximate solutions can be used rather than deterministic algorithms with exact solutions. The GA is one of them.

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

For encoding the problem, chromosomes with constant length have been used. The length of the chromosome is the number of nodes on the network. The genes of a chromosome represent nodes included in a path between a designated pair of source and destination nodes.

The crossover used in the algorithm is identical to the method proposed by Ahn and Ramakrishna (2002) except a difference on implementation. To produce the initial population DFS (depth of first search), the algorithm was reorganized to make a random selection of the nodes from source to destination and the same approach was also used in the mutation phase to produce alternative paths from the mutation point to the destination point.

The remaining parts of this paper is organized as follows: a description of the shortest driving time problem and the genetic algorithm proposed to solve this problem; the basic design of the proposed route guidance system; the experimental results of the genetic algorithm obtained from the networks with different sizes; and the general conclusions of the study.

THE PROPOSED GENETIC ALGORITHM FOR THE ROUTE GUIDANCE SYSTEM

Shortest driving time

It is possible to describe this network with a 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) ∈ 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 as follows (Ahn and Ramakrishna, 2002):

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

In the shortest driving time problem, the cost which is Tij defines the driving time from node i to node j. Using these definitions, the shortest driving time problem can be formulated as a combinatorial optimization problem, minimizing the objective function as follows (Ahn and Ramakrishna, 2002):

$$\text{Minimize} \quad \sum_{i=B}^{V} \sum_{\substack{i=B \\ i \neq V}}^{V} T_{ij} I_{ij} \qquad (1)$$

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

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

In the 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

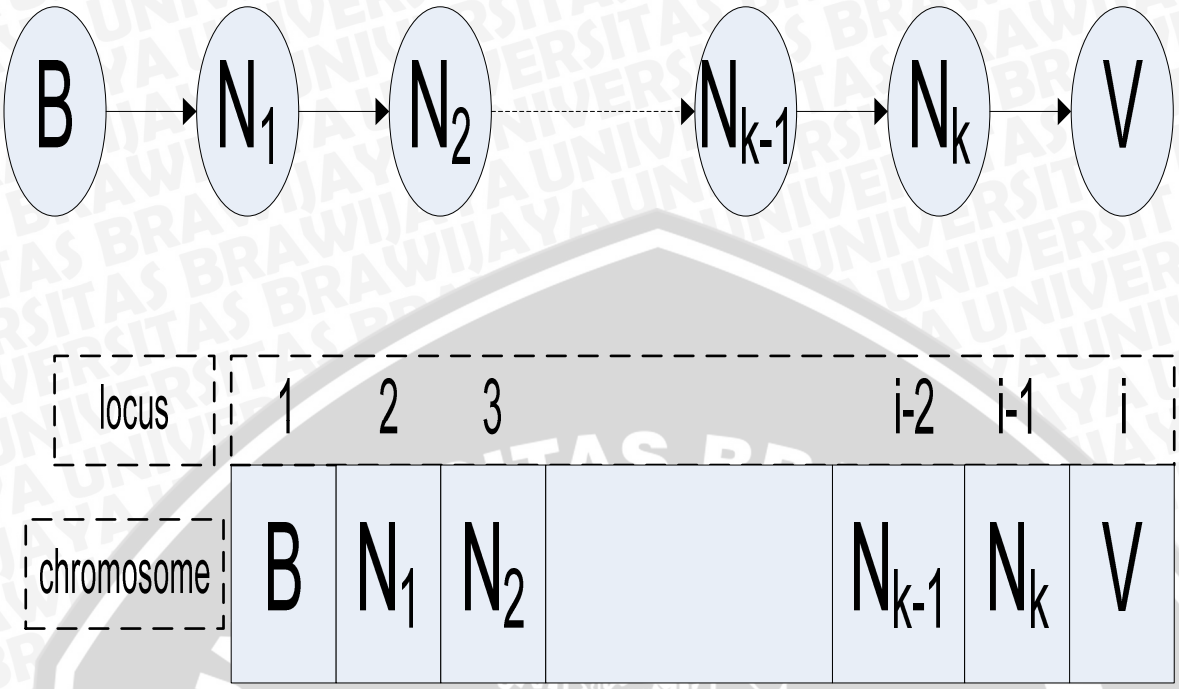

Figure 2. Chromosome, encoding a routing path.

to node j are considered. In this case, driving time can be formulated as follows:

$$Tij = \frac{dij}{vij \times (1 - yij)} \quad (4)$$

Genetic representation

In the chromosome structure of the 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, and 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 are assigned by a zero value. The chromosome encoding the proposed GA is shown in Figure 2.

Initialization of the population

To produce the initial population of the DFS (depth of first search), the algorithm is reorganized to produce random paths from source to destination. The pseudo code for population initialization is as follows:

{Step 1: Store the source node in the gene of the first locus of the chromosome.
Step 2: Randomly select a node among the nodes that the current node is directly connected to and not visited before.
Step 3: If there is no node to select, cancel the chromosome and go to Step 1.
Step 4: Store the selected chromosome in the gene of the next locus of the chromosome.
Step 5: If the selected node is not the destination node, go to Step2.
Step 6: If the selected node is the destination node, store the node in the gene of the next locus of the chromosome.}

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 by Ahn and Ramakrishna (2002) rearranged according to the formula given in “the shortest driving time” to compute the shortest driving time is as follows:

fi: fitness function of the i th chromosome;
gi (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.

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

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, the probability (p) of the n number of chromosomes with the fitness function of f is calculated. However, the probability of the k th chromosome is calculated as follows:

$$pk = \frac{fk}{\sum_{i=1}^{n} fi} \quad (6)$$

Subsequently, 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:

$$ck = \sum_{i-1}^{k} pi \quad (7)$$

After that, a random number between 0 and 1 is generated, and the particular cumulative sum that has the number is searched. If the generated random number is equal or less than the first cumulative value, the first chromosome is passed on to the new generation directly. Otherwise, the chromosome with greater cumulative sum is passed on to the new generation. This process is continued as population size increases.

Crossover

Crossover operation is applied to obtain better chromosomes. The crossover used in the proposed algorithm is identical to the method proposed by Ahn and Ramakrishna (2002), except the difference in implementation. Ahn and Ramakrishna (2002) searched the same nodes on two chromosomes in crossover toindicate the potential crossover points, and thus, select one of the points randomly among them. In the first-matched gene search method proposed in this study, the first genes matched on two chromosomes as crossover points were selected. The difference of the crossover phase used in this study from the classical crossover is that crossover points do not have to be in the same locus of chromosomes. The crossover points may be different for each parent chromosomes in the crossover phase, in that the crossover is done in a loop that is repeated as a 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 the 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 are obtained. If these child chromosomes are infeasible, a repair function for dealing with the infeasible chromosomes is performed. The pseudo code used for the crossover phase is as follows:

{Step 1 : Generate a random number between 0 and 1.
Step 2 : If the random number is smaller than the crossover rate, go to Step 3, otherwise go to the next cycle of the loop.
Step 3 : Select two chromosomes randomly from the population.
Step 4 : Search for the matching gene starting from the gene of the second locus of the first chromosome. Select the locus numbers of the chromosomes in which the first matched genes are included as crossover points.
Step 5 : Starting from the crossover point, exchange the genes between the chromosomes.
Step 6 : If the newly generated chromosomes have loops (feasible), remove the loops.
Step 7: Move the possible zero gene values, which may occur after removing loops, to the highest locus numbers of the chromosomes.
Step 8 : Pass the newly generated chromosomes to the population.}

Steps of the crossover phase are shown in Figure 3.

Mutation

Mutation operation maintains the genetic diversity of the population and changes the genes of the selected chromosomes, thereby keeping them away from local optima. In this study, mutation is done in a loop that is repeated as a 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 the mutation rate. If so, a

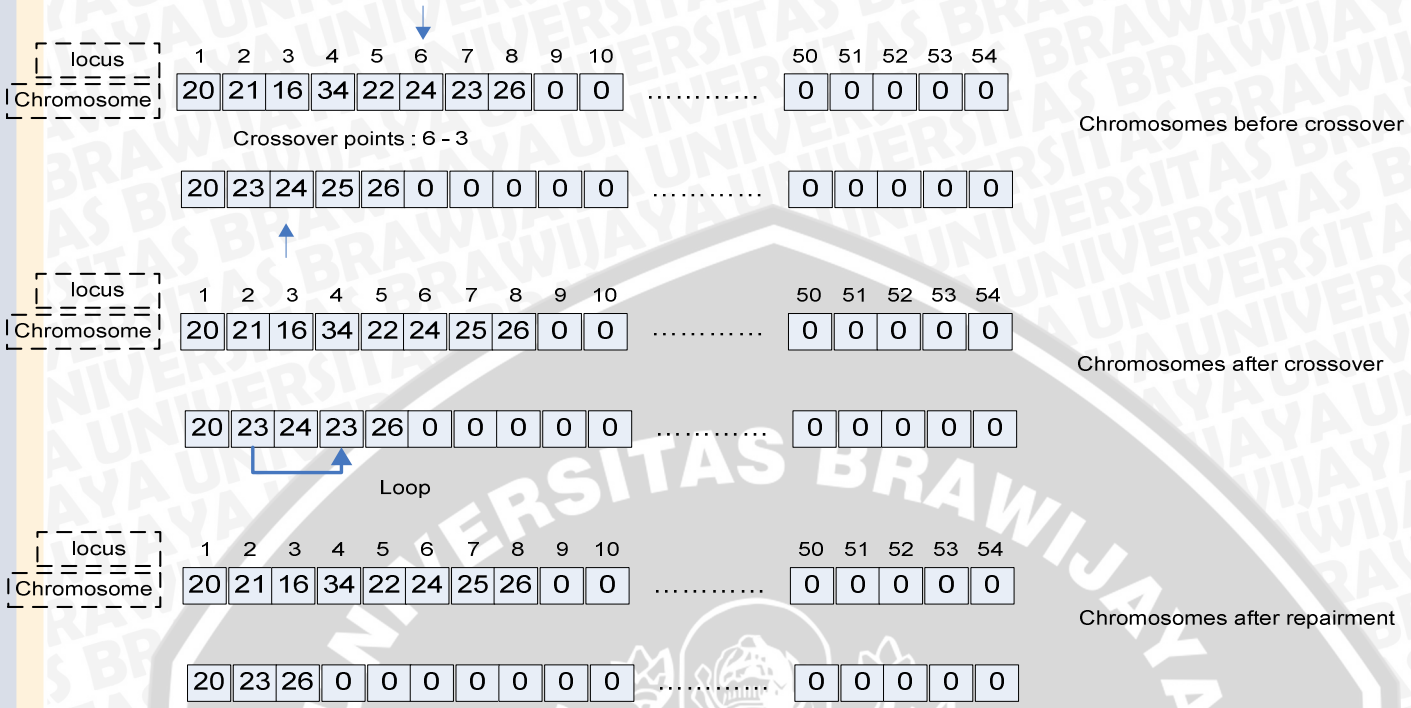

Figure 3. Crossover phase.

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 the algorithm described in the "initialization of the population" from the mutation node to the destination node. This random generated path is exchanged with the genes starting from the mutation point, due to the fact that the mutated chromosome may be feasible. In this case, a repair function for dealing with these infeasible chromosomes is performed. The pseudo code used for the mutation phase is as follows:

{Step 1: Generate a random number between 0 and 1.
Step 2: If the random number is smaller than the mutation rate, go to Step 3, otherwise go to the next cycle of the loop.
Step 3: Select a chromosome randomly from the population.
Step 4: Select a random gene from the chromosome as the mutation point, excluding the source and destination genes.
Step 5: Generate a random path from the mutation node to the destination node.
Step 6 : Exchange the generated path with the genes starting from the mutation point.
Step 7: If the newly generated chromosomes have loops (feasible), remove the loops.
Step 8: Move the possible zero gene values which may occur after removing the loops to the highest locus numbers of the chromosome.
Step 9: Pass the mutated chromosome to the population.}

Steps of the mutation phase are shown in Figure 4.

Steps of the proposed genetic algorithm and termination

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 a follows:

{Step 1: Initialize the crossover rate, mutation rate and population size.
Step 2: Read the graph.
Step 3: Create the initial population.
Step 4: Calculate the fitness values of the chromosomes.
Step 5: Counter = 0, Generation = 1.
Step 6: The values are repeated in the 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 and minimum_fitness of the value [Generation 1]== minimum_fitness_value

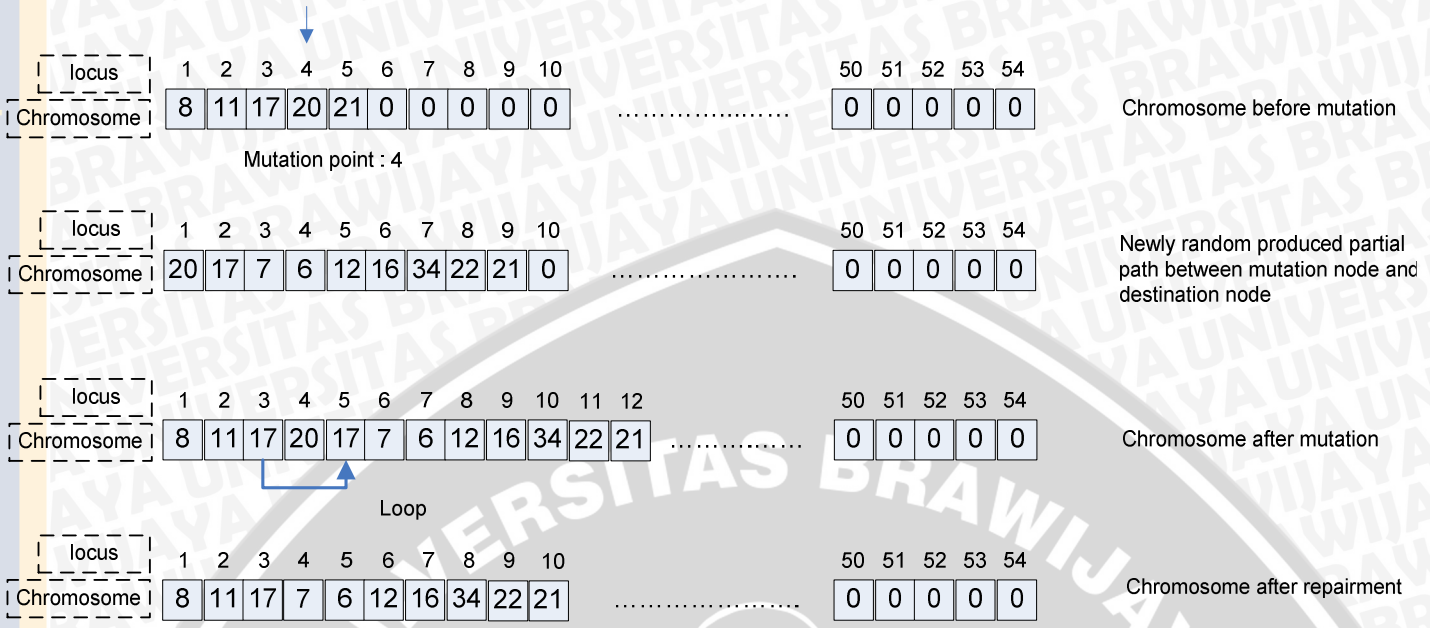

Figure 4. Mutation phase.

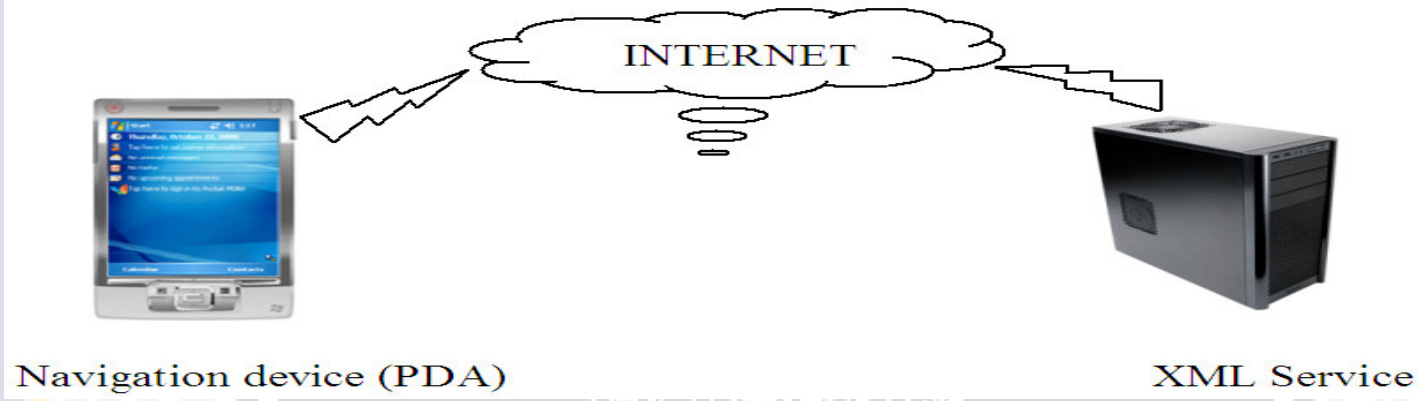

Figure 5. Route guidance system.

[Generation-2]) Counter++; If (Counter>10), stop loop.
Step 12: Counter = 0;
Step 13: Generation ++; Go to Step 7.}

Basic principles of the designed route guidance system

The route guidance system proposed was recommended to the drivers on the path with the least driving time from source to destination by considering the parameters that affect driving time, like traffic density and allowable velocity, in real-time. The system had been designed for navigation devices and for pocket computers that use processor capacities even on large networks with thousands of nodes. Real-time traffic conditions are obtained from a XML service and this service produces XML data based on the traffic condition periodically.Client devices also take the data periodically. The maps that are supposed to be used in the route guidance system should be loaded on the device in advance and therefore, only the traffic density should be obtained from the XML service over internet connection in real-time. The route guidance system is shown in Figure 5 and the traffic density is simulated in the map loaded on the navigation device. Colors of the roads on the map are based on the 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 the final velocit y limits. The color codes of these

Table 1. Velocity ranges for coloring the map in the route guidance system.

Colors	Velocity limit ranges (km/h)
Red	0 to 30
Orange	30 to 50
Yellow	50 to 70
White	70 to 90

Figure 6. Map screen for the route guidance system.

groups are described in Table 1. A sample map loaded on a pocket pc is shown in Figure 6, while the obtained road after calculation is shown to users in blue color.

Table 2. Effect of the crossover rate.

Crossover rate (%)	Average generation number	Average difference (%)
10	16	42
20	15	37
30	14	36
40	14	34
50	14	29
60	14	28
70	14	23
80	14	22
90	13	21
100	13	15

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. Random generated graphs with 10, 50, 250 and 1000 nodes are used. Distances, velocity limits and updated traffic density obtained from XML service are periodically generated randomly. The experiments done 100 times for each case are given on the result tables. On a graph with 50 nodes, when the population size was 100 and the mutation rate was set to be 5%, the average difference of the exact solution found by Dijkstra's algorithm and the approximate solution found by the proposed genetic algorithm was shown on Table 2 in order to confirm the effect of the crossover operator.

As given in Table 2, by increasing the crossover rate, the average difference of the exact and approximate solutions had decreased and no notable difference had been seen on average generations. When the crossover rate was 70%, the average difference was decreased from 42 to 23% and after this point, it was not less than 15% even on 100% crossover rate. With the increase in the crossover rate, the processor and memory consumption increased. Thus, it is feasible to select the crossover rate between ranges of 70 and 80%. On a graph with 50 nodes, the average difference of the exactsolution found by Dijkstra's algorithm and the approximate solution found by the proposed genetic algorithm was shown on Table 3 in order to show the effect of the mutation operator when the population size was 100 and the crossover rate was set to be 75%. As given in Table 3, the algorithm, found in the approximate solutions with average difference between the ranges of 2 and 5%, started from the mutation rate of 30%. Variation on the mutation rate affected the average

Table 3. Effect of the mutation rate.

Mutation rate (%)	Average generation number	Average difference (%)
1	20	71
5	19	34
10	18	23
20	18	12
30	17	5
40	19	4
50	20	2
60	18	2
70	18	2
80	19	2
90	20	2
100	20	2

Table 4. 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

difference rather than the average generations. By increasing the mutation rate from 1 to 30%, the average difference of the exact and approximate solutions was cut from 71 to 5%. Considering the resources consumption, it was feasible to select the mutation rate at about 30%. Table 4 shows the average generations of the genetic algorithm to find the shortest driving time. As given in Table 4, even the number of nodes grew to 1000, while the average generation to find the optimum path was not much than 36.4, which is in a worst case, and by increasing the population size, it was seen that the algorithm found solutions in fewer generations. When the number of nodes grew 10 times from 10 to 1000, the time needed for finding the solution grew only almost twice.

Fitness function evolution of the proposed genetic algorithm is shown in Figure 7. It can be seen on the graph that on each step to the next generation, the minimum fitness function and average fitness function of the chromosomes in the population converge with each other. The average fitness values of the proposed genetic algorithm that converges with the exact solution found by Dijkstra algorithm is shown in Figure 8. Table 5 shows the average difference of the exact and approximate routes found by the proposed genetic algorithm. According to the results given in Table 5, the approximate solutions found by the proposed genetic algorithm were very close to the exact solutions with node number 10 and 50. On a graph with 10 nodes, the exact solutions were obtained when the population size was 50 or greater than 50. When the node number grew to 50, the exact solutions were obtained with the population starting from 400. By increasing the population size, it was seen that the algorithm got closer to the exact solutions. When the node number was 1000, the average difference of the exact and approximate routes was cut from 159.3 to 12.4% with the increase of the population size from 30 to 800.

Conclusions

An efficient route guidance system should provide driving advice considering the influential factors of traffic flow such as traffic density and allowable velocity limits of the roads obtained from internet based information providers rather than showing only the shortest path from source to destination. On handheld devices, it is not proper to find the exact solutions when the amount of the data being processed is too large. The GA presented in this paper finds the acceptable approximate solutions effectively even on large networks, while considering real-time information. Results obtained from the experiments show

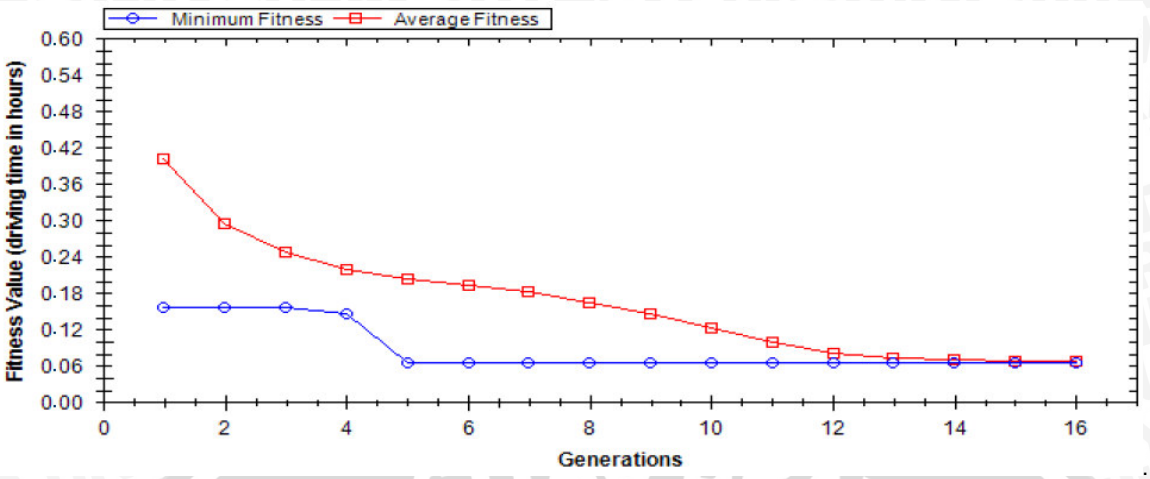

Figure 7. Convergence property of the genetic algorithm.

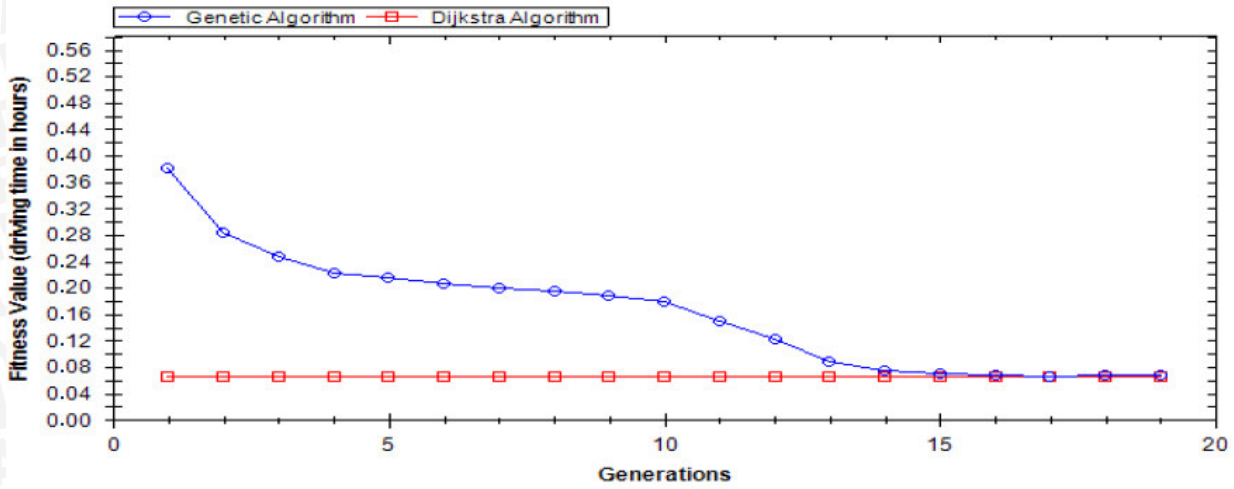

Figure 8. Convergence property of the proposed genetic algorithm and Dijkstra algorithm.

Table 5. Average difference of the exact 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

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

Ahn CW, Ramakrishna RS (2002). A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations. IEEE Trans. Evol. Comput., 6: 566-579.

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

Cedric D, Pawan D (2003). Genetic algorithms for rerouting shortest paths in dynamic and stochastic networks. Eur. J. Oper. Res., 144: 27-38.

De Jong K (1988). Learning with Genetic Algorithms: An Overview. Machine Learn., 3: 121-138.

Dijkstra EW (1959). A Note on Two Problems in Connection with Graphs. Numerische Math., 1: 269-271.

Goldberg DE (1989). Genetic Algorithms in Search Optimizations and Machine Learning. Boston, Addison-Wesley Longman Publishing Co.

Hasan BS, Khamees MA, Mahmoud ASH (2007). A Heuristic Genetic Algorithm for the Single Source Shortest Path Problem. In Computer Systems and Applications, Proceedings of the 5th IEEE/ACS International Conference. Amman, Jordan, pp. 187-194.

Holland JH (1975). Adaptation in natural and artificial system. Ann Arbor, The University of Michigan Press.

Inagaki J, Haseyama M, Kitajima H (1999). A genetic algorithm for determining multiple routes and its applications. In Circuits and Systems, Proceedings of the 1999 IEEE International Symposium. Orlando, FL, USA, pp. 137-140.

Lin CH, Yu JL, Liu JC, Lai WS, Ho CH (2009). Genetic Algorithm for Shortest Driving Time in Intelligent Transportation Systems. Int. J. Hybrid Inf. Tech., 2 (1): 21-30.

Munemoto M, Takai Y, Sato YA (1998). A migration scheme for the genetic adaptive routing algorithm. In Proceedings of the Systems, Man, and Cybernetics 1998 IEEE International Conference. San Diego, CA, USA, pp. 2774-2779.

Pallottino S, Grazia M (1996). Shortest path algorithms in transportation models: Classical and innovative aspects. In Proceedings of the 25th Anniversary Meeting of the Centre for Research on Transportation on Equilibrium and Advanced Transportation Modeling. Montreal, Canada, pp. 245-281.

Whitley D (1994). A genetic algorithm tutorial. Statist. and Comput., 4: 65-85.

Xu MH, Liu YQ, Huang QL, Zhang YX, Luan GF (2007). An improved Dijkstra's shortest path algorithm for sparse network. Appl. Math. Comput., 185: 247-254.