Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 765135, 6 pages
http://dx.doi.org/10.1155/2013/765135

Research Article

Dynamic Route Guidance Using Improved Genetic Algorithms

Zhanke Yu,[1] Mingfang Ni,[1] Zeyan Wang,[1] and Yanhua Zhang[2]

[1] *Institute of Communication Engineering, PLA University of Science and Technology, NanJing 210007, China*
[2] *Lightning Protection Center, Meteorology Bureau of Jiangsu Province, NanJing 210009, China*

Correspondence should be addressed to Zhanke Yu; jackty_2004@163.com

Received 27 September 2012; Accepted 30 December 2012

Academic Editor: Rui Mu

This paper presents an improved genetic algorithm (IGA) for dynamic route guidance algorithm. The proposed IGA design a vicinity crossover technique and a greedy backward mutation technique to increase the population diversity and strengthen local search ability. The steady-state reproduction is introduced to protect the optimized genetic individuals. Furthermore the junction delay is introduced to the fitness function. The simulation results show the effectiveness of the proposed algorithm.

1. Introduction

Online route guidance is one of the most desirable features in intelligent transportation systems (ITSs) [1]. Dynamic route guidance algorithm compute routes with minimum travel time by taking into account the rapid changes in the network traffic conditions and guide the behavior of travelers by providing them with optimal route based on real-time traffic information. As a result the travel time can be saved and the traffic congestion can be avoided.

Various algorithms are available for computing the optimal route. The most popular algorithm is the Dijkstra's algorithm. Many variations to the Dijkstra's algorithm such as bidirectional search and binary heap implementation have been proposed to improve its response time. The A^* algorithm, which is widely used in vehicle navigation, is an improved version of the Dijkstra's algorithm [2, 3]. It makes use of an appropriate heuristic function to search the most promising nodes first thereby reducing the computation time. However, the traditional optimal algorithms often cannot be used because they are too computationally intensive to be feasible for real-time operations. A number of heuristic search strategies have been developed for increasing the computational efficiency of route search. Most of these heuristic search algorithms originated in the artificial intelligence field, such as genetic algorithm (GA) [4]. GA is an intelligence search algorithm based on the hypothesis of natural selections and natural genetics and has been successfully applied to various areas of optimization [5–10]. GA-based approaches have several advantages. Naturally, they cannot only treat the discrete variables but also overcome the dimensionality problem. In addition, they have the capability to search for the global optimum or quasi optimums within a reasonable computation time. But in engineering practice, premature convergence often happens, and sometimes the speed of convergence is very slow. Accordingly, researchers have been designing the improved genetic algorithms. Zou proposes an improved genetic algorithm for dynamic route guidance algorithm that generate initial population by random A^* algorithm to increase the population diversity [8]. Kanoh proposes a method for finding the quasishortest route within a given time using a genetic algorithm. In the proposed method, he improves the search rate and quality of solutions by giving direction to the search, using viruses [9].

In this paper, an improved genetic algorithm (IGA) for dynamic route guidance, which can overcome the aforementioned problems of the conventional GA to some extents, is developed. The proposed IGA incorporates the following three main features. First, a vicinity crossover scheme is devised to increase the diversity of population. Second, a greedy backward mutation scheme is developed to strengthen the local search ability. Third, the steady-state reproduction is introduced to protect the optimized genetic individuals.

In order to test the performance of the proposed algorithm, we implement the algorithm in matlab 2009. Simulation experiments demonstrate the effectiveness of the algorithm by evaluating it on random road topologies.

The paper is organized as follows. Problem definition and mathematical formulation is introduced in Section 2. The improved genetic algorithm for dynamic route guidance algorithm is developed in Section 3. The numerical simulation is reported in Section 4. Finally, conclusions are presented in Section 5.

2. Problem Definition and Mathematical Formulation

A road traffic network is represented by a digraph $G(V, E)$ that consists of a set of nodes V and a set of links E. Denote the number of nodes $|V| = n$ and the number of links $|E| = m$. A link $E_{ij} \in E$ is directed from node i to node j. Each link has an associated average travel time function $t(E_{ij})$ and delay function $d(E_{ij})$. Let $T(E_{ij})$ denote the weight of link E_{ij}. It can be expressed as

$$T\left(E_{ij}\right) = t\left(E_{ij}\right) + d\left(E_{ij}\right). \tag{1}$$

A path from an origin (o) to destination (d) may be defined as a sequential list of links: $(o, j), \ldots, (i, d)$ and the weight of the path is the sum of weights on the individual links. The problem is to find the path P that has the minimum total weight from the origin node to the destination node

$$\min \sum_{E_{ij} \in P} T\left(E_{ij}\right), \tag{2}$$

where $i, j \in V$ and $i \neq j$.

2.1. Link Travel Time Function. Using BPR model the average travel time for a vehicle can be expressed as

$$t\left(E_{ij}\right) = t_0\left(1 + 0.15\left(\frac{q\left(E_{ij}\right)}{c\left(E_{ij}\right)}\right)^4\right), \tag{3}$$

where t_0 = free flow travel time on link E_{ij}, in minute; $q(E_{ij})$ = volume of traffic on link E_{ij} per hour; $c(E_{ij})$ = capacity of link E_{ij} per hour; $t(E_{ij})$ = the average travel time for a vehicle on link E_{ij}, in minute.

2.2. Delay Functions for Signalized Intersections. Over the past few decades, a number of delay formulas have been proposed to account for delay at signalized intersections. Among the better known of these delay formulas are Webster and HCM 2000.

As for nonsaturated intersections, Webster model can be used. Average delay at intersection can be expressed as

$$d\left(E_{ij}\right) = 0.9\left(\frac{T\left(1-\lambda^2\right)}{2\left(1-\lambda X\right)} + \frac{X^2}{2Q\left(1-X\right)}\right), \tag{4}$$

where $d(E_{ij})$ = average delay at intersection, in seconds; T = cycle length of the traffic light, in seconds; λ = green time of the traffic light/cycle length of the traffic light; Q = traffic volume on entering link, in vehicle per hour; X = intersection saturation degree.

This model can be applied under saturated ($X < 0.8$) conditions only.

As for saturated intersections ($X > 0.8$), average delay at intersection is estimated as

$$\begin{aligned} d\left(E_{ij}\right) &= d_1\left(E_{ij}\right) + d_2\left(E_{ij}\right), \\ d_1 &= 0.38T\frac{(1-\lambda)^2}{(1-\lambda X)}, \\ d_2 &= 173X^2\left((X-1) + \sqrt{(X-1)^2 + \frac{16X}{S}}\right), \end{aligned} \tag{5}$$

where $d(E_{ij})$ = average delay at intersection, in seconds; $d_1(E_{ij})$ = uniform delay at intersection, in seconds; $d_2(E_{ij})$ = oversaturation delay at intersection, in seconds; S = saturate volume of intersection, in vehicle per hour; X = intersection saturation degree.

Table 1: Characteristics of the IGA.

Encoding:	Path string, no duplicate individuals
Selection:	Roulette
Crossover:	Vicinity crossover
Mutation:	Greedy backward mutation
	Vicinity crossover
Improvements:	Greedy backward mutation
	Steady-state reproduction

3. The Improved Genetic Algorithm

By introducing the following new techniques, the performance of a simple genetic algorithm for dynamic route guidance algorithm has been improved essentially. Various characteristics of this proposed IGA are shown in Table 1.

3.1. Coding and Initial Population. We regard each route from the origin node to the destination node as a chromosome and express it as a sequence of nodes. The first gene in the chromosome is always the origin node, and the last gene in the chromosome is always the destination node. Since different paths may have different number of intermediate nodes, the chromosomes will be of variable length. However, the maximum length of a chromosome cannot exceed the total number of nodes in the network. Any repeated nodes in the chromosome signify that the path represented by the chromosome contains a loop and in network routing, any loop should be eliminated. At the beginning, the population is filled with chromosomes that represent random paths.

The algorithm to generate the random path is adapted from [10]. The algorithm goes as follows.

Step 1. Start from the origin node.

Step 2. Randomly choose, with equal probability, one of the nodes that are connected to the current node.

Step 3. If the chosen node has not been visited before, mark that node as the next node in the path. Otherwise, find another node.

Step 4. If all the neighboring nodes have been visited, go back to Step 1.

Step 5. Otherwise, repeat Step 2 by using the next node as the current node. Do this until the destination node is found.

3.2. Fitness Function. For the dynamic route guidance problem, the lower the total weight of the path, the better the solution. The fitness function is defined as follows:

$$f = \frac{1}{\sum_{E_{ij} \in P} T\left(E_{ij}\right)}. \tag{6}$$

3.3. Selection. In this GA, roulette wheel (Monte Carlo) method is adopted as the selection operator. Chromosomes with better fitness will have a higher probability to be selected.

3.4. Steady-State Reproduction. To avoid the highest fitness chromosome destroyed in the process of crossover and mutation, we adopt steady-state reproduction and replace the worst chromosome produced by the genetic operators with the best chromosome in the previous population.

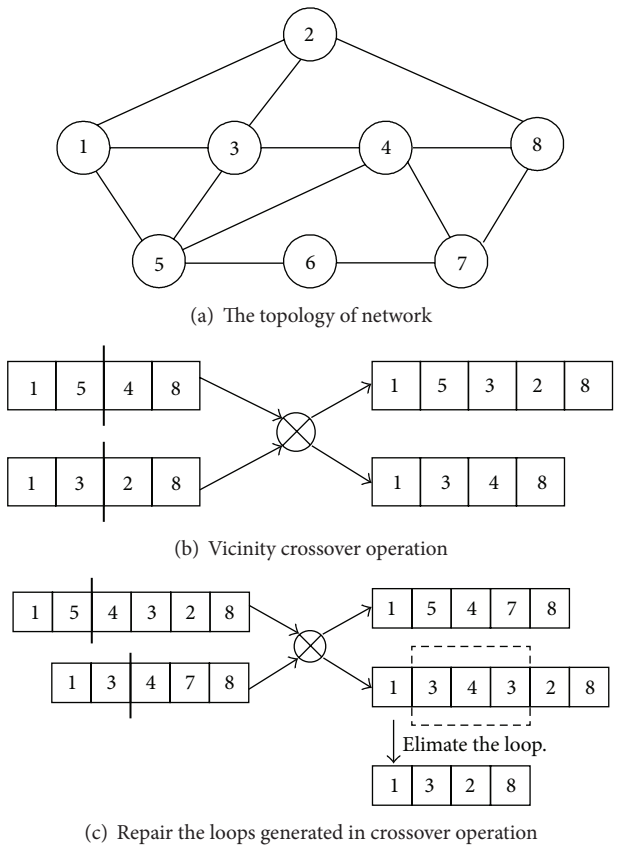
3.5. Vicinity Crossover Strategy. The common way of crossover is the two chromosomes selected must have at least one common node other than the origin and destination nodes. The common node is called the crossover point. Crossover operation will be carried out on crossover point. In this paper, we propose a new vicinity crossover strategy. The main idea is crossover operation will be carried out not on common point but on vicinity point. The vicinity point is the two adjacent points that the distance between them less than the given maximal vicinity value in the networks. The distance between vicinity points is vicinity value. To ensure that the paths generated by the crossover operation are still valid paths, the two chromosomes selected must have at least one vicinity node other than the origin and destination nodes. If more than one vicinity node exists, one of them will be randomly chosen with equal probability. For the sake of clearly describing the concept of vicinity crossover strategy, we consider the following example. The topology of network is shown in Figure 1(a). Assume origin node is 1, destination node is 8 and maximal vicinity value is 3.

Assume we have the following parent chromosomes.

Parent chromosome1 = [1 5 | 4 8].

Parent chromosome2 = [1 3 | 2 8].

where node 1 and node 8 are the origin node and destination node respectively. In this example, the vicinity nodes are node 5 of parent chromosome1 and node 2 of parent chromosome2, node 3 of parent chromosome2 and node 4 of parent chromosome1. Therefore, crossover operation will exchange the first portion of chromosome1 with the second portion of chromosome2 and vice versa. The vicinity value



(a) The topology of network

(b) Vicinity crossover operation

(c) Repair the loops generated in crossover operation

Figure 1: The process of vicinity crossover operation.

of node 5 and node 2 is 2. Hence $5 \rightarrow 3 \rightarrow 2$ is a generated vicinity path. The vicinity value of node 3 and node 4 is 1.

As a result, the following child chromosomes will be generated.

Child chromosome1: [1 5 3 2 8].

Child chromosome2: [1 3 4 8].

These two chromosomes would then become new members of the population as shown in Figure 1(b). It is possible that loops may occur after crossover operation is performed. Loops in a chromosome can be repaired by performing a search along the chromosome to find repeated nodes. The nodes in between the repeated nodes are then eliminated. For example, in Figure 1(c), assume that we have the following chromosome that contains a loop.

Chromosome with loop: [1 3 4 3 2 8].

In this case, there are two node 3 in the chromosome which signifies that the path contains a loop. This chromosome can be fixed by eliminating one of the node 3 and all the other nodes in between the two node 3's. The fixed chromosome would become like this.

Fixed chromosome: [1 3 2 8].

The fixed chromosome can be searched again just in case there are multiple loops in the chromosome. The vicinity crossover strategy will increase population diversity.

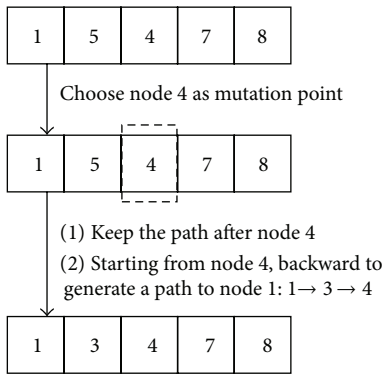


Figure 2: The process of greedy backward mutation.

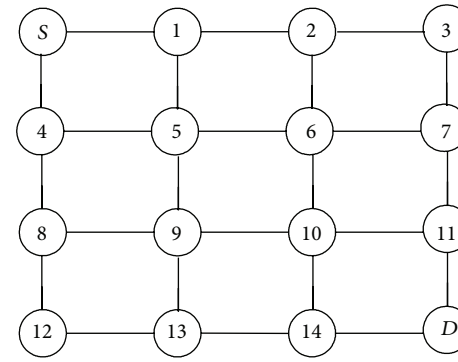


Figure 3: Virtual map of square matrix.

3.6. Greedy Backward Mutation Strategy. Instead of bitwise mutation, a new greedy backward mutation strategy is used. The main idea is the following. For each chromosome that is chosen to be mutated, if the correlation degree of intermediate nodes in the path from origin to destination (i.e., the origin and destination node cannot be chosen as the mutation point) is greater than 2, then the intermediate nodes will be chosen as candidate nodes. A mutation point will be chosen randomly, with equal probability, among the candidate nodes. Once the mutation point is chosen, the chromosome will be changed starting from the node before the mutation point and backwards. To explain this procedure, we still use the network as shown in Figure 1(a) and the procedure of greedy backward mutation is shown in Figure 2.

Assume that the following chromosome has been chosen to be mutated.

Original chromosome: [1 5 4 7 8].

Where 1 and 8 are the origin node and the destination node, respectively. Assume also that the node 4 has been chosen as the mutation point. The mutated chromosome would become like this.

Mutated chromosome: [1 x_1 $x_2 \cdots$ 4 7 8].

The mutated chromosome now contains a new backward path from 4 to 1, where x_i is the ith new node in the path. The new path is generated randomly; the same way as the paths in the initial population is generated.

4. Simulation Results

To test the effectiveness of the proposed IGA for dynamic route guidance, we implement the algorithm and conduct a series of simulation experiments. The algorithm has been coded in Matlab 2009 and implemented on an Intel Core 2, CPU 2.53 Ghz, RAM 2 GB, Windows XP System. Several main performance metrics are considered: number of generation, population size, convergence ability, and convergence speed.

We performed a set of experiments on a virtual square matrix map. The virtual maps of square matrix had sizes of 4×4, 8×8, 16×16, and 32×32. As shown in Figure 3, the origin was at the upper left corner and the destination was at the lower right corner. The distances between nodes were varied from 10 to 50.

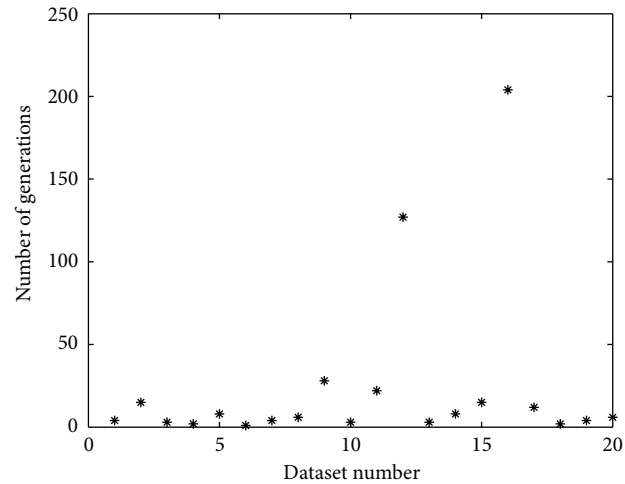


Figure 4: The number of generations required to find a feasible solution.

We generate 20 random datasets using the 4×4 virtual square matrix maps. Figure 4 shows the number of generations required to find a feasible solution. From the numerical results, we observe that all 20 datasets are able to converge to a feasible solution and the proposed IGA is correct. From the result, it can also be seen that most of time, the number of generations required to find a feasible solution is quite low and acceptable. However, in some cases, the number of generations required to find a feasible solution can exceed 200. The difference is quite large. The fact is probably caused by the state of initial population. The quality of initial population may be very low and takes many generations to find a feasible solution.

To solve the problem above, we can adjust the population size. Figure 5 shows the effect of population size on the average number of generations needed to find a solution. We execute the algorithm 10 times and record the average value. From Figure 5, we can conclude that the average number of

Table 2: Simulation parameters for IGA and SGA.

	IGA	SGA
Population size	60	60
Selection mode	Roulette	Roulette
	Vicinity crossover	One-point crossover
Crossover operator	Crossover rate $p_c = 0.9$	Crossover rate $p_c = 0.9$
	Maximal vicinity value = 3	
Mutation operator	Greedy backward mutation	One-point mutation
	Mutation rate $p_m = 0.01$	Mutation rate $p_m = 0.01$
Termination	500	500
Execution run	20	20

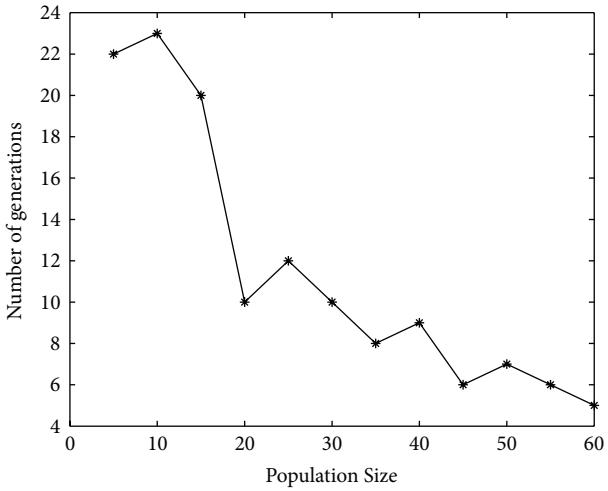


Figure 5: Average number of generations for different population size.

Table 3: Average generation to find optimal path.

Number of nodes	IGA	SGA
16	4.5	4.5
64	5.5	6.5
256	10	15.5
1024	18.5	39

generations become less with the increase of population size and is acceptable most of time.

We compare our improved genetic algorithm (IGA) with simple genetic algorithm (SGA) on virtual square matrix maps. Table 2 showed all of the simulation parameters for IGA and SGA.

Table 3 gives the average number of generational processes to find the optimal path. It shows that our IGA converges very fast, less than 20 generations even when the number of nodes growing to 1024. Also for a complex map with thousands of nodes, the average number of generational processes for the genetic algorithm beginning to converge is still small enough for real time applications.

Table 4 shows the experiment results of the minimal path weight and average path weight for our IGA and SGA for different network sizes. IGA achieves better minimum weight and average weight than SGA for all scenarios.

Table 4: The comparison of IGA and SGA.

Number of nodes	IGA Mini_weight	SGA Mini_weight	IGA Avg_weight	SGA Avg_weight
16	126	126	128.2	129.6
64	201	207	212.4	224.2
256	392	396	403.9	418.3
1024	784	791	795.5	824.3

5. Conclusion

This paper presented an improved genetic algorithm for dynamic route guidance algorithm. Several details of the genetic algorithm such as vicinity crossover scheme, greedy backward mutation scheme, and steady-state reproduction have been specifically designed for solving this problem while other details are adapted from previous works. The simulation results show that the proposed algorithm works well in finding an optimal path for real-time applications and converges much faster to better solutions. The performance gets better with larger population size.

Acknowledgments

This work supported by the National Nature Science Foundation of China (no. 70971136) and supported by the Youth Foundation of Institute of Sciences, PLA University of Science and Technology (no. KYLYZL001235).

References

[1] G. Dimitrakopoulos and P. Demestichas, "Intelligent transportation systems: systems based on cognitive networking principles and management functionality," *IEEE Vehicular Technology Magazine*, vol. 5, no. 1, pp. 77–84, 2010.

[2] I. Chabini and S. Lan, "Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 60–74, 2002.

[3] G. R. Jagadeesh, T. Srikanthan, and K. H. Quek, "Heuristic techniques for accelerating hierarchical routing on road networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 4, pp. 301–309, 2002.

[4] L. Fua, D. Sunb, and L. R. Rilettc, "Heuristic shortest path algorithms for transportation applications: state of the art," *Computers and Operations Research*, vol. 33, pp. 3324–3343, 2006.

[5] S. Yussof and O. H. See, "QoS routing for multiple additive QoS parameters using genetic algorithm," in *Proceedings of the 13th IEEE International Conference on Networks jointly held with the 7th IEEE Malaysia International Conference on Communications*, pp. 99–104, November 2005.

[6] H. R. Varia and S. L. Dhingra, "Dynamic optimal traffic assignment and signal time optimization using genetic algorithms," *Computer-Aided Civil and Infrastructure Engineering*, vol. 19, no. 4, pp. 260–273, 2004.

[7] N. Huimin, C. Mingming, and Z. Minghui, "Optimization theory and method of train operation scheme for urban rail transit," *China Railway Science*, vol. 32, pp. 128–133, 2011 (Chinese).

[8] L. Zou, J. Xu, and L. Zhu, "Application of genetic algorithm in dynamic route guidance system," *Journal of Transportation Systems Engineering and Information Technology*, vol. 7, no. 3, pp. 45–48, 2007 (Chinese).

[9] H. Kanoh, "Dynamic route planning for car navigation systems using virus genetic algorithms," *International Journal of Knowledge-Based and Intelligent Engineering Systems*, vol. 11, no. 1, pp. 65–78, 2007.

[10] C. W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 566–579, 2002.