World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:5, 2007

Dynamic Network Routing Method Based on Chromosome Learning

Xun Liang

***Abstract*—In this paper, we probe into the traffic assignment problem by the chromosome-learning-based path finding method in simulation, which is to model the driver' behavior in the with-in-a-day process. By simply making a combination and a change of the traffic route chromosomes, the driver at the intersection chooses his next route. The various crossover and mutation rules are proposed with extensive examples.

***Keywords*—Chromosome learning, crossover, mutation, traffic path finding

I. Introduction

The dynamic in this paper is a within-a-day process, not a day-to-day process [1-2]. For example, after a special event like earthquake, there are some lanes closed and people worriedly drive back to home. The driver listen to the traffic information over the radio, and the driver could also change their choices of roads. This forms a within-a-day transient process, typically not repetitive. The purpose of studying the with-in-a-day dynamics is to provide the scenarios so that the people who need it can make decision based on it. For example, after earthquake, the ambulance may try to reach the patients as soon as possible. This would put them into an advantageous situation that the driver foresee the traffic congest in advance and deploy their emergency vehicles to avoid the most severe congestion.

Clearly, the mathematical static user equilibrium (UE) method [3] cannot be used here, neither does the dynamic user equilibrium method, which solves the dynamic traffic assignment problem using calculation variations along the time horizon [4-5]. However, this method is too time-consuming. Up to now, it looks that the most applicable and acceptable solution is the simulation. There are already a lot of simulation tools in the past 20 years. Typical software includes ATIS, TRABSYT, INTRAS, TEXAS, TRANSIMS, PASSER, CONTRAM, SATURN, INTEGRATION, and DYNASMART. However, they only provide the day-to-day dynamics for the driver's behavior [2, 6-9].

Since the chromosome-learning-based (CL-based) path finding method employs the probability in choosing the routes, it is very similar to the stochastic user equilibrium (SUE) method. The differences are (1) in the CL the formation of probabilities is based on "the-shortest-time-path produces more children than other paths", while the SUE uses the probability based on a logit model; (2) the CL uses simple arc exchanges among shortest paths (not exhausting all the possible routes) by the idea of route chromosome crossover, hence saving simulation time, while the SUE does not incorporate the short path information in determining probabilities.

The CL has 3 operations: reproduction, crossover and mutation [10]. In the reproduction, the path with shortest time gets the most proportion in the next generation (i.e., more the driver use the path). This means, if at the current time a driver takes that path, the driver is expected to use the least time among the others. The reproduction operation will permit and encourage more the driver in the next generation to go along that path. But it is not necessary that all of the driver in next generation will take the best path since (1) the reproduction operation will produce every kind of descendents based on their parent proportion, (2) the result by the reproduction operation will experience crossover and mutation before the driver are put in use. In the mutation, some drivers may take a complete a new arc with a very small probability.

II. Within-a-Day Dynamic Traffic Path Finding Model

Suppose the traffic network is given. Each length of the arc $p{\sim}q$ is also known as $l^{p\sim q}$. Vehicle birth and death is given by the Origin-Destination (O-D) pairs. Vehicle initial path assignment is given by the shortest time path method [11]. Vehicle location (count from the end of each arc) is $x_i^{p\sim q}(t) = \max\left\{ x_i^{p\sim q}(t) - v_i^{p\sim q}(t)\Delta t, 0 \right\}$ where $x_i^{p\sim q}(t)$ is the location of the vehicle i at time t counted from the end of the arc $p{\sim}q$ the vehicle i is in, $v_i^{p\sim q}(t)$ is the speed of the vehicle i at time t in arc $p{\sim}q$. Vehicle velocity is given by

$$\frac{v_i^{p\sim q}(t) - v_{(\mathrm{A}),\,\mathrm{Min}}^{p\sim q}}{v_{(\mathrm{A}),\,\mathrm{Free}}^{p\sim q} - v_{(\mathrm{A}),\,\mathrm{Min}}^{p\sim q}} = \left(1 - \frac{k_{(\mathrm{A})}^{p\sim q}}{k_{(\mathrm{A}),\,\mathrm{Jam}}^{p\sim q}} \right)^{\alpha}$$

where $v_{(\mathrm{A}),\,\mathrm{Free}}^{p\sim q}$ is the aggregated speed for the free traffic flow, $v_{(\mathrm{A}),\,\mathrm{Free}}^{p\sim q}$ is the aggregated minimum speed, k is the density, is a user-defined parameter. Number of vehicles in arc

Manuscript received December 10, 2003. The project was sponsored by the Scientific Research Foundation for ROCS, SEM.

Xun Liang is with the Institute of Computer Science and Technology, Peking University, Beijing 100871, China, and the Department of Management Science and Engineering, Stanford University, CA 95035, USA (phone: +86-10-8252 9988-5228; fax: +86-10-82529207; e-mail: liangxun@icst.pku.edu.cn).

p-q at time *t* is $n_{(\mathrm{A})}^{p\sim q}(t)=\min\left\{\frac{l^{p\sim q}}{l_{\mathrm{vehicle}}},\ n_{(\mathrm{A}),\mathrm{NewlyGen}}^{p\sim q}(t),\ \sum_{q'\in N} n_{(\mathrm{A}),\mathrm{In}}^{q'\sim p}(t)\delta^{q'\sim p}(t)\right\}$ where $l^{p\sim q}$ is the length of the arc $p\sim q$, l_{vehicle} is the length occupied by an average vehicle, $n_{(\mathrm{A}),\,\mathrm{NewlyGen}}^{p\sim q}(t)$ is the number of the vehicles newly generated in arc $p\sim q$ at time *t*, $n_{(\mathrm{A}),\,\mathrm{In}}^{q'\sim p}(t)$ is the number of the vehicles newly entered arc $p\sim q$ from other arc $q'\sim p$ at time *t*, and $\delta^{q'\sim p}=1$ is *q'* is directly connected with *p*, or 0 otherwise. Suppose at starting point, we have *n* vehicles heading to both directions. The *n* vehicles start their journey one by one. The starting time interval is *k* minute(s). Our simulation time length is *l* minutes even some of the vehicle may not arrive at their destinations. After a driver approaches an intersection, the driver examines the travel time for each past ancestor's route. Tending to choose the shortest time route with the biggest probability [12], the driver does not refrain himself from a combination of the best shortest time routes (crossover), neither from a suddenly invention of taking a new arc (mutation).

III. Dynamic Traffic Network Routing Based on Chromosome Learning

We use the California South Bay Area as an example (see Figure 1).

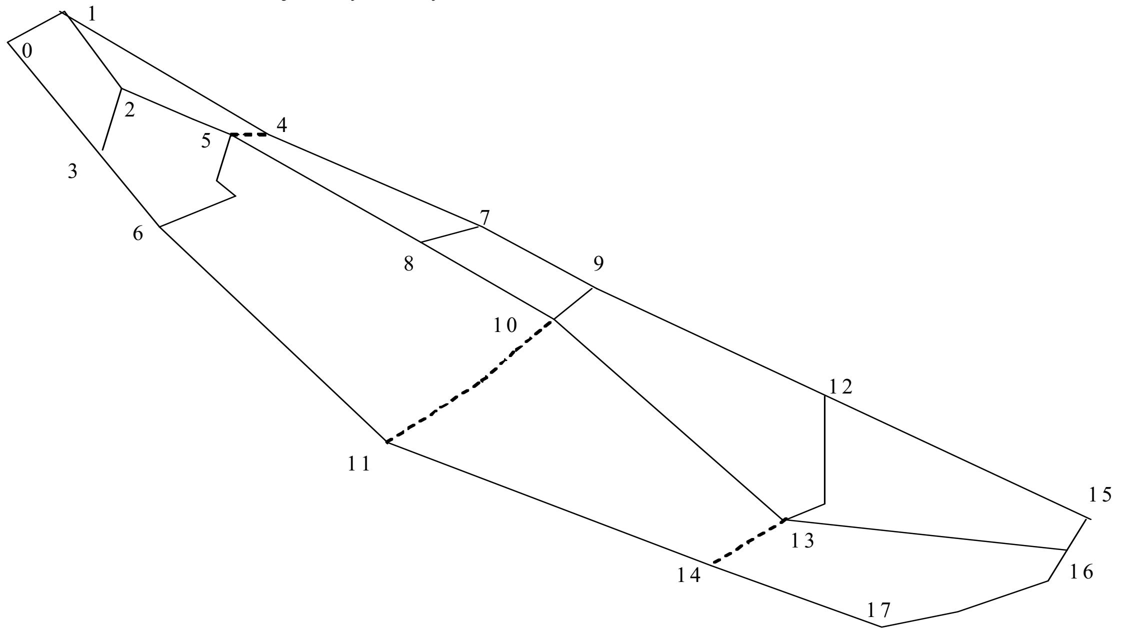

Figure 1. The California South Bay Area traffic network. After a special event, half of the lanes are closed in dashed arcs.

Gene set *G* contains all the arcs. A gene is an arc. Since this is a directed arc (both sides have different traffic flow, thus leading to different travel time on that arc), we may just mention the ending node of the gene when referring this gene. This is to say, you may think both arc *p~q* and node *q* are the same gene if your the driver in simulation is driving from *p* to *q*; or you may think both arc *p~q* and node *p* are the same gene if your the driver in simulation is driving from *q* to *p*. Throughout this paper, we use both equivalent notations (see Table 1).

Table 1. The number of vehicles in arc *p~q* at time *t*, and travel time. The time formula is $T^{p\sim q}(N(p, q, t))$.

p	*q*	# of vehicles in *p~q* in + direction	# of vehicles in *p~q* in – direction	Time, + direction	Time, - direction
0	1	130	151	4.9	5.0
0	3	120	200	4.8	5.3
1	2	100	150	4.6	5.0
1	4	201	312	5.3	5.7
2	5	120	152	4.8	5.0
2	3	67	83	4.2	4.4
3	6	128	38	4.9	3.6
4	5	20	34	3.0	3.5
4	7	179	200	5.2	5.3
5	6	124	198	4.8	5.3
5	8	256	283	5.5	5.6
6	11	302	298	5.7	5.7
7	8	35	63	3.6	4.1
7	9	120	230	4.8	5.4
8	10	300	384	5.7	6.0
9	10	120	321	4.8	5.8
9	12	87	41	4.5	3.7
10	11	139	210	4.9	5.3
10	13	325	279	5.8	5.6
11	14	421	394	6.0	6.0
12	13	201	176	5.3	5.2

12	15	235	203	5.5	5.3
13	14	104	96	4.6	4.6
13	16	236	271	5.5	5.6
14	17	230	202	5.4	5.3
15	16	67	72	4.2	4.3
16	17	205	238	5.3	5.5

Route pools Z^{rs} ($r\in R$; $s\in S$) is classified by origin-destination pair (r, s), is composed of paths $k_{i^{rs}}$ = {ordered arc $p\sim q$ series} for driver i^{rs}, $i^{rs}\in\{1, 2, \ldots, \#\text{ of driver with } (r, s)\}$. Obviously, any arc $p\sim q$ in $k_{i^{rs}}$ must be in G, i.e., $p\sim q\in G$.

Throughout the process, the number of members in route population Z^{rs} is always equal to the # of the driver with (r, s), including those who are on the road (living chromosomes) and those who are out of the road (died chromosomes), but not including those who will start their journey from r towards s (unborn chromosomes). That is, there is a one-to-one corresponding between the current and past the driver with (r, s) and the members of route population Z^{rs} at any time t. Each driver has his own (different or same) chromosome or string in Z^{rs}.

The obvious relationship between the gene set and route population is that the members in route population Z^{rs} ($r\in R$, $s\in S$) must be composed by the connectable elements in the gene set. For example, a driver i starts from the arc 15~12, to his destination 6~3. The driver may take the route {15~12, 12~9, 9~7, 7~4, 4~5, 5~6, 6~3}. This string is a chromosome or a member in route population Z^{rs}. Obviously, the chromosomes thus defined in the route population Z^{rs} may have different lengths of genes.

Chromosomes are produced in the birth process of new the driver. When a new the driver is about to start his journey, the driver is given a shortest route based on solving the current UE. The fittest produces the most. Gene crossover is performed at the sites located at or after the current intersection node, and performed at all time t, but only to the driver at an intersection. Thus, it improves the simulation speed. Note that the size of the population is always increasing. Descendants take all the advantages of their precedents. Mutation is performed on a gene-by-gene basis. We may assume the probability of mutation is p_m (a user-defined probability). For each gene, if it should be mutated, all the genes including and after this gene should be removed and a randomly chosen (connectable) chromosome from G is attached to the cut place of the mutating chromosome.

IV. Gene Operators

A. *Rules of Crossover*

Conservation rule. The past arcs are the same between two chromosomes — then simply match the past portion of chromosomes, then randomly choose any (sub)strings of the remaining ones to crossover.

1-4-7-9-12-13-14
1-4-7-8-10-11-14
a b

By two simple loop node matching searching, the program can find that there are two sites that can crossover: site a with node 4, site b with node 7. By a (biased-)coin flipping, the driver may choose one route to continue his journey (exhibiting as go to the first arc of that route).

Crossover with different lengths is like

1-4-5-6-11-14
1-2-5-4-7-8-10-9-12-13-14

Randomly choose any (sub)strings of the remaining ones to crossover as long as nodes are matched. (Note that crossover does not mean that only the old chromosomes exchange their genes. The driver do may result in completely new routes.)

Regret rule. Suppose that the driver started from 1 and took arc 1~4 and route

1-4-5-6-11-14

at that time (since the first arc is determined by "shortest time" — a universal rule to the driver). After the driver approach intersection 4, the driver deletes all the connections with node 1 since it is finished (we can see that by deleting all the row containing node 1, we eliminate the possibility that the driver comes back to a node which the driver has used).

There is another chromosome chosen randomly from its "brothers" and "parents" to be crossovered.

1-2-5-4-7-8-10-9-12-13-14

Suppose the driver is already at intersection 4. By randomly choosing crossover sites 6 in the first chromosome and 10 in the second, we obtain *one* temporary string (not necessary a chromosome if it is not disconnect). The process is like this:

(original route) 4-5-6-11-14
4-7-8-10-9-12-13-14
(crossovered string) 4-7-8-10-11 – 14

Thus we obtain a new route for this the driver, and the simulation program puts him on his new way until the next intersection.

Mostly, the crossover string is not a connectable route. For example, if we choose 7 and 6 to crossover.

(original route) 4-5-6-11-14
4-7-8-10-9-12-13-14
(crossovered string) 4-7-11-14.

This is not feasible between 7 and 11.

The regret rule says that the driver will want to connect back to his original route as soon as possible at the crossover point (i.e., disconnected point). Thus, the driver builds circles alternatively with the centers at the two disconnected points 7 and 11, with radii 1 Hamming distance (that is, one arc away from 7 or 11), starting from the current route to connecting his original route. In our instance, write $C_{7,0} = \{7\}$, $C_{11,0} = \{11\}$. We know $C_{7,0} \cap C_{11,0} = \phi$. This is the starting point of the regret rule.

(1) The driver starts with 7, builds a circle $C_{7,1} = \{4, 8, 9\}$ with center 7, deleting 4 and have $C_{7,1} = \{8, 9\}$.

(2) Since $C_{7,1} \cap C_{11,0} = \phi$, the driver builds the circle $C_{11,1} = \{6, 10\}$ with the center 11.

(3) Since $C_{7,1} \cap C_{11,1} = \phi$, we continue to build the circle

$C_{7,2}$ = {1, 5, 5, 10, 10, 12}, deleting one 1, one 5, one 10 and have $C_{7,2}$ = {5, 10, 12}.

(4) Since $C_{7,2} \cap C_{11,1}$ = { 10 }, the driver finds a connecting way by retracing the bridge of 10 to both sides:

$C_{11,1}$ = {6, 10} $\rightarrow C_{11,0}$ = {11}

$C_{7,2}$ = {5, 10, 12} $\rightarrow C_{7,1}$ = {8, 9} $\rightarrow C_{7,0}$ = {7}.

There is no difference from the pure network point of view. However, to hold a direct reasonability, the driver compares the travel time 3.6+5.7=9.3 and 4.8+4.8=9.6 between 7-8-10 and 7-9-10, and decides to use 7-8-10 for connection. In summary, his new route is

(fixed crossover string) 4-7-8-10-11-14 (a new chromosome)

Nearsighted meticulous calculation rule. Suppose that a meticulously the driver meets the following case after crossover operation, 4-7-11-14.

The nearsighted meticulous calculation rule says that this the driver will calculate the time the driver will use to connect his crossovered route and the original route.

We also use C as our circles but this time C represents travel time. The Hamming distance will not be calculated. Instead, concrete travel time on that arc is used. In our instance, we write $C_{7,0}$ = {7(0)}, $C_{11,0}$ = {11(0)}. 0 means the driver is there. We know $C_{7,0} \cap C_{11,0} = \phi$. Then

(1) The driver starts with 7, builds a circle $C_{7,1}$ = {4, 8, 9} with center 7, deletes 4 and has $C_{7,1}$ = {8(3.6), 9(4.8)}.

(2) Since $C_{7,1} \cap C_{11,0} = \phi$, the driver builds the circle $C_{11,1}$ = {6(5.7), 10(4.9) } with the center 11.

(3) Since $C_{7,1} \cap C_{11,1} = \phi$, we continue to build circle $C_{7,2}$ = {1, 5, 5, 10, 10, 12}, deleting one 1, one 5, one 10 and have $C_{7,2}$ = {5(9.2), 10(9.3), 12(8.5)}, where 9.2 = 8(3.6)+5(5.6), 9.3 = min {8(3.6)+10(5.7), 9(4.8)+10(4.8)}, and delete the memory of arc connection 7-9-10 (that is, only record the shortest time path, and forget the useless path before finding a bridge), 8.5 = 9(4.8)+12(3.7).

(4) Since $C_{7,2} \cap C_{11,1}$ = {10}, the driver finds a connecting bridge 10 and by retracing this bridge to both sides:

$C_{11,1}$ = {6(5.7), 10(4.9)} $\rightarrow C_{11,0}$ = {11}

$C_{7,2}$ = {5(9.2), 10(9.3), 12(8.5)} $\rightarrow C_{7,1}$ = {8(gives 9.3), 9(gives larger number)} $\rightarrow C_{7,0}$ = {7}.

The driver finds a new route (fixed crossover string) 4-7-8-10-11-14 (a new chromosome).

The only difference between the nearsighted meticulous calculation rule and the regret rule is that the nearsighted meticulous calculation rule has to continue since the above route may not be the best one in time. This is to accomplish by further expand the radii of circles until the Hamming distance reaches min{H^*, H_1} where H^* is a user-defined number, H_1 is the shortest Hamming distance between to two disconnected points 7 and 11.

(5) Further expanding the radius of 11, the driver finds 5, 8, 9, and 13. Then the driver calculates the respective time to arrive those points. Again, further expanding the radius of 7, the driver finds 11. Then the driver terminates his searching.

Since in the operation of one generation, we do not have to mend the chromosome immediately after crossover since the mutation is the next step. We would normally first let it go through mutation.

If it is not chosen, we go back and mend the disconnected part using the above rules. If it is chosen but the mutation makes another gene disconnected, we also go back and mend the disconnected part using the above rules. Otherwise, we wait it mutated first and then repair the genes.

B. Rules of Mutation

Mutation of a route is with a small probability. Mathematically, if a node p is chosen to mutate, we extract a subset M_p from the gene set with one end of p. Suppose q is its leading node. Then our target set to choose for mutation is $M_p \backslash \{q\text{-}p\}$. With equal probability, we choose one of the nodes in set $M_p \backslash \{q\text{-}p\}$. After mutation, the following rules are for making them feasible.

Stubbornness rule. If the probability chooses one gene to mutate, it has to choose at least one more gene that is neighboring it in the other end of the new gene. By searching in set G, we may find a new series of genes (subchromosome) that leads the driver back to his route or to his destination.

When searching this subchromosome, always choose the gene, which is able to immediately go back to the original route if this gene exists. This can be done by simply search G. For example, a driver is driving on the route 16-13-14-11-6. At intersection 13, a mutation operation changes the gene of 14 to the gene of 10. We start search our gene set G. We find that there are 4 genes connecting to node 10: 10~13, 10~9, 10~8, 10~11. It is not difficult for the computer itself to recognize that the 10~13 is his past arc. Among the other 3 arcs, 10~11 leads him immediately back to his original route. Hence we use 10~11 and our mutation operation ends with two new genes of 13~10 and 10~11.

From the connection point of view, a mutation fixation is a projection from one point to the discrete set. Although this projection does not demand the perpendicularity, it does demand the shortest length. For example, suppose a the driver is at intersection 4, by mutation the driver is driven to node 7. Now, the driver wants to project himself back to his original route with the shortest time (excluding his coming arc). We formulate it as follows.

$$\text{Point: } 7 \longrightarrow \text{Projected set: } \{5, 6, 11, 14\}$$

This time, the driver only uses radius method on one side that is node 7. Then the driver finds

7-8-5: 3.6 + 5.6 = 9.2

7-8-10-11: 3.6 + 5.7 + 4.9 > 9.2 (note that we already know 7-8-10 < 7-9-10, we omit 7-9-10 here.)

7-8-10-13-14: 3.6 + 5.7 + 5.8 + (·) > 9.2

Hence the driver takes the route 7-8-5 and gets back to his original route.

Global eyesight rule. Sometimes the stubbornness rule is not intelligent enough since there exist better new routes leading the driver to his destination. The global eyesight rule says that when a the driver is driven by mutation to a new point not on his original route, the driver joins in the group whose origin is the point the driver is on. If this group does not exist, the driver himself forms a group. That is, the driver has to use the UE method and starts his journey there as if newly starting. In the above example, his new origin is node 7, and his destination is still node 14.

V. A Complete Example

Say we have a population of routes, which starts from the same origin 1 and ends with the same destination 14. The population consists of 5 chromosomes

1-4-7-9-12-13-14
1-3-6-11-14
1-2-5-8-7-9-10-13-14
1-4-7-8-10-11-14
1-2-5-4-7-8-10-9-12-13-14

Step 1. Start with 1. Find the route through where the shortest time is expected. Say it is 1-4-5-6-11-14.

Step 2. When the driver approach node 4, the driver must judge the route to continue.

1-4-5-6-11-14
1-4-7-9-12-13-14
1-3-6-11-14
1-2-5-8-7-9-10-13-14
1-4-7-8-10-11-14
1-2-5-4-7-8-10-9-12-13-14

Since the crossover occurs only at or after node 4, only 4 of 6 chromosomes are feasible (in which the host chromosome must be included). Then we have

1-4-5-6-11-14 (host chromosome)
1-4-7-9-12-13-14
1-4-7-8-10-11-14
1-2-5-4-7-8-10-9-12-13-14

Crossover occurs at the places between 4 and 14. There are many combinations, but not all of them are feasible. Actually, only a small portion of them is feasible. The crossover rules can make the route feasible.

Suppose the host chromosome chooses the chromosome 1-4-7-8-10-11-14 to crossover, and crossover sites are 6 to the host chromosome and 10 to the spouse chromosome. The next generation of the host chromosome is 1-4-5-6~11-14 (host chromosome), where ~ denotes the place which needs examination. In the meantime, we put {1-4-5-6} into the population pool $Z^{1,6}$. Since 11-14 is a simple arc, we don't need to have a pool $Z^{11,14}$ and put it into it. Also we put the handicapped chromosome 1-4-5-6~11-14 into pool $Z^{1,14}$.

Step 3. Mutation. Suppose the host chromosome meets a mutation signal, which asks it to mutate 11 to 10. By Stubbornness Rule, it projects 10 onto "discrete one-dimensional space" 1-4-5-6~11-14 (host chromosome) and find the shortest path to get to this space is through 10-13-14. Hence, we have 1-4-5-6 ~ 10-13-14 (host chromosome).

Step 4. Keeping all the repairing rules in mind, we examine the handicapped part 6~10. By Regret Rule, we connect 6~10 by 6-11-10. Therefore, we finally finish the operation of this generation by the following chromosome (mutation should be kept to a small level), 1-4-5-6-11-10-13-14 (host chromosome).

VI. Simulation

We write the above operations by C++. After a special event, half of the lanes on roads 4-5, 10-11 and 13-14 are closed. W simulate 2 hours for the transient process, and find the total cars leaving the origins are among them, 928 are still in the half-way back to their home, and 1983 cars arrives to home. Examples are given in Table 2.

Table 2. Examples of route changes.

Origin	Destination	Route (*t*)	Route (*t*+1)	Route (*t*+2)
6	7	6-5-4-7	6-5-8-7	
6	12	6-5-4-7-9-12	6-5-8-7-9-12	
6	16	6-5-4-7-9-12-15-16	6-5-8-7-9-12-15-16	
6	17	6-11-14-17		
9	1	9-7-4-1		
9	3	9-7-4-5-2-3	9-7-8-5-2-3	
9	8	9-7-8		
9	13	9-12-13		
9	14	9-10-11-14	9-10-13-14	9-10-13-16-17-14
9	16	9-12-15-16		
9	17	9-10-11-14-17	9-10-13-14-17	

From the examples, we can see that the drivers keep changing the routes according to the road information they get on their way back home.

VII. Concluding Remarks

The traffic information is already summarized inexplicitly as a sort of condensed information in chromosome and its reproduction. As a result, the chromosome learning employs the probability incorporated with all past route traffic inexplicitly. Its probabilities are simply based on the past information its precedents provide.

References

[1] T. Y. Hu & H. S. Mahmassani, Evolution of network flows under real-time information: day-to-day dynamic simulation assignment framework, *Transportation Research Record*, 1990, 1493, 46-56.

[2] M. L. Hazelton, Day-to-day variation in Markovian traffic assignment models, *Transportation Research*, 2002, 36B, 637-648.

[3] Y. Sheffi, *Urban transportation networks: equilibrium analysis with mathematical programming methods*, New Jersey: Prentice-Hall, 1985.

[4] H. N. Koutsopoulos, A. Polydoropoulou & M. Ben-Akiva, Travel simulators for data collection on the driver behavior in the presence of information, *Transportation Research C*, 1995, 3, 143-159.

[5] D. E. Glodberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, New York: Addison-Wesley, 1989.

[6] V. Astarita, Node and link models for network traffic flow simulation, *Mathematical and Computer Modelling*, 2002, 35, 643-656.

[7] D. Boyce & D. H. Lee, B. Ran, Analytical models of the dynamic traffic assignment problem, *Networks and Spatial Economics*, 2001, 1, 377-390.

[8] M. Carey, & E. Subrahmanian, An approach to modelling time-varying flows on congested networks, *Transportation Research*, 2000, 34B, 157-183.

[9] A. Faghri, R. Nanda, & K, Hamad, Development of a dynamic traffic simulation model in a near system optimal route guidance system, *Civil Engineering and Environmental Systems*, 2002, 19, 141-167.

[10] S. Nakayama, & R. Kitamura, Route choice model with inductive learning, *Transportation Research Record*, 2000, 1725, 63-70.
[11] S. Nakayama, R. Kitamura, & S. Fujii, Drivers' route choice rules and network behavior - Do drivers become rational and homogeneous through learning, *Transportation Research Record*, 2001, 1752, 62-68.
[12] M. Rickert, & K Nagel, Dynamic traffic assignment on parallel computers in TRANSIMS, *Future Generation Computer Systems*, 2001 17, 637-648.

APPENDIX

The model is as follows:

$$\min_x \sum_a \int_0^{x_a} t_a(\omega)\,d\omega$$

s.t.

$$\sum_k f_k^{rs} = q^{rs}, \quad \forall r,s$$

$$f_k^{rs} \geq 0, \quad \forall k,r,s$$

$$x_a = \sum_r \sum_s \sum_k f_k^{rs}\delta_{a,k}^{rs}, \quad \forall a$$

where

R	Set of origin nodes; $R \subseteq N$
S	Set of destination nodes; $S \subseteq N$
K^{rs}	Set of paths connecting origin-destination (O-D) pair r-s, $r\in R$, $s\in S$
x_a	Flow on arc a; $x=(\ldots, x_a, \ldots)$
t_a	Travel time on arc a; $t=(\ldots, t_a, \ldots)$
f_k^{rs}	Flow on path k connecting origin-destination pair r-s, $r\in R$, $s\in S$; $c^{rs}=(\ldots, c_k^{rs}, \ldots)$; $c=(\ldots, c^{rs}, \ldots)^\tau$ where $(.)^\tau$ is the transpose of (.).
c_k^{rs}	Travel time on path k connecting origin-destination pair r-s, $r\in R$, $s\in S$; $c^{rs}=(\ldots, c_k^{rs}, \ldots)$; $c=(\ldots, c^{rs}, \ldots)^\tau$ where $(.)^\tau$ is the transpose of (.).
q^{rs}	Trip rate between O-D pair r-s, $r\in R$, $s\in S$; $(\boldsymbol{q})^{rs} = q^{rs}$
$\delta_{a,k}^{rs}$	Indicator variable: $\delta_{a,k}^{rs} = \begin{cases} 1, & \text{if link } a \text{ is on path } k \text{ between O-D pair } r\text{-}s \\ 0, & \text{otherwise} \end{cases}$

Steps of minimizing process:

(0) Initialization. Set counter n=0.

(1) Perform calculation of all-or-nothing assignment based on $t_a^{(n)} = t_a^{(n)}(x_a^{(n)}), \forall a$. This requires solving the sub-problem for each r-s pair.

$$\min_x \sum_a c_k^{rs} g_k^{rs}$$

s.t.

$$\sum_k g_k^{rs} = q^{rs}, \quad \forall r,s \quad g_k^{rs} \geq 0, \quad \forall k,r,s$$

The solving of this programming is equivalent to finding a direction $y^{(n)} - x^{(n)}$ where vector $x^{(n)}$ is composed of $x_a^{(n)}$'s and vector $y^{(n)}$ is composed of $y_a^{(n)}$'s, which is implemented by performing all-or-nothing assignment based on $\{t_a^{(n)}\}$. Hence, a set of auxiliary flow $\{y_a^{(n)}\}$ is introduced, $y_a^{(n)} = \sum_r \sum_s \sum_k g_k^{rs}\delta_{a,k}^{rs}, \quad \forall a$. The direction finding is accomplished by the shortest path method. Steps are

1) Initialize every node by label T where T represents the calculation for this node has not been finished. Initialize every node by index (-1) in I, where I records the index for the shortest coming-way connecting node.
2) Start with the origin node r. Let $P(r)$=0, L=L-{0}. Let v_i =r.
3) Consider the nodes v_j with label T and also connecting to a node v_i with label P.
4) Assign nodes v_j with label $T(v_j)$ = min { $T(v_j)$, $P(v_i)$+t_{ij} }. Assign the node with the smallest $T(v_j)$ among j's as $P(v_j)$= $T(v_j)$, where label P means that its shortest length for node v_j has been determined. $I(v_j)$={i: $P(v_j)$ = min { $T(v_j)$+ t_{ij} }, $i\in S$-L } where S is the index set for all nodes, and L is index set for the nodes with label T. L=L-{i}
5) If L=ϕ, stop. Otherwise, go back to step 3).

(2) Line search. Find $\alpha^{(n)}$ that solves

$$\min_{0\leq\alpha\leq 1} \sum_a \int_0^{x_a^{(n)}+\alpha(y_a^{(n)}-x_a^{(n)})} t_a(\omega)\,d\omega$$

Differentiating it w.r.t. to α leads to:

$$\sum_a (y_a^{(n)} - x_a^{(n)})t_a(x_a^{(n)} + \alpha(y_a^{(n)} - x_a^{(n)})) = 0$$

Hence, $\alpha^{(n)}$=α can be obtained by solving it.

(3) Set $x_a^{(n+1)} = x_a^{(n)} + \alpha^{(n)}(y_a^{(n)} - x_a^{(n)}), \quad \forall a$.

(4) Calculate $t_a^{(n+1)} = t_a(x_a^{(n)}), \forall a$.

Xun Liang (M'04) received his BS and PhD degrees in computer engineering from Tsinghua University, Beijing, China, in 1989 and 1993 respectively, and an MBA from Stanford University, California, USA, in 1999.

He was a PostDoc research fellow, in Peking University, from 1993 to 1995, and in the University of New Brunswick, from 1995 to 1997, respectively. He worked as a Senior Software Engineer, System Architect respectively, from 1997 to 2003 in the high-tech IT corporations in Silicon Valley. He is currently an Associate Professor at Peking University, Beijing 100871, China. He has published over 60 papers and two books *eFinance - Theory and Applications*, *Web Financial Information Mining*. His research interests include internet-based financial information systems, data mining, neural networks and genetic algorithms.

Dr. Liang is a member of International Neural Networks Society, IEEE Neural Networks Society, and IJIT.