Journal of Intelligent & Fuzzy Systems 33 (2017) 3095–3102
DOI:10.3233/JIFS-169361
IOS Press

A novel model and its algorithms for the shortest path problem of dynamic weight-varying networks in Intelligent Transportation Systems

Zhong-Zhong Jiang[a,b,*], Yi-Ru Jiao[a], Ying Sheng[c] and Xiaohong Chen[d,e]
[a]*School of Administration Business, Northeastern University, Shenyang, China*
[b]*Institute of Behavioral and Service Operations Management, Northeastern University, Shenyang, China*
[c]*Deparment of Mathematics, College of Sciences, Northeastern University, Shenyang, China*
[d]*School of Business, Central South University, Changsha, China*
[e]*Key Laboratory of Hunan Province for Mobile Business Intelligence, Hunan University of Commerce, Changsha, China*

Abstract. Intelligent Transportation Systems (ITS) are defined as those systems utilizing synergistic technologies and systems engineering concepts to develop and improve transportation systems. In this paper, a novel route selection problem based on the envisaged driving mode with dynamic signals in ITS is proposed. It belongs to a kind of the shortest path problem of dynamic weight-varying networks, and the arc-weights of the network vary with the arc-chosen, so it cannot be solved by the existing greedy algorithms. According to the characteristics of the proposed problem, firstly, a dynamic programming model for the driving mode on a single path is established. Secondly, two algorithms for solving the route selection problem based on the former mode are developed. One is a brute-force algorithm based on matrix expansion with the computational complexity of $O\ (Nt \times n^2)$. The other is an improved adaptive ant colony algorithm with the complexity of $O\ (Nc \times m \times n^2)$. Finally, the computational experiments show the advantages and disadvantages of the two effective algorithms with numerical examples.

Keywords: Intelligent Transportation Systems, shortest path problem, dynamic weight-varying networks, brute-force algorithm, ant colony algorithm

1. Introduction

Traffic lights, also known as traffic control signals, are signaling devices positioned at road intersections to control flows of traffic [1]. It is designed to balance growth of vehicles and road capacity [2]. However, frequent stop and waiting not only intensifies the consumption of vehicles, but also weakens the mobility of traffic flows. Mobility is vital in modern societies. One of the ways to alleviate congestion and to utilize the existing infrastructure more efficiently is the route guidance system [3], which is the core of Intelligent Transportation Systems (ITS) and can influence travelers' behaviors, reduce travelers' anxiety to unknown traffic conditions, and reasonably distribute flows on the whole traffic network [4].

In recent years, Big Data has drawn huge attention in numerous fields [5–7]. In transportation, proper locations of sensors on a traffic network and data fusion techniques are collecting more useful data for controlling [8, 9]. A variety of inventions emerge

*Corresponding author. Zhong-Zhong Jiang, School of Administration Business, and Institute of Behavioral and Service Operations Management, Northeastern University, Shenyang 110169, China. E-mail: zzjiang@mail.neu.edu.cn.

Table 1
Problem classification

Information quality	Deterministic input	Stochastic input
Information evolution		
Input known beforehand	Static & deterministic	Static & stochastic
Input changes over time	Dynamic & deterministic	Dynamic & stochastic

one after another, such as novel localization algorithms [10], Smart-Eye for ITS [11], new cloud-based service framework [12], multilayered vehicular data cloud platform [13]. The reform of traffic data collection and processing methods is leading a deep change of ITS, for example, dynamic signals are more conveniently obtained in ITS.

Based on the above mentioned development, this paper is motivated by a driving mode with dynamic signals. In this driving mode, navigation devices obtain positions and speeds of vehicles, locations and states of the traffic lights, whereas the route guidance system guides drivers to select specific driving choices, in order to avoid waiting at red lights and run without any break. In this paper, we mainly focus on how to select an optimal route in the dynamic networks. A novel problem is proposed, which aims to find the shortest path in a digraph whose arc weights change with segments-chosen.

The remainder of this paper is organized as follows. Section 2 reviews different shortest path problems in ITS. Section 3 presents a dynamic programming model for the driving mode applied on a single path. In Section 4, two algorithms to solve route selection problem are developed. Then Section 5 conducts computational experiments to test the two algorithms. Finally, the conclusion is given in Section 6.

2. Route selection problems in ITS

In graph theory, there existing many algorithms for the general shortest path problem. While in ITS, changeful networks and real-time solution make the problem much more difficult [14]. In order to study more clearly, Pillic et al. classified vehicle routing problems by information evolution and quality as Table 1 [15]. It can be also applied to the shortest path problem.

Static and deterministic shortest path problem belongs to the kind of classical problems. Zhan et al. ever studied different shortest path algorithms in real traffic network in 1998 [16, 17]. Their research shows that though Dijkstra algorithm had been proposed for 39 years, it remained one of the fastest algorithms on one-to-all and one-to-one shortest path problem. However, Dijkstra cannot meet the real-time demand when the number of vertices is relatively large. Many later algorithms, no matter for the shortest path or for the k shortest paths, cannot break the real-time barrier. Afterwards, soft computing methods such as genetic algorithm and neural network came up [18–22], whose computation time are not sensitive to the size of networks, and can improve the computing efficiency [23].

Static and stochastic problems are characterized by arc weights partially known as random variables with definite distribution. This problem can be determined by setting each random arc weight to its expected value and solving an equivalent deterministic problem, and thereby be solved by generalized algorithms [24].

In dynamic problems, part or all of the arc weights change dynamically during the execution of the routes. For problems whose weights vary with external variables (usually with time), a generally accepted algorithm is to find the priori least expected time routes from all origins to a single destination for each departure time and define lower bounds on the expected times of these routes, then determine an exact solution [25]. While for the problem whose weights vary with the segments-chosen, the existing greedy algorithms are no longer applicable because it is not Markovian.

Thus, our work differs from these contributions in that we propose a novel route selection problem, which is a kind of the shortest path problem of dynamic weight-varying networks, and its arc-weights vary with the execution itself. Furthermore, two new algorithms are developed in order to solve it.

3. Dynamic programming model

3.1. Problem description

In the instant of setting out, the navigation device acquires initial states of traffic lights. Later throughout the trip, it is not necessary to get real-time states because of the periodicity of traffic lights. In this model, without considering yellow lights, we define the lights cycle with the instant green-to-red as the starting point.

If the traffic light is green when the vehicle reaching the maximum limited speed, the time-consumed in this travel is the actual time, which can be roughly described by the ratio of distance to speed when assuming driving uniformly. If it is red, when the vehicle drives as usual, taking the 'stop-wait-restart-accelerate' mode, the time-consumed in this travel must be longer than the sum of the running time and the waiting time because of the recovery (delay time). While when the vehicle drives based on dynamic signals without standstill and waiting, ideally, no excrescent time would cost, and the vehicle would end the travel at the instant the light changes to green.

3.2. Dynamic programming model

Divide the travel from the start to the end into n stages according to the traffic lights. The end of stage k is light k. Define l_k as the distance of stage k, v_k as the maximum limited speed (The lower between road speed limit and limited speed caused by congestion), C_k as the cycle of light k (such as C_1, C_2, C_3 in Fig. 1). These data can be held in the navigation device. Δt_{k0} is defined as the state of light k in the instant of setting out, $0 \le \Delta t_{k0} < C_k$ (such as Δt_{10}, Δt_{20}, Δt_{30} in Fig. 1). It needs to be acquired in real time. If $0 \le \Delta t_{k0} < C_k/2$, the light has changed to be red for Δt_{k0}; if $C_k/2 \le \Delta t_{k0} < C_k$, the light has changed to be green for $\Delta t_{k0} - C_k/2$.

Firstly, let Δt_k be the state variable, indicating the state of light k when the vehicle starts in stage k (such as Δt_2, Δt_3 in Fig. 1).

$$\Delta t_k = \begin{cases} \Delta t_{10}, \ k = 1 \\ C_k - \left(\sum_{i=1}^{k-1} t_i + \Delta t_{k0} \right) \backslash C_k, \ k = 2, 3, \ldots, n \end{cases}$$

(\: to get the remainder, the same below)

Secondly, let u_k be the decision variable, indicating the driving mode in stage k. It decides the time consumed in the stage, t_k. Figure 1 shows these different situations.

(1) When $C_k/2 \le (l_k/v_k + \Delta t_k)\backslash C_k < C_k$, that is, the light is green when the vehicle reaches at the maximum limited speed.

$$u_k = u_1, \ t_k = l_k/v_k$$

(2) When $0 \le (l_k/v_k + \Delta t_k)\backslash C_k < C_k/2$, that is, the light is red when the vehicle reaches at the maximum limited speed.

① $u_k = u_2, t_k = l_k/v_k + [C_k/2 - (l_k/v_k + \Delta t_k)\backslash C_k] + t_D$, t_D is the delay time for speed recovery. Usually $t_D \neq 0$.

② $u_k = u_3, t_k = l_k/v_k + [C_k/2 - (l_k/v_k + \Delta t_k)\backslash C_k] + t_E$, t_E is the time exceeding the green light time point when taking the dynamic driving mode. Ideally, $t_E = 0$.

Afterwards, let the time cost in the total travel be the optimal value function:

$$T = \min \sum_{i=1}^{k} t_i(u_i, \ \Delta t_k)$$

Finally, the recursive equation can be obtained:

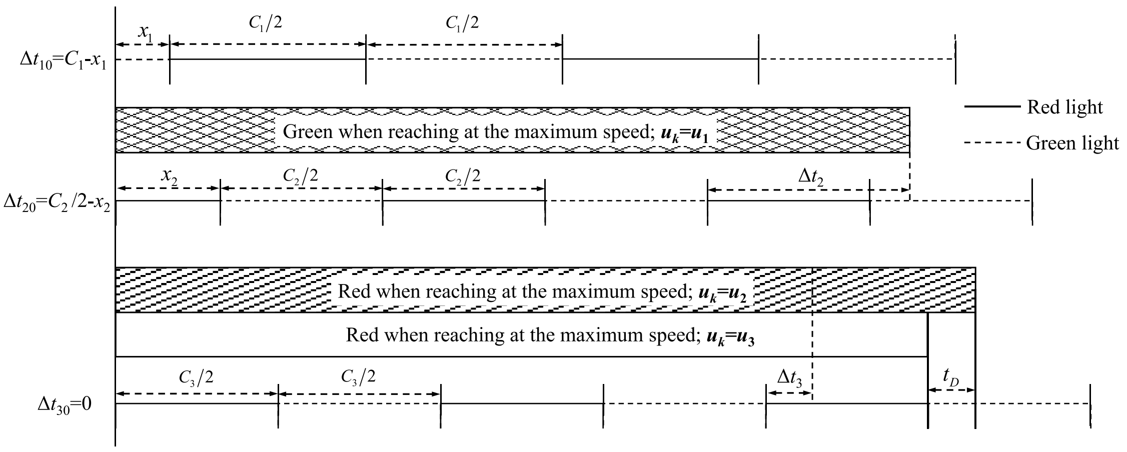

$\Delta t_{10}=C_1$-x_1
x_1
$C_1/2$
$C_1/2$
Green when reaching at the maximum speed; u_k=u_1
Red light
Green light
$\Delta t_{20}=C_2/2$-x_2
x_2
$C_2/2$
$C_2/2$
Δt_2
Red when reaching at the maximum speed; u_k=u_2
Red when reaching at the maximum speed; u_k=u_3
Δt_{30}=0
$C_3/2$
$C_3/2$
Δt_3
t_D

Fig. 1. Time consumed in different driving modes.

$$\begin{cases} f_k(\Delta t_k) = \min\limits_{u_k=\{u_1,u_2,u_3\}} \\ \qquad \{t_k(u_k,\ \Delta t_k) + f_{k-1}(\Delta t_{k-1})\} \\ k = 1, 2, \ldots,\ n \\ f_0(\Delta t_0) = f_0(\Delta t_{10}) = 0 \end{cases}$$

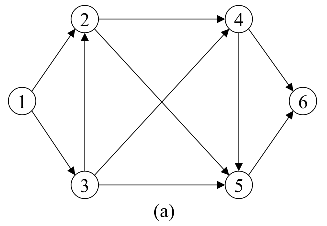
4. Two route selection algorithms

4.1. A brute-force algorithm based on matrix expansion

Usually, an exact route selection problem is solved by labeling. Since the problem discussed in this article does not satisfy the optimal principle, so we try to develop a brute-force algorithm (BFA). Firstly, BFA is to find all routes from the origin node to the destination node. Then it calculates the time consumed throughout different routes and selects the least travel time path.

Here, a method based on matrix expansion is used to find all routes between two nodes. In Fig. 2 (a), a digraph with 6 nodes is shown. Figure 2 (b) lists the simple expanding of route matrix from node 1 to node 6.

In fact, the rout matrix expending includes replicating and piecing. In this matrix, if the last node in a raw is not the destination and has m connected nodes, replicate the row for m times and piece the connected matrix next to these rows. If the last node in a row is the destination, the connected matrix would be 0 or Inf. So that we can obtain all routes until the connected matrix only contains 0 or Inf elements.



(a)

1	

1	2
1	3

1	2	4
1	2	5
1	3	2
1	3	4
1	3	5

1	2	4	5
1	2	4	6
1	2	5	6
1	3	2	4
1	3	2	5
1	3	4	5
1	3	4	6
1	3	5	6

1	2	4	5	6
1	2	4	6	0
1	2	5	6	0
1	3	2	4	5
1	3	2	4	6
1	3	2	5	6
1	3	4	5	6
1	3	4	6	0
1	3	5	6	0

(b)

Fig. 2. The digraph and its route matrix expending.

The steps of BFA are as follows:

Step 1. Generate adjacency matrix A. If node i is connected with node j, $A(i,\ j) = j$. If not, $A(i,\ j) = 0$.

Step 2. Let current matrix = [origin node], cycle and traverse each node, expend the current matrix until all routes are found.

Step 3. Give each arc the same weight $w_{ij} = M$. Set C_i as the cycle of each traffic lights and Δt_{i0} as the state in the instant of the vehicle setting out. Take l_{ij} and v_{ij} as the distance and the maximum limited speed of $(v_i,\ v_j)$ (The distance and speed of two disconnected nodes is ∞).

Step 4. Calculate time cost throughout each path.

Step 5. Select the path with the least travel time.

4.2. An improved adaptive ant colony algorithm

To simulate complex traffic and transportation processes, the swarm intelligence algorithm is well used [25]. Various changes in the route selection can be imitated by swarm's behaviors. Ant colony algorithm is an evolutionary computation proposed by Italian researchers Dorigo [26]. Generally, the ant system of the original ant colony algorithm is static. However, many real problems are dynamic, and more and more ant systems have been improved with different strategies and successfully applied to the dynamic optimization problems [27].

Ant system is originally designed for Traveling Salesman Problem (TSP). According to the characteristics of the problem, we redesign the ant system differing from two aspects. One is to replace randomly placing ants in different cities with placing ants in the same starting city. The other is to add a step to adjust arc weights in the searching loop. Based on these enhancements, we develop an improved adaptive ant colony algorithm ($IAACA$) to solve the shortest path problem in a digraph whose arc weights dynamically change with segments-chosen.

4.2.1. Algorithm symbols

Without loss of generality, the road network can be described as $\{V,\ E,\ W_k | k = 1, 2, \ldots,\ m\}$. Here, $V = \{v_0,\ v_1, \ldots,\ v_n\}$ is a finite point set, v_0 and v_n are the origin and destination points, respectively. $E = V \times V$ is the corresponding edge set. $W_k = \{w_{ij} | (v_i,\ v_j) \in E\}$ is the weight matrix when ant k is searching, it represents the least time cost by ant k during $(v_i,\ v_j)$. Initially, $w_{ij} = M$, and M is an arbitrarily large positive number. With edges being selected by the ant, the weights are modified continuously when the ant is at intersection i: $w_{ij} = t_{ij}(\Delta t_j)$, $v_j \in allowed_i$. $allowed_i =$ {intersections connected with v_i} − {intersections the ant k has gone}.

Assuming there is a path $P = \{(v_0,\ v_{i1}),\ (v_{i1},\ v_{i2}), \ldots,\ (v_{in},\ v_{i(n+1)})\}$, the time consumed during the total travel is $T(P) = w_{0i1} + w_{i1i2} + \ldots + w_{ini(n+1)}$. So, $IAACA$ is to find the path P, $T(P)$ of which is the least of all paths from the start to the end.

$$\Delta t_j = \begin{cases} \Delta t_{10}, \ j = 1 \\ C_j - \left(\sum_{i=1}^{j-1} t_i + \Delta t_{j0}\right) \backslash C_j, \ j \neq 1 \end{cases}$$

$$t_{ij}(\Delta t_j) = \begin{cases} l_{ij}/v_{ij}, & C_j/2 \leq (l_{ij}/v_{ij} + \Delta t_j)\backslash C_j < C_j \\ l_k/v_k + \left[\frac{C_k}{2} - (l_k/v_k + \Delta t_k)\backslash C_k\right], & 0 \leq (l_{ij}/v_{ij} + \Delta t_j)\backslash C_j < \frac{C_j}{2} \end{cases}$$

4.2.2. Weight-varying ant system

All m ants concentrate on the origin at first. Before ant i sets out, the weights of edges connected to v_0 should be adjusted. After that, ant i randomly selects an edge according to the selection strategy. Then ant i moves to the other node of this edge, weights of the edges allowed are modified, ant i selects one from them, and so on. The process would stop when searching to the destination. So that ant i obtains a solution. This time we should update the pheromone amount of edges in this path. After ant i finished the search, ant j sets out and finds a solution through the same process as ant i. Until all m ants completed the search, we can obtain m solutions, including some repetition. Then we can search the m solutions for a local optimum and update global pheromone amount based on the local optimum. Continually iterate as above until the maximum number of iteration is reached. The global optimum is the path with the minimum $T(P)$ of all local optima.

The steps of IAACA are as follow:

Step 1. Information initialization. Set the iteration number $Nc = 0$; maximum number of iteration $Nc_{\max}$. Give each edge the same amount of pheromone $\tau(i,\ j) = c$ and the same weight $w_{ij} = M$. Set C_i as the cycle of the traffic lights and Δt_{i0} as the state of the traffic lights in the instant of the vehicle setting out. Take l_{ij} and v_{ij} as the distance and the maximum limited speed of $(v_i,\ v_j)$ (the distance and speed of two disconnected nodes is ∞).

Step 2. Searching loop. Place m ants at the starting node v_i and perform the following operations:

Step 2.1. Determine nodes connected with the current node to form the allowed set, and adjust each weight in the set.

If $w_{ij} = t_{ij}(\Delta t_j)$, $v_j \in allowed_i$.

Step 2.2. Select the next node according to the following strategy, and then ant k transfers to the node.

$$p_{ij}^k(u_{ij},\ \Delta t_i) = \begin{cases} \dfrac{[\tau(i,\ j)]^\alpha [\eta(i,\ j)]^\beta}{\sum_{v_j \in allowed_k} [\tau(i,\ z)]^\alpha [\eta(i,\ z)]^\beta}, & v_j \in allowed_k \\ 0 & \text{, else} \end{cases}$$

Here, $\tau(i,\ j)$ indicates the pheromone amount left by ants passing through the road; $\eta(i,\ j)$ indicates the local heuristic information, define $\eta(i,\ j) = 1/w_{ij}(u_{ij},\ \Delta t_i)$, that is, if the time cost in $(v_i,\ v_j)$ is shorter, then $\eta(i,\ j)$ is larger, $p_{ij}^k(u_{ij},\ \Delta t_j)$ is also larger. α and β reflect the relative influence of pheromone amount $\tau(i,\ j)$ and heuristic information $\eta(i,\ j)$ on ant decision-making. Use roulette to select v_j.

Step 2.3. Select an edge $(v_i,\ v_s)$ every time, update the pheromone amount $\tau(i,\ s)$, so that the probability of other edges to be selected could be increased.

$$\tau(i,\ s) = \alpha\tau(i,\ s) \ 0 < \alpha < 1$$

Step 2.4. If v_j is not the end node, continue to search the following nodes, go to **step 2.1**. If v_j is the end node, that means ant k has found a solution, save the path P_k and the time it consumes T_k.

Step 3. After m ants have completed the search, update the global pheromone amount:

$$\tau(i,\ j) = \rho\tau(i,\ j) + \sum_{k=1}^{m} \Delta\tau(i,\ j)^k \ 0 < \rho < 1$$

Here, ρ indicates the residual degree of pheromone after volatilization. If ant k has passed through $(v_i,\ v_j)$, then $\Delta\tau(i,\ j)^k = Q/T_k$. If not, $\Delta\tau(i,\ j)^k = 0$. Q is a constant. It means the pheromone amount left on the path that an ant searches for at a time.

Step 4. Obtain the local optimum in m solutions, $T_{Nc} = \min\{T_k,\ k = 1, 2, \ldots m\}$.

Step 5. If $Nc_{\max}$ is reached, Exit. If not, $Nc = Nc + 1$, go to Step 2.

Step 6. Among all the local optima, the path with the minimum T(P) is the global optimum.

5. Computational experiments

5.1. Computational complexity

The two algorithms in Section 4 are both consisted of several parts. Tables 2 and 3 describe the complexity of different parts. Where n is the number of nodes; Nt is traversal times of BFA. If node i connects nt_i nodes, $Nt = \prod_{i=1}^{n-1} nt_i$, representing the scale of the problem; p is the number of all paths from origin to destination; Nc is iteration times; m is ant number in each iteration.

The two tables show that the computation complexity of BFA is $O(n \times Nt \times n)$, while that of $IAACA$ is $O(Nc \times m \times n^2)$. We can deduce that when the problem scale is small, the n and Nt is small,

Table 2
Computational complexity of *BFA*

Brute-force algorithm based on matrix expansion	
Generating adjacency matrix	$O(n^2)$
Searching all routes	$O(n \times Nt \times n)$
Calculate each path's cost time	$O(p \times n)$

Table 3
Computational complexity of *IAACA*

The improved adaptive ant colony algorithm	
Information initialization	$O(n^2)$
Searching loop	$O(m \times n^2)$
pheromone updating globally	$O(m \times n)$

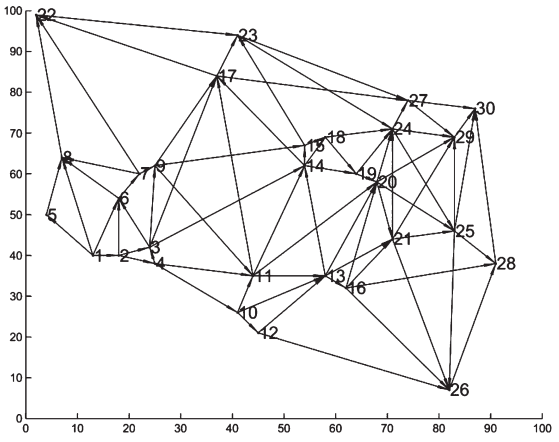


Fig. 3. The directed graph generated from the Oliver 30.

BFA not only can ensure the precise result, but also has a lower complexity. However, when the scale is larger, the Nt increases rapidly, $IAACA$ shows its superiority in efficiency.

5.2. Computational examples

Since there is no benchmark for the problem in this paper, we use the instance *Oliver30* in *TSPLIB*, sort and label vertices by the distance from the origin, then randomly generate the directed graph as shown in Fig. 3. In this digraph, the direction of the arcs is always from the smaller labeled nodes to the larger, so that in the following computation the destination can represent the network's size.

We randomly assign 90 s, 100 s and 120 s the three sizes of cycle to each traffic lights in the intersection, and set the initial state for each traffic lights. For the sake of convenience, we set speed limits of the route segments as $v_{ij} = 20\,m/s$.

The parameters of $IAACA$ are set as: the number of ants follows the label of destination, which represents the network size, the iterations are 40 times, the pheromone updating coefficient $\alpha = 0.5$, the heuristic information influence coefficient $\beta = 0.5$, and the pheromone evaporation $\rho = 0.8$.

Let node 1 as the origin point. After running in the environment with MATLAB R2014a, Intel i5-4210u CPU 1.70 Hz 2.40 Hz, the average of 10 times computation in the Table 4 are obtained. For a more intuitive observation, we also illustrate the results in the Fig. 4.

It can be seen from Fig. 4 that when the size of network size increases, the problem scale Nt also increases, especially it shows an exponential growth trend when the size gets to about 25. In Table 4,

Table 4
Computational results of the two algorithms

Destination	*BFA*		*IAACA*		Difference	
	Travel length/min	Computation time/s	Travel length/min	Computation time/s	Travel length/min	Computation time/s
1	0.00	0.0000	0.00	0.0014	0.00	–0.0014
2	4.17	0.0008	4.17	0.0359	0.00	–0.0351
3	9.44	0.0009	9.44	0.0283	0.00	–0.0274
4	10.23	0.0011	10.23	0.0345	0.00	–0.0334
5	11.21	0.0028	11.21	0.1295	0.00	–0.1267
6	12.60	0.0032	12.60	0.1690	0.00	–0.1658
7	18.61	0.0008	18.61	0.0239	0.00	–0.0231
8	21.35	0.0026	21.35	0.0541	0.00	–0.0515
9	21.61	0.0028	21.61	0.0794	0.00	–0.0766
10	27.37	0.0052	27.37	0.1259	0.00	–0.1208
11	26.26	0.0070	26.26	0.1662	0.00	–0.1592
12	32.70	0.0059	32.70	0.1015	0.00	–0.0956
13	38.29	0.0121	38.29	0.2579	0.00	–0.2459
14	39.48	0.0135	39.48	0.3403	0.00	–0.3268
15	43.65	0.0178	43.65	0.2758	0.00	–0.2580
16	43.12	0.0255	43.12	0.2932	0.00	–0.2677
17	43.21	0.0384	43.21	0.3467	0.00	–0.3083
18	46.85	0.0554	46.85	0.4054	0.00	–0.3500
19	47.98	0.0769	47.98	0.3581	0.00	–0.2813
20	52.38	0.0772	52.38	0.4768	0.00	–0.3996
21	52.00	0.1136	52.00	0.5059	0.00	–0.3922
22	51.37	0.1604	51.37	0.4360	0.00	–0.2756
23	52.79	0.1472	52.79	0.5865	0.00	–0.4392
24	58.02	0.6832	58.02	0.5351	0.00	0.1481
25	62.14	1.2178	63.47	0.6595	1.33	0.5583
26	66.64	1.1314	66.64	0.6210	0.00	0.5104
27	64.36	1.5369	64.36	0.7219	0.00	0.8150
28	67.80	3.8304	69.97	0.8749	2.17	2.9555
29	67.88	6.1299	68.41	0.8276	0.53	5.3023
30	74.60	25.0059	74.67	0.8408	0.07	24.1651

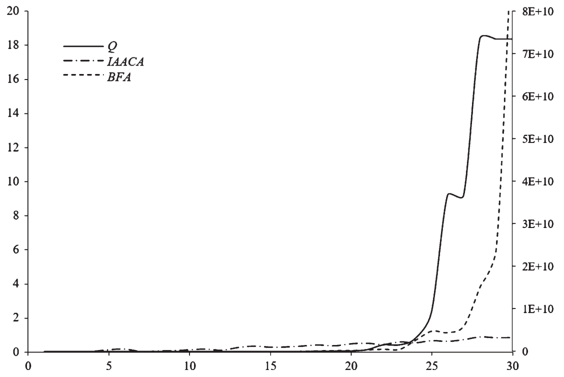


Fig. 4. Computation time of the two algorithms varying with the size of network.

though the *IAACA* cannot find the optimal solution each time, the overall optimal rates are still satisfactory because there are only four suboptimal solutions in 300 experiments. As for computation time of the two algorithms, the experiment results are consistent with the complexity analysis. When the size of network is larger than 24, *Nt* grows to about 3.06E+9, and the computation time of *BFA* is greater than that of *IAACA*; when it reaches 28, *Nt* grows to about 7.34E+10, and the computation time of the *BFA* suddenly increases. Therefore, compared with *IAACA*, *BFA* will fail to solve the larger scale problem due to data overflown.

6. Conclusions

In this paper, we discuss a novel dynamic shortest path problem for ITS that is not Markovian, in which networks' arc-weights vary with the execution itself. So the existing greedy algorithms are not applicable. The problem is derived by an envisaged driving mode with dynamic signals: drivers select specific driving choice guided by devices to run without any break. We first establish the dynamic programming model for this driving mode on a single path, and then develop two new algorithms to solve it. One is a brute-force algorithm (*BFA*) with the computational complexity of $O(n \times Nt \times n)$. The other is an improved adaptive ant colony algorithm (*IAACA*) with the complexity of $O(Nc \times m \times n^2)$. The computational experiments show that the two algorithms can both solve the problem effectively. When the scale of problem is small, the *BFA* can obtain the optimal solution, as well as has a less computing time compared with the *IAACA*. When the scale of problem is large, the computation time of *BFA* increases sharply due to the rapid growth of the amount of all paths, while the *IAACA* shows a superior performance. The proposed model and developed algorithms in this paper will provide the theoretical guideline and decision support for ITS.

Acknowledgments

This work was partly supported by the Funds for the National Natural Science Foundation of China (Nos. 71671033, 71371190, 71210003, 71431006), and the Fundamental Research Funds for the Central Universities (Nos. N160601001, N150605001).

References

[1] Traffic light. In Wikipedia, The Free Encyclopedia. Retrieved 12:36, February 14, 2017, from https://en.wikipedia.org/w/index.php?title=Traffic_light&oldid=765302777, (2017).

[2] M. Collotta, L.B. Lucia and P. Giovanni, A novel approach for dynamic traffic lights management based on Wireless Sensor Networks and multiple fuzzy logic controllers, *Expert Systems with Applications* **42**(13) (2015), 5403–5415.

[3] J. Wahle, O. Annen and C. Schuster, A dynamic route guidance system based on real traffic data, *European Journal of Operational Research* **131**(2) (2001), 302–308.

[4] Z.S. Yang, *Theory and Model of Urban Traffic Flow Guidance System*, China Communications Press, Beijing, 2000.

[5] C.L.P. Chen and C.Y. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on Big Data, *Information Sciences* **275(1)** (2014), 314–347.

[6] F.-M. Manuel, A survey on fractal dimension for fractal structures, *Applied Mathematics and Nonlinear Sciences* **1**(2) (2016), 437–472.

[7] B. Li and N. You, Research on the dynamic evolution behavior of group loitering air vehicles, *Applied Mathematics and Nonlinear Sciences* **1**(2) (2016), 353–358.

[8] M. Gentili and P.B. Mirchandani, Locating sensors on traffic networks: Models, challenges and research opportunities, *Transportation Research Part C: Emerging Technologies* **24**(1) (2012), 227–255.

[9] C. Bachmann, B. Abdulhai and M.J. Roorda, A comparative assessment of multi-sensor data fusion techniques for freeway traffic speed estimation using microsimulation modeling, *Transportation Research Part C: Emerging Technologies* **26**(1) (2013), 33–48.

[10] A. Amini, R.M. Vaghefi and M. Jesus, Improving GPS-based vehicle positioning for intelligent transportation systems, *Intelligent Vehicles Symposium Proceedings, 2014 IEEE IEEE*, 2014.

[11] D. Singh, G. Tripathi and M.A. Wadud, SEE: A Smart-Eye for Intelligent Transportation System, *The 2nd International Symposium on Advanced and Applied Convergence (ISAAC 2014)*, 2014.

[12] C.Y. Hsu, C.S. Yang and L.C. Yu, Development of a cloud-based service framework for energy conservation in a sustainable intelligent transportation system, *International Journal of Production Economics* **164**(1) (2015), 454–461.

[13] W. He, G. Yan and L.D. Xu, Developing vehicular data cloud services in the IoT environment, *IEEE Transactions on Industrial Informatics* **10**(2) (2014), 1587–1595.

[14] M. Patriksson, *The traffic assignment problem: Models and methods*, Courier Dover Publications, 2015.

[15] V. Pillac, M. Gendreau and C. Gueret, A review of dynamic vehicle routing problems, *European Journal of Operational Research* **225**(1) (2013), 1–11.

[16] F.B. Zhan and C.E. Noon, Shortest path algorithms: An evaluation using real road networks, *Transportation Science* **32**(1) (1998), 65–73.

[17] F.B. Zhan, Three fastest shortest path algorithms on real road networks: Data structures and procedures, *Journal of Geographic Information and Decision Analysis* **1**(1) (1997), 69–82.

[18] 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* **6**(6) (2002), 566–579.

[19] M.K.M. Ali and F. Kamoun, Neural networks for shortest path computation and routing in computer networks, *IEEE Transactions on Neural Networks* **4**(6) (1993), 941–954.

[20] Z.-Z. Jiang, Z.-P. Fan, W.H. Ip and X.H. Chen, Fuzzy multi-objective modeling and optimization for one-shot multi-attribute exchanges with indivisible demand, *IEEE Transactions on Fuzzy Systems* **24**(3) (2016), 708–723.

[21] Z.-Z. Jiang, W.H. Ip, H.C.W. Lau and Z.-P. Fan, Multi-objective optimization matching for one-shot multi-attribute exchanges with quantity discounts in E-brokerage, *Expert Systems with Applications* **38**(4) (2011), 4169–4180.

[22] H.C.W. Lau, Z.-Z. Jiang, W.H. Ip and D.W. Wang, A credibility-based fuzzy location model with Hurwicz criteria for the design of distribution systems in B2C e-commerce, *Computers & Industrial Engineering* **59**(4) (2010), 873–886.

[23] E. Avineri, Soft computing applications in traffic and transport systems: A review, *Soft computing: Methodologies and Applications* (2005), 17–25.

[24] E.D. Miller-Hooks and H.S. Mahmassani, Least expected time paths in stochastic, time-varying transportation networks, *Transportation Science* **34**(2) (2000), 198–215.

[25] D. Teodorović, Swarm intelligence systems for transportation engineering: Principles and applications, *Transportation Research Part C: Emerging Technologies* **16**(6) (2008), 651–667.

[26] M. Dorigo, V. Maniezzo and A. Colorni, Ant system: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **26**(1) (1996), 29–41.

[27] H.M. Kammoun, I. Kallel and J. Casillas, Adapt-Traf: An adaptive multiagent road traffic management system based on hybrid ant-hierarchical fuzzy model, *Transportation Research Part C: Emerging Technologies* **42**(1) (2014), 147–167.