Hindawi
Scientific Programming
Volume 2021, Article ID 9922466, 19 pages
https://doi.org/10.1155/2021/9922466

Review Article

Improved AND/OR Tree Search Algorithm in Analysis of Stochastic and Time-Dependent Shortest Path Problem

Zhi-ying Xie,[1,2] **Yuan-Rong He**,[1,2] **Yuan-tong Jiang**,[3,4] **and Chih-Cheng Chen**[5,6]

[1]*School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China*
[2]*Big Data Institute of Natural Hazards Monitoring for Digital Fujian, Xiamen, Fujian 361024, China*
[3]*School of Cultural Industries and Tourism, Xiamen University of Technology, Xiamen 361024, China*
[4]*Research Center of Cultural Industries, Fujian Social Science Research Base, Xiamen 361024, China*
[5]*School of Ocean Information Engineering, Jimei University, Xiamen 361021, China*
[6]*Department of Aeronautical Engineering, Chaoyang University of Technology, Taichung 413, Taiwan*

Correspondence should be addressed to Yuan-Rong He; 2012112001@xmut.edu.cn and Chih-Cheng Chen; ccc@gm.cyut.edu.tw

Received 5 April 2021; Revised 9 June 2021; Accepted 14 July 2021; Published 28 July 2021

Academic Editor: Qianchuan Zhao

Real-time vehicle guidance effectively reduces traffic jams and improves the operational efficiency of urban transportation. The trip time on a route is considered as a random process that changes with time, and the shortest path selection requires a random dynamic model and the solution of a decision-making problem. Thus, the shortest trip time is the criterion to determine the dynamic path selection by a random dynamic programming (DP) model which discretizes the trip times in the continuous segments on the route. In this study, a numerical model of random dynamic programming is established by using a probability tree model and an AND/OR (AO*) algorithm to select the path of the shortest trip time. The results show that the branches of the probability tree are only accumulated on the "quantity" and do not cause a "qualitative" change. The inefficient accumulation of "quantity" affects the efficiency of the algorithm, so it is important to separate the accumulation of "quantity" from node expansion. The accumulation of "quantity" changes the trip time according to the entering time into a segment, which demands an improved AO* algorithm. The new AO* algorithm balances between efficiency and the trip time and provides the optimal real-time vehicle guidance on the road.

1. Introduction

Vehicle-to-everything (V2X) and Internet of Things (IoT) collect a large amount of observation data of traffic from multisource sensors and devices and require big data technology [1–4]. The collected data may be used to provide the optimal path for vehicles, especially unmanned vehicles. In unmanned driving, dynamic path planning is required in two aspects: global path planning [5] and obstacle avoidance [6]. The algorithms of obstacle avoidance in path planning [7–9] and global path planning [10–12] are attracting considerable interest. However, the randomness of the traffic data from the variable traffic network environment complicates the algorithms, which still need more research than before.

The trip time in a segment of a vehicle path is regarded as a time-dependent random variable, and its probability distribution depends on the entering time of a vehicle to the road segment [13]. Thus, the problem to obtain the shortest time of the vehicle's trip in the segment is the shortest path problem of a stochastic time-dependence (STD) network. Hall [14] stated that a label setting (LS) or a label correcting (LC) algorithm was not suitable for the shortest path as Bellman's optimality principle does not hold randomness [15].

The optimal path selection is not a simple shortest path problem, but an adaptive decision-making problem with time. The adaptive decision-making problem cannot be solved by a fixed shortest path method. The optimal path selection strategy from a starting to the next target node considers the time to move between them. The path selection is not only needed from the starting node but from each decision point (intersection) to reach the target node in the shortest time. The optimal path must be selected to have the shortest trip time to the target node or intersection, and the subsequent selection will be no longer affected by the previous selection process. For this type of decision-making, dynamic programming is regarded to be appropriate. Thus, a problem of stochastic and time-dependent shortest path (STDSP) requires a model of dynamic programming.

This study aims to suggest an AND/OR (AO*) algorithm to solve an STDSP problem in finding the optimal path with the shortest trip time between nodes or intersections. The complexity of the shortest path problem in an STD network was analyzed by using a probability tree diagram and a first in first out (FIFO) algorithm based on the nonsatisfaction and nonuniqueness of the trip times. Then, a model of stochastic dynamic programming for dynamic path selection was applied to the estimation of the minimum trip times. Based on the heuristic function and the two-point discretization of the trip time between segments (between nodes or intersections), an AO* algorithm, a heuristic search algorithm was proposed to solve the STDSP problem by using the STD programming model with the consideration of the balances between the efficiency and the optimal trip. With the results, the STDSP problem was defined with the analysis of the complexity of the STDSP problem. Then, the algorithm was validated with real-time observation data. Finally, the proposed AO* algorithm was improved again for suggesting the final algorithm.

2. Theoretical Background

Bellman's optimality principle is used to find the shortest path between any of the two nodes in a network and applied to a static shortest path algorithm. When the trip time between the nodes is constant, that is, the path is time-independent, an efficient labeling algorithm solves the shortest path problem. Wu et al. demonstrated [16] that the Bellman optimal principle in the static network is also applicable to the STD and FIFO networks. In the FIFO network, the algorithm of the shortest path in the static network can be used for solving the shortest path in a dynamic network. Thus, LS or LC algorithms solve such shortest path problems. When a trip time is regarded as a random variable with a known probability distribution with the objective function, the random shortest path problem is transformed into a deterministic one for obtaining a desired shortest path. However, a random trip time can be obtained by an efficient labeling algorithm by solving the expected shortest path problem [17]. The shortest time is obtained as the solution of an objective function of the path between nodes, which does not consider the random characteristics of the network [18].

In the actual traffic, the minimum trip time, as well as the minimum risk, is considered [19]. Therefore, the minimum expectation-mean square error [20] and α-reliable path problem need to be considered [21]. The path objective function of the minimum expectation-mean-square error path is a linear combination of the expected value and the mean square error. Thus, the proportional coefficient of the linear combination needs an artificial setting that is arbitrary and cannot uniformly explain the path selection.

Dynamic programming is appropriate for multistage decision-making problems. Steinmetz et al. [22] modeled a dynamic path selection as a Markov decision process (MDP) and believed that each segment on the road corresponded to blocking and nonblocking. When the current state of each segment is known, an intersection to pass is selected. Hall [14] suggested the use of dynamic programming to solve the time-dependent path selection and proposed the algorithm that combined a k-shortest path and a branch and bound method. This algorithm had low operating efficiency and could not satisfy the requirement in real time. When Wu et al. [16] summarized the rules for solving STDP problems using the LC algorithm which had randomness. They believed that replacing the definite value with the expected value of a random variable allowed similar results to the definite situation by assuming consistency. However, the assumption is not applicable in the actual transportation network as the real situation has randomness.

Fu and Rilett [23] regarded a random time-dependence problem of the pate selection by Hall [14] to be a continuous random process. They transformed the time-dependence problem into an extreme value problem to estimate the trip time on the path. Based on the trip time and its variance in each segment, a series of probability-based approximation models (PAMs) were established by using the k-shortest path. Then, a heuristic algorithm was used to solve the STDSP problem. Fu [24] proposed an adaptive strategy and used dynamic programming for the PAM. However, his adaptive LC algorithm did not adopt the nondifferentiability of the extreme value problem in the use of Rosenblueth's two-point estimation.

Global path planning is based on an algorithm of optimal path selection which is divided into intelligent and graph search algorithms. Graph search algorithms [10] include Dijkstra's algorithm and A* algorithm. As an A* algorithm uses heuristic estimation, it reduces the amount of search, improves efficiency, and guarantees the optimality of the path. However, the efficiency is lower in a complex environment of a large scale. Intelligent algorithm [11, 12] simulates the biological evolution and biomimetic nature of insect foraging and nesting, mainly including genetic algorithm, ant colony algorithm, particle swarm optimization, etc. This is appropriate for solving and optimizing complex problems by having the characteristics of potential parallelism but has slow operation speed and premature solution and does not consider randomness and time dependence.

Previous studies proposed good solutions for the shortest path problem with randomness but not for the problems of time dependence. A problem of randomness with time dependence requires a different approach to obtain the solution. Therefore, this study aims to find a solution to the trip time and path selection by considering the random processes that change with time, that is, time-dependent. Thus, the shortest path problem is considered as a segmented decision-making problem by using dynamic programming.

3. Problem Descriptions

The real-time navigation of a vehicle requires planning an optimal path from the current to the target position (node) based on real-time information. By updating the trip times in each segment, the guidance strategy shows the shortest path with an appropriate algorithm. When the strategy does not consider the variability of a driving route, it only proposes a suboptimal path. Variability is solved with dynamic programming that decomposes a complex problem into a series of simple problems. The optimal path of a route is obtained by decomposed into segments between the nearest two nodes. For the shortest path problem of the nodes, the principle of stochastic dynamic programming is considered for dynamic path selection.

3.1. Definition of STDSP. The real-time navigation of a vehicle requires planning an optimal path from the current to the target position (node) based on real-time information. By updating the trip times in each segment, the guidance strategy shows the shortest path with an appropriate algorithm. When the strategy does not consider the variability of a driving route, it only proposes a suboptimal path. Variability is solved with dynamic programming that decomposes a complex problem into a series of simple problems. The optimal path of a route is obtained by decomposed into segments between the nearest two nodes. For the shortest path problem of the nodes, the principle of stochastic dynamic programming is considered for dynamic path selection.

The traffic network is a spatiotemporal network. The spatiotemporal network is expressed as

$$G_T = (V_T, A_T), \tag{1}$$

where $V_T = \{(i,t) | i \in V, t \in T\}$ and $A_T = \{(i,t), (j, t+\tau_{ij}(t)) | i, j \in V, t, t+\tau_{ij}(t) \in T\}$.

Here, (i,t) refers to a point of (space, time). The defined G_T does not have a closed-loop and avoids a cyclic search. When the successor node is defined as $\Gamma(i,t)$, the next node is $\Gamma^{-1}(i,t)$. $X_a(t)$ represents the trip time in a segment at time t, and$\mu_a(t)$, $\sigma_a(t)$, respectively, represent the expectation and variance at time t. $f_{X_a(t)}(x)$ represents a probability density function.

There are three possible directions at the intersections: going straight, turning left, and turning right. The signal control system develops from the point control of "single point fixed cycle" to the direction of line control and surface control. The signal cycle of intersection will be adjusted according to the traffic flow. Due to its complexity, most intersections are still controlled by point control in reality, so this article only studies the situation that the signal cycle does not change.

The problem is represented by the average of past historical data as $d_{ij}(t)$ which means the delay of intersection with the next node j on node i at time t. Then, the strategy of the path selection problem is defined as

$$\begin{aligned} &p: V_T \longrightarrow V, \\ &p(i,t) \in \Gamma(i,t). \end{aligned} \tag{2}$$

It is a mapping strategy from the node of a spatiotemporal network to the node of a nontemporal network. When n_k is the k-th node to arrive, and its corresponding arrival time is t_k, then $(n_{(k+1)}, t_{(k+1)}) = p(n_k, t_k)$. $((n_k, t_k), (n_{(k+1)}, t_{(k+1)}))$ is random due to the randomness of the trip time of $t_{(k+1)}$ to node $n_{(k+1)}$. It can be used as a predicted value of the time to reach $n_{(k+1)}$ and from node n_k.

3.2. The Complexity Caused by the Branch of Probability. Since the trip times on a route are random, the time to reach a certain node is uncertain. Thus, total trip time is decomposed into several arrival times with the probability of different shortest paths.

Figure 1 shows an example. If the departure time from node 1 is 0, the probability to reach node 2 at $t = 10$ and $t = 45$ is 0.5, respectively. There are five routes from node 1 to node 6 to choose from. The probability to reach node 6 at $t = 25$ and $t = 60$ is 0.5, respectively, and the expected trip time is 42.5 s.

Path (1, 2, 3, 6) is the same as path (1, 2, 4, 6) but has a different probability and expected trip time of 57.5. The expected trip times of path (1, 3, 6), path (1, 3, 2, 4, 6), and path (1, 5, 6) are 95, 75, and 70, respectively. The paths are not the optimal trip from node 1 to node 6. The trip from node 1 at $t = 0$ to node 2 at $t = 10$, node 3 at $t = 15$, and node 6 at $t = 20$ sequentially has the probability of 0.5. For some reason, when the trip reaches node 2 at $t = 45$ due to the entry time limit on the road section (3, 6) and changes its path to node 4 at $t = 50$ and then to node 6 at $t = 60$, the probability is also 0.5 and the expected trip time from node 1 to node 6 is 40. This path has the shortest time, but it is not the same as the path in the static state that is expressed as a path (1, 2, 3, 6|4, 6).

In this case, the search process of the shortest path from node 1 to node 6 can be resolved by using an AND/OR tree. The decomposition needs to define the state representation of the problem. The problem has two types of states. One is the state when it reaches a certain node, and the other is the selection state when starting from a certain node. The two types of states alternately make a pair. The decomposition process of the original problem is shown in Figure 2. The numbers in gray and white ellipses represent (node, arrival time) and (node, arrival tine, next node). The search process starting from node 1 at time 0 includes the subprocesses to nodes 2, 3, and 5, which requires "OR" node. Starting from node 1 to node 2 at time 0, due to the randomness, there are two arrival times to node 2. When there is the shortest path from node 2 to node 6, it is indicated with a certain probability, and the shortest trip time from node 1 to node 6 can be calculated. In this case, this process needs an AND node, and the trip time is the sum of that between the previous nodes. Through AND/OR tree decomposition, every feasible path from node 1 to node 6 can be found, and the optimal path by dynamic path selection at node 2 can also be found by a thick line in Figure 2.

In decomposing the STD shortest path, the AND nodes are not appearing in the static shortest path problem. If dynamic path selection is not performed at node 2, the

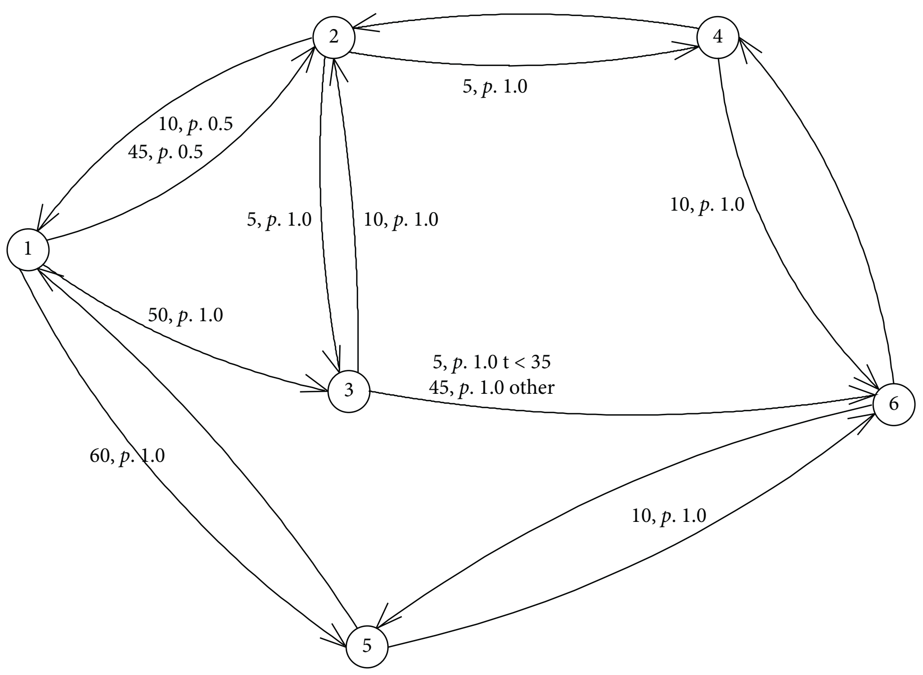


Figure 1: An STD net.

optimal route is (1,2,4,6), and the estimated trip time is 42.5 that is longer than that of the optimal path (1,2,3, 6|4, 6) with the trip time of 40. This shows that the STD problem is complicated due to time correlation although the estimated trip time is used as the optimal evaluation criterion.

3.3. FIFO Complexity Caused by Unsatisfactory Conditions. Kaufman and Smith [16] defined the FIFO condition when solving the time-dependent shortest path, that is, for any $(i, j) \in A$, $s + d_{ij}(s) \le t + d_{ij}(t)$ for all $s, t \in T$, when $s \le t$ is always true. As long as the FIFO conditions are met, efficient labeling algorithms solve the shortest time-dependent path problem.

Since the time-dependent trip time of the (2,3) section does not meet the FIFO condition ($2 + 4$ is not less than $3 + 2$), the shortest path trip time is 6 if the Dijkstra algorithm is used (Figure 3). However, the trip from nodes 1 to 3 has the shortest trip time of 5, which means that nodes 1 to 2 are not the shortest path. This does not conform to the Bellman optimality principle. When the FIFO condition is met (Figure 3(a)), the Bellman optimality principle is also satisfied. Miller-Hooks and Mahmassani [17] extended the FIFO condition with the time-dependency and defined it as

$$\begin{aligned}&\forall (i, j) \in A,\\ &\operatorname{Prob}\left\{s + \tau_{ij}(s) \le t + \tau_{ij}(t)\right\} = 1, \quad \forall s \le t,\end{aligned} \tag{3}$$

where $\tau_{ij}(t)$ is the trip time on (i, j). This implies that the STD network with an independent trip time is calculated with the assumption of consistency between the paths.

In Figure 4(b), since the trip time of (2,3) section does not meet the extended FIFO condition, the expected shortest path from nodes 1 to 3 at time 0 is 5, and its route is ((1,0), (2,3)&&(2,4), (3,5)); the expected shortest path from nodes 1 to 2 at time 0 is 2.5, and its route is ((1,0), (2, 2) &&(2,3)). The shortest path from nodes 1 to 2 is inconsistent with that from nodes 1 to 3, which means that the expected trip time is not simply equal to the expected trip time. The situation that does not meet the definition of the extended FIFO is common in STD networks. Thus, it is unreasonable that the non-FIFO section of the shortest path is excluded.

4. Dynamic Programming Model and AO* Algorithm

4.1. Dynamic Programming Model. An optimal strategy $p^*(i,t) \in \Gamma(i,t)$ of the path from a node to the target node in the shortest trip time is described as follows:

$$\begin{aligned}p^*(i,t) \in \arg\min\Big\{&\mu_a(t) + v^*\left(j, t + d_{ij}(t) + X_a(t)\right) + d_{ij}(t) \mid \forall a\\ &= \left((i,t), \left(j, t + d_{ij}(t) + X_a(t)\right)\right) \text{ and } \left(j, t + d_{ij}(t) + X_a(t)\right) \in \Gamma(i,t)\Big\},\end{aligned} \tag{4}$$

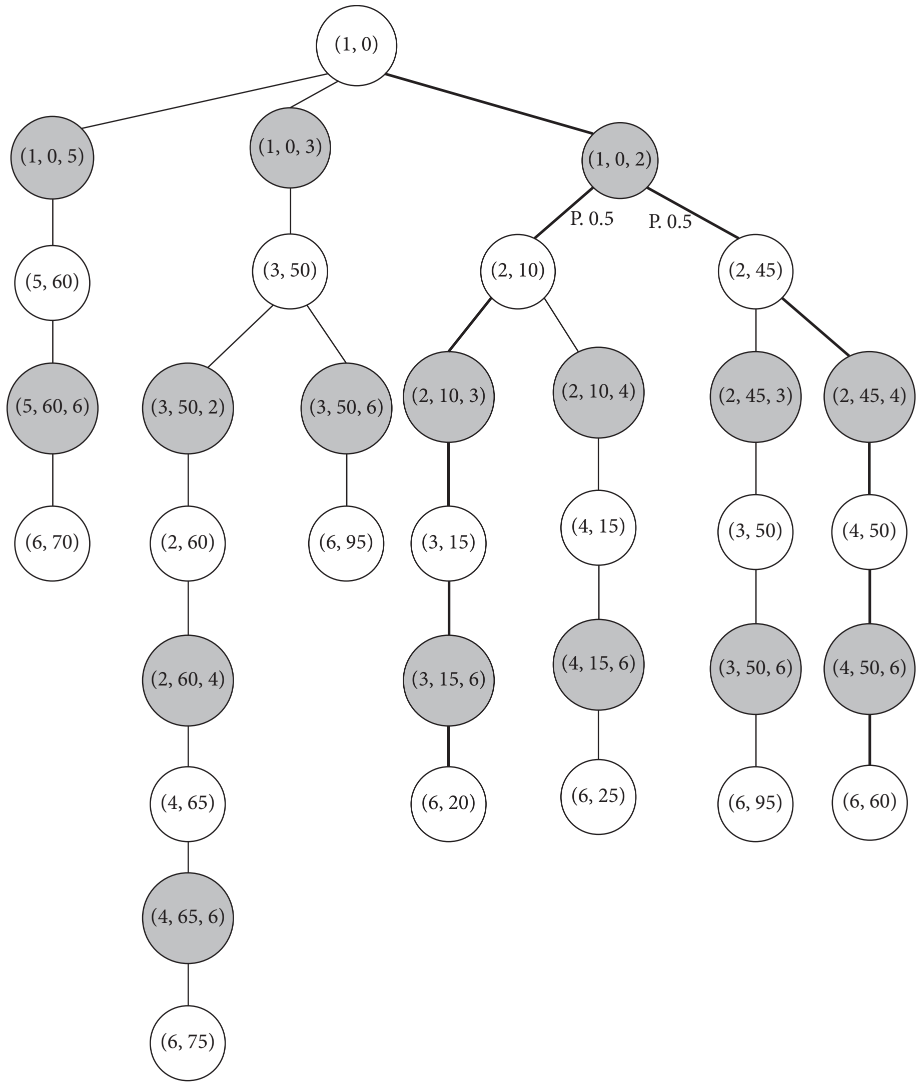


Figure 2: STD AND/OR graph decomposition of the shortest path problem.

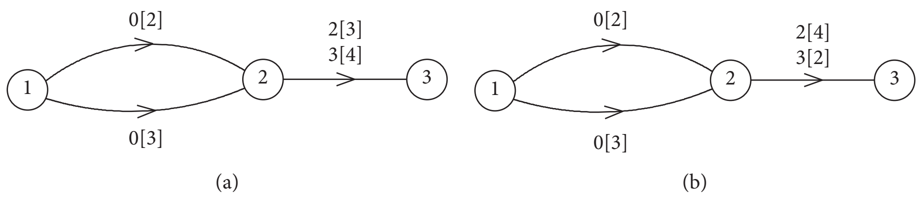


Figure 3: Time-dependent dynamic network. (a) FIFO definition. (b) Non-FIFO definition.

where $v^*(i,t)$ is the shortest trip time t from nodes i to d at time t. Then,

$$v^*(i,t) = E(Y_i(t)), \tag{5}$$

$$Y_i(t) = \min\left\{ \begin{array}{c} v^*\left(j, t + d_{ij}(t) + X_a(t)\right) + X_a(t) + d_{ij}(t) \mid \\ \forall a = \left((i,t), \left(j, t + d_{ij}(t) + X_a(t)\right)\right) \text{ and } \left(j, t + d_{ij}(t) + X_a(t)\right) \in \Gamma(i,t) \end{array} \right\}, \tag{6}$$

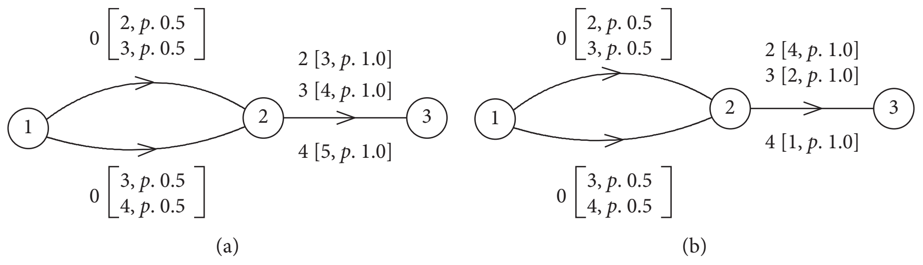


FIGURE 4: Simple random time-dependent net. (a) Meet the definition of extended FIFO. (b) Non-FIFO definition.

with boundary conditions $v^*(d,\cdot)=0$, the recursive process of dynamic programming is used to solve equations (5) and (6). However, obtaining the random variable is difficult as the extreme function. $Y_i(t)$ has at least one random variable. The statistics $F(y)$ is defined as

$$F_{Y_i(t)}(y)=1-\left[1-F_{X_{a1}(t)}(y)\right]\left[1-F_{X_{a2}(t)}(y)\right]\cdots\cdot\left[1-F_{X_{a|\Gamma(i,t)|}(t)}(y)\right], \tag{7}$$

where $|\Gamma(i,t)|$ represents the out-degree of the spatiotemporal node (i,t). Although the trip time in each independent segment follows an asymptotically normal distribution [25], the random variables do not meet the requirements of independent and identical distribution:

$$E(Y_i(t))=\int_{-\infty}^{+\infty} y\, dF_{Y_i(t)}(y). \tag{8}$$

By equation (8), it is very difficult to solve $E(Y_i(t))$. Fu [24], based on Rosenblueth's two-point estimation method, calculates the expectation and variance of random variable function from the expectation and variance of random variable, so as to effectively avoid solving the distribution of random variable function. However, the correctness of the equation is questionable as the extreme values were not differentiable, and Rosenblueth used a series to expand and approximate the first few terms for approximation. As we do not assume the dependence of road trip time on time, the time of entering a segment does not have to be considered for recursing. This study starts from the Rosenblueth two-point estimation method [26] to study the discrete method of solving $E(Y_i(t))$.

4.2. Numerical Stochastic Dynamic Programming Model. The density of the random variable X is defined as the function $f_X(x)$ and the random variable $Z=h(X)$, a function of the random variable. The two-point estimation method uses the probability of two points to express $f_X(x)$, and the probability of these two points meets the first three moments of $f_X(x)$. $E(Z^k)$, $k=1,2$, is calculated from two discrete values if $Z=h(X_1,X_2,\ldots,X_n)$ is a function of n random variables. Then, $E(Z^k)$, $k=1,2$, is calculated from 2^n discrete values.

Figure 5 depicts the typical situation of a route selection when the vehicle is at node (i,t). At the node, there are three possibilities to move to the next node: going straight, turning left, and turning right.

When $Y_i^k(t)$ is defined as the loop variable with the minimum value of $Y_i(t)$ and the number of loops of k, equation (6) is written as follows:

$$Y_i^1(t)=v^*\left(j_1,t+\tau_{ij_1}\right)+X_{ij_1}\left(t+d_{ij_1}(t)\right)+d_{ij_1}(t), \tag{9}$$

$$Y_i^k(t)=\min\left\{Y_i^{k-1}(t),v^*\left(j_k,t+\tau_{ij_k}\right)+X_{ij_k}\left(t+d_{ij_k}(t)\right)+d_{ij_k}(t)\right\},\quad k=2,\ldots,|\Gamma(i,t)|. \tag{10}$$

The expected calculation corresponding to equations (9) and (10) is then

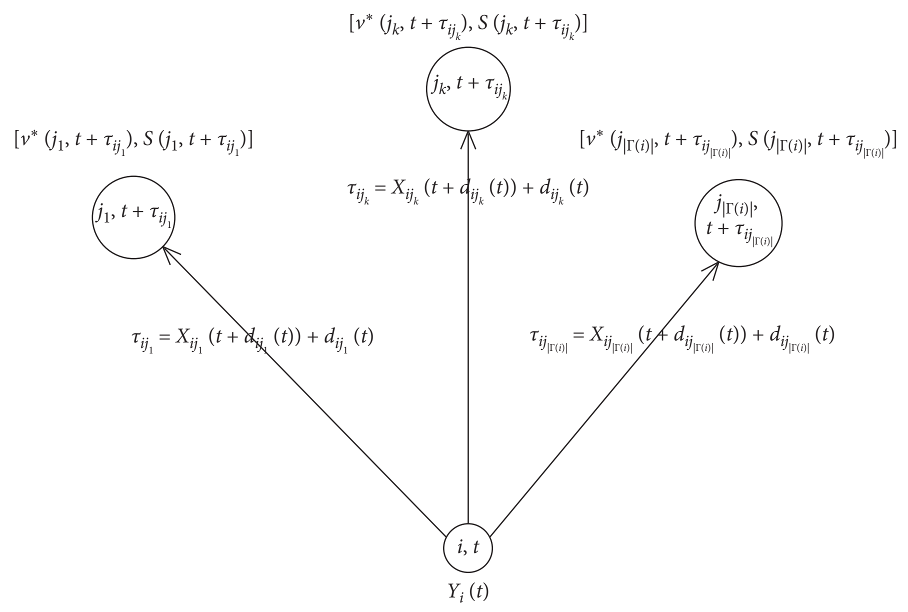
$$\begin{aligned}v_k^*(i,t)&=E\left[Y_i^k(t)\right]\\&=E\left[\min\left\{Y_i^{k-1}(t),v^*\left(j_k,t+\tau_{ij_k}\right)+X_{ij_k}\left(t+d_{ij_k}(t)\right)+d_{ij_k}(t)\right\}\right]\\&=\min\left\{v_{k-1}^*(i,t),v^*\left(j_k,t+\tau_{ij_k}\right)+E\left[X_{ij_k}\left(t+d_{ij_k}(t)\right)\right]+d_{ij_k}(t)\right\},\quad k=2,\ldots,|\Gamma(i,t)|,\end{aligned} \tag{11}$$

$$v_1^*(i,t)=E\left[Y_i^1(t)\right]=v^*\left(j_1,t+\tau_{ij_1}\right)+E\left[X_{ij_1}\left(t+d_{ij_1}(t)\right)\right]+d_{ij_1}(t). \tag{12}$$

The trip time of the shortest path from nodes i to d at time t is calculated as

$$v^*(i,t)=E\left[Y_i(t)\right]=v^*_{|\Gamma(i,t)|}(i,t). \tag{13}$$

Equations (10) and (11) describe the reverse recursive process of dynamic programming. In the dynamic path selection, the trip time in a segment $X_a(t)$ is calculated in a continuous random process. The time t changes



$[v^* (j_k, t+\tau_{ij_k}), S(j_k, t+\tau_{ij_k})]$
$j_k, t+\tau_{ij_k}$
$[v^* (j_1, t+\tau_{ij_1}), S(j_1, t+\tau_{ij_1})]$
$[v^* (j_{|\Gamma(i)|}, t+\tau_{ij_{|\Gamma(i)|}}), S(j_{|\Gamma(i)|}, t+\tau_{ij_{|\Gamma(i)|}})]$
$j_1, t+\tau_{ij_1}$
$\tau_{ij_k} = X_{ij_k}(t+d_{ij_k}(t)) + d_{ij_k}(t)$
$j_{|\Gamma(i)|}, t+\tau_{ij_{|\Gamma(i)|}}$
$\tau_{ij_1} = X_{ij_1}(t+d_{ij_1}(t)) + d_{ij_1}(t)$
$\tau_{ij_{|\Gamma(i)|}} = X_{ij_{|\Gamma(i)|}}(t+d_{ij_{|\Gamma(i)|}}(t)) + d_{ij_{|\Gamma(i)|}}(t)$
i, t
$Y_i(t)$

FIGURE 5: Dynamic path choice at node (i, t).

continuously, and thus the unrestricted boundary condition $v^*(d,\cdot) = 0$ to determine that t has the arbitrariness of the time during the recursion. However, this reverse recursion does not meet the requirement in this study. Thus, the sequential recursive process is adopted.

However, if the trip time in a segment is calculated by a heuristic function instead of $v^*(j_k, t+\tau_{ij_k})$, a sequential recursive process for the path can be obtained from equation (11). If the appropriate heuristic function is used, the optimal path for the search can be determined. Then, when the boundary conditions are reached, $v^*(j_k, t+\tau_{ij_k})$ is deduced along the path, and then $v^*(i,t)$ is deduced inversely.

Rosenblueth's two-point estimation method discretizes the continuous distribution of random variable Z into a simple two-point distribution and is expressed as

$$p_1 = P(Z = \mu_Z - \sigma_Z) = \frac{1}{2},$$
$$p_2 = P(Z = \mu_Z - \sigma_Z) = \frac{1}{2}, \tag{14}$$

where μ_Z is the expectation and σ_Z is the standard deviation.

The trip time $X_a(t)$ of a segment is discretized into a simple two-point distribution using Rosenblueth's two-point estimation method. That is,

$$p_1 = P\left(X_a(t) = \mu_{X_a}(t) + \sigma_{X_a}(t)\right) = \frac{1}{2},$$
$$p_2 = P\left(X_a(t) = \mu_{X_a}(t) - \sigma_{X_a}(t)\right) = \frac{1}{2}, \tag{15}$$

where $h(j,t)$ represents the heuristic trip time in the segment of (j, t) with the probability of 0.5. See

$$c(i,t,j) = \mu_{X_{ij}}\left(t + d_{ij}(t)\right) + d_{ij}(t). \tag{16}$$

Equation (11) is discretized as

$$v_k^*(i,t) = \min\left\{ v_{(k-1)}^*(i,t), c(i,t,j_k) + \frac{1}{2}\sum_{m=1,2} h\left(j_k, t + c(i,t,j_k) + (-1)^m \cdot \sigma_{X_{ij_k}}\left(t + d_{ij_k}(t)\right)\right)\right\}, \quad k = 2,\ldots,|\Gamma(i,t)|. \tag{17}$$

Similarly, equation (11) is discretized as

$$v_1^*(i,t) = E\left[Y_i^1(t)\right] = c(i,t,j_1) + \frac{1}{2}\sum_{m=1,2} h \left(j_1, t + c(i,t,j_1) + (-1)^m \cdot \sigma_{X_{ij_1}}\left(t + d_{ij_1}(t)\right)\right). \tag{18}$$

Equations (17) and (18) with boundary condition $v^*(d,\cdot) = 0$ constitute a numerical model of stochastic dynamic programming. The solutions of equations (10) and (11) need the selection of heuristic trip times that are modified when the boundary conditions are reached. Dynamic programming is not an algorithm [27], but a modeling method. Based on the heuristic trip time, the AO* algorithm is effective for solving the random dynamic programming models.

4.3. AO Algorithm.* With the complexity of the STD shortest path problem, an AND/OR tree is used to decompose its solution process. The AND/OR tree inversely calculates the trip time from a precursor node to a successor node in a dynamic programming model that uses equations (17) and (18). The AO* algorithm accompanies an AND/OR graph search in artificial intelligence. Bander and White [28] first applied the algorithm to a nonstatic random shortest path problem and constructed a solution graph. The result was appropriate for a dynamic selection model, as the path was not a single one but a hyperpath. Different path options are decided by the time of entering into a segment.

4.3.1. Graph Heuristic Search and Agreement with AND/OR Graph. The heuristic search algorithm for AND/OR graphs was formally named AO* by Nilsson in 1980 [28]. Whether a node on the AND/OR graph is solved is determined by its successor nodes. An "AND" node becomes solvable only when all its successor nodes are solvable. An "OR" node is solvable when one of the successor nodes is solvable. In the growth process of the graph as the search process, nodes are continuously added to the AND/OR graph. The generated graph is denoted as G_T' and then $G_T' \subseteq G_T$. The demarking process is a retrospective recursive process in which the precursor nodes are solvable by the solvable successor nodes. These are used repeatedly in the search process of the AND/OR graph until a node is marked as a solvable or unsolvable node.

The result of the heuristic search of the AND/OR graph finds the optimal solution graph with the least cost (shortest time). For the AND/OR graph, the nodes that are expected to be part of the optimal solution graph for expansion must be selected. The AND/OR graph with the solvable successor and their precursor nodes is called the hope graph (potential solution graph). In the search process, as new nodes join, the trip time is constantly changing. Therefore, the graph is also constantly changing with the constant graph of heuristic search.

The total cost (time) of the heuristic search of the AND/OR graph is obtained by calculating the trip time in the graph. With $c(i,t,j)$, the trip time from a node $(i,\ t)$ to a successor node $(j,\ t_j)$, the trip time is calculated as follows:

(1) If the node $(i,\ t)$ is the target (successor) node, its trip time is a function $f(i,t) = 0$.

(2) If the node $(i,\ t)$ is an AND node, its trip time is a function as follows:

$$f(i,t) = \sum_{k=1}^{|\Gamma(i,t)|} cc\left(i,t,j_k\right) + f\left(j_k, t_{j_k}\right), \left(j_k, t_{j_k}\right) \in \Gamma(i,t). \tag{19}$$

(3) If the node (i,t) is an OR node, its trip time is a function as follows:

$$f(i,t) = \min\left\{c\left(i,t,j_k\right) + f\left(j_k, t_{j_k}\right), \left(j_k, t_{j_k}\right) \in \Gamma(i,t)\right\}. \tag{20}$$

(4) If $(i,\ t)$ is not scalable and it is not the target node; then, its trip time is defined as $f(i,t) = \infty$.

The trip time of the precursor node is deduced from that of the successor node. Thus, with the optimal solution graph, the trip times from successor nodes to the precursor node are obtained, which is repeated layer by layer, and finally, the starting node can be found. This process solves equations (17) and (18).

4.3.2. Algorithm. With the random variables of the trip time in continuous segments by using the Rosenblueth two-point estimation method, the solution of the random dynamic programming model is regarded as the search process of the AND/OR graph. The flowchart of the AO* algorithm is shown in Figure 6. It includes two processes: generating the successor nodes and the reverse correction of the trip time.

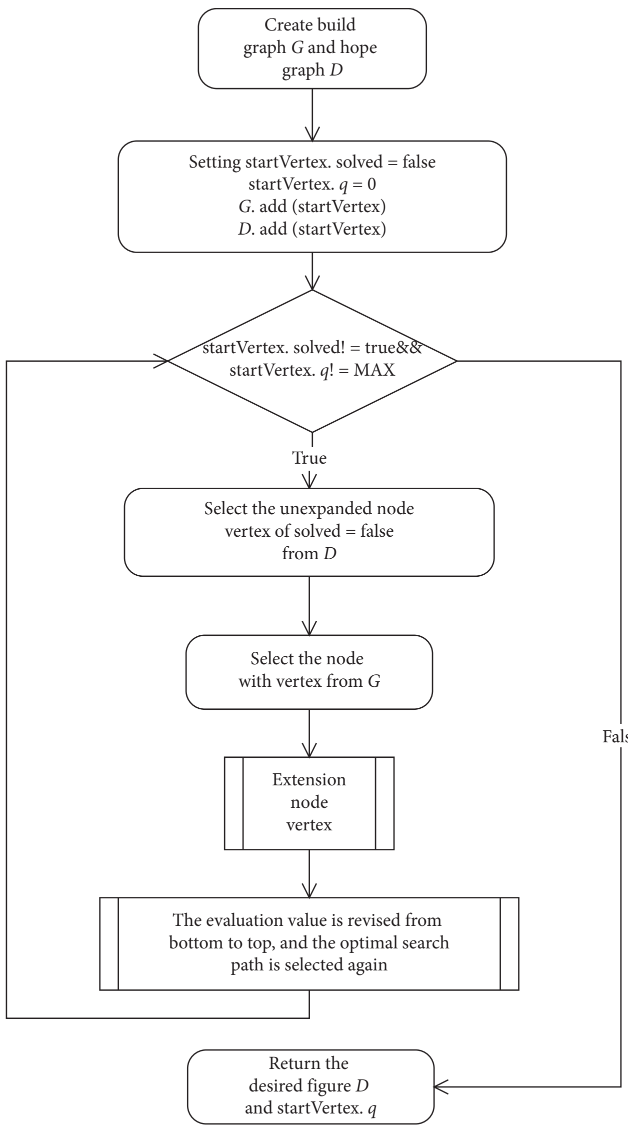
In the algorithm, $q(i,t)$ represents the trip time from a node $(i,\ t)$. $h(i,t)$ represents the heuristic function of the node $(i,\ t)$. We use the Euclidean distance from a node $(i,\ t)$ to the target successor node and the maximum free flow velocity in each segment. When the flow speed is zero at the starting node $(i,\ t)$,

$$h(i,t) = \frac{\text{Dist}(i,d)}{u_0}. \tag{21}$$

Meuleau et al. [29] proved that the AO* algorithm must be able to terminate on the optimal solution when the heuristic function calculates the shorter trip time than the real trip time and there is an optimal solution. As the heuristic function in this study satisfies this condition, the optimal decision $p^*(i,t) \in \Gamma(i,t)$ at the node $(i,\ t)$ is found by the AO* algorithm, which presents the path of the shortest trip time from nodes s to d.

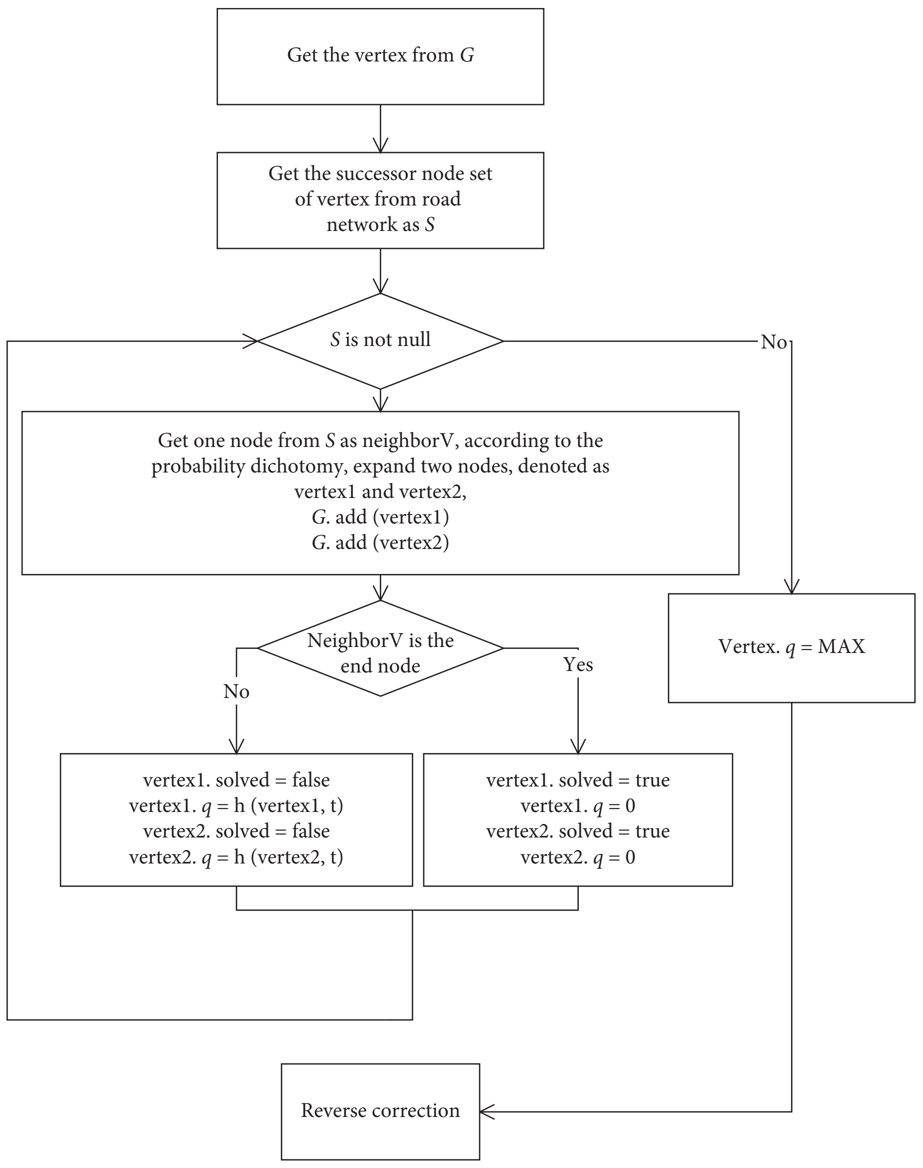
The AO* algorithm does not allow loops in the search process. Finding the following successor nodes requires operating on the nodes that are not the precursors. Due to the monotonicity of time, there is no possibility of loops in the spatiotemporal traffic network G_T. The heuristic search does not find all nodes on G_T but those in the optimal path. The search is processed based upon heuristic information, so the algorithm is efficient in finding the optimal path.

4.4. Improving AO Algorithm.* Since the increased number of successor nodes affects the efficiency of the algorithm, the probability tree diagram is necessary to consider during the search process. When the variance of the trip times in segments is not large, the conditional probability has little effect on the final optimal path selection. In some segments, the past trip times are almost constant. In this case, the conditional probability does not affect significantly. When the variance becomes large, the conditional probability of the branches is considered, which simplifies the process significantly with a less number of nodes and the improvement of the efficiency of the algorithm. Figure 7 shows


Create build graph G and hope graph D
Setting startVertex. solved = false
startVertex. $q = 0$
G. add (startVertex)
D. add (startVertex)
startVertex. solved! = true&&
startVertex. q! = MAX
True
Select the unexpanded node vertex of solved = false from D
Select the node with vertex from G
Extension node vertex
The evaluation value is revised from bottom to top, and the optimal search path is selected again
False
Return the desired figure D and startVertex. q

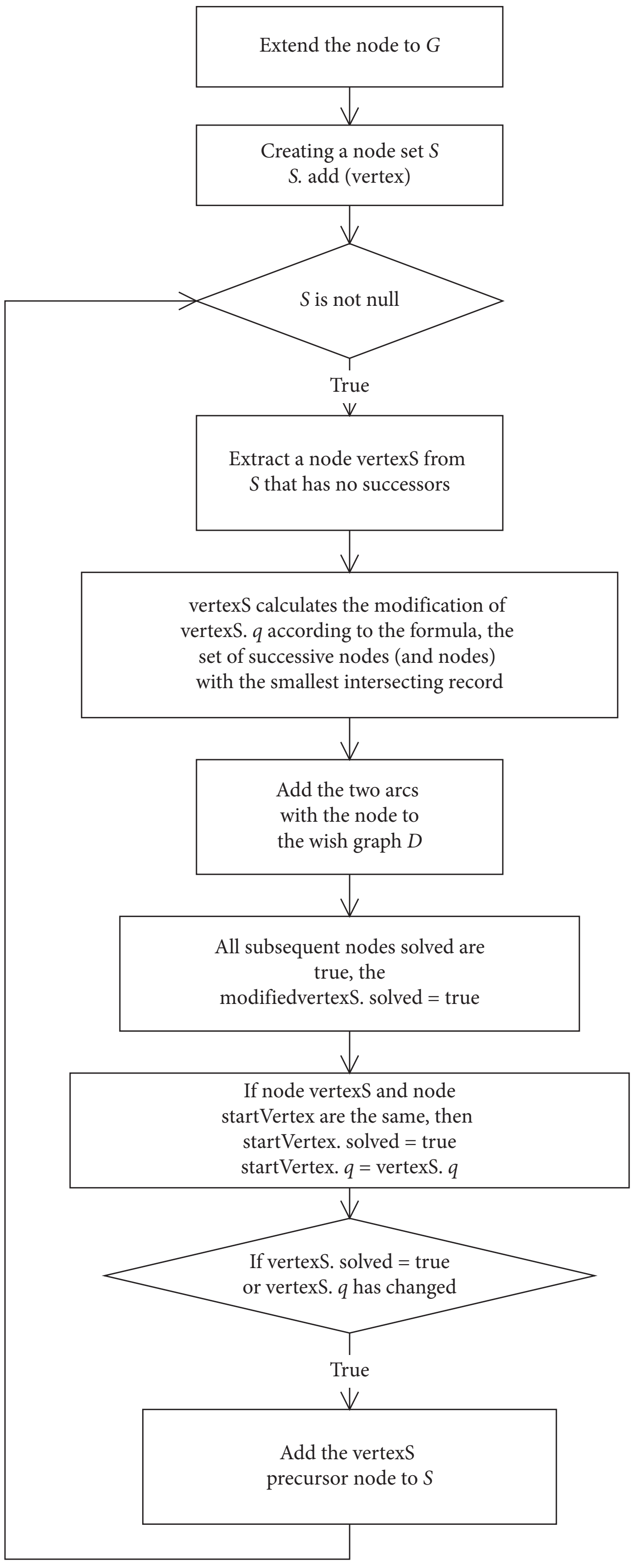
(a)

FIGURE 6: Continued.


Get the vertex from G
Get the successor node set of vertex from road network as S
S is not null
No
Get one node from S as neighborV, according to the probability dichotomy, expand two nodes, denoted as vertex1 and vertex2, G. add (vertex1) G. add (vertex2)
NeighborV is the end node
No
Yes
vertex1. solved = false vertex1. q = h (vertex1, t) vertex2. solved = false vertex2. q = h (vertex2, t)
vertex1. solved = true vertex1. q = 0 vertex2. solved = true vertex2. q = 0
Vertex. q = MAX
Reverse correction

(b)

Figure 6: Continued.


Extend the node to *G*
Creating a node set *S*
S. add (vertex)
S is not null
True
Extract a node vertexS from *S* that has no successors
vertexS calculates the modification of vertexS. *q* according to the formula, the set of successive nodes (and nodes) with the smallest intersecting record
Add the two arcs with the node to the wish graph *D*
All subsequent nodes solved are true, the modifiedvertexS. solved = true
If node vertexS and node startVertex are the same, then startVertex. solved = true startVertex. *q* = vertexS. *q*
If vertexS. solved = true or vertexS. *q* has changed
True
Add the vertexS precursor node to *S*

(c)

Figure 6: Heuristic AND/OR graph search flowchart. (a) General flowchart. (b) Node expansion process. (c) Reverse correction process.

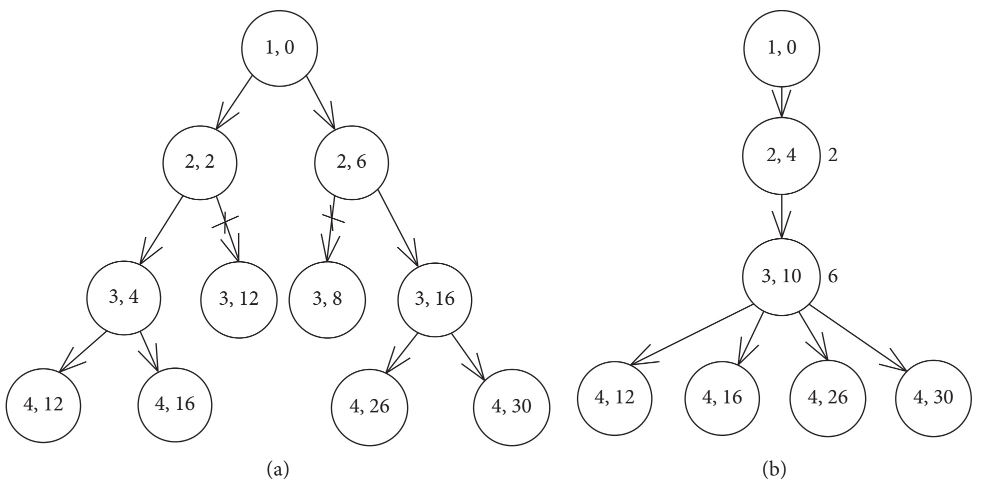


Figure 7: Judging the node expansion diagram based on the variance and trip time. (a) Multilevel expansion of the nodes. (b) Limiting the expansion.

diagrams on how to judge for expanding the nodes increasing the number of nodes) based on the variance of trip times. The branch in Figure 7(a) has the multilevel expansion of the nodes, while that in Figure 7(b) limits the expansion. In each node (circle), the trip times and the variance of them are shown. Figure 7(b) shows that limiting the expansion only needs the nodes with either the longest or the shortest trip time with the estimated trip time unchanged when compared to Figure 7(a) that has the expansion of the odes. Based on this result, the flowchart in Figure 6 is modified and shown in Figure 8.

5. Case Studies and Discussion

5.1. Application of Improved AO* Algorithm. The traffic data in Shenzhen was used to calculate the trip times of each segment on a route with the improved AO* algorithm. Figure 7 shows the expected trip times (in parenthesis) on various routes and times in a day. In nonpeak times such as 7:00, 9:30, 14:00, and 20:30, the path was chosen for a smaller number of segments than in peak times such as 8: 30, 11:00, 12:30, and 18:00 when the road has traffic congestion. When there are many vehicles in a segment, another segment with fewer vehicles is chosen for a shorter trip time.

The calculation result at 12:30 (Figure 9(e)) suggested two different choices at node 138. Of course, all the nodes shown in Figure 9 have the same kind of selection as node 138, but the result of selection is the same road segment in space, and the difference is only in time. It can be considered that the difference in time is only the difference in "quantity," and the accumulation of "quantity" will cause the change in "quality," such as the appearance of different road segment selection results. If there is no choice for a segment, then this means that the change in "quantity" is not enough to cause that in "quality." If there are two paths on which it takes 45 (going straight) and 50 seconds (turning right) to reach node 138 from node 28, the optimal path will be the path for a shorter total trip time. Of course, the path choice at node 138 depends on the road condition of the segment (nodes 28-138) and the estimated arrival time at node 138.

5.2. Performance of Improved AO* Algorithm. Randomness changes the trip time even in the same segment at the same period. Then, many solution graph nodes need to be considered in the search of the algorithm. Table 1 shows the statistics of the AO* algorithm search process. When the number of nodes is increased, the number of solution graph nodes and the time for search increased. That is, the efficiency of the search decreased. The expansion of solution graph nodes affects the efficiency of the algorithm based on the probability of each branch (the branch).

Table 2 shows the result from the search process by using the improved algorithm. The increase of the number of nodes does not affect significantly the numbers of the solution graph nodes, the generated nodes, the iteration, and the time for search. This indicates the advantages of the heuristic search and the function used.

The data relationship in the two tables is shown in Figure 10. It can be clearly seen that the efficiency of the unrestricted algorithm decreases rapidly as the number of road network nodes increases. The efficiency of the improved algorithm does not change much with the increase in the number of road network nodes, which is also the characteristic of heuristic search.

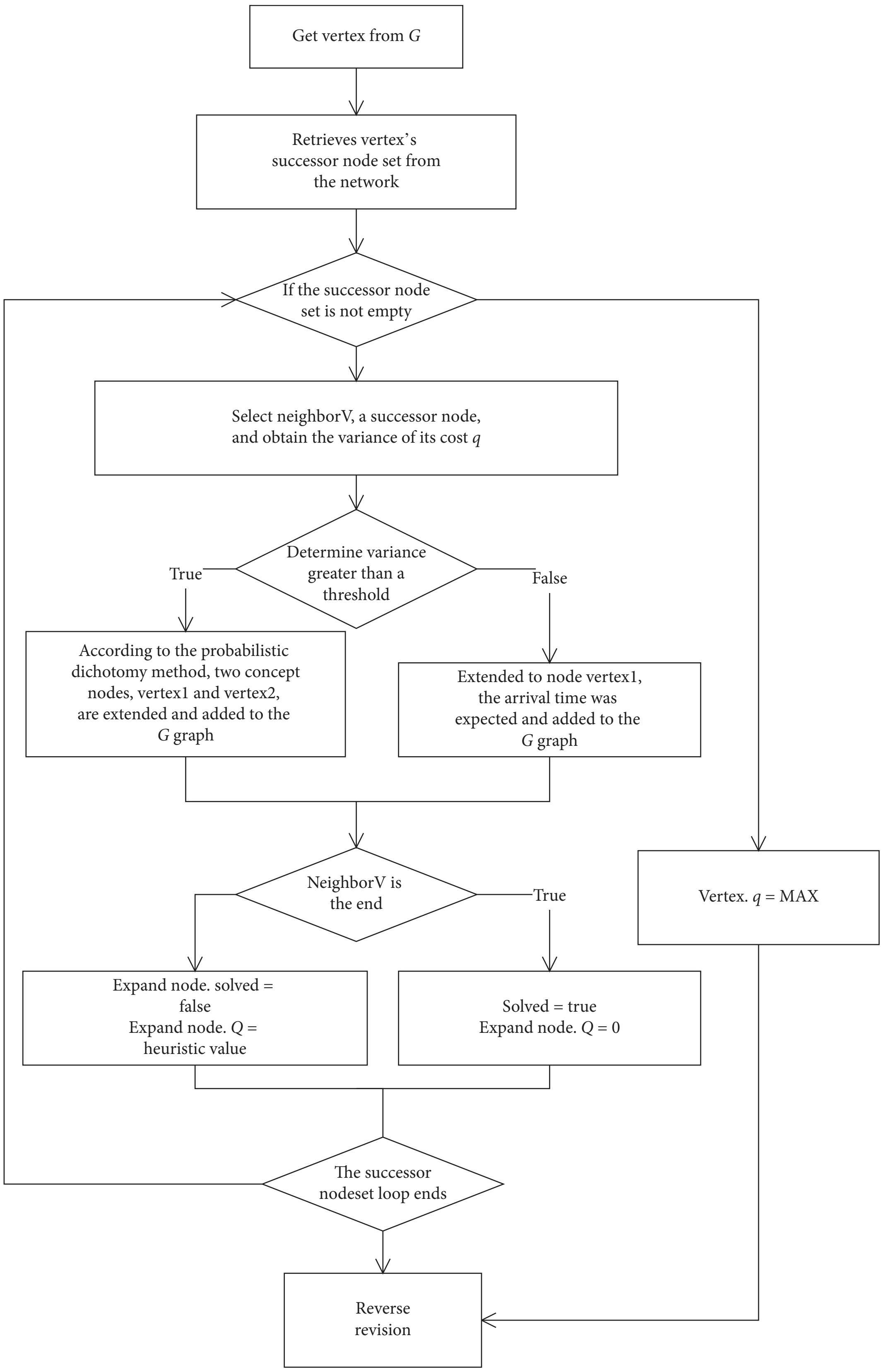
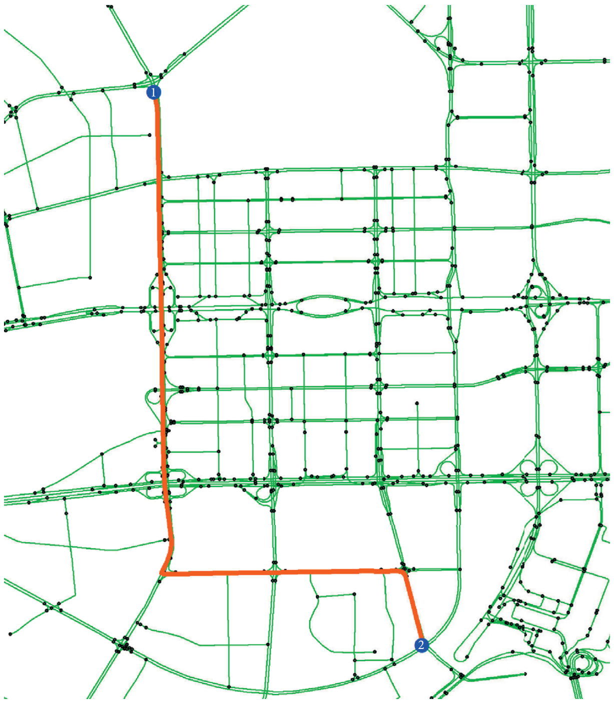
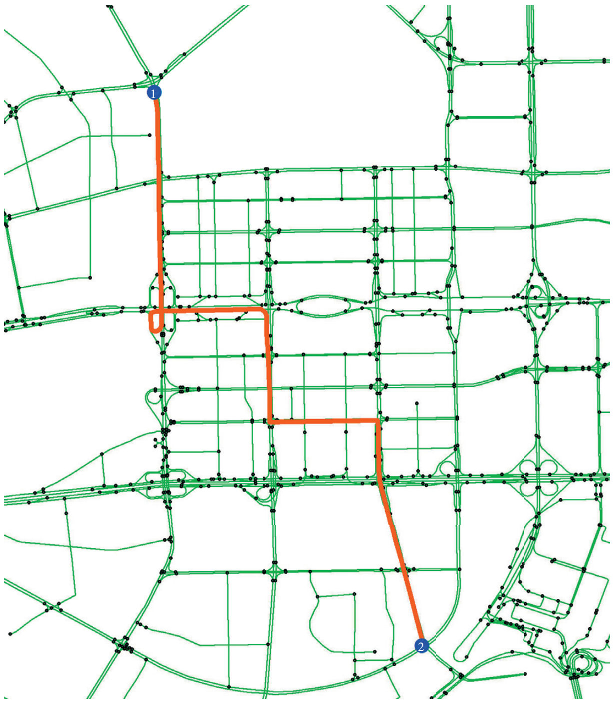

Get vertex from *G*
Retrieves vertex's successor node set from the network
If the successor node set is not empty
Select neighborV, a successor node, and obtain the variance of its cost *q*
Determine variance greater than a threshold
True
False
According to the probabilistic dichotomy method, two concept nodes, vertex1 and vertex2, are extended and added to the *G* graph
Extended to node vertex1, the arrival time was expected and added to the *G* graph
NeighborV is the end
True
Vertex. *q* = MAX
Expand node. solved = false Expand node. *Q* = heuristic value
Solved = true Expand node. *Q* = 0
The successor nodeset loop ends
Reverse revision

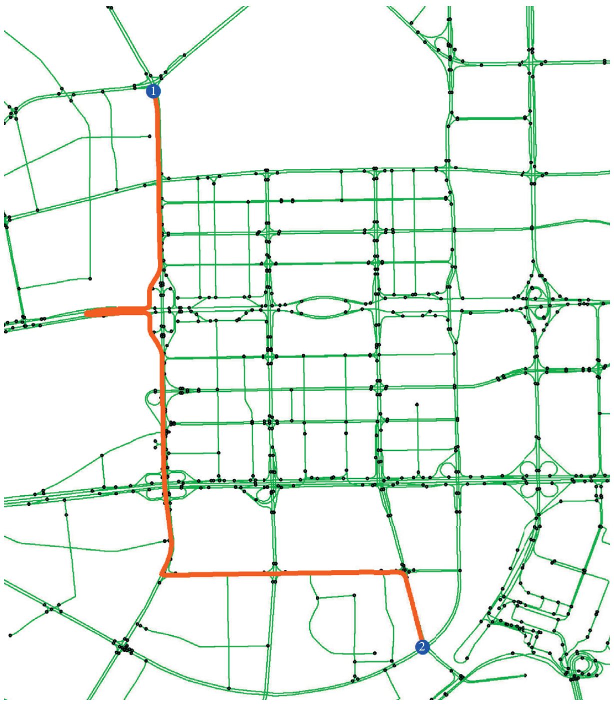
FIGURE 8: The flowchart for the limited expansion search process.



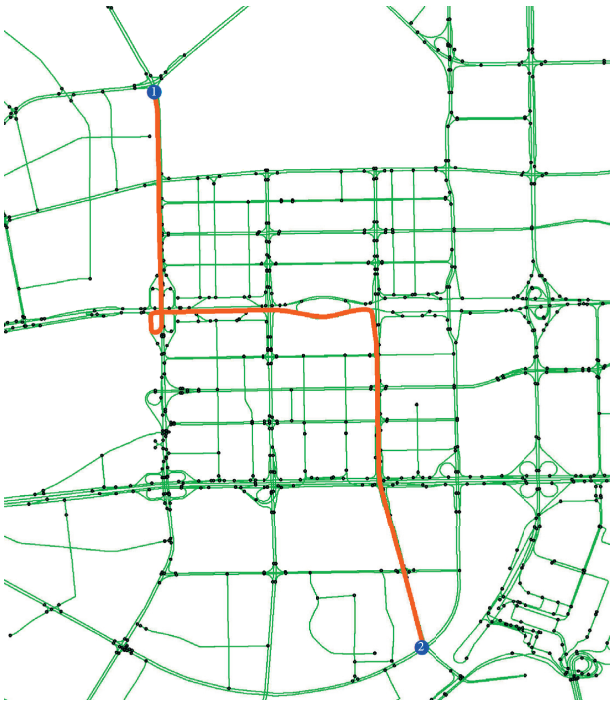
(a)



(b)

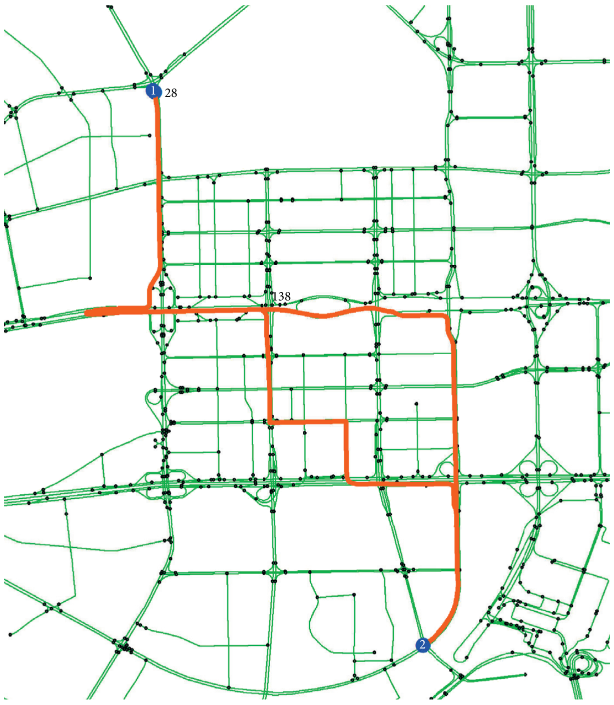


(c)

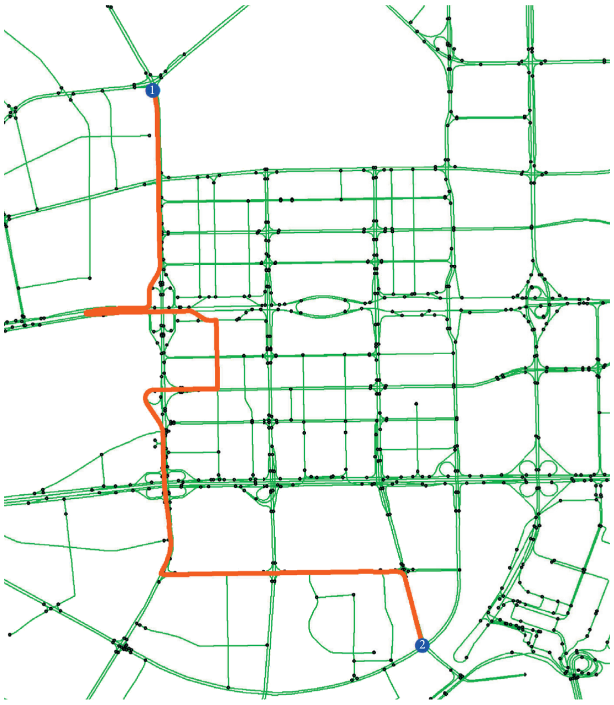


(d)

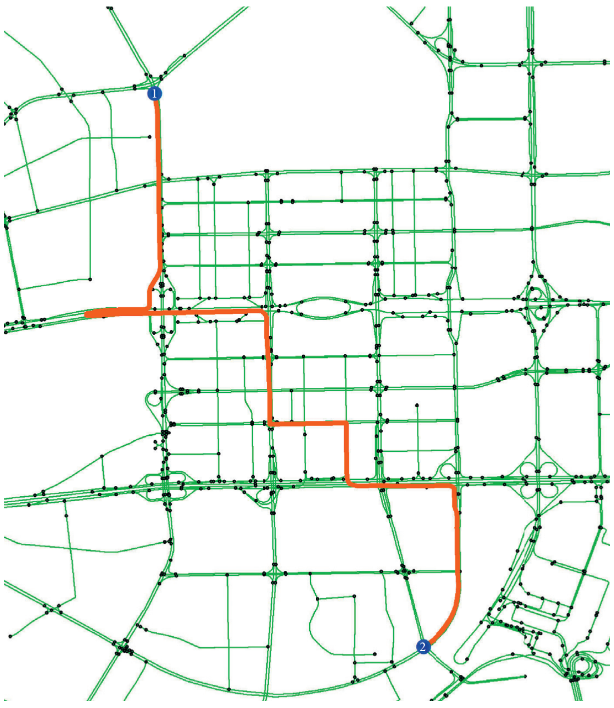
Figure 9: Continued.



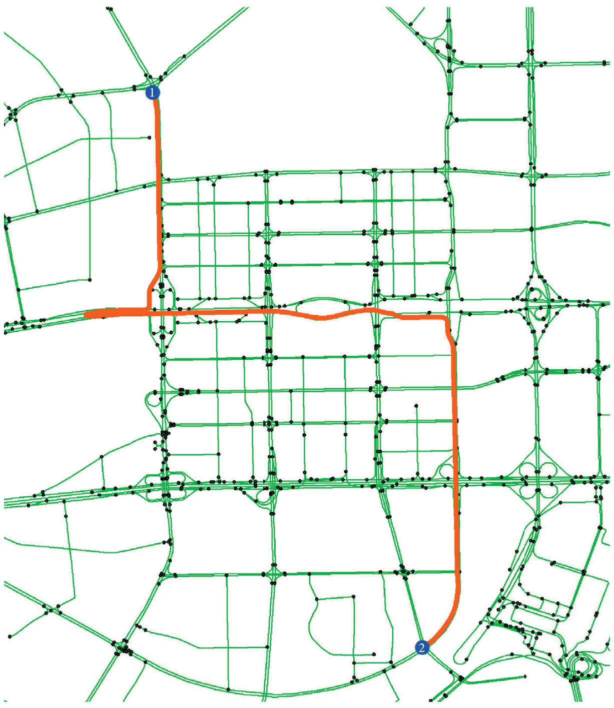
(e)



(f)



(g)



(h)

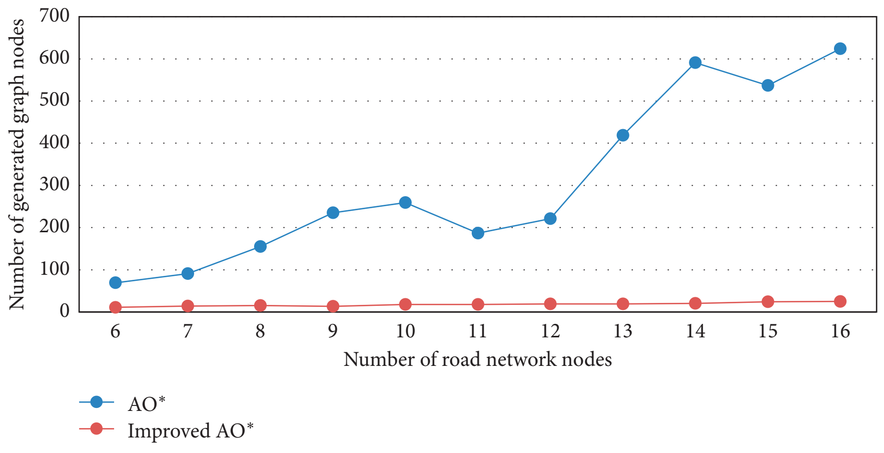
Figure 9: Path selection result: (a) at 07:00 (292 s), (b) 08:30 (352 s), (c) 09:30 (320 s), (d) 11:00 (340 s), (e) 12:30 (339 s), (f) 14:00 (318 s), (g) 18:00 (343 s), and (h) 20:30 (300 s).

Table 1: The result of the AO* algorithm search.

Number of nodes in an optimal path	Number of solution graph nodes	Number of generated graph nodes	Number of iterations from the starting node	Time for search (s)
6	33	69	17	0.7
7	45	91	26	1.1
8	63	155	29	2
9	97	235	59	4.2
10	100	259	51	4.3
11	96	187	49	4.7
12	78	221	51	8.1
13	145	419	47	17.1
14	226	591	78	34.1
15	220	537	81	49.5
16	253	624	89	51

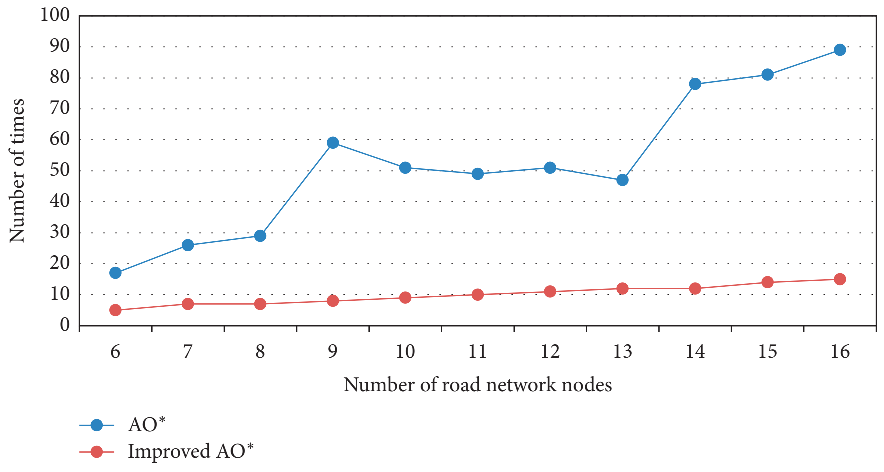
Table 2: The result from the improved AO* algorithm search.

Number of nodes in an optimal path	Number of solution graph nodes	Number of generated graph nodes	Number of iterations from the starting node	Time for search (s)
6	7	11	5	0.2
7	8	14	7	0.3
8	8	15	7	0.3
9	9	13	8	0.3
10	10	18	9	0.4
11	11	18	10	0.4
12	12	19	11	0.4
13	15	19	12	0.4
14	14	20	12	0.4
15	17	24	14	0.5
16	16	25	15	0.5

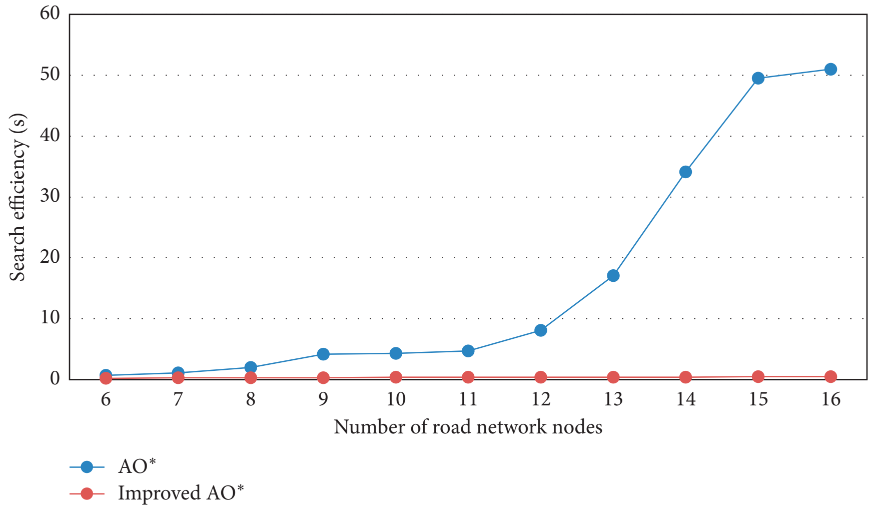


(a)

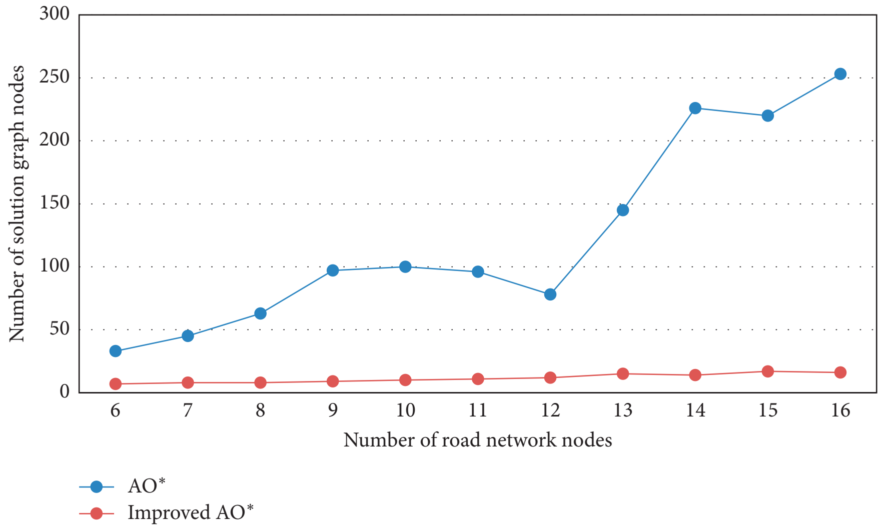
Figure 10: Continued.



(b)



(c)



(d)

Figure 10: The comparison of the efficiency of the original and improved AO* algorithm. (a) Comparison of the generated graph nodes. (b) Comparison of iteration from the starting node. (c) Comparison of running time. (d) Comparison of solution graph nodes.

6. Conclusion

The trip times in the segments on a route by a vehicle are estimated in a random process with randomness. Each node has a different probability of arriving at the target node in the shortest time according to the entry time. To calculate the trip time, a new algorithm of the optimal path selection was proposed and the following were investigated.

(1) The complexity of a random dynamic shortest path problem is necessary to consider. The variability of the trip time between nodes is caused by the insufficiency of FIFO to explain the complexity. The dynamic model of the shortest path selection considers the fact that the shortest trip time on a route is not equal to the sum of the trip times in the segments.

(2) The random shortest path selection is rather a decision-making problem. That is, at each segment, the choice of going straight, turning left, and turning right needs to be made according to the estimated shortest trip time. Stochastic dynamic programming is required to model for the solution of such a stochastic decision-making problem.

(3) The trip times between nodes on a route are not discrete, and that in the shortest path is the extreme value function with random variables. As the calculation of the probability density of the trip times is complicated, this study adopted Rosenblueth's two-point estimation method. The trip time as a random variable is discretized so that the expectation of the shortest trip time is unchanged.

(4) The probability density affects the "quantity" of the estimation of the trip times, not the "quality." The "quantity" influences the efficiency of an algorithm. The "quantity" does not mean the increasing number of nodes but possible successor nodes. The increase in the number of selections affects the efficiency of the estimation of the trip time significantly. Therefore, we proposed the improved AO* algorithm for enhancing the efficiency for finding the optimal trip time to apply the algorithm to the real data process. The result proves the superiority of the improved AO* algorithm and the basis for applying this to the real-time navigation system of a vehicle.

Data Availability

The data can be accessed at https://github.com/ricebow/Improved-AND-OR-Tree-Search. There are no restrictions on data access. The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the Fujian Province Natural Fund Project (Grant no. 2020J01263), Science and Technology Planning Foreign Cooperation Project of Longyan (Grant no. 2019LYF7003), Open Fund Project of Fujian University Engineering Research Center for Disaster Prevention and Mitigation of Southeast Coastal Engineering Structure of Putian University (Grant no. 2019005), Open Foundation Project of Fujian Provincial Key Laboratory of Higher Education (Putian University) (Grant no. ST19004), and High-Level Talents Program of Xiamen University of Technology (YKJ20010R).

References

[1] Q. Li and D. Li, "Big data GIS," *Geomatics and Information Science of Wuhan University*, vol. 39, no. 6, pp. 641–644, 2014.

[2] H. Lu, Z. Sun, and W. Qu, "Big data and its applications in urban intelligent transportation system," in *Proceedings of the 3rd International Conference on Intelligent Transportation, Big Data & Smart City(ICITBS 2018)*, Xiamen, China, January 2018.

[3] S. Wan, Y. Zhao, T. Wang, Z. Gu, Q. H. Abbasi, and K.-K. R. Choo, "Multi-dimensional data indexing and range query processing via Voronoi diagram for internet of things," *Future Generation Computer Systems*, vol. 91, pp. 382–391, 2019.

[4] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, "An edge traffic flow detection scheme based on deep learning in an intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1840–1852, 2021.

[5] Y. Luo, S. Shao, and Y. Zhang, "Location and path planning of mobile robots based on data fusion," *Journal of Computer Applications*, vol. 30, no. 11, pp. 3091–3093, 2010.

[6] W. Xu, J. Pan, J. Wei, and J. M. Dolan, "Motion planning under uncertainty for on-road autonomous driving," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2507–2512, Hong Kong, China, June 2014.

[7] Y. Rasekhipour, A. Khajepour, S.-K. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1255–1267, May 2017.

[8] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 285–292, Singapore, May 2017.

[9] T. Y. Abdalla, A. Abed, and A. A. Ahmed, "Mobile robot navigation using PSO-optimized fuzzy artificial potential field with fuzzy control," *Journal of Intelligent and Fuzzy Systems*, vol. 32, no. 6, pp. 3893–3908, 2017.

[10] S. M. Persson and I. Sharf, "Sampling-based A∗ algorithm for robot path-planning," *The International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683–1708, 2014.

[11] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 10, pp. 4813–4821, 2011.

[12] A. Lissovoi and C. Witt, "Runtime analysis of ant colony optimization on dynamic shortest path problems," *Theoretical Computer Science*, vol. 561, pp. 73–85, 2015.

[13] Q. Li, H. Li, Z. Xie, and D. Xu, “On the road trip time for dynamic route choice,” *Geomatics and Information Science of Wuhan University*, vol. 31, no. 6, pp. 519–522, 2006.

[14] R. W. Hall, “The fastest path through a network with random time-dependent travel times,” *Transportation Science*, vol. 20, no. 3, pp. 182–188, 1986.

[15] R. Bellman, “On a routing problem,” *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[16] J. Wu, S. Jin, H. Ji, and T. Srikanthan, “Algorithm for time-dependent shortest safe path on transportation networks,” *Procedia Computer Science*, vol. 4, no. 4, pp. 958–966, 2011.

[17] E. D. Miller-Hooks and H. S. Mahmassani, “Least expected time paths in stochastic, time-varying transportation networks,” *Transportation Science*, vol. 34, no. 2, pp. 198–215, 2000.

[18] S. Gao and I. Chabini, “Optimal routing policy problems in stochastic time-dependent networks,” *Transportation Research Part B: Methodological*, vol. 40, no. 2, pp. 93–122, 2006.

[19] X. Wu and Y. Nie, “Modeling heterogeneous risk-taking behavior in route choice: a stochastic dominance approach,” *Transportation Research Part A: Policy and Practice*, vol. 45, no. 9, pp. 896–915, 2011.

[20] A. Khani and S. D. Boyles, “An exact algorithm for the mean-standard deviation shortest path problem,” *Transportation Research Part B: Methodological*, vol. 81, no. 4, pp. 252–266, 2015.

[21] Y. Nie and X. Wu, “Shortest path problem considering on-time arrival probability,” *Transportation Research Part B: Methodological*, vol. 43, no. 6, pp. 597–613, 2009.

[22] M. Steinmetz, J. Hoffmann, and O. Buffet, “Goal probability analysis in probabilistic planning: exploring and enhancing the state of the art,” *Journal of Artificial Intelligence Research*, vol. 57, pp. 229–271, 2016.

[23] L. Fu and L. R. Rilett, “Expected shortest paths in dynamic and stochastic traffic networks,” *Transportation Research Part B: Methodological*, vol. 32, no. 7, pp. 499–516, 1998.

[24] L. Fu, “An adaptive routing algorithm for in-vehicle route guidance systems with real-time information,” *Transportation Research Part B: Methodological*, vol. 35, no. 8, pp. 749–765, 2001.

[25] K. Jeffrey P, “Stochastic analysis of link trip times-distributions, moments, and approximations,” Doctoral Dissertation, The Pennsylvania State University, State College, PA, US, 2001.

[26] Y. Zhao and T. Ono, “New point estimates for probability moments,” *Journal of Engineering Mechanics*, vol. 126, no. 4, 2000.

[27] J. Zietz, “Dynamic programming: an introduction by example,” *The Journal of Economic Education*, vol. 38, no. 2, pp. 165–186, 2007.

[28] J. L. Bander and C. C. White, “A heuristic search approach for a nonstationary stochastic shortest path problem with terminal cost,” *Transportation Science*, vol. 36, no. 2, pp. 218–230, 2002.

[29] N. Meuleau, E. Benazera, R. I. Brafman, E. A. Hansen, and Mausam, “A heuristic search approach to planning with continuous resources in stochastic domains,” *Journal of Artificial Intelligence Research*, vol. 34, pp. 27–59, 2009.