Hindawi
Scientific Programming
Volume 2021, Article ID 9589476, 13 pages
https://doi.org/10.1155/2021/9589476

Research Article

AGVs Route Planning Based on Region-Segmentation Dynamic Programming in Smart Road Network Systems

Zheng Zhang, Juan Chen, and Qing Guo

College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China

Correspondence should be addressed to Juan Chen; jchen@mail.buct.edu.cn

Received 10 August 2021; Accepted 15 September 2021; Published 22 October 2021

Academic Editor: Wenbing Zhao

In this paper, a route-planning approach is proposed based on the region-segmentation Dynamic Programming (DP) algorithm for Automated Guided Vehicles (AGVs) in large Smart Road Network Systems (SRNSs) to deal with the problem of low route computation efficiency of the classical DP algorithm. We introduced an improved Markov Decision Process (MDP) to describe SRNSs, in which the SRNSs are divided into several regions according to the AGVs' start nodes and their goal nodes to improve the route-planning efficiency. Moreover, the route with the minimum number of turns is selected to reduce the system running time and energy cost in the following way: first, all the equidistant shortest routes are acquired from the AGVs' start nodes to their goal nodes using the improved DP algorithm; then, the routes are screened by calculating the angular deviation between all feasible routes and AGVs' initial directions, and the route with the fewest number of turns is taken as the shortest-time route. The simulation results verified that the proposed method can effectively solve the route-planning problem of AGVs in current SRNSs.

1. Introduction

Smart Road Network Systems (SRNSs) are important parts of the Smart World. Automated Storage and Retrieval Systems (AS/RSs) and Container Terminal Systems are typical Industry 4.0 application scenarios. Automated Guided Vehicles (AGVs) are the main tools, which enable automatic access to goods transportation without human labor. With the development of Smart City and Internet of Things technology, the route-planning algorithms of AGVs in SRNSs are commonly used not only in logistics industries and smart factories but also in intelligent transportation systems [1], energy transmission systems [2], and even network planning [3]. With the wide application of AGVs route planning, the goal of route planning is not only to find the route with the shortest distance from AGVs' start nodes to their goal nodes but also to minimize the system operating time and reduce energy cost. Moreover, the speed of the route-planning algorithm is also important. Therefore, AGVs route-planning problem in SRNSs has been widely studied.

The AGVs route planning in SRNSs belongs to the single-source shortest routing problem, and it has similarities with trajectory optimization. They both can be solved using graph search algorithms [4, 5], sampling-based methods [6], and intelligent algorithms [7]. Dijkstra's algorithm is a classical route search algorithm, which has a simple structure and is robust and easy to implement, and it can meet requirements in practical scenarios. However, Dijkstra's algorithm belongs to Breadth-First Search (BFS), and thus it is not suitable for applying in scenarios with high real-time requirements [8–10]. In addition, when there are multiple equidistant shortest routes between the start node and the goal node, Dijkstra's algorithm can find only one of them. Even though there is improved Dijkstra's algorithm [11] that can obtain all equidistant shortest routes through iterations, the time complexity of Dijkstra's algorithm is $O(n^2)$, where n indicates the number of nodes in SRNSs. Therefore, it is not suitable for route planning in large SRNSs. AGVs interact with SRNSs all the time and an AGV has state s_t at each time t. An AGV takes action a on state s_t, and then it transforms to a new state s_{t+1} by taking reward r_{t+1}, and a sequence (i.e., s_{t-1}, a_{t-1}, r_t, s_t, a_t, r_{t+1}...) is built by repeating the operation. The process of computing the shortest routes for AGVs is called the sequence decision process, and Markov Decision Process (MDP) is a typically

formulaic method for it. SRNSs described in this paper consist of several regularly arranged nodes, and the workflow can be described using MDP. However, we cannot completely copy the classical MDP to model the environment of the characteristics of different SRNSs.

Dynamic Programming (DP) algorithm is a branch of operations research, which was proposed by Bellman et al. [12] in the 1950s. DP algorithm is a method for solving optimization problems of the multistep decision process. DP can transform multistage problems into a series of single-stage problems [13], and it is suitable for solving optimization problems in large-scale environments. DP algorithm is the basic of Reinforcement Learning (RL) [14], and RL is an unsupervised algorithm based on the principles of reward and punishment. However, it is suitable for solving model-free problems, an AGV does not know which state has the bigger reward, and it needs to find the optimal policy to the goal state through trial and error [15]. Moreover, the classical DP algorithm iterates the sample set by randomly selecting starting point until the value function approaches the optimal policy. With the expansion of the scales of route-planning environments, the training time can become larger, and the computation amount of the DP algorithm will increase exponentially with the increasing number of nodes [16].

With the development of new communication technologies (e.g., 5G) and the increase of nodes' number in SRNSs, the real-time requirements for AGVs route-planning algorithm are becoming higher, and the classical DP algorithm is not suitable for the current development of Smart World. We propose a route-planning approach based on the region-segmentation DP algorithm for AGVs in SRNSs to address the low-efficiency problem of the classical DP algorithm. Because the classical MDP fails to accurately model the current SRNSs, we propose an improved MDP model for SRNSs. Since the number of samples is one of the important factors that affect the efficiency of the algorithm, a region-segmentation-based DP algorithm is proposed. Firstly, SRNSs are divided into several regions according to AGVs' start nodes and their goal nodes, and the objective is to reduce the number of training samples. Then, each node is assigned a value, and the DP algorithm is used to compute the value function of each region, and the objective is to find all equidistant shortest routes from the AGVs' start nodes to their goal nodes. Because AGVs can take more time and energy to turn, we introduce the following steps to determine routes with the least turns: (a) screening routes by calculating the angle deviation between all candidate routes and an AGV's initial direction and (b) choosing a route among remaining routes with the fewest number of turns and considering it the optimal route for the AGV. Moreover, we design different route-planning strategies according to whether an AGV is loaded or not, which improves the flexibility and efficiency of SRNSs, and the more nodes in the SRNSs, the more efficient our approach.

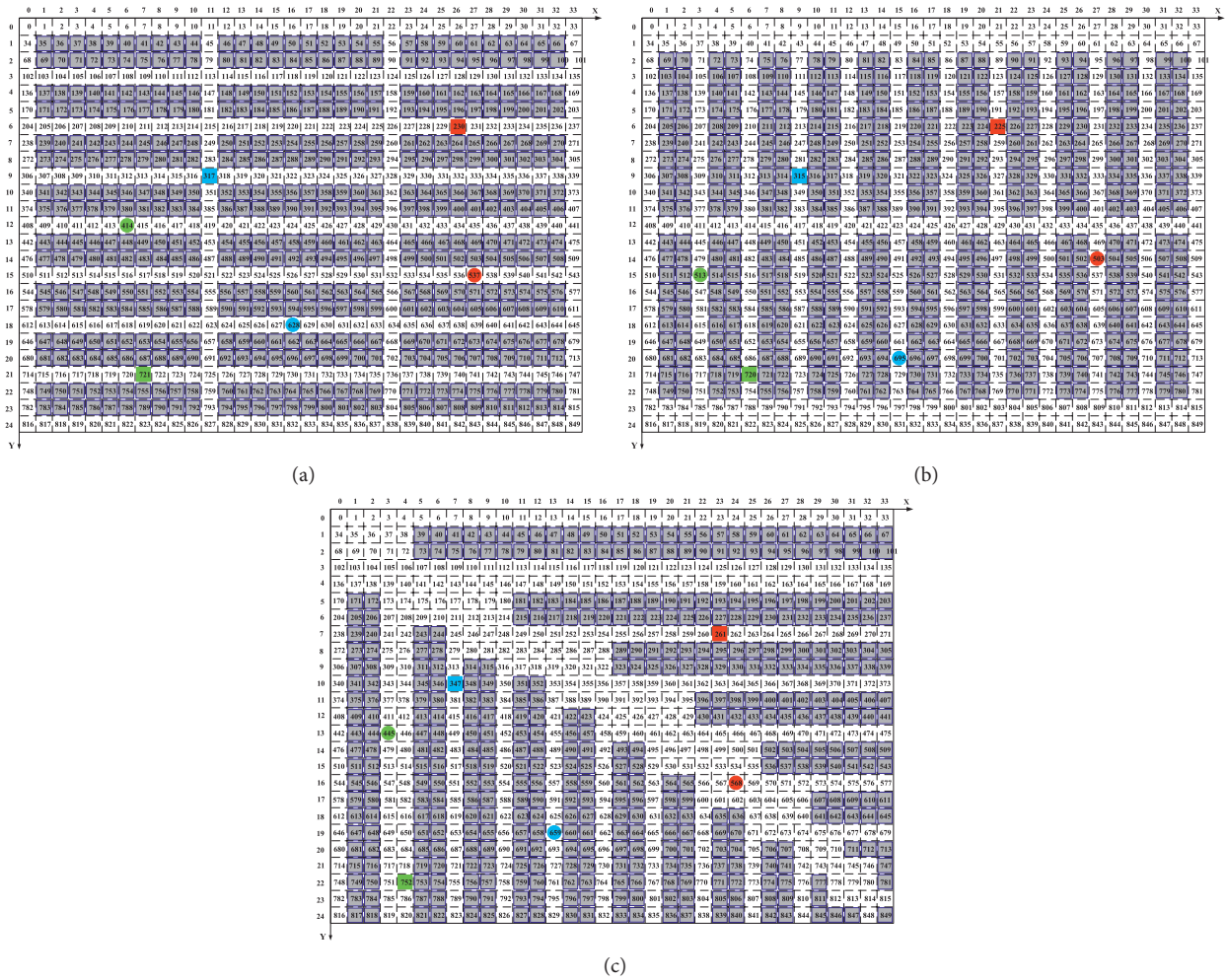
2. Related Works

The AGVs route-planning problem can be solved using Dijkstra's algorithm [17], A* algorithm [18], D* algorithm [19], Probabilistic Roadmap (PRM) [20], Rapid Random Tree (RRT) [21], Artificial Potential Field (APF) [22], neural network [23], genetic algorithm [24], and ant colony algorithm [25]. Dijkstra's algorithm is a classical shortest route search algorithm, and it is simple and stable for performance. Guo et al. [11] improved Dijkstra's algorithm and found the shortest routes with the least distance and time consumption for AGVs in AS/RSs. Li et al. [26] assumed that all obstacles were convex and found the shortest routes for AGVs without collision. However, the above achievements were all obtained in small-scale environments without considering the case of large-scale SRNSs (i.e., a large number of nodes and large size of the space). With the rapid development of SRNSs, the calculation amount of AGVs route planning will increase exponentially. But classical Dijkstra's algorithm does not consider the scenarios with a large number of nodes, and the system may be locked due to too much computation.

Yuan et al. [27] proposed Approximate Dynamic Programming (ADP), and they applied it to crane scheduling to reduce the operation time of the AS/RSs. Novoa et al. [28] focused on Vehicle Routing Problem (VRP) from the perspective of ADP, and they proved that a DP-based algorithm could be applied to vehicle routing planning. Cimen et al. [29] proposed a vehicle routing optimization algorithm by combining ADP and MDP. Bahlawan et al. [30] introduced a method of system operation management and energy optimization based on the DP algorithm to minimize the energy cost of factories. Horiguchi et al. [31] focused on the routing problem of network packets, and they redefined the energy function by combining the DP algorithm and neural network. They proposed a method to find the optimal balance between queues' length and the routes' distance. Although many achievements have proved that the DP algorithm can be used to solve optimization problems in various situations, the classical DP algorithm needs to iterate the values of each node in the whole space many times until a stable and optimal policy is obtained. To reduce the computing time of DP, Ulmer et al. [32] proposed an offline value function prediction method, which introduced MDP into state space and combined action space with reward information. Desai et al. [33] proposed a preprocessing method of a random dynamic network according to vehicles' starting time and location. But they failed to fundamentally solve the problem of longer DP algorithm iteration time.

3. Rasterizing the SRNS as the Model

There are a large number of nodes arranged according to the regular form of rows and columns in SRNSs and nodes and the lines between nodes (i.e., the driving paths of AGVs) are typically presented as a regular rectangle. Therefore, SRNSs can be rasterized, and the grids' locations can be stored with spatial coordinates. Figures 1(a)–1(c) are $m \times n$ SRNSs constructed using grid methods [34], where m and n represent the numbers of rows and columns of the SRNSs, respectively. We establish the Cartesian coordinate system of the SRNSs: the upper-left corner is selected as the origin $(x = 0, y = 0)$, the lower-



(a) (b) (c)

Figure 1: Rasterize large SRNSs as a model. (a) Horizontal row SRNSs; (b) vertical column SRNSs; (c) fish-bone SRNSs.

right corner defines the workspace size $(x = n - 1, y = m - 1)$, and each node corresponds to a coordinate (a, b) $(a \le n - 1,\ b \le m - 1)$. Grey grids represent obstacle nodes, white grids indicate the nodes that are available for AGVs, and numbers in grids indicate the number of nodes (i.e., the number of nodes starting from $(x = 0, y = 0)$ and marked as 1, 2, 3, . . ., n). Red, green, and blue solid squares represent the start nodes of AGVs, and solid circles of the corresponding color represent their goal nodes. The distance between every two adjacent nodes is equal and the SRNS is bidirectional (i.e., AGVs can travel bidirectionally), and each node can only be occupied by one AGV. The size of every obstacle is the same in SRNSs and the number of grids AGVs passed represents the travel distance for AGVs.

Figures 1(a) and 1(b) are commonly used SRNSs (e.g., AS/RSs), in which the shelves are arranged according to the rules of row and column, and there is a channel on both sides of each row (or column) of shelf for AGVs access. Figure 1(c) shows recently presented SRNSs due to the rapid development of the Smart World. In these fish-bone SRNSs, shelves are arranged on both sides of the main diagonal of the SRNSs, since it is inconvenient for AGV access and the signal may be blocked, and it can lead to system deadlock of SRNSs; this environment model has no practical application.

3.1. General Markov Decision Process Framework. Markov Decision Process (MDP) is a quintuple (i.e., $\langle S, A, P, R, \gamma \rangle$), where S is the finite states set, A is the actions set, and P is the state transition probability matrix. Given the current state s and the next state s', the state transition probability (i.e., $P_{ss'} = P[S_{t+1} = s' | S_t = s]$) represents the transition probability from state s to state s'. R is the rewards set; γ is the discount factor, which denotes that the influence of the goal state on each state is gradually weakened with the increase of distance between a node and the goal node, $\gamma \in [0, 1]$. There are two kinds of rewards of R set: (1) immediate reward R_t, which denotes the expectation of rewards from state s to state s', that is, $R_t = E[R_{t+1} | S_t = s]$, and (2) accumulated reward G_t, which denotes the weighted sum of each immediate reward from the start state to the goal state.

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \quad (1)$$

3.2. The Markov Decision Process in SRNSs. State value function $V(s)$ represents the expectation of accumulated reward under state s:

$$V(s) = E[G_t|S_t = s]. \tag{2}$$

Bellman equation [12] is the core of all MDP-based algorithms. AGVs can obtain the best route from start nodes to goal nodes according to Bellman equation. The derivation process of the Bellman equation is as follows:

$$\begin{aligned} V(s) &= E[G_t|S_t = s] \\ &= E\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots |S_t = s\right] \\ &= E\left[R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} \ldots\right)|S_t = s\right] \\ &= E\left[R_{t+1} + \gamma G_{t+1}|S_t = s\right] \\ &= E\left[R_{t+1} + \gamma V\left(S_{t+1}\right)|S_t = s\right]. \end{aligned} \tag{3}$$

In equation (3), the value of state s consists of (i) the expectation of immediate reward and (ii) the expectation of the state's value for the next moment. State-policy transition probability $P^{\pi}_{s,s'} = \sum_{a\in A}\pi(a|s)P^{a}_{s,s'}$ represents the probability of an AGV transforming from state s to state s'.

3.2.1. Smart Road Network System-Reward Matrix (SRNS-RM). AGVs can obtain different immediate rewards at different nodes, and the reward/penalty obtained by AGVs is defined as follows: (1) if AGVs enter grey solid grids (i.e., obstacle nodes), they will get penalty: $-r$, where r is a positive integer; (2) if AGVs enter white grids, they will get penalty: $-\varepsilon$, where ε is a positive integer and $\varepsilon \ll r$; (3) if AGVs reach the goal node, they can obtain reward $+ r^2$. Given an $m \times n$ SRNS, where m indicates the number of rows, that is, the number of nodes in a column, and n indicates the number of columns, that is, the number of nodes in a row, the SRNS-RM is

$$\text{SRNS} - \text{RM} = \begin{bmatrix} r_{11} = -\varepsilon & r_{12} = -\varepsilon & r_{13} = -r & \ldots & r_{1n} = -\varepsilon \\ r_{21} = -\varepsilon & r_{22} = -r & r_{23} - r & \ldots & r_{2n} = -\varepsilon \\ r_{31} = -\varepsilon & r_{32} = -\varepsilon & r_{33} = -r & \ldots & r_{3n} = -\varepsilon \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ r_{m1} = -\varepsilon & r_{m2} = -\varepsilon & r_{m3} = -\varepsilon & \ldots & r_{mn} = +r^2 \end{bmatrix}. \tag{4}$$

In equation (4), the dimension of SRNS-RM is $m \times n$, which is the same as the SRNS. The elements (i.e., r_{ij}) in SRNS-RM ($i = 1, 2 \ldots m$; $j = 1, 2 \ldots n$) indicate the rewards of an AGV reaching nodes in row i and column j. Consider that there are four obstacle nodes in the SRNS, that is, *SRNS-RM*$_{13} = -r$, *SRNS-RM*$_{22} = -r$, *SRNS-RM*$_{23} = -r$, and *SRNS-*RM33 $= -r$.

3.2.2. Smart Road Network System-Transition Probability Matrix (SRNS-TPM). The adjacencies of nodes in SRNSs are as follows: (1) there are no upper adjacent nodes for the nodes on the upper edge but only left, lower, and right adjacent nodes; (2) there are no left adjacent nodes for the node on the left edge but only top, right, and bottom adjacent nodes; (3) there are no lower adjacent nodes for the node on the lower edge but only upper, left, and right adjacent nodes; (4) there are no right adjacent nodes for the node on the right edge but only top, left, and bottom adjacent nodes; (5) there are four adjacent nodes except for the nodes on edges. In principle, the probabilities are the same for an AGV entering any adjacent node in an $m \times n$ SRNS, that is, $P_{ij} = P[S_{t+1} = j|S_t = i] = \delta$, where $0 \le \delta \le 1$. The SRNS-TPM is

$$\text{SRNS} - \text{TPM} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \ldots & P_{1n} \\ P_{21} & P_{22} & P_{23} & \ldots & P_{2n} \\ P_{31} & P_{32} & P_{33} & \ldots & P_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ P_{m1} & P_{m2} & P_{m3} & \ldots & P_{mn} \end{bmatrix}. \tag{5}$$

In equation (5), P_{ij} indicates each element of SRNS-TPM, where i indicates AGV's current state (i.e., S_t), and j indicates its next state (i.e., S_{t+1}). The dimension of SRNS-TPM is $m \times n$, which is the same as the SRNS and $P_{11} = P_{12} = \cdots = P_{mn} = \delta$; that is, the probability of AGV entering each adjacent node is the same.

3.2.3. Criteria of the Optimal Policy Selection. The objective of the classical MDP is to find the optimal policy that can reach the goal to maximize the accumulated reward G_t (see equation (2)). However, AGVs route planning in SRNSs is a multiobjective combinatorial optimization problem. On the one hand, the total routes' length should be minimized; on the other hand, total routes' length should be minimized, and AGVs' traveling time should be the least. Therefore, the criteria for choosing the optimal policy need to be modified.

When an AGV enters a node, it uploads coordinates of the current node to the server through 5G, Bluetooth, or other wireless technologies, so that the server can know the location of the AGV in SRNSs. We ignore the size of nodes in SRNSs, and consider that the distance between every two adjacent nodes is L_n; the longitudinal length of the AGV is L_c; and the traveling speed of the AGV is fixed as v. When the head of the AGV enters a node, the node is occupied, and the node is released when the tail of the AGV leaves. As shown in Figure 2, the time for the AGV to pass through every two adjacent nodes is $t_c = L_n + L_c/v$.

In many SRNSs (e.g., AS/RSs), an AGV can only travel in four directions, north, east, south, and west, and can only turn at nodes. It requires deceleration, stopping, autobiography, and acceleration, and the total time for turning is t_u. The route length from the start node to the goal node is shown in equation (6). Given that the starting time of an AGV is 0 s, the total time for the AGV traveling to the goal node is shown in equation (7).

$$L = (n_c - 1) \times L_n, \tag{6}$$

$$T = (n_c - 1) \times t_c + n_u \times t_u. \tag{7}$$

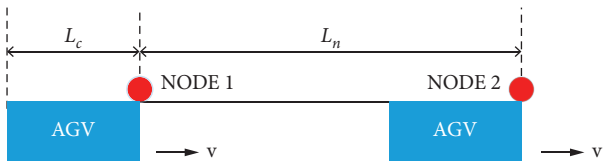


Figure 2: Time of AGVs passing through every two adjacent nodes.

In equations (6) and (7), n_c denotes the total number of nodes in the route and n_u denotes the number of turns in the route.

Given that the computing time of the server is t_s, the criteria of the optimal policy selection in SRNSs are as follows:

$$\min \quad {}_{\pi}t_s, \tag{8}$$

$$\min \quad {}_{\pi}L = (n_c - 1) \times L_n, \tag{9}$$

$$\min \quad {}_{\pi}T = (n_c - 1) \times t_c + n_u \times t_u, \tag{10}$$

$$\text{s.t.} \quad \begin{cases} t_s > 0 \\ n_c > 1 \; an \; d \; L_n > 0 \\ 0 \le n_u \le n_p \\ t_u > 0 \end{cases} n_p = 1, 2, \ldots, n. \tag{11}$$

Equation (8) indicates that we should reduce route computing time of the server; equation (9) indicates that we should shorten routes' distance of AGVs; equation (10) indicates that we should minimize the total travel time of AGVs. In equation (11), n_p denotes the number of nodes in an AGV's route. Due to the fewer turns in AGVs' routes, with less energy cost of the SRNSs, reducing the number of turns in AGVs routes can reduce the energy cost.

3.3. AGVs Route-Planning Approach in SRNSs. The flowchart of AGVs route-planning approach proposed in this paper is shown in Figure 3, which consists of four parts: (1) route-planning region segmentation; (2) computing all shortest routes by training value functions of the regions; (3) obtaining the feasible routes according to AGVs' status; and (4) selecting collision-free routes with the shortest length and the least travel time.

3.3.1. Route-Planning Region Segmentation. The most effective way to speed up training is to reduce the number of samples. We here proposed a region-segmentation-based DP algorithm, in which the SRNSs are divided into regions according to AGVs' start nodes and their goal nodes. The objective is to accelerate the convergence of the DP algorithm by excluding irrelevant nodes. Given the coordinates of an AGV's start node (X_s, Y_s) and the goal node (X_g, Y_g), the left boundary of an SRNS X_l and the right boundary X_r, the upper left coordinates of the AGV's route-planning region $(X^{ul}_{AGV_i}, Y^{ul}_{AGV_i})$, the upper-right coordinates $(X^{ur}_{AGV_i}, Y^{ur}_{AGV_i})$, the coordinates of the lower-left corner $(X^{dl}_{AGV_i}, Y^{dl}_{AGV_i})$, the coordinates of the lower right corner $(X^{dr}_{AGV_i}, Y^{dr}_{AGV_i})$, the length of the region along the x-direction L_{row}, and the length along the y-direction L_{col}, the AGVs' route-planning region is defined as follows:

(i) If $X_g - X_l < X_r - X_g$, that is, an AGV's goal node is closer to the left boundary of the SRNS, then

$$\left(X^{ul}_{AGV_i}, Y^{ul}_{AGV_i}\right) = \left(X_l, \min\left(Y_s, Y_g\right)\right), \tag{12}$$

$$\left(X^{ur}_{AGV_i}, Y^{ur}_{AGV_i}\right) = \left(\max\left(X_s, X_g\right), \min\left(Y_s, Y_g\right)\right), \tag{13}$$

$$\left(X^{dl}_{AGV_i}, Y^{dl}_{AGV_i}\right) = \left(X_l, \max\left(Y_s, Y_g\right)\right), \tag{14}$$

$$\left(X^{dr}_{AGV_i}, Y^{dr}_{AGV_i}\right) = \left(\max\left(X_s, X_g\right), \max\left(Y_s, Y_g\right)\right), \tag{15}$$

$$L_{row} = \max\left(X_s, X_g\right) - X_l, \tag{16}$$

$$L_{col} = \left|Y_g - Y_s\right|. \tag{17}$$

(ii) If $X_g - X_l > X_r - X_g$, that is, an AGV's goal node is closer to the right boundary, then

$$\left(X^{ul}_{AGV_i}, Y^{ul}_{AGV_i}\right) = \left(\min\left(X_s, X_g\right), \min\left(Y_s, Y_g\right)\right), \tag{18}$$

$$\left(X^{ur}_{AGV_i}, Y^{ur}_{AGV_i}\right) = \left(\min\left(X_s, X_g\right), \max\left(Y_s, Y_g\right)\right), \tag{19}$$

$$\left(X^{dl}_{AGV_i}, Y^{dl}_{AGV_i}\right) = \left(X_r, \min\left(Y_s, Y_g\right)\right), \tag{20}$$

$$\left(X^{dr}_{AGV_i}, Y^{dr}_{AGV_i}\right) = \left(X_r, \max\left(Y_s, Y_g\right)\right), \tag{21}$$

$$L_{row} = X_r - \max\left(X_s, X_g\right), \tag{22}$$

$$L_{col} = \left|Y_g - Y_s\right|. \tag{23}$$

The conventional region-segmentation methods only focus on locations of AGVs' start nodes and their goal nodes, but we take the relationship between them and the left (and right) boundaries of SRNSs into consideration. Our approach can reduce the number of samples without losing the accuracy of route planning. As shown in Figure 4, given that the start node of an AGV is node 43 $(x = 3, y = 5)$ and the goal node is node 20 $(x = 4, y = 2)$, the blue line indicates the feasible route from the start node to the goal node; the yellow shaded area represents the route-planning region obtained using the conventional region-segmentation methods, which does not include the feasible route, and the area in the red box indicates the region in our approach, which contains the feasible route.

3.3.2. Training Value Function of the Route-Planning Region. There are two kinds of commonly used DP algorithms: policy iteration-based DP algorithm and value iteration-based DP algorithm. The time of computing route using the DP algorithm based on policy iteration and value iteration in

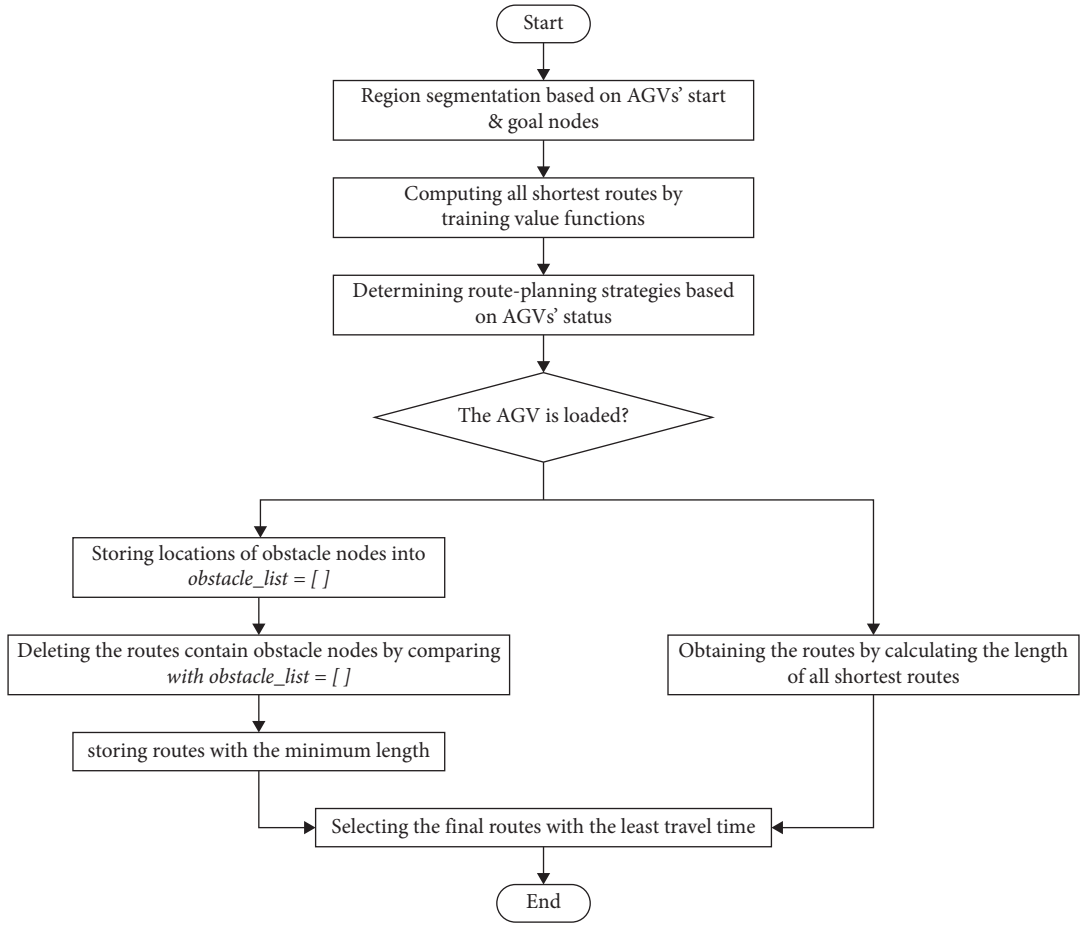


Figure 3: Flowchart of the AGVs route-planning approach proposed in this paper.

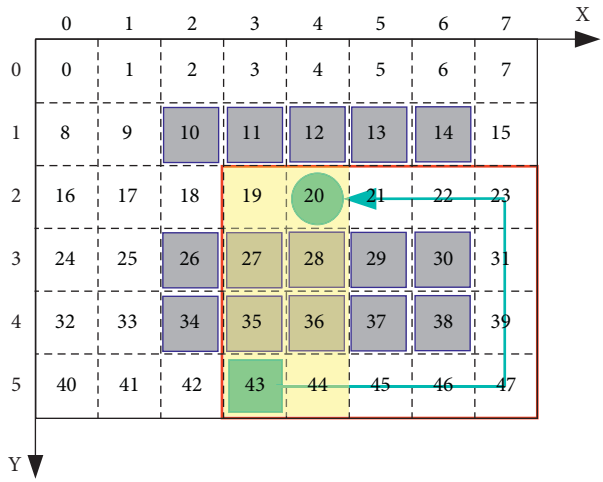


Figure 4: Conventional and improved region segmentation.

an $m \times n$ SRNS (see Figure 1(a)) is shown in Figure 5, where the blue line denotes the training time of the DP algorithm based on policy iteration, and the red line denotes that of the DP algorithm based on value iteration.

As can be seen from Figure 5, the training time of the DP algorithm based on value iteration is smaller than that of the DP algorithm based on policy iteration. In this scenario, the average time for route computing is improved by nearly **97.39%**, and when the scales of SRNSs become larger, the difference of computing time between them becomes bigger. We employ the value iteration-based DP algorithm because it can adjust the policy while training rather than after the value function converges, which greatly reduces the number of iterations and significantly improves the efficiency of the algorithm.

The calculation of the optimal value function (i.e., $V^*(s)$) is a recursive process (refer to equation (3)). We need to evaluate the rewards of all adjacent states based on the current state, and the state with the largest reward is taken as the next state of the AGV. The method of computing the relationship between the current node and all adjacent nodes is

$$V^{T+1}(s) = \sum_{s+1} p(s_{t+1}|s_t, a_t)\left[r_{a_t}^{s_{t+1}} + \gamma \times V^T(s_{t+1})\right], \quad (24)$$

where $p(s_{t+1}|s_t, a_t)$ indicates the probability of taking action a_t to transfer to an adjacent state s_{t+1} based on the current state s_t; $r_{a_t}^{s_{t+1}}$ is the reward of taking action a_t to transfer to state s_{t+1}; γ is the discount factor, where $\gamma \in [0, 1]$; $V^T(s_{t+1})$ is the value of each state adjacent to state s_t.

Without considering obstacle nodes in SRNSs, the probability is the same for an AGV traveling to surrounding nodes; that is, $p(s_{t+1}|s_t, a_t) = 1/4$ and $\gamma = 1$. Equation (24) can be rewritten as

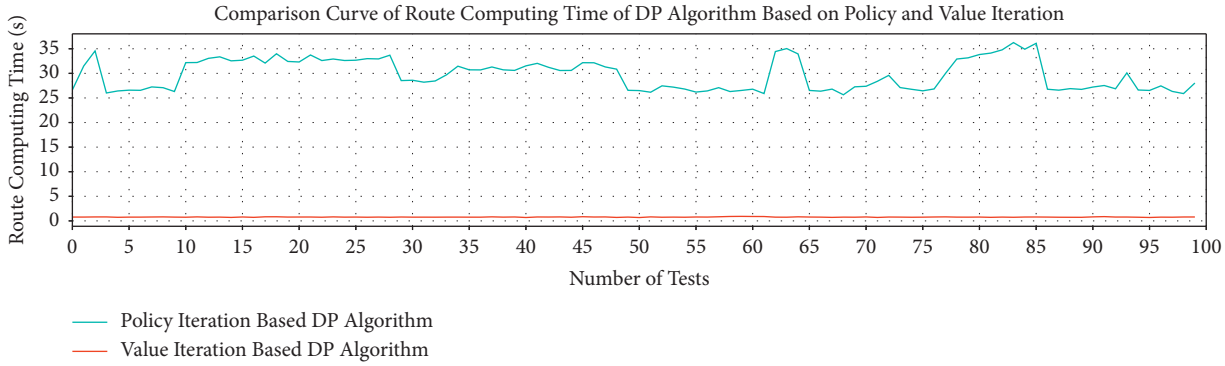


Figure 5: Curve of convergences between the DP algorithms based on policy iteration and value iteration.

$$V^{T+1} = \frac{1}{4} \times \left[\left(-1 + 1 \times V_u^T\right) + \left(-1 + 1 \times V_d^T\right) + \left(-1 + 1 \times V_l^T\right) + \left(-1 + 1 \times V_r^T\right)\right]. \quad (25)$$

In equation (25), V_u^T, V_d^T, V_l^T, and V_r^T denote the state values of upper, lower, left, and right adjacent nodes of AGV's current nodes, respectively.

3.3.3. Route-Planning Strategies Based on Status of AGVs. There are multiple obstacles on AGVs' routes, and the route-planning methods of AGVs in different status can be different. Taking AS/RSs as an example [35], as shown in Figure 6, the height of the bottom shelf from the ground is 75 cm, and the height of each no-load AGV is 50 cm. If an AGV needs to load, the tray on the AGV will be lifted to hold the bottom to lift the shelf. Because the height from the bottom shelf to the ground is greater than the height of a no-load AGV, a no-load AGV can pass under the shelve, whereas a loaded AGV (i.e., an AGV that is carrying a shelve) can not. So the loaded AGVs should avoid collision with not only other AGVs but also shelves placed above the nodes.

If AGVs are loaded, the route-planning method is as follows: (1) storing locations of obstacle nodes into a list *obstacle_list* = *[ ]* while initializing the environment; (2) repeating equation (25) to compute the value function of the route-planning regions and obtaining all feasible routes from AGVs' start nodes to their goal nodes; (3) comparing nodes in all feasible routes with nodes in *obstacle_list* = *[ ]* and deleting the routes that contain obstacle nodes; and (4) calculating the length of each collision-free route and storing routes with the minimum length. If AGVs are no-load ones, it is not necessary to establish *obstacle_list* = *[ ]* while initializing the environment, and we need not compare feasible routes to *obstacle_list* = *[ ]* but only to calculate the length of all feasible routes.

3.3.4. Obtaining Collision-Free Routes with the Shortest Length and the Least Travel Time. SRNSs are shaped like chessboards, and AGVs can travel in four directions: north, east, south, and west. Therefore, there may be more than one shortest-distance route from AGVs' start nodes to their goal nodes. In some large SRNSs (e.g., AS/RSs), AGVs can only turn at nodes, and the time of the process becomes longer with the increase of AGVs' turning angles. The fewer angles, the smaller the time cost and the higher the system efficiency. We can find routes with the shortest travel time for AGVs through the following steps:

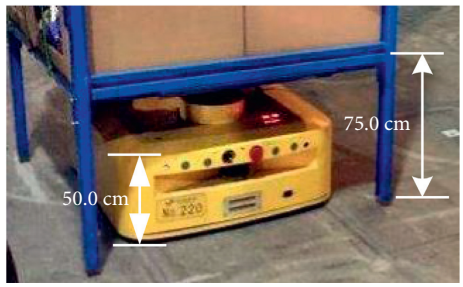


Figure 6: Height of bottom shelf and no-load AGV in AS/RSs.

Step 1. Comparing the angles of each feasible route with AGVs' initial directions to determine their departure routes:

$$\theta_{init} = \min\left(\left|\theta_1^{start} - \theta_{AGV}^{start}\right|, \left|\theta_2^{start} - \theta_{AGV}^{start}\right|, \left|\theta_3^{start} - \theta_{AGV}^{start}\right|, \left|\theta_4^{start} - \theta_{AGV}^{start}\right|\right). \quad (26)$$

In equation (26), θ_{init} indicates the departure angle of AGVs from start nodes. We take the southward as 0° and start nodes as the origin; θ_1^{start} indicates the angle of route 1, θ_2^{start} indicates the angle of route 2, θ_3^{start} indicates the angle of route 3, and θ_4^{start} indicates the angle of route 4; θ_{AGV}^{start} indicates the angle of AGVs at start nodes.

Step 2. Computing the routes with the shortest travel time, that is, the routes with the fewest turns: traversing each node of feasible routes obtained in Step 1, deleting routes that contain nodes in *obstacle_list* = *[ ]*, and then judging whether a node is a turn or not in remaining routes. The method of determining whether a node is a turn or not is as follows: considering the X and Y coordinates of upstream and downstream nodes, if a node satisfies equation (27), it is a turn:

$$\left|X_{i+1} - X_i\right| \neq \left|X_i - X_{i-1}\right| \text{or} \left|Y_{i+1} - Y_i\right| \neq \left|Y_i - Y_{i-1}\right|. \quad (27)$$

In equation (27), i denotes the number of the current node, X_i is the X-coordinate of the current node, and Y_i is the Y-coordinate; $i+1$ denotes the number of the downstream node, X_{i+1} is the X-coordinate, and Y_{i+1} is the Y-coordinate; $i-1$ denotes the number of the upstream node, X_{i-1} is the X-coordinate, and Y_{i-1} is the Y-coordinate.

4. Simulation Case Studies

Simulation OS is *Windows 10*, *Intel Xeon W-2145 CPU @ 3.70 GHz x64-based processor*; programming software is *Python 3.7.3*; rasterized AS/RSs environment parameters are *25×34 Horizontal Row*, with 850 nodes in total, $L_c = 2.5\ m$ (note: the proposed algorithm is suitable for not only AS/RSs but also all SRNSs); $v = 0.5\ m/s$, and the times for AGVs traveling between two adjacent nodes (i.e., t_c) and turning at nodes (i.e., t_u) are both *5 s*.

Case 1. Route planning for an AGV in AS/RSs from a designated start node to a designated goal node. The numbers of obstacle nodes of Figure 7 are as follows: *obstacle_list* = [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, . . ., 805, 806, 807, 808, 809, 810, 811, 812, 813, 814]. The start node of an AGV is node 56 *(x = 22, y = 1)*, and the goal node is node 639 *(x = 27, y = 18)*. The process of route planning is as follows.

4.1. Determining the Route-Planning Region for the AGV. Based on the start node and the goal node of every AGV and the AS/RS's boundary, we use equations (12)–(17) (or equations (18)–(23)) to obtain the region-segmentation result, as shown in the blue shaded part in Figure 7.

4.2. Modeling the AS/RS Using the Improved MDP. According to the definition in equations (16) and (17) (or equations (22) and (23)), we can obtain the following: $L_{row} = 12$, $L_{\text{col}} = 18$, $r = 10$, and $\varepsilon = 1$, and the probability of AGV traveling towards the node adjacent to the current node (i.e., state transition) $p = 1/4$. Based on equations (4) and (5), SRNS-RM and SRNS-TPM are as follows:

$$PBS - RM = \begin{bmatrix} r_{1,1} = -1 & r_{1,2} = -10 & r_{1,3} = -10 & \cdots & r_{1,12} = -1 \\ r_{2,1} = -1 & r_{2,2} = -10 & r_{2,3} = -10 & \cdots & r_{2,12} = -1 \\ r_{3,1} = -1 & r_{3,2} = -1 & r_{3,3} = -1 & \cdots & r_{3,12} = -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{18,1} = -1 & r_{18,2} = -1 & r_{18,3} = -1 & \cdots & r_{18,12} = +100 \end{bmatrix}, PBS - TPM$$
$$= \begin{bmatrix} P_{1,1} = \frac{1}{4} & P_{1,2} = \frac{1}{4} & P_{1,3} = \frac{1}{4} & \cdots & P_{1,12} = \frac{1}{4} \\ P_{2,1} = \frac{1}{4} & P_{2,2} = \frac{1}{4} & P_{2,2} = \frac{1}{4} & P_{2,3} = \frac{1}{4} & P_{2,12} = \frac{1}{4} \\ P_{3,1} = \frac{1}{4} & P_{3,2} = \frac{1}{4} & P_{3,2} = \frac{1}{4} & P_{3,3} = \frac{1}{4} & P_{3,12} = \frac{1}{4} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ P_{18,1} = \frac{1}{4} & P_{18,2} = \frac{1}{4} & P_{18,2} = \frac{1}{4} & P_{18,3} = \frac{1}{4} & P_{18,12} = \frac{1}{4} \end{bmatrix}. \quad (28)$$

4.3. Obtaining All the Equidistant Shortest Routes. By using the value iteration-based DP algorithm (see equation (25)), we can obtain all the feasible routes from AGV's start node to its goal node, as shown in Figures 8(a)–8(f).

4.4. Selecting a Route with the Least Travel Time. The route of an AGV in SRNS can be presented as *S(T)*, where *S* indicates the set of nodes' numbers in AGV's route, and *T* indicates the set of time when the AGV passes the nodes. We can choose

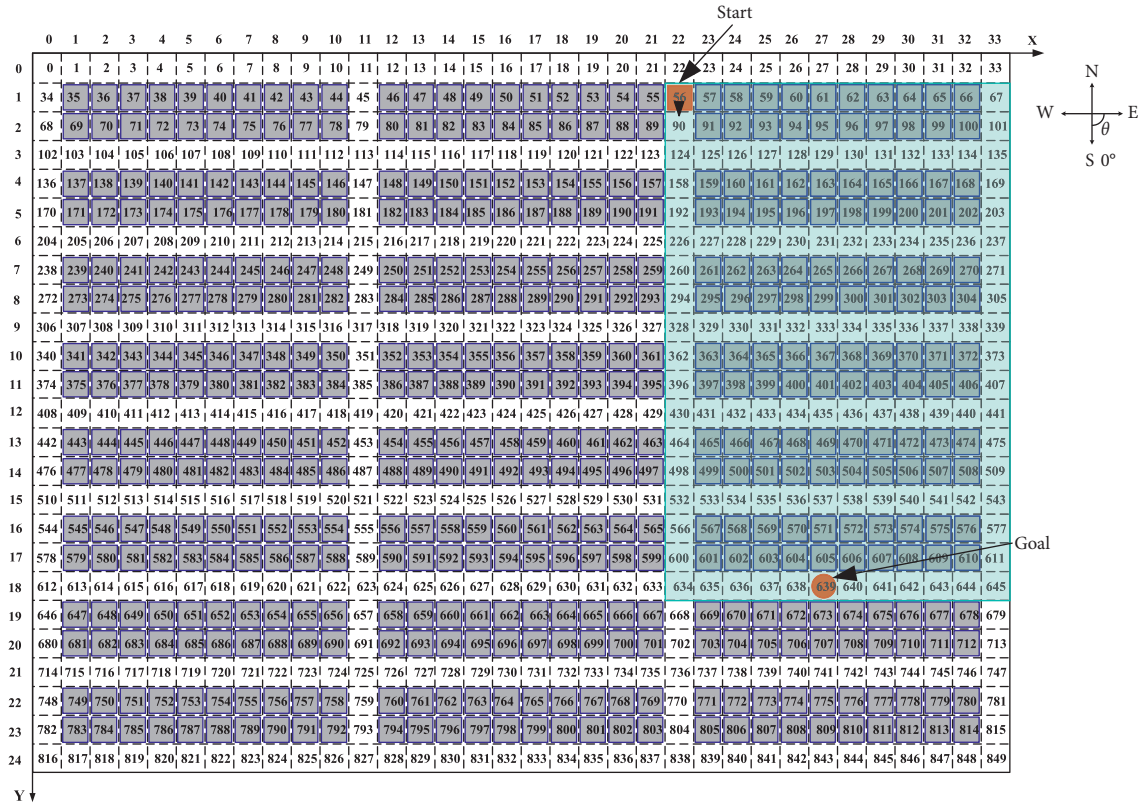


Figure 7: AGV's route-planning region (from node 56 $(x = 22, y = 1)$ to node 639 $(x = 27, y = 18)$).

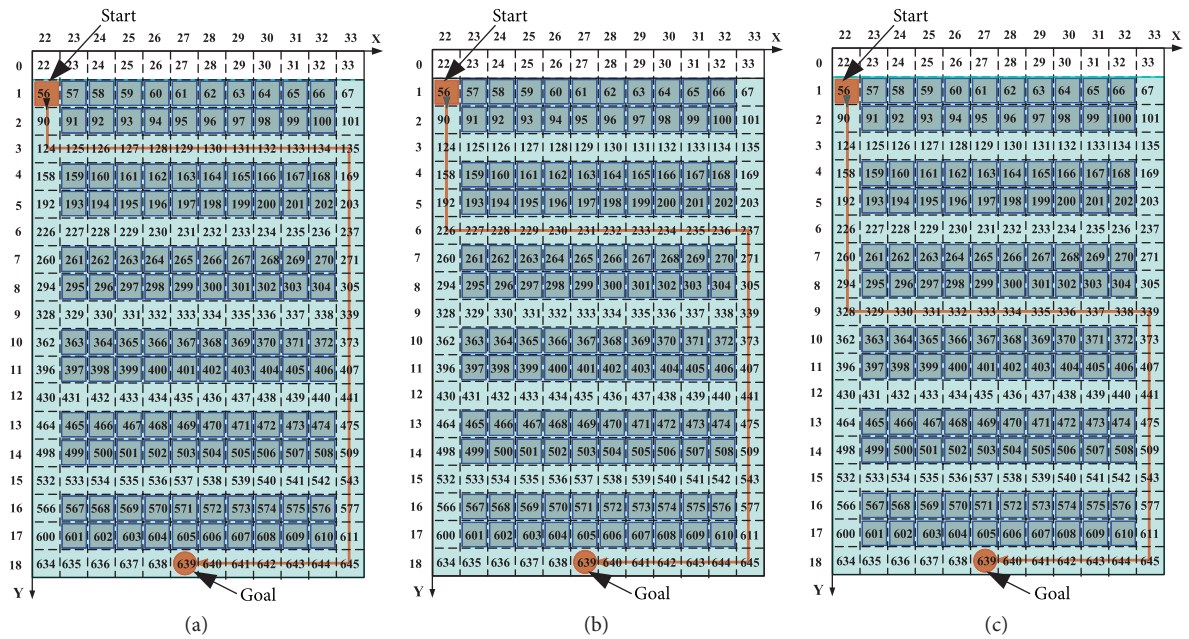


Figure 8: Continued.

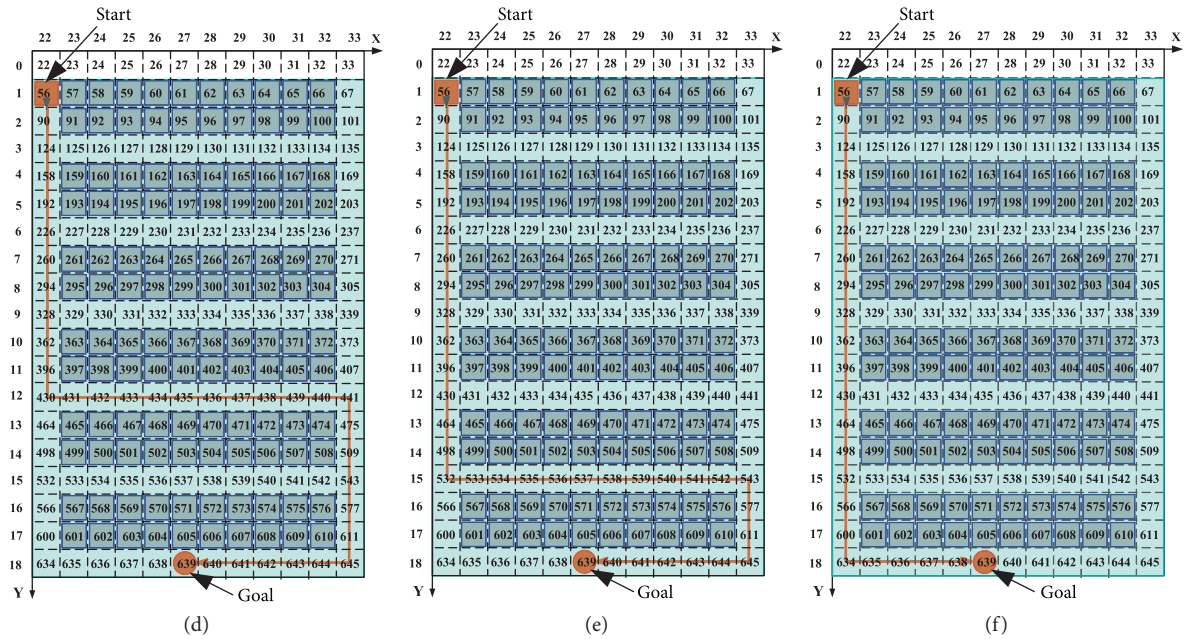


Figure 8: All feasible routes from AGV's start node to its goal node.

routes with the least AGV turning angle using (22) and obtain a route with the least turning nodes using equation (23), as shown in Figure 9. The comparison of route computing time between the algorithm proposed in this paper and the classical DP algorithm is shown in Figure 10, where the blue line represents the classical DP algorithm and the red the algorithm proposed in this paper.

As shown in Figure 10, the route computing time of the classical DP algorithm is **0.8756 s**, the time of the algorithm proposed in this paper is **0.1233 s**, and it reduces the route computing time (i.e., t_c) by nearly **85.92%**. Then, we compare the route computing time of the classical DP algorithm and our algorithm in SRNSs of different sizes (see Figure 11).

As shown in Figure 11, the algorithm proposed in this paper greatly reduces route computing time of AGVs and significantly improves route-planning algorithm's efficiency, and the larger the SRNSs, the more obvious the advantages of the proposed algorithm.

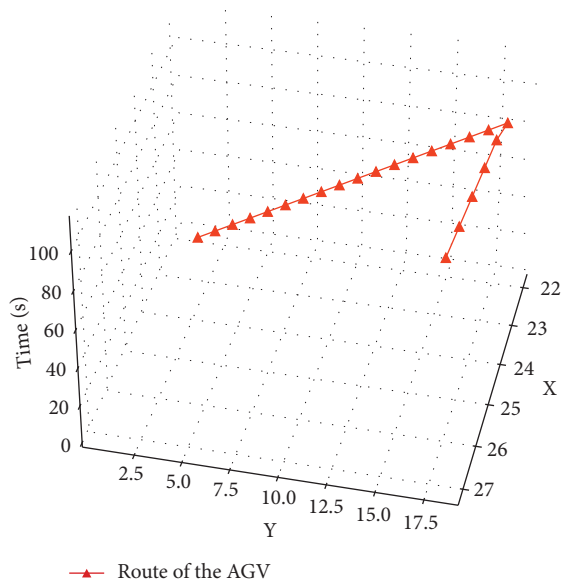


Figure 9: AGV's route that has the shortest distance and the least travel time.

Case 2. Route planning for loaded AGVs and a no-load AGV in AS/RSs. Consider the start node of an AGV to be node 23 *(x = 23, y = 0)* and the goal node to be node 435 *(x = 27, y = 12)*. If the AGV is loaded, it should avoid shelves located on nodes while traveling, and we need to build *obstacle_list = [ ]*. Then, we compute route-planning regions using equations (8)–(13) (or equations (14)–(19)) and repeatedly employ equation (21) to obtain all the feasible routes for the AGV from the start node to the goal node. Finally, we obtain the route with the shortest travel time according to equations (22) and (23), as indicated by the red line in Figure 12. If the AGV is a no-load one, it does not need to avoid shelves while traveling. We omit the process of building *obstacle_list = [ ]* and directly use the region-segmentation DP algorithm to obtain the route with the shortest length and travel time, as the blue line shows in Figure 12.

As shown in Figure 12, although the start node and the goal node of loaded and no-load AGV are the same, the number of turns in routes, the length of the route, and the traveling time of the AGV are different, and the comparison results are as follows.

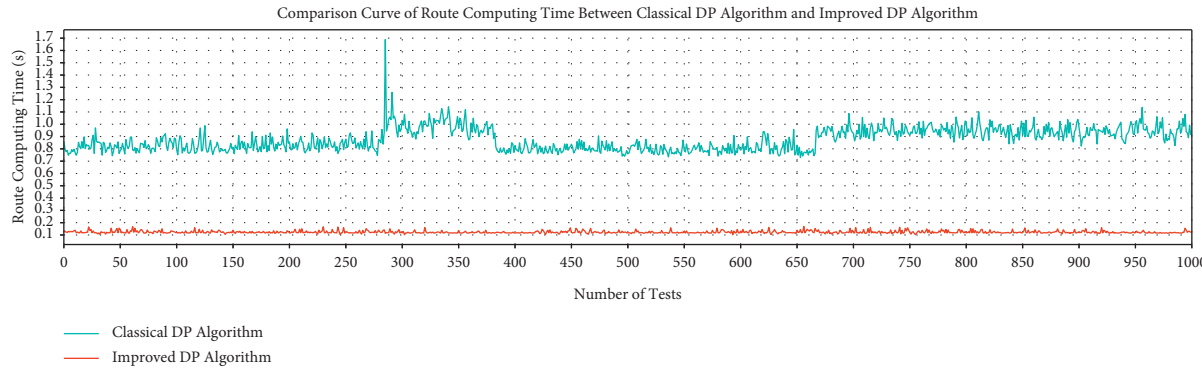


Figure 10: Comparison of route computing time between the classical DP algorithm and our algorithm.

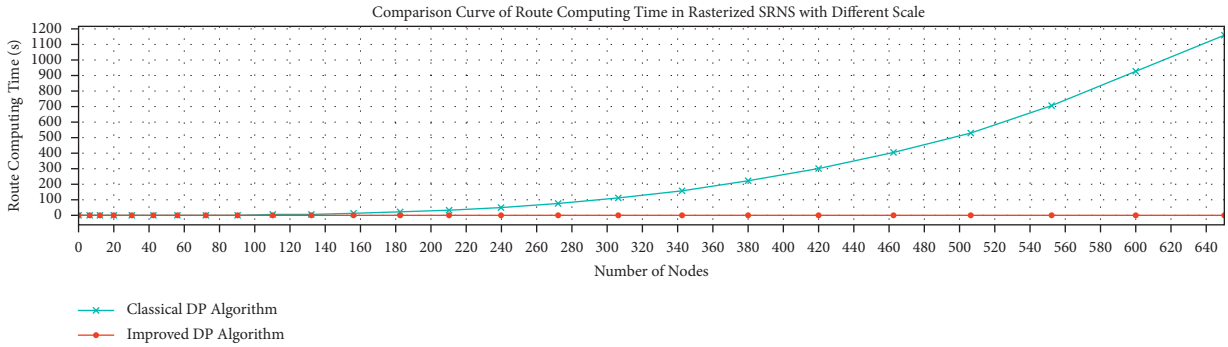


Figure 11: Comparison of route computing time between the classical DP algorithm and our algorithm.

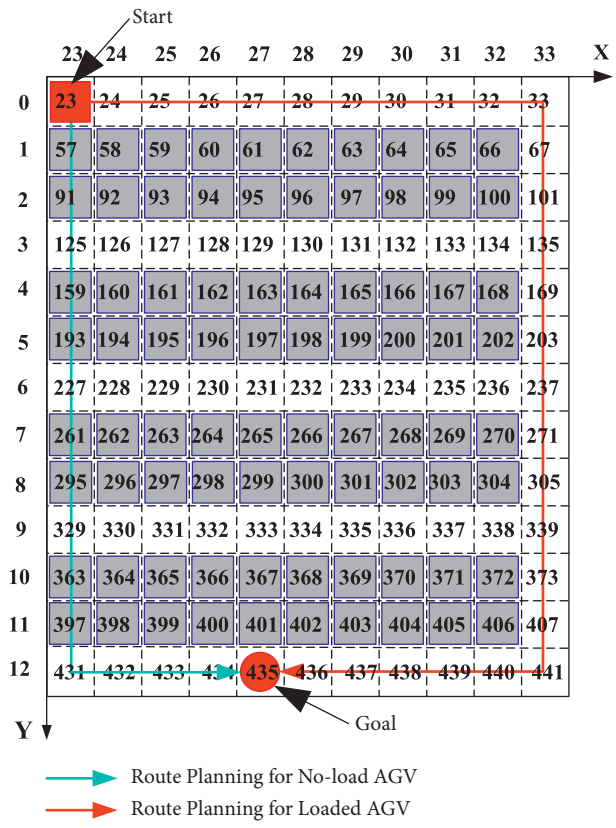


Figure 12: Route-planning results according to AGV's status (loaded or no-load).

Table 1: All feasible routes from AGV's start node to its goal node.

	Number of turns	Length of the route (m)	AGV's travel time (s)
Loaded AGV	2	70	150
No-load AGV	1	40	85

As shown in Table 1, (i) when the AGV is loaded, the number of turns is 2, the length of the route is 70m, and AGV's travel time is 150 s; (ii) when the AGV is a no-load one, the number of turns is 1, the length of the route is 40m, and AGV's travel time is 85s. In this scenario, the algorithm proposed in this paper improves the routes computing time (i.e., $\mathbf{t}_c$) by nearly 43.33%.

5. Conclusions

This paper proposes an AGVs route-planning approach in large SRNSs based on a region-segmentation Dynamic Programming algorithm. First, we use the improved MDP to model the large SRNSs. Then, the large SRNSs are divided into several route-planning regions according to AGVs' start nodes and their goal nodes, the objective is to narrow the range of searching routes and reduce the number of samples, and this step significantly improved the route-planning speed. Since more travel time and energy cost can be caused by AGVs' turns, we compute the absolute value of angle deviation between each feasible route to AGVs' initial directions to choose the candidate routes, and then the route

with the least turns is selected as the optimal one. Compared with the conventional Dynamic Programming algorithm, the algorithm proposed in this paper greatly improves the efficiency of route planning in large SRNSs.

Data Availability

The data set used to support the simulation studies of this paper is available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Ning, G. Han, P. Duan, and J. Tan, "A global and dynamitic route planning application for smart transportation," in *Proceedings of the First International Conference on Computational Intelligence Theory, Systems and Applications (CCITSA)*, December 2015.

[2] K. Li, W. Ni, and E. Tovar, "On-board deep Q-network for UAV-assisted online power transfer and data collection," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12215–12226, 2019.

[3] M. Ramasaravanan and M. Kirithikadevi, "End-to-End trust based transmission optimization in smartgrid network architecture," *International Journal of Computer Trends and Technology*, vol. 5, no. 3, 2013.

[4] R. Katsuki, T. Tasaki, and T. Watanabe, "Graph search based local path planning with adaptive node sampling," in *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium*, Changshu, China, June 2018.

[5] D. Kularatne, S. Bhattacharya, and M. A. Hsieh, "Going with the flow: a graph based approach to optimal path planning in general flows," *Autonomous Robots*, vol. 42, no. 7, 2018.

[6] E. Glorieux, P. Franciosa, and D. Ceglarek, "Coverage path planning with targeted viewpoint sampling for robotic free-form surface inspection," *Robotics and Computer-Integrated Manufacturing*, vol. 61, no. Feb., pp. 1–11, 2020.

[7] X. Wang, Y. Yan, and X. Gu, "Spot welding robot path planning using intelligent algorithm," *Journal of Manufacturing Processes*, vol. 42, no. June, pp. 1–10, 2019.

[8] N. Sugianti, A. Mardhiyah, and N. R. Fadilah, "Komparasi kinerja algoritma BFS, Dijkstra, greedy BFS, dan A∗ dalam melakukan pathfinding," *JISKA (Jurnal Informatika Sunan Kalijaga)*, vol. 5, no. 3, 2020.

[9] K. Niemeyer and C. J. Sung, "DRGEP-based mechanism reduction strategies: graph search algorithms and skeletal primary reference fuel mechanisms," in *Proceedings of the 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Orlando, Florida, July 2016.

[10] S. Permana, K. Bintoro, B. Arifitama, and A. Syahputra, "Comparative analysis of pathfinding algorithms A∗, Dijkstra, and BFS on maze runner game," *International Journal of Information System & Technology*, vol. 1, no. 2, 2018.

[11] Q. Guo, Z. Zhang, and Y. Xu, "Path-planning of automated guided vehicle based on improved Dijkstra algorithm," in *Proceedings of the 29th Chinese Control and Decision Conf.*, pp. 7280–7285, Chongqing, China, May 2017.

[12] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[13] R. I. Shreeve, "Introduction to dynamic programming, by leon cooper and mary W. Cooper. Pp 289. £15 hardback, £8·50 paperback. 1981. ISBN 0-08-0250645 (pergamon)," *The Mathematical Gazette*, vol. 66, no. 436, pp. 174-175, 1982.

[14] D. Bertsekas, "Multiagent reinforcement learning: rollout and policy iteration," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 249–272, 2021.

[15] A. G. Barto, "Reinforcement learning and dynamic programming," *Analysis, Design and Evaluation of Man-Machine Systems*, Elsevier Science, Amsterdam, Netherlands, pp. 407–412, 1995.

[16] Y. D. Tian, "A simple analysis of AlphaGo," *Acta Automatica Sinica*, vol. 42, no. 5, pp. 671–675, 2016.

[17] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, 1959.

[18] A. R. Leach and A. P. Lemon, "Exploring the conformational space of protein side chains using dead-end elimination and the A∗ algorithm," *Proteins-structure Function & Bioinformatics*, vol. 33, no. 2, pp. 227–239, 2015.

[19] T. Oral and F. Polat, "MOD∗ lite: an incremental path planning algorithm taking care of multiple objectives," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 245–257, 2015.

[20] H. Akbaripour and E. Masehian, "Semi-lazy probabilistic roadmap: a parameter-tuned, resilient and robust path planning method for manipulator robots," *International Journal of Advanced Manufacturing Technology*, vol. 89, no. 5-8, pp. 1401–1430, 2016.

[21] T. Xu, Y. Xu, D. Wang, S. Chen, W. Zhang, and L. Feng, "Path planning for autonomous articulated vehicle based on improved goal-directed rapid-exploring random tree," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–14, 2020.

[22] M. Drust, T. Dietz, and A. Verl, "Dynamic and interactive path planning and collision avoidance for an industrial robot using artificial potential Field based method," in *Proceedings of the 9th International Conference Mechatronics 2011*, Warsaw, Poland, September 2011.

[23] Y. Zhang, S. Li, and H. Guo, "A type of biased consensus-based distributed neural network for path planning," *Nonlinear Dynamics*, vol. 89, pp. 1803–1815, 2017.

[24] A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm," *Computers & Electrical Engineering*, vol. 38, no. 6, pp. 1564–1572, 2012.

[25] G. Liu, H. Hou, and J. Liu, "Convergence analysis and improvement method of ant colony algorithm of path planning for mobile robot," in *Proceedings of the 2nd International Conference on Artificial Intelligence*, pp. 3839–3842, Management Science and Electronic Commerce (AIMSEC), Zhengzhou, China, August 2011.

[26] L. He, P. Lou, X. Qiao, and R. Liu, "Conflict-free automated guided vehicles routing based on time window," *Computer Integrated Manufacturing Systems*, vol. 16, pp. 2630–2634, 2010.

[27] Y. Yuan and L. Tang, "Novel time-space network flow formulation and approximate dynamic programming approach for the crane scheduling in a coil warehouse," *European Journal of Operational Research*, vol. 262, no. 2, pp. 424–437, 2017.

[28] C. Novoa and S. Robert, "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands," *European Journal of Operational Research*, vol. 196, no. 2, pp. 509–515, 2009.

[29] M. Çimen and S. Mehmet, "Time-dependent green vehicle routing problem with stochastic vehicle speeds: an approximate dynamic programming algorithm," *Transportation Research Part D: Transport and Environment*, vol. 54, pp. 82–98, 2017.

[30] H. Bahlawan, M. Morini, M. Pinelli, and P. R. Spina, "Dynamic programming based methodology for the optimization of the sizing and operation of hybrid energy plants," *Applied Thermal Engineering*, vol. 160, Article ID 113967, 2019.

[31] T. Horiguchi, H. Takahashi, K. Hayashi, and C. Yamaguchi, "Dynamic programming for optimal packet routing control using two neural networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 339, no. 3-4, pp. 653–664, 2004.

[32] M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and B. W. Thomas, "On modeling stochastic dynamic vehicle routing problems," *EURO Journal on Transportation and Logistics*, vol. 9, no. 2, Article ID 100008, 2020.

[33] S. Desai and G. J. Lim, "Solution time reduction techniques of a stochastic dynamic programming approach for hazardous material route selection problem," *Computers & Industrial Engineering*, vol. 65, no. 4, pp. 634–645, 2013.

[34] C. B. Browne, E. Powley, and D. Whitehouse, "A survey of Monte Carlo tree search methods," *IEEE Trans Comput Intell AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[35] Z. Zhang, Q. Guo, J. Chen, and P. Yuan, "Collision-free route planning for multiple AGVs in an automated warehouse based on collision classification," *IEEE Access*, vol. 6, pp. 26022–26035, 2018.