Research Article

A Bi-Level Probabilistic Path Planning Algorithm for Multiple Robots with Motion Uncertainty

Jingchuan Wang ¹, Ruochen Tai ², and Jingwen Xu ¹

¹Department of Automation, Key Laboratory of System Control and Information Processing of Ministry of Education of China, Institute of Medical Robotics, Shanghai Jiao Tong University, Shanghai 200240, China

²School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 637121, Singapore

Correspondence should be addressed to Jingchuan Wang; jchwang@sjtu.edu.cn

Received 18 April 2020; Accepted 15 May 2020; Published 5 June 2020

Guest Editor: Ning Wang

Copyright © 2020 Jingchuan Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

For improving the system efficiency when there are motion uncertainties among robots in the warehouse environment, this paper proposes a bi-level probabilistic path planning algorithm. In the proposed algorithm, the map is partitioned into multiple interconnected districts and the architecture of proposed algorithm is composed of topology level and route level generating from above map: in the topology level, the order of passing districts is planned combined with the district crowdedness to achieve the district equilibrium and reduce the influence of robots under motion uncertainty. And in the route level, a MDP method combined with probability of motion uncertainty is proposed to plan path for all robots in each district separately. At the same time, the number of steps for each planning is dependent on the probability to decrease the number of planning. The conflict avoidance is proved, and optimization is discussed for the proposed algorithm. Simulation results show that the proposed algorithm achieves improved system efficiency and also has acceptable real-time performance.

1. Introduction

In the running of multiple robots in warehouses, motion uncertainties happen unavoidably since there exist some disabled components of robots, the instability of networks, and the interference of people walking. Under this case, robots would not follow the designed paths and the coupled relationship between temporal and space domain for paths is broken. And there is no doubt that other robots are disturbed by the ones where motion uncertainties happen. Finally, the whole multirobot system would be influenced and this brings about chaos or even breakdown of the whole system. It can be seen that this poses a great challenge to the path planning of multiple robots in warehouses. Therefore, taking motion uncertainty into consideration in the path planning of multiple robots is an issue worthy of attention and research.

As a kind of method taking motion uncertainty into consideration for the discretized map, Ma et al. [1] designed

the path planning algorithm which contains relevant restricted principles that an agent cannot enter the vertex that is occupied by some other agents at the last time step. The conflicts can still occur and this would cause the breakdown of the multirobot system in warehouses. Another kind of method which can deal with the issues caused by motion uncertainties needs to replan the paths of multiple robots when the samplings of robots contain abnormal information. Li et al. [2] proposed a two-stage trajectory planning scheme where a directed graph with cost variable connection is established and cost variable depends on the reachability, the risk of collision, and different state constraints. Gombolay et al. [3] presented a centralized algorithm that handles tightly intercoupled temporal and spatial constraints and scales to environment with large number of robots, which aims at improving the efficiency firstly. However, the above method cannot guarantee the real-time performance. In [4], the authors proposed a hybrid architecture which combines centralized coordination with distributed freedom of action

to achieve an appropriate interplay. The architecture takes the new sampling for the state of robots at each time step, and path planning algorithm would be adopted to coordinate paths of multiple robots when sampling information containing motion uncertainties. However, the real-time performance cannot be guaranteed and the efficiency of the whole system is reduced due to too frequent planning. The algorithm introduced in [5, 6] based on generalized probabilistic roadmaps (GPRM) also coordinates paths of multiple robots under motion uncertainty by using the feedback information. In the algorithm, the passive planning strategy is introduced in the multiple traveling salesman problem (MTSP) solution methodology. However, the system efficiency is low and the scalability of the algorithm cannot be guaranteed because far more points need to be sampled. A novel method in [7] based on the sampling considering the motion uncertainty would propagate this delay. Robots' paths need to be replanned adopting the Pareto-optimal plan repair scheme. However, the problem concerned with the decreased system efficiency is still not solved.

As a classical method devoted to solving the uncertainty problem, the Markov decision process (MDP) is introduced to the field of multiple agents. Corresponding multiagent MDPs (MMDPs) have been designed for multiple agent planning. In [8], approximating methods are introduced to be combined with MMDPs for the path planning problem of a team of homogeneous microair vehicles (MAVs) towards a set of goals. This algorithm can guarantee that there are no conflicts. Ma et al. [9] proposed the use of robust plan-execution policies to control how each robot proceeds along its path with a 2-level MDP solver that generates valid plans. To optimize the solutions, a novel optimal solver is designed in [10] for the problem of motion uncertainties in the transition-independent MMDPs. In [11], an equilibrium policy has been proposed in MMDPs. Based on the notion of macroactions, the path planning problem is transferred into the decentralized partially observable Markov decision processes (DecPOMDPs) in [12, 13]. Multiagent reinforcement learning (MARL) [14–20] solves the learning task in multiagent systems (MASSs). Nevertheless, the reasoning agent would deal with learning tasks which grow exponentially according to the number of agents. Therefore, the computational complexity of these methods based on MDPs is so high that the real-time scalability performance cannot be guaranteed.

Recently, the planning methods based on homotopy classes have been proposed to solve the problem of motion uncertainties. In [21, 22], the coordination space is established, which is the base for the later path planning. By selecting homotopy classes of the paths which are collision-free, a centralized controller is designed for the path planning problem under motion uncertainties. Furthermore, the encoding based on priorities, which is called the priority graphs, is introduced for the homotopic solutions to solve the coordination problem in [23]. For [21–23], the planning method based on homotopy classes can be proved that there are no conflicts for the coordinating paths and make sure that all robots will eventually reach their destinations even when some robots are temporarily stopped by a delay disturbance. However, the quality of solutions is

decreased to a great extent since this kind of method only considers the next step of motions and all the robots would follow the previous planned paths.

Although the above methods ensure that conflict can be avoided among the planning of multiple agents under motion uncertainty, problem still exists. The efficiency of a multirobot system with motion uncertainty cannot be guaranteed. Therefore, the first objective of this paper is to design an algorithm which can plan conflict-free paths for multiple robots even when motion uncertainty happens. And another objective is to improve the system efficiency that includes the makespan of the whole system and the number of planning.

To attain these goals, in this paper, the map is partitioned into multiple districts and we propose a bi-level planning architecture that is composed of topology level and route level generating from above map.

In the topology level, the passing orders of districts are planned based on the weight of paths in topology map. And in this process, the weight changes dynamically according to the distribution of robots; the aim of planning in the topology level is to achieve the district equilibrium, which can in turn reduce the influence of robots under motion uncertainty and promote the system efficiency. In the route level, a MDP-based probabilistic method is proposed to plan path for all robots in each district separately. The MDP is combined with the probability of motion uncertainty to improve efficiency. And the number of steps for each planning is dependent on the probability to decrease the number of planning. We proved that the designed algorithm is conflict-free. And we also validated the system efficiency improvement of designed algorithm from the optimization discussion and simulation.

The remainder of this paper is organized as follows. Section 2 provides the problem statement. In Section 3, the path planning algorithm for multiple robots is introduced. Section 4 proves that the proposed algorithm is conflict-free and discusses the optimization. Simulation validation results are reported in Section 5. Section 6 concludes our research.

2. Problem Formulation

The problem that this paper attempts to solve is the path planning of multiple AGVs with motion uncertainty in warehouses. In this problem, warehouse environment is discretized and simplified as a graph $G(V, E)$ whose vertices V correspond to locations and whose edges E correspond to transitions between locations. Figure 1 shows the discretized map.

Let $R = \{r_{(1)}, r_{(2)}, \dots, r_{(n)}\}$ be a set of robots. Time is discretized into time steps. To formally describe a plan, the path for a robot $r_{(i)}$ is a map $\mathbf{p}_{(i)}$ which means that the location $p_{(i)}(t)$ of $r_{(i)}$ in each time step $t \in N^*$ belongs to the vertices V . A robot can either remain stationary or move to an adjacent vertex. Robots are allowed to move simultaneously but restricted to move on the same edge in opposite directions or move at the same vertex simultaneously to ensure that there are no conflicts among multiple robots. Figure 2 gives the illustrations of these restricted conditions.

FIGURE 1: The discretized map of the warehouse. The orange and blue squares represent pickup points of the shelves and storage points, respectively. Other squares represent feasible ways.

FIGURE 2: Restricted condition for motions of multiple robots in warehouses.

Let $T = \{t_{(1)}, \dots, t_{(n)}\}$ be a set of tasks which are generated continually. For each robot that has been assigned a task, it has a start position (pickup point) and a goal position (storage point). And the makespan of all the tasks is M . During the running of robots, motion uncertainty, which means that robots have to stop immediately, could happen at any time. And the probability of the uncertainty is p_u .

This paper assumes the following about the scheduling of multiple AGVs in warehouses:

- (i) All feasible routes are two-way bidirectional streets, which means that two robots can drive side by side and robots can drive in both directions on each feasible way.
- (ii) The warehouse is made up of shelves and feasible ways, which is shown in Figure 1 (not limited to this scenario).
- (iii) The locations of all the robots are known.
- (iv) The probability of the uncertainty is known.
- (v) The robots in the state of delay will return to normal. And after the recovery, these robots will be put into operation in the warehouse system.
- (vi) The task allocation is not taken into consideration in this paper.

In this paper, letters of the subscripts which are not in the parentheses are used for identification. Letters or numbers of the subscripts which are in the parentheses are used for index.

Therefore, the problem studied in this paper can be formulated in the following equation:

$$\min M \text{ s.t. } \begin{cases} p_{(i)}(t) \neq p_{(j)}(t+1), & \forall i, j \in [1, n], i \neq j, \\ p_{(j)}(t) \neq p_{(i)}(t+1), & \forall i, j \in [1, n], i \neq j, \\ p_{(i)}(t) \neq p_{(j)}(t), & \forall i, j \in [1, n], i \neq j. \end{cases} \quad (1)$$

3. The Bi-Level Path Planning Algorithm for Multiple Robots

In this section, the bi-level path planning algorithm is introduced. The aim of this structure is that district equilibrium would be taken into consideration in the topology level for reducing the influence of robots under motion uncertainty and distributed path planning in each district can improve system efficiency in the route level even when motion uncertainty occurs. The architecture of the algorithm is shown in Figure 3. It is divided into two levels generating from the map partition: topology level and route level. In the

FIGURE 3: The bi-level architecture of the algorithm.

topology level, the algorithm plans the order of passing districts. In the route level, the algorithm plans paths for robots in each district separately, considering the motion uncertainty. And the introduction of map partition is as follows.

3.1. Map Partition. In this paper, the discretized map in Figure 1 is partitioned into multiple districts $D = \{d_{(1)}, \dots, d_{(M)}\}$ as shown in Figure 3. For the topology level, the generation of the topology map $G'(V', E')$ is listed as follows: (1) each vertex $v' \in V'$ corresponds to district $d_{(s)}$; (2) each edge $(v'_{(s)}, v'_{(s')}) \in E'$ represents that districts $d_{(s)}$ and $d_{(s')}$ are connected. Let the horizontal edges and vertical edges be E'_h and E'_v . For the route level, all feasible ways of each district in the discretized map of Figure 1 are shown in the route map.

3.2. Topology Level. In the topology level, the first step of the planning problem introduced in Section 3 is to plan the passing orders of districts for the robots which have been assigned tasks. At each time step, there might be multiple tasks assigned to robots. The number of these tasks is N and the robots corresponding to these tasks are $R_t = \{r_{(1)}, \dots, r_{(t)}\} (t \leq n)$. According to the task $t_{(i)}$ which has been assigned to the robot $r_{(i)}$, the districts which contain the start position and goal position of $t_{(i)}$ are

$d_{(s_{(i)})}$ and $d_{(g_{(i)})}$, respectively (here $s_{(i)}$ and $g_{(i)}$ represent the index of the corresponding district). To formally describe the plan results, the passing order of districts for a robot $r_{(i)}$ is a map $\mathbf{o}_{(i)}: N^* \rightarrow D$.

In this paper, $T (T \in N^*)$ is defined as the macroscopical time, which is the independent discrete variable in the topology level. With the increase of T , the robot will move from the start district $d_{(s_{(i)})}$ to the goal district $d_{(g_{(i)})}$. Different from the discrete time step, the macroscopic time represents the temporal space and corresponds to the district in topology level. Therefore, the change of T represents the transfer of the robot in the topology level. And if districts $\mathbf{o}_{(i)}(T)$ and $\mathbf{o}_{(i)}(T+1)$ are different, then the corresponding edge that robot $r_{(i)}$ will pass in the graph $G'(V', E')$ is defined as follows:

$$E_{(i)}(T) = (\mathbf{o}_{(i)}(T), \mathbf{o}_{(i)}(T+1)). \quad (2)$$

In this formulation, the edge $E_{(i)}(T)$ represents the transition from $\mathbf{o}_{(i)}(T)$ to $\mathbf{o}_{(i)}(T+1)$, which is the moving direction for the robot $r_{(i)}$. For the robot $r_{(i)}$ which has been assigned the task, there is the corresponding macroscopical time $T_{s_{(i)}}$ and $T_{g_{(i)}}$ when $r_{(i)}$ reaches the start district $d_{(s_{(i)})}$ and goal district $d_{(g_{(i)})}$, respectively. Then, the planning problem concerned with the districts can be formulated as follows:

$$\min \sum_{i=1}^t \sum_{T=T_{s_{(i)}}}^{T_{g_{(i)}}} L(E_{(i)}(T)) (1 + C(T, E_{(i)}(T))) \text{ s.t. } \begin{cases} \mathbf{o}_{(i)}(T_{s_{(i)}}) = d_{(s_{(i)})}, & 1 \leq i \leq t, \\ \mathbf{o}_{(i)}(T_{g_{(i)}}) = d_{(g_{(i)})}, & 1 \leq i \leq t. \end{cases} \quad (3)$$

In this formulation of the optimization objective, $L(\cdot)$ is the length of the edge $E_{(i)}(T)$, which is the corresponding distance between the center of the two districts. Due to the length difference of the edges in the graph $G'(V', E')$, the durations of passing these edges are different. In our problem, the length of the horizontal edge is twice that of the

vertical edge. Therefore, the duration of passing the horizontal edge is twice that of the vertical edge. For the other environment map with different layout, the same principle applies. According to this principle, the passing order for the robot $r_{(i)}$ should satisfy the following:

$$o_{(i)}(T+1) = \begin{cases} o_{(i)}(T), & (o_{(i)}(T-1), o_{(i)}(T)) \in E'_h \text{ and } o_{(i)}(T-2) \neq o_{(i)}(T-1), \\ \text{next district}, & \text{otherwise.} \end{cases} \quad (4)$$

Based on this principle, equation (2) can be rewritten as

$$E_{(i)}(T+1) = \begin{cases} E_{(i)}(T), & E_{(i)}(T-1) \in E'_h \text{ and } E_{(i)}(T-2) \neq E_{(i)}(T-1), \\ \text{next edge}, & \text{otherwise.} \end{cases} \quad (5)$$

In the optimization objective of equation (3), $C(\cdot)$ represents the crowdedness of the edge $E_{(i)}(T)$ at macroscopic time T (here we assume that $o_{(i)}(T)$ and $o_{(i)}(T+1)$ are different). And it is defined as follows:

$$C(T, E_{(i)}(T)) = \frac{NC_{(o_{(i)}(T))}(T) + NC_{(o_{(i)}(T+1))}(T)}{2}, \quad (6)$$

where $NC_{(o_{(i)}(T))}(T)$ and $NC_{(o_{(i)}(T+1))}(T)$ are the normalized crowdedness of the district $o_{(i)}(T)$ and $o_{(i)}(T+1)$, respectively. And the crowdedness of each district $o_{(i)}(T)$ is denoted as $C_{(o_{(i)}(T))}(T)$, which represents the number of robots in each district at the macroscopic time T . It can be obtained by the passing orders for each robot. Furthermore, these values are changing dynamically since robots are moving. Then, the method of normalization can be adopted to calculate $NC_{(o_{(i)}(T))}(T)$ and $NC_{(o_{(i)}(T+1))}(T)$.

Furthermore, due to the motion uncertainty of robots themselves, robots would not arrive at the planned district precisely. Therefore, at each macroscopic time, there are multiple robots which are assigned new tasks and need new path planning in current warehouse environment plan, the normalized crowdedness in equation (6) should be recalculated according to the distribution and passing orders of all robots in topology level. To be specific, assuming that one robot is located at district $d_{(s1)}$ and will move to district $d_{(s2)}$ when there are some new tasks and the planning is needed at the macroscopic time T , if the edge corresponding to the transition between $d_{(s1)}$ and $d_{(s2)}$ belongs to E'_h , then this robot would still stay at this edge at next time $(T+1)$ in equation (6). If the edge belongs to E'_v , then this robot would move to next edge at macroscopic time $(T+1)$.

Therefore, the optimization objective in equation (3) considers the length of the edges which robots would pass and the crowdedness of the related districts, whose aim is to make sure that the paths of robots would not be too long and the districts which have been planned for these robots would not be too crowded. And this design can guarantee the balance among different districts and efficiency of the whole warehousing system.

The constraints listed in equation (3) make sure that the algorithm can plan paths for robots from their start position to the goal position in topology level.

To solve the optimization problem in equation (3), K shortest path planning algorithm [24] is adopted to obtain the multiple shortest paths for each robot which needs to be planned at the moment. According to the planned paths, the passing order can be generated by using equations (4) and (5). The optimization objective can be calculated based on the rules introduced above, and the optimal solution among all the combinations can be selected, which contains the information concerned with the passing order for the robots assigned with new tasks in the topology level.

Figure 4 shows an example of the planning results in topology level. And it can be seen that the planning results follow the analysis above; the algorithm in the topology level prefers to choose the paths which are shorter and less crowded since this can minimize the optimization objective mentioned in equation (3).

3.3. Route Level. In the bi-level architecture, when the planning of the topology level has been finished, according to the distribution of the robots, the planning of each district would be conducted for the robots in this district distributedly.

In the route level, the route map is made up of all the feasible ways in each district. And the local route planning is to plan the actions of all the robots in the district being planned.

In the planning of each district, considering the motion uncertainty of robots and the probability p_u , the planning step α is introduced that should be intuitively related to p_u (α not only refers to the planning step at each planning but also refers to the interval between each two plannings). The aim of this idea is that the planning step can be adjusted dynamically based on p_u and some redundant planning in [11, 12] would not be needed anymore. Here, we define the planning step as the follows:

FIGURE 4: Illustration of the planning result in topology level. Red squares represent the robots which need to be planned. Green squares represent the other robots. The red line shows the planned passing order for each robot assigned with new task. According to the number of robots, the grey shade implies the crowdedness of each district. The fewer the number of robots, the lighter the color.

$$\alpha = \lfloor \frac{1}{p_u} \rfloor. \quad (7)$$

It can be seen from this definition that there is an inverse relationship between the planning step S and the probability of motion uncertainty p_u . When p_u gets larger, which means that there is higher probability that robots would not follow the previous planned path, the planning step α becomes smaller and the algorithm would plan paths for robots more conservatively. The first aim of computation for probability of motion uncertainty is to guarantee the real-time performance of the planning in route level since the traditional replanning all the paths from the current position to the goal position. However, the planning step is dependent on the probability of the motion uncertainty. Another aim of this is the planning step is adjusted based on p_u , which can prevent the situation that frequent replanning might destroy the optimized planned paths.

In the route level, the planning of each district can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\pi_{(x)}} & E(R(S_{(1)}) + \lambda \cdot R(S_{(2)}) + \dots + \lambda^{(\alpha-1)} R(S_{(\alpha)})), \\ \text{s.t.} & \begin{cases} p_{(i)}(t) \neq p_{(j)}(t+1), & \forall i, j \in [1, m], i \neq j, \\ p_{(j)}(t) \neq p_{(i)}(t+1), & \forall i, j \in [1, m], i \neq j, \\ p_{(i)}(t) \neq p_{(j)}(t), & \forall i, j \in [1, m], i \neq j. \end{cases} \end{aligned} \quad (8)$$

In the formulation of this optimization objective, $\pi_{(x)}$ represents one policy and there are k policies which are $\pi_{(1)}, \pi_{(2)}, \dots, \pi_{(k)}$. Each policy contains the information concerned with actions of the robots at each time step in the district. And for each policy, $\pi_{(x)} = (\pi_{(x),(1)}, \pi_{(x),(2)}, \dots, \pi_{(x),(\alpha)})$ where $\pi_{(x),(1)}$ indicates the policy at the first time step for all the robots in the district being planned. And $\pi_{(x),(1)} = (\pi_{(x),(1)}^{r_{(1)}}, \pi_{(x),(1)}^{r_{(2)}}, \dots, \pi_{(x),(1)}^{r_{(m)}})$ ($m \leq t$) where $\pi_{(x),(1)}^{r_{(1)}}$ indicates the policy at the first time step for the robot $r_{(1)}$ and m is the number of the robots located in the district being planned. And $\pi_{(x)}^{r_{(1)}} = (\pi_{(x),(1)}^{r_{(1)}}, \pi_{(x),(2)}^{r_{(1)}}, \dots, \pi_{(x),(\alpha)}^{r_{(1)}})$ represent the α step policy of $r_{(1)}$.

$E(\cdot)$ indicates the expectation of the reward which is similar to that of MDP. The aim of designing this objective function is that this can obtain the probabilistic optimal solution since there exists motion uncertainty of robots. λ ($0 < \lambda < 1$) is the discount factor. λ has relationship with p_u as equation (9). The aim of equation (9) is that the discount factor can be adjusted dynamically based on and when is high; the long-term reward is not reliable anymore and the path planning based on MDP is concerned more on the optimization of total short-term reward.

$S_{(1)}, \dots, S_{(\alpha)}$ represents the state from the start time step when the current district needs to be planned to the time step which is $(\alpha - 1)$ time steps after the start time step. And for $S_{(1)}$, it contains states for all the robots in the district, which means that $S_{(1)} = (s_{(1)}^{r_{(1)}}, s_{(1)}^{r_{(2)}}, \dots, s_{(1)}^{r_{(m)}})$. The state $s_{(1)}^{r_{(1)}}$ indicates the state for the robot $r_{(1)}$ after the action $\pi_{(1),(1)}^{r_{(1)}}$ is taken. $R(\cdot)$ defines the reward for each state based on the distribution of the robots in the corresponding district.

$$\lambda = 1 - p_u. \quad (9)$$

Therefore, according to the Markov decision process in [8], the optimization objective in equation (8) implies the expectation of the rewards for the state from the start time step to after $(\alpha - 1)$ time steps. And it can be rewritten as follows:

$$\min_{\pi_{(x)}} E(R(S_{(1)}) + \lambda \cdot R(S_{(2)}) + \dots + \lambda^{(\alpha-1)} R(S_{(\alpha)})) = \min_{\pi_{(x)}} \sum_{q=1}^{\alpha} \lambda^{(q-1)} \left(\sum_{S'_{(q)}} P_{S_{(q-1)}\pi_{(x),(q-1)}}(S'_{(q)}) \cdot R(S'_{(q)}) \right), \quad (10)$$

where $S'_{(q)}$ indicates the possible state after the action $\pi_{(x),(q-1)}$ is taken at the state $S_{(q-1)}$ since there exists the motion uncertainty of robots, $R(S'_{(q)})$ represents the reward of the robots in this district at the state $S'_{(q)}$, and $P_{S_{(q-1)\pi_{(x),(q-1)}}}(S'_{(q)})$ is the probability that the state of robots in this district would turn to $S'_{(q)}$ when the action $\pi_{(x),(q-1)}$ is taken at the state $S_{(q-1)}$. The probability can be calculated as follows:

$$P_{S_{(q-1)\pi_{(x),(q-1)}}}(S'_{(q)}) = \prod_{j=1}^m P_{s_{(q-1),\pi_{(x),(q-1)}}}^{r_{(j)}}(s'_{(q)}), \quad (11)$$

where $s'_{(q-1)}$ is the state of the robot $r_{(j)}$ after the action $\pi_{(x),(q-1)}$ is taken at state $s_{(q-1)}$ and $P_{s_{(q-1),\pi_{(x),(q-1)}}}^{r_{(j)}}(s'_{(q)})$ is the probability that the state of the robot $r_{(j)}$ will change into $s'_{(q)}$ after the action $\pi_{(x),(q-1)}$ is taken at state $s_{(q-1)}$. And this probability is dependent on the motion uncertainty and can be represented as follows:

$$P_{s_{(q-1),\pi_{(x),(q-1)}}}^{r_{(j)}}(s'_{(q)}) = \begin{cases} p_u, & \pi_{(x),(q-1)}^{r_{(j)}} = \text{move and } r_{(j)} \text{ stays still,} \\ 1 - p_u, & \pi_{(x),(q-1)}^{r_{(j)}} = \text{move and } r_{(j)} \text{ moves,} \\ 1, & \pi_{(x),(q-1)}^{r_{(j)}} = \text{waiting.} \end{cases} \quad (12)$$

For the reward $R(S'_{(q)})$ of the policy $\pi_{(x)}$, it is related to the crowdedness of the district and the distances among robots in the district, which can be defined as follows:

$$\varepsilon = \begin{cases} \infty, & p_{(i)}(t) = p_{(j)}(t+1), p_{(j)}(t) = p_{(i)}(t+1) \text{ or } p_{(i)}(t) = p_{(j)}(t), \exists i, j \in [1, m], \\ e^{1/(1-p_u)}, & p_{(i)}(t) = p_{(j)}(t), \exists i, j \in [1, m], \\ 1, & \text{otherwise.} \end{cases} \quad (16)$$

It can be seen from equation (16) that when there would be a conflict between robot $r_{(i)}$ and $r_{(j)}$, ε is set at ∞ . When a robot would move to the location at next time step where another robot is staying at for now, then ε is set at $e^{1/(1-p_u)}$. Otherwise, ε is set at 1.

According to equations (13) to (16), the first aim of this design is to prevent that each district would be too crowded since this would cause the coordination space to be too small, which means that once there happens motion uncertainty in some robots, it would be hard for the algorithm to find the solution and the system efficiency would be decreased. Another aim for the set of ε is to prevent the conflicts and make sure that one robot would not choose the action which might cause it to be next to another one. And this is beneficial for the motion coordination of robots since the motion uncertainty of one would cause it to still stay at the previous location, which increases the probability of the conflicts. Therefore, this setting can decrease the probability of the situation like that.

To solve the optimization problem in equation (8), the policy set π needs to be generated in the district and the

$$R(S'_{(q)}) = Cd'_{(x)} \cdot \left(\varepsilon \cdot \sum_{h=1}^m \text{value}(s_{(q)}^{r_{(h)}}) \right), \quad (13)$$

where $Cd'_{(x)}$ is the normalized crowdedness of the district being planned when the policy $\pi_{(x)}$ is taken. And the crowdedness $Cd_{(x)}$ of the district being planned can be calculated as follows:

$$Cd_{(x)} = \frac{m}{\sum_{i=1}^m \sum_{j=1}^m \text{distance}_{(i)(j)}}, \quad (14)$$

where $\text{distance}_{(i)(j)}$ is the distance between $r_{(i)}$ and $r_{(j)}$ at state $S'_{(q)}$. Then, the crowdedness $Cd_{(x)}$ needs to be normalized by using the principle of the softmax according to the following formula:

$$Cd'_{(x)} = \frac{e^{Cd_{(x)}}}{\sum_{h=1}^k e^{Cd_{(h)}}}. \quad (15)$$

In equation (13), $\text{value}(s_{(q)}^{r_{(h)}})$ indicates the value of the robot $r_{(h)}$ at state $s_{(q)}^{r_{(h)}}$. It is defined as the distance between the robot $r_{(h)}$ and the next district which $r_{(h)}$ heads to since the passing order has been planned in the topology level. The rule for calculating this distance is based on the Manhattan distance.

To avoid the conflicts among multiple robots, the factor ε is introduced. And ε in equation (13) is defined as follows:

policy with the minimum objective would be selected as the solution. And the principle for the candidate action set of the policies is that robots would not be far away from the next district after taking this action. There are two benefits for this principle. Firstly, the farther the distance from the next district, the larger the objective function in equation (8). Secondly, due to the introduction of this principle, the candidate actions will decrease, which can in turn lower the computational complexity. According to above principle, there are two kinds of candidate action sets when robot is in different locations of each district:

- (1) When robot is in any location of district and not coming into the boundary, the candidate action set of robot is denoted as {left, up, still}. The action left means that robot drives to the other side of two-way street at next time step, up means that robot follows current side of street, and still means that robot stays still at next time step. For example, in Figure 5, robots $r_{(1)}$ and $r_{(3)}$ are in the district and not coming into boundary at next time step. Therefore, their

FIGURE 5: Illustration of the candidate actions. Green squares represent the robots in the district. The blue arrow indicates the next district where these robots are heading to. Therefore, the orange arrows for each robot represent the candidate actions they can take. Besides the actions represented by these orange arrows, staying still is also included in the candidate actions for each time step.

candidate action set includes left, up, and still as orange arrows show. And as shown in Figure 5, the boundary between two districts is defined as an area composed of four grids, and one half of the boundary belongs to one district and the other belongs to the neighboring district. Considering that path planning in the route level is conducted separately in each district, robot should only follow one direction (drive to the right) in the process of traversing the boundary to the neighboring district, for avoiding the conflict.

- (2) When robot is in the boundary of district or coming into the boundary, the candidate action set of robot is denoted as $\{\text{up}, \text{still}\}$ or $\{\text{left}, \text{still}\}$ according to the rule that robot should only drive to the right in the process of traversing the boundary. For example, robot $r_{(2)}$ in Figure 5 is coming into the boundary at next time step and its candidate action set is $\{\text{left}, \text{still}\}$ which means that it needs to drive to the right side of two-way street firstly. $r_{(4)}$ and $r_{(5)}$ are in the boundary and their candidate action set is $\{\text{up}, \text{still}\}$.

According to the above candidate action set, the generation process of robot $r_{(i)}$ and policy $\pi_{(q)}^{r_{(i)}}$ is shown in Figure 6. And it can obtain multiple policies for each robot. Then, π can be generated with the combination of multiple policies of each robot in the district.

4. Discussion and Analysis

4.1. Conflict Avoidance. In this paper, we propose a bi-level path planning algorithm. The architecture of the proposed

method is divided into topology level and route level. And in the topology level, the passing order of districts is planned for robots. There is no conflict in topology level because the districts are allowed to traverse for multiple robots at the same time. Therefore, the conflict avoidance of the proposed algorithm in the paper mainly depends on whether or not it could plan conflict-free path in route level. And we try to prove it in the following proposition.

Proposition 1. *The path planning in route level could plan conflict-free paths for all robots in the district.*

Proof. In route level of the proposed method, the path for robots in the district is planned for many times and α steps with MDP at each time. Therefore, we try to use mathematical induction to prove the conflict-free path can always be planned by the proposed method in the district at each time. And we firstly describe some symbols of some variables to facilitate the expression of proof.

For any district $d_{(s)} \in D$ with m robots, the route map can be denoted as $G_p(V_p, E_p)$. And robots in the district can be denoted as $R_p = \{r_{(1)}, \dots, r_{(m)}\}$. n_{re} represents the n_{re} th path planning in each district. In the n_{re} th planning: the beginning of time step is denoted as $t_{n_{re}}$, and the planning step is α . The solution with MDP is $\pi_{n_{re}}$ and planned path of $\pi_{n_{re}}$ for any robot $r_{(i)}$ is $[p_{(i)}(t_{n_{re}}), p_{(i)}(t_{n_{re}} + 1), \dots, p_{(i)}(t_{n_{re}} + \alpha)]$. The proof detail is as follows:

- (a) Firstly, when $n_{re} = 1$, prove that the path planning is conflict-free:

At the first time of path planning, robots are leaving the boundary in each district. These robots of each district are heading to different directions and located in the different sides of two-way street in the map. Therefore, there always exists one solution π_1 that makes sure that all robots go on with their current side of two-way street in the α time steps and no robots from the opposite direction run in the same side of two-way street. And the planned paths for robots meet equation (17). It can be concluded that reward of the above solution π_1 according to equation (16) will not be ∞ and π_1 could be feasible solution of the optimization objective, which could prove that the proposed method could plan conflict-free path for all robots when $n_{re} = 1$.

$$\begin{cases} p_{(i)}(t) \neq p_{(j)}(t+1) \text{ and } p_{(j)}(t) \neq p_{(i)}(t+1), & \forall i, j \in R_p, t_{n_{re}} \leq t \leq t_{n_{re}+1}, \\ p_{(i)}(t) \neq p_{(j)}(t), & \forall i, j \in R_p, t_{n_{re}} \leq t \leq t_{n_{re}+1}. \end{cases} \quad (17)$$

- (b) Secondly, when $n_{re} = k_1$, suppose that the planned path in the k_1 th planning is conflict-free and prove

the planned path in the $(k_1 + 1)$ th planning is also conflict-free:

Considering that the path planning in the k_1 th planning is conflict-free, the ending location $p_{(i)}(t_{k_1} + \alpha)$ of planned path for any robot $r_{(i)} \in R_p$ satisfies equation (17), which means that there does not exist the situation of direction conflict and location. On the other hand, the ending location $p_{(i)}(t_{k_1} + \alpha)$ of the k_1 th path planning is equivalent to the beginning location $p_{(i)}(t_{k_1+1})$ of the $(k_1 + 1)$ th path planning. If there exists $v1_i, v2_i \in R_p$ where $v1_i$ represents the next location from with the action of going up and $v2_i$ represents the next location from $p_{(i)}(t_{k_1+1})$ with the action of going left, equation (18)

$$\begin{cases} p_{(h)}(t_{n_e}) \neq v2_i, & \forall h \in R_p \text{ and } h \neq i \text{ if } p_{(j)}(t_{n_e}) = v1_i, \exists j \in R_p \text{ and } j \neq i, \\ p_{(h)}(t_{n_e}) \neq v1_i, & \forall h \in R_p \text{ and } h \neq i \text{ if } p_{(j)}(t_{n_e}) = v2_i, \exists j \in R_p \text{ and } j \neq i, \end{cases} \quad (18)$$

and when robots go to the boundary in the last time planning, considering that the robots always follow one direction when they are coming into boundary or traversing on the boundary, the district-distributed way of the proposed method in route level will not have conflict between robots from different districts.

To summarize, there always would be conflict-free solution for path planning in the route level with the proposed method. $\square$

4.2. Optimization. For a multirobot system with each robot subject to uncertainty, directly computing the optimal solution is very difficult and it must recompute many times during the running of robots. Therefore, it is hard to directly prove the optimization or near optimization. In this section, we will discuss the optimization of the proposed method. Considering that the architecture of proposed method in this paper is divided into two levels and paths are planned separately, we discuss the optimization of proposed method from following two parts:

- (1) For path planning in topology level: in the topology level, the optimal orders of districts for all robots can be obtained and the reason is as follows. Firstly, K shortest path planning algorithm [24] is adopted to obtain the multiple orders of passing districts for each robot. And combinations of orders of passing districts among robots can be generated. When K is large enough, all possible combinations of orders of passing districts can be obtained. Secondly, equation

can be concluded. It means that each robot $r_{(i)}$ can go left or up without conflict with other robots. Therefore, it can construct a candidate solution π_{k_1} that makes robots heading to the same direction run in the same side of two-way streets in the α steps of path. And when robots heading to the different direction run in the same side of two-way street, the planned path will not have conflict with other robots. Therefore, it can be proved that the planned path in the $(k_1 + 1)$ th planning is conflict-free.

(3) is put up with in the topology level, which aims to minimize the corresponding distance between the center of the two districts and total crowdedness between districts among all the robots assigned with new tasks. And above optimization objective can be calculated based on all possible combinations of orders of passing of districts for robots.

- (2) For path planning in route level: the proposed method takes path planning for all robots in each district for many times and plans paths of α steps for the robots at each time until all robots leave the district. So, it is difficult to directly prove the optimization. In this part, we compare above method with existing method in [8] to discuss the optimization of the proposed method. Similar to the proposed method, the existing method also takes path planning in each district for many times. The difference is that at each time planning, the existing method plans the path of static α_{global} steps with the MDP method and executes the first α steps of them. The optimization objective is shown in equation (19). α_{global} can make sure that the destination of planned path is the location where robots are leaving from the current district. In the existing method in [8], α is also intuitively related to p_u following equation (7). The existing method in [8] has proved the optimization.

$$\min_{\pi_{(x)}} \sum_{q=1}^{\alpha} \lambda^{(q-1)} \left(\sum_{S'_{(q)}} P_{S_{(q-1)\pi_{(x)},(q-1)}}(S'_{(q)}) \cdot R(S'_{(q)}) \right) + \sum_{q=1}^{\alpha_{\text{global}} - \alpha} \lambda^{(\alpha+q-1)} \left(\sum_{S'_{(q)}} P_{S_{(q-1)\pi_{(x)},(q-1)}}(S'_{(q)}) \cdot R(S'_{(q)}) \right). \quad (19)$$

For discussing the optimization of proposed method, we try to discuss the similarity of first α steps for solution at each

time planning compared with existing method in [8] in the following two situations:

FIGURE 6: Generation of policy $\pi_{(q)}^{r(i)}$ for $r(i)$ in each district. Assuming that the candidate actions in each district include left, up, and still, the generation of policy $\pi_{(q)}^{r(i)}$ for robot $r(i)$ is shown in this figure. In this tree structure, different level of branches corresponds to different steps. Each circle represents the action for $r(i)$ in the district at the corresponding time step. For example, the action left means that all the robots in the district should go left. A complete policy $\pi_{(q)}^{r(i)}$ from the first time step to the α th time step is illustrated in the blue box. Multiple policies for each robot can be generated from this tree structure.

- (a) When the motion certainty probability p_u is low ($p_u \leq 0.2$):

In this situation, α is close to α_{global} according to (7). And it is obvious that the first α steps of solution from the proposed method and existing method in [6] are close to the same. For example, Figure 7 shows a situation where three robots are located at different states, and Table 1 shows the different solutions of path planning for the situation in Figure 7 using above two methods when p_u is different (α_{global} is set to 10 for the existing method in [8]). As shown in Table 1, when p_u is 0.1 or 0.2, first α steps of the proposed method are the same as the existing method in [6].

- (b) When the motion certainty probability p_u is high ($p_u > 2$):

In this situation, consider that both the proposed method and existing method in [8] choose the MDP method where the discount factor λ exists in optimization objective and has relationship with p_u according to equation (9). Therefore, when p_u is high, λ^α in (19) will be lower. This means the last ($\alpha_{global} - \alpha$) steps of solution have little influence on the computation for equation (19). And in this situation, the first α steps of the proposed method are close to the existing method in [6]. For example, in Table 1, when α is close to α_{global} according to (19), it is obvious that the solution from the proposed method is close to first α steps of solution from the existing method in [8]. Figure 7 shows a situation where three robots are located at different states, and Table 1 shows the different solutions of above two methods when p_u is different. As shown in Table 1, in the example of Figure 7, when p_u is 0.4 or 0.5, although the solution of above two methods is not exactly the same, the solution from the proposed method is

FIGURE 7: Illustration of the example of trying to plan path for three robots in the district. The original location of robots is shown in the figure. And the blue arrow indicates the next district where these three robots are heading to. $v_1 \sim v_{11}$ represent different locations in the district and v_6 is the destination where robots are leaving from the current district.

close to the first α steps of solution from the existing method in [8].

5. Simulation Result and Analysis

In order to evaluate the proposed method, simulations were implemented under the Matlab environment with Intel 3.60 GHz Core i7-4790U CPU and 8G RAM. The map of warehouse is shown in Figure 1.

The motion uncertainty p_u ranges from 0 to (1/2) to simulate the situations under different motion modes. The number of robots is set at 10, 20, 30, 40, and 50. The number of tasks is 1000.

The makespan, number of planning steps, and computation time are contrasted with those of the existing method in [21, 22] under a different number of robots. And for the existing method in [21, 22], the probability corresponding to the disturbance intensity whether during the following second the robot will be prevented from moving is set to p_u and the

TABLE 1: The solution of path planning for three robots which are in different locations as shown in Figure 7.

	The solution of the proposed method	The first α -step solution of the existing method in [8]
$p_u = 0.1$ $\alpha = 10$ $\lambda = 0.9$	$r_{(1)}: (v_0, \text{up}), (v_2, \text{up}), (v_5, \text{left}), (v_4, \text{up})$ $(v_7, \text{left}), (v_6, \text{left})$ $r_{(2)}: (v_{12}, \text{up}), (v_9, \text{up}), (v_7, \text{left}), (v_6, \text{left})$ $r_{(3)}: (v_{13}, \text{up}), (v_{11}, \text{up}), (v_{10}, \text{left}), (v_8, \text{still})$ $(v_8, \text{left}), (v_7, \text{left}), (v_6, \text{left})$	$r_{(1)}: (v_0, \text{up}), (v_2, \text{up}), (v_5, \text{left}), (v_4, \text{up})$ $(v_7, \text{left}), (v_6, \text{left})$ $r_{(2)}: (v_{12}, \text{up}), (v_9, \text{up}), (v_7, \text{left}), (v_6, \text{left})$ $r_{(3)}: (v_{13}, \text{up}), (v_{11}, \text{up}), (v_{10}, \text{left}), (v_8, \text{still})$ $(v_8, \text{left}), (v_7, \text{left}), (v_6, \text{left})$
$p_u = 0.2$ $\alpha = 5$ $\lambda = 0.8$	$r_{(1)}: (v_0, \text{up}), (v_2, \text{up}), (v_5, \text{left}), (v_4, \text{up})$ (v_7, left) $r_{(2)}: (v_{12}, \text{up}), (v_9, \text{up}), (v_7, \text{left}), (v_6, \text{up})$ $r_{(3)}: (v_{13}, \text{up}), (v_{11}, \text{up}), (v_{10}, \text{left}), (v_8, \text{still})$ (v_8, left)	$r_{(1)}: (v_0, \text{up}), (v_2, \text{up}), (v_5, \text{left}), (v_4, \text{up}), (v_7, \text{left})$ $r_{(2)}: (v_{12}, \text{up}), (v_9, \text{up}), (v_7, \text{left}), (v_6, \text{up})$ $r_{(3)}: (v_{13}, \text{up}), (v_{11}, \text{up}), (v_{10}, \text{left}), (v_8, \text{still})$ (v_8, left)
$p_u = 0.3$ $\alpha = 3$ $\lambda = 0.7$	$r_{(1)}: (v_0, \text{up}), (v_2, \text{up}), (v_5, \text{up})$ $r_{(2)}: (v_{12}, \text{up}), (v_9, \text{up}), (v_7, \text{left})$ $r_{(3)}: (v_{13}, \text{up}), (v_{11}, \text{up}), (v_{10}, \text{left})$	$r_{(1)}: (v_0, \text{up}), (v_2, \text{up}), (v_5, \text{up})$ $r_{(2)}: (v_{12}, \text{up}), (v_9, \text{up}), (v_7, \text{left})$ $r_{(3)}: (v_{13}, \text{up}), (v_{11}, \text{up}), (v_{10}, \text{left})$
$p_u = 0.4$ $\alpha = 3$ $\lambda = 0.6$	$r_{(1)}: (v_0, \text{up}), (v_2, \text{up}), (v_5, \text{up})$ $r_{(2)}: (v_{12}, \text{up}), (v_9, \text{up}), (v_7, \text{left})$ $r_{(3)}: (v_{13}, \text{up}), (v_{11}, \text{up}), (v_{10}, \text{left})$	$r_{(1)}: (v_0, \text{up}), (v_2, \text{up}), (v_5, \text{left})$ $r_{(2)}: (v_{12}, \text{up}), (v_9, \text{up}), (v_7, \text{left})$ $r_{(3)}: (v_{13}, \text{up}), (v_{11}, \text{up}), (v_{10}, \text{left})$
$p_u = 0.5$ $\alpha = 2$ $\lambda = 0.5$	$r_{(1)}: (v_0, \text{up}), (v_2, \text{up})$ $r_{(2)}: (v_{12}, \text{up}), (v_9, \text{up})$ $r_{(3)}: (v_{13}, \text{up}), (v_{11}, \text{up})$	$r_{(1)}: (v_0, \text{up}), (v_2, \text{up})$ $r_{(2)}: (v_{12}, \text{up}), (v_9, \text{up})$ $r_{(3)}: (v_{13}, \text{up}), (v_{11}, \text{up})$

lower bound which represents the best possible travel time is set to the shortest distance from start to end for each robot. The average of 20 repeated simulations is shown in Tables 2–4. The tasks have been generated before simulations.

5.1. Makespan. Table 2 shows the simulation results of average makespan of the existing method and the proposed method. Standard deviations are also shown in Table 2. Decrease percentage of makespan (the last column in Table 2) is calculated according to the following formula:

$$\text{decrease percentage} = \frac{\text{makespan}_1 - \text{makespan}_2}{\text{makespan}_1} \times 100\%, \quad (20)$$

where makespan_1 and makespan_2 represent the makespan of the existing method in [21, 22] and the proposed method, respectively. Graphical summaries of simulation results of makespan are shown in Figure 8.

5.1.1. Average Time. Compared to the existing method in [21, 22], it can be seen that the proposed method in this paper effectively decreases the makespan. When the number of robots is relatively small, the decrease percentage is low because the robots are located sparsely in the warehouse environment and the motion uncertainties of robots would not influence too many other normal robots, which means that most of the normal robots could still follow the previous planned paths. As the number of robots increases, the decrease percentage of makespan is greatly improved because the number of influenced robots would increase further, which means that the proposed method would replan to find more optimal paths based on the probability p_u for more robots to avoid robots which are in the motion uncertainty. Under this circumstance, by combining MDP, redundant waiting of the method in [21, 22] is not necessary. When the value of p_u increases, the

proposed method could still improve the system efficiency and has a smaller variance compared to the existing method in [21, 22]. The percentage of decrease can be up to 27.9% when the number of robots is 50 and p_u is 0.5.

5.1.2. Variance. Compared to the existing method in [21, 22], the variance of the proposed method is smaller. Furthermore, with the increase of the number of robots, the variance in the results of the proposed method increases less than that of the existing method in [21, 22]. The comparison between the results of the proposed method and the existing method in [21, 22] means that the proposed method has less randomness and is more stable. The reason is that the method in this paper would optimize paths of all the robots depending on the probability p_u .

In the proposed method, unlike the existing method in [21, 22] that would solve the problem of motion uncertainty through a waiting strategy, the proposed method to coordinate paths of robots under motion uncertainties adopts the strategy of replanning new paths to choose the probabilistic optimal solution. Therefore, when motion uncertainty happens, all the normal robots would follow the paths that could reach the destination as soon as possible from the global view instead of adopting the strategy of going forward and waiting. Therefore, compared to the existing methods in [21, 22], the decrease of the efficiency induced by the motion uncertainties could be lowered. In this way, the efficiency of the whole system could be improved when motion uncertainties happen by adopting the proposed method. Compared to the existing methods, the efficiency could be increased, which is beneficial for the overall running of the system.

5.2. Number of Planning Steps. Percentage of planning (listed in Table 2) is calculated according to the following formula:

TABLE 2: Simulation results in a warehouse environment of 50×34 4-neighbor grids: makespan (standard deviation).

p_u	Robot number	Existing method in [21, 22]	Proposed method	Decrease percentage (%)
0.1	10	19794.5 (272.4)	19022.4 (187.1)	3.9
	20	13582.6 (285.6)	12210.8 (193.5)	10.1
	30	11394.5 (294.2)	9297.9 (203.8)	18.4
	40	9019.3 (314.8)	7044.1 (209.4)	21.9
	50	7768.4 (332.9)	5818.5 (216.7)	25.1
0.2	10	22956.7 (280.4)	22061.4 (193.5)	3.9
	20	15636.6 (290.1)	14041.7 (199.4)	10.2
	30	13103.2 (301.5)	10666.1 (204.5)	18.6
	40	9019.3 (317.6)	8702.4 (210.4)	22.2
	50	9587.2 (334.9)	7123.3 (220.7)	25.7
0.3	10	25895.6 (287.5)	24859.8 (197.2)	4.0
	20	16879.3 (298.4)	15140.7 (200.1)	10.4
	30	14374.5 (309.7)	11657.7 (208.7)	18.9
	40	12287.3 (338.6)	9510.4 (218.5)	22.6
	50	10625.4 (344.8)	7820.4 (229.5)	26.4
0.4	10	28190.4 (294.1)	27034.6 (196.5)	4.1
	20	18724.5 (302.6)	16777.2 (200.4)	10.4
	30	16394.6 (315.3)	13246.8 (206.8)	19.2
	40	14274.8 (334.7)	10991.6 (218.5))	23.0
	50	12463.4 (354.7)	9085.8 (230.5))	27.1
0.5	10	31852.4 (300.6)	30514.6 (199.2)	4.2
	20	19952.6 (310.9)	17857.6 (201.5)	10.5
	30	17365.4 (325.6)	13979.1 (208.7)	19.5
	40	15284.7 (344.8)	11692.8 (221.8))	23.5
	50	13561.2 (356.3)	9777.6 (231.5)	27.9

$$\text{planning percentage} = \frac{N_{re}}{\text{makespan}} \times 100\%, \quad (21)$$

where N_{re} and makespan represent the number of planning and the makespan of corresponding method, respectively. The makespan of each method is given in Table 2. Graphical summaries of simulation results of planning percentage are shown in Figure 9. It can be seen from Table 3 that the percentage of planning of the proposed method is much more smaller than that of the method in [21, 22]. It has to plan at each time step since the method of [21, 22] is a kind of reactive method. Therefore, the percentage is 100%. On the contrary, the proposed method in the route level would only plan when robots have finished the path of planning step α or when the motion uncertainty happens. Therefore, the percentage of planning would decrease, compared with the existing methods in [21, 22].

Another conclusion we can get from Table 3 is that when the probability p_u is small, the percentage reduction compared with the method in [21, 22] is larger and when the probability p_u increases, the percentage reduction becomes smaller. The reason for this is that when p_u is small, there would not be too many time steps when motion uncertainties happen, which means that there would not be too many time steps when planning is needed. Therefore, the percentage of planning is relatively small, which causes the percentage reduction to be larger. When p_u becomes larger, the time steps when motion uncertainties happen become more, which causes more planning. Therefore, the percentage of planning increases, which causes the percentage reduction to be smaller. However, it is noteworthy that even

if the percentage of planning becomes larger when p_u increases, the percentage of the proposed method is still fewer than that of the method in [21, 22].

5.3. Computation Time. Table 4 shows the average of the computation time under different p_u . The statistics in the table demonstrate that the proposed method could satisfy the requirement of the real-time performance. The reason that the proposed method is higher than that of the existing method in [21, 22] is that the existing method only plans for the next time step while the proposed method needs to plan the whole path to the destination. Therefore, the computation of the proposed method would be larger than that of the existing method. However, as shown in Figure 10, the trend of computation time is more close to linear relationship for the proposed method and increasing speed will reduce slowly with robot number increasing. On the other hand, even if the computation time is somewhat larger, the proposed method could still plan conflict-free paths for all the robots in real time, which can be applied for the lifelong scheduling in warehouses.

The real-time performance of the bi-level probabilistic path planning algorithm for multiple robots with motion uncertainty is guaranteed by the bi-level planning architecture and the distributed planning in each district. The topology level would plan the passing orders of districts for all robots assigned with tasks and the path planning is conducted in each district for route level. The planning in each district only needs to take the actions of the current district being planned into consideration. Therefore, the

TABLE 3: Simulation results in a warehouse environment of 50×34 4-neighbor grids: number of planning (standard deviation).

p_u	Robot number	Existing method in [21, 22]	Planning percentage (%)	Proposed method	Planning percentage (%)
0.1	10	19794.3 (272.4)	100	2587.0 (102.4)	13.6
	20	13582.6 (285.6)	100	1721.7 (104.1)	14.1
	30	11394.5 (294.2)	100	1348.2 (107.3)	14.5
	40	9019.3 (314.8)	100	1056.6 (109.9)	15.0
	50	7768.4 (332.9)	100	907.7 (112.6)	15.6
0.2	10	22956.7 (280.4)	100	5405.0 (104.3)	24.5
	20	15636.6 (290.1)	100	3510.4 (106.4)	25.0
	30	13103.2 (301.5)	100	2815.9 (109.6)	26.4
	40	11185.6 (317.6)	100	2427.9 (112.1)	27.9
	50	9587.2 (334.9)	100	2108.5 (115.9)	29.6
0.3	10	25895.6 (287.5)	100	8825.2 (106.5)	35.5
	20	16879.3 (298.4)	100	5708.1 (108.5)	37.7
	30	14374.5 (309.7)	100	4558.2 (110.8)	39.1
	40	12287.3 (328.6)	100	3937.3 (113.4)	41.4
	50	10625.6 (344.8)	100	3456.6 (116.5)	44.2
0.4	10	28190.4 (294.1)	100	12544.1 (108.6)	46.4
	20	18724.5 (302.6)	100	8170.5 (110.7)	48.7
	30	16394.6 (315.3)	100	6861.8 (113.6)	51.8
	40	14274.8 (334.7)	100	6034.4 (116.8)	54.9
	50	12463.4 (354.7)	100	5324.3 (119.9)	58.6
0.5	10	31852.4 (300.6)	100	17728.9 (110.3)	58.1
	20	19952.6 (310.9)	100	10750.3 (112.9)	60.2
	30	17365.4 (325.6)	100	8764.9 (115.6)	62.7
	40	15284.7 (344.8)	100	7647.1 (118.4)	65.4
	50	13561.2 (356.3)	100	6736.8 (122.4)	68.9

TABLE 4: Simulation results in a warehouse environment of 50×34 4-neighbor grids: computation time (standard deviation).

p_u	Robot number	Existing method in [21, 22]	Proposed method
0.1	10	0.0015s (0.000154)	0.0083s (0.000356)
	20	0.0021s (0.000202)	0.1287s (0.000391)
	30	0.0026s (0.000283)	0.2176s (0.000514)
	40	0.0033s (0.000362)	0.3025s (0.000698)
	50	0.0041s (0.000451)	0.3786s (0.000824)
0.2	10	0.0016s (0.000156)	0.0041s (0.000322)
	20	0.0022s (0.000209)	0.095s (0.000371)
	30	0.0027s (0.000291)	0.1756s (0.000432)
	40	0.0034s (0.000369)	0.2476s (0.000599)
	50	0.0042s (0.000468)	0.3012s (0.000774)
0.3	10	0.0017s (0.000162)	0.0029s (0.000316)
	20	0.0023s (0.000218)	0.0523s (0.000354)
	30	0.0028s (0.000401)	0.1026s (0.000407)
	40	0.0035s (0.000379)	0.1424s (0.000558)
	50	0.0043s (0.000481)	0.1672s (0.000732)
0.4	10	0.0019s (0.000171)	0.0026s (0.000301)
	20	0.0025s (0.000226)	0.0278s (0.000334)
	30	0.0030s (0.000311)	0.0478s (0.000386)
	40	0.0037s (0.000397)	0.0623s (0.000514)
	50	0.0045s (0.000492)	0.0745s (0.000702)
0.5	10	0.0021s (0.000178)	0.0021s (0.000289)
	20	0.0027s (0.000235)	0.0312s (0.000318)
	30	0.0032s (0.000323)	0.0558s (0.000371)
	40	0.0039s (0.000398)	0.0797s (0.000501)
	50	0.0047s (0.000506)	0.0976s (0.000688)

FIGURE 8: Decrease percentage of average makespan in a warehouse environment of 50×34 4-neighbor grids.

FIGURE 9: Planning percentage in a warehouse environment of 50×34 4-neighbor grids. (a) $p_u = 0.1$. (b) $p_u = 0.2$. (c) $p_u = 0.3$. (d) $p_u = 0.4$. (e) $p_u = 0.5$.

FIGURE 10: Computation time in a warehouse environment of 50×34 4-neighbor grids. (a) $p_u = 0.1$. (b) $p_u = 0.2$. (c) $p_u = 0.3$. (d) $p_u = 0.4$. (e) $p_u = 0.5$.

number of planning depends on p_u , which means that the computation is adjusted by p_u .

6. Conclusion

For improving the system efficiency when there are motion uncertainties among robots in the warehouse environment, this paper proposes a bi-level probabilistic path planning algorithm. In the proposed algorithm, the map is partitioned into multiple interconnected districts and the architecture of proposed algorithm is composed of topology level and route level generating from above map: in the topology level, the district crowdedness is taken into consideration to optimize the passing orders of districts and achieve the district equilibrium; in the route level, path planning is conducted in each district with a MDP-based probabilistic method, where the planning step is related to the probability of the motion uncertainty to improve the efficiency and guaranteeing the real-time performance.

Furthermore, the conflict avoidance of designed algorithm is proved and optimization is discussed. The simulation results validate that the designed algorithm can improve the system efficiency and the percentage can be up to 27.9% in the designed warehouses environment of this paper, which also accords with optimization discussion. It can be also seen that the number of planning has been reduced to a large extent.

Our future work will involve the further optimization of paths by taking the interaction of flow among different districts into consideration since the actions are only obtained by the distributed planning in each district. Another aspect is that the real experiments are needed to verify the effectiveness of the proposed method.

Data Availability

The source code and simulation data of the proposed method and compared method in [18, 19] are available

from the corresponding author upon request. They can also be found at https://github.com/SourceCode2020/Bi-level_Probabilistic_Path_Planning_Algorithm.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by the National Natural Science Foundation of China under grant no. 61773261.

References

- [1] H. Ma, T. K. S. Kumar, and S. Koenig, "Multi-agent path finding with delay probabilities," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 3605–3612, San Francisco, CA, USA, February 2017.
- [2] G. Li, H. P. Hildre, and H. Zhang, "Toward time-optimal trajectory planning for autonomous ship maneuvering in close-range encounters," *IEEE Journal of Oceanic Engineering*, pp. 1–16, 2019.
- [3] M. Gombolay, R. Wilcox, and J. Shah, "Fast scheduling of multi-robot teams with temporospatial constraints," in *Proceedings of the Robotics: Science and Systems Conference*, Berlin, Germany, June 2013.
- [4] H. Kowshik, D. Caveney, and P. R. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 804–818, 2011.
- [5] S. Chakravorty and S. Kumar, "Generalized sampling-based motion planners," *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, vol. 45, no. 4, pp. 647–662, 2015.
- [6] S. Kumar and S. Chakravorty, "Multi-agent generalized probabilistic RoadMaps: MAGPRM," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3747–3753, Vilamoura, Portugal, October 2012.
- [7] A. W. ter Mors, "Conflict-free route planning in dynamic environments," in *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2166–2171, San Francisco, CA, USA, September 2011.
- [8] L. Liu and N. Michael, "An MDP-based approximation method for goal constrained multi-MAV planning under action uncertainty," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 56–62, Stockholm, Sweden, May 2006.
- [9] H. Ma, T. K. S. Kumar, and S. Koenig, "Multi-agent path finding with delay probabilities," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, CA, USA, February 2017.
- [10] J. Scharpf, D. M. Roijers, F. A. Oliehoek, M. T. J. Spaan, and M. M. de Weerd, "Solving transition-independent multi-agent MDPs with sparse interactions," in *Proc. 30th AAAI Conf. Artif. Intell. (AAAI)*, May, pp. 3174–3180, 2016.
- [11] Y. Hu, Y. Gao, and B. An, "Multiagent reinforcement learning with unshared value functions," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 647–662, 2015.
- [12] C. An, G. Konidaris, A. Anders, G. Cruz, J. P. How, and L. P. Kaelbling, "Policy search for multi-robot coordination under uncertainty," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1760–1778, 2016.
- [13] C. Amato, G. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling, "Planning for decentralized control of multiple robots under uncertainty," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1241–1248, Seattle, WA, USA, May 2015.
- [14] F. L. Da Silva, R. Glatt, and A. H. R. Costa, "MOO-MDP: an object-oriented representation for cooperative multiagent reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 49, no. 2, pp. 567–579, 2019.
- [15] L. Bu, R. Babu, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.
- [16] M. L. Koga, V. Freire, and A. H. R. Costa, "Stochastic abstract policies: generalizing knowledge to improve reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 45, no. 1, pp. 77–88, 2015.
- [17] F. L. Silva, R. Glatt, and A. H. R. Costa, "Simultaneously learning and advising in multiagent reinforcement learning," in *Proceedings of the Sixteenth International Conference on Autonomous Agents and Multiagent Systems*, pp. 1100–1108, Sao Paulo, Brazil, May 2017.
- [18] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2012.
- [19] M. Geist and O. Pietquin, "Algorithmic survey of parametric value function approximation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 845–867, 2013.
- [20] F. L. Silva and A. H. R. Costa, "Accelerating multiagent reinforcement learning through transfer learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 5034–5035, San Francisco, CA, USA, February 2017.
- [21] M. Čáp, J. Gregoire, and E. Frazzoli, "Provably safe and deadlock-free execution of multi-robot plans under delaying disturbances," in *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5113–5118, Daejeon, South Korea, October 2016.
- [22] J. Gregoire, M. Čáp, and E. Frazzoli, "Locally-optimal multi-robot navigation under delaying disturbances using homotopy constraints," *Autonomous Robots*, vol. 42, no. 4, pp. 895–907, 2018.
- [23] J. Gregoire, "Priority-based coordination of mobile robots," 2014, <http://arxiv.org/abs/1410.0879>.
- [24] J. Hershberger, M. Maxel, and S. Suri, "Finding the k shortest simple paths: a new algorithm and its implementation," *ACM Transactions on Algorithms*, vol. 3, no. 4, p. 45, 2007.