Applied Intelligence (2022) 52:429–451
https://doi.org/10.1007/s10489-021-02397-0

Prioritized planning algorithm for multi-robot collision avoidance based on artificial untraversable vertex

Haodong Li[1] · Tao Zhao[1] · Songyi Dian[1]

Accepted: 27 March 2021 / Published online: 3 May 2021

Abstract
This paper presents a method to avoid collisions and deadlocks between mobile robots working collaboratively in a shared physical environment. Based on the shared knowledge of the robot's direction and coordinates, we define five conflict types between robots and propose a new concept named Artificial Untraversable Vertex (AUV) to resolve the potential conflicts. Since conflict avoidance between robots is typically a real-time process, a heuristic search algorithm D* Lite with fast replanning characteristics is introduced. Once a robot finds that it may collide with another robot while moving along the preplanned path, a new conflict-free path can be calculated based on the AUV and D* Lite. The experimental results demonstrate that the proposed Multi-Robot Path Planning (MRPP) method can effectively avoid collisions and deadlocks between mobile robots.

Keywords Multi-robot path planning · Conflict type · Collision avoidance · D* lite · Artificial untraversable vertex

1 Introduction

Multi-Robot Systems (MRSs) have advantages over single-robot systems in terms of spatial distribution [1], robustness [2], scalability [3], and flexibility [4]. During the last decade, MRS has evolved from an isolated anecdotal laboratory system to robust deployment in many fields [5], especially when a given task requires the cooperation of robot teams, such as cleaning [6], service [7], space-exploring [8, 9], warehousing [10], coverage [11], search-and-rescue [12], and military [13]. To date, many research efforts have focused on different research topics related to MRS, including communication techniques [14], vision systems [15], and navigation [16, 17]. Mobile navigation is typically realized through perception, planning, and control [18]. The objective of perception is to obtain environmental information around the robot for path planning, and the objective of control is to make the mobile robot follow the planned path. Therefore, path planning is a fundamental problem of MRS, and its objective is to move all robots working in a shared environment to their respective goal positions while meeting performance constraints (e.g., travel time) and safety constraints (e.g., collision avoidance) [19]. This paper focuses on a decentralized MRPP approach whose objective is to avoid potential collisions and deadlocks [20] with other robots in real-time while minimizing the deviation from the preplanned optimal path.

The collision between robots is a crucial problem in the path planning and motion control of MRSs and has not received sufficient attention. Obstacle and interrobot collision avoidance directly impact system safety, so it should be the highest priority among all objectives [21]. A simple method of avoiding collisions is to treat other moving robots as obstacles, namely, untraversable vertices [22] in the grid map. However, as shown in Fig. 1, the method cannot prevent robot 2 (R2) from planning D3 as the next vertex, which is likely to cause another collision with robot 1 (R1).

The deadlock between robots is another vital issue in MRPP since it can degrade system performance or even crash the system [23]. As shown in Fig. 2, the deadlock between mobile robots typically occurs when robots block each other in a way so that none of them can follow its predefined path without causing a collision.

Moreover, the prioritized planning algorithm typically plans each robot's path individually and then combines other methods or strategies to resolve the conflicts between robots. The established paths are typically optimal in the grid environment, so minimizing the deviation between the modified path and the initial path helps maintain the path optimality. As shown in Fig. 3, R1 collides with R2 as it follows the

✉ Tao Zhao
zhaotaozhaogang@126.com

[1] College of Electrical Engineering, Sichuan University, Chengdu 610065, China

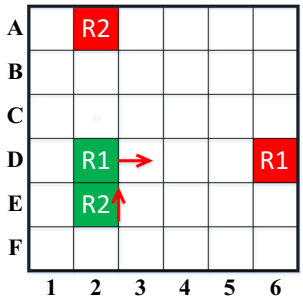


Fig. 1 A problematic situation of collision avoidance by treating other robots as obstacles. The green and the red vertices denote the robot positions and goal positions, and the arrow denotes the direction of the robot

preplanned path and vice versa. A decentralized approach may make the optimal decision locally to move R1 to vertex C3 and then head south to reach the target, but it causes unnecessary deviations if R3 can move diagonally.

The motivation of this research is to solve the challenges mentioned above in implementing the prioritized MRPP approach, that is, (1) what is the effect when a robot lies in the path of another robot, namely, interrobot collision avoidance, (2) how to move robots when they block each other, namely, deadlock avoidance, and (3) how to ensure that the actions taken to avoid collisions and deadlocks with other robots do not exceed the minimum effort required, that is, to minimize the deviation between the modified path and preplanned path. This paper proposes a Multi-Robot D* Lite (MR-D* Lite) algorithm to solve these problems. The main idea of MR-D* Lite is to employ the D* Lite algorithm to calculate the initial path of each robot individually, and then combine the AUV to modify the path of the robot with a potential collision or deadlock risk. The main contributions of this paper can be summarized as follows:

- Many researchers focus on collision avoidance between robots, but deadlock avoidance and path deviation are typically not considered simultaneously. This paper defines five types of potential conflicts between mobile robots and proposes three types of AUVs to resolve the conflicts. The AUV considers both the position and direction of robots, thereby solving the conflicts between robots with a minor or minimum deviation from the predetermined optimal path.
- The dynamic characteristics of D* Lite are used to address the conflicts between robots while minimizing the time to replan a new collision-free and deadlock-free path for each robot.

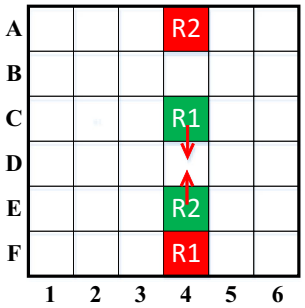


Fig. 3 A problematic situation of decentralized methods where the shortest path is the objective [24]

The remainder of this paper is organized as follows: Section 2 introduces the related works, problem definition, and underlying algorithm. Section 3 provides the proposed method. The results and the conclusion are presented in Sections 4 and 5, respectively.

2 Preliminaries

2.1 Related works

MRPP can be characterized into two categories: centralized and decentralized approaches [25, 26]. The centralized approach treats the path planning issue as an optimization problem via a central server, which has comprehensive knowledge about all robots' workspace (e.g., a 2D grid map) and intents (e.g., speed, start position, and goal position). The centralized approach mainly includes four categories [27]: protocol-based [28], conflict-based search [29], increasing cost tree search [30], and A* search expansion [31]. It plans collision-free

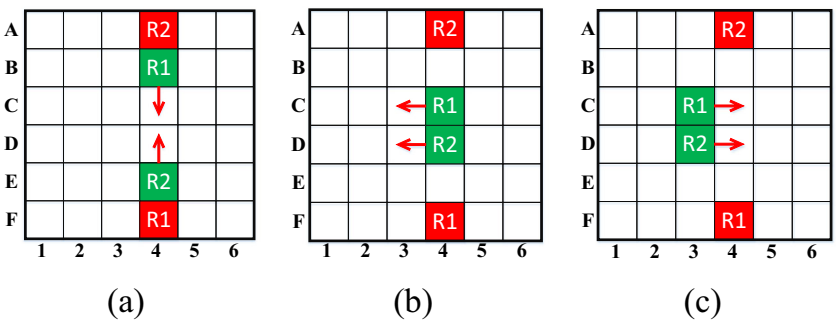


Fig. 2 Illustration of the deadlock between robots [24]. **a** The initial state of robots. **b** and **c** The deadlock condition is encountered and repeated in-between (**b**) and (**c**) infinitely

paths for all robots simultaneously and can produce globally optimal plans. However, the time complexity is exponentially related to the composite configuration space dimensions and the number of robots. Thus, it is typically applicable to simple problems that involve two or three robots, and its performance can be low if tasks/goals are reassigned frequently. Moreover, the approach relies heavily on a reliable communication network between the central server and robots in practice. Thus, MRS will breakdown if the central server or communication network fails. Furthermore, the centralized approach is inapplicable when the environment is unknown and unstructured [32]. In recent years, nature-inspired optimization algorithms have been applied to various versions of centralized approaches to address the MRPP optimization problem [21], which include Particle Swarm Optimization (PSO) [33], Gravitational Search Algorithms (GSA) [34], firefly optimization [35], Charged System Search (CSS) [36], Ant Colony Optimization (ACO) [37], Artificial Bee Colony (ABC) [38], and Genetic Algorithm (GA) [39]. Most of these algorithms can find better paths as the number of iterations increases, but they are computationally expensive.

Compared with centralized approaches, some researchers have proposed decentralized conflict avoidance strategies, where each robot takes the observable states (e.g., velocities, shapes, and positions) of other robots as inputs to make autonomous decisions. In contrast to a centralized approach, a decentralized approach can better respond to unknown or dynamic environments and has better reliability, flexibility, adaptability, and fault tolerance [40]. It can be further categorized into two approaches: the path coordination approach and the prioritized planning approach. In a path coordination approach, the robot moves in a coordinated path. If a potential collision is detected, it changes robot velocity to predefined values or even stops robots to avoid collision [19]. However, stopping or slowing down increases the time for the robot to complete the task. In practice, a widely used decentralized approach for MRPPs that has proven effective is prioritized planning. The prioritized planning algorithm assigns a unique priority to each robot and proceeds sequentially from the robot with the highest priority to the one with the lowest priority [41]. The prioritized planning approaches plan the path for each robot individually and then combine other methods to avoid collisions between robots, which include Artificial Potential Field (APF) [42], A* [43], Revised Prioritized Planning (RPP) [44], Safe Interval Path Planning (SIPP) [45], and Hybrid Path Planning (HPP) [46]. Nonetheless, the work proposed above typically focuses on interrobot collision avoidance, but deadlock avoidance and path deviation are typically not considered simultaneously.

2.2 Problem definition

We model each robot as a point that can move freely in a two-dimensional space. It is assumed that each robot is equipped with several sensors by which a robot can build a global map of the working environment. The robot may find new obstacles or robots when moving, so it also builds a local map to save dynamic information on the surrounding environment. Due to the various local information surrounding the robots, each robot's grid map may vary slightly. All the maps mentioned above are composed of finite vertices, and some of the vertices are occupied by obstacles. The configuration of obstacles in different vertices may vary in shape, position, and occupancy level. However, this paper assumes that a vertex is either unknown, free, or entirely occupied by obstacles. The occupied vertices of the global and local maps can represent the shape and location of obstacles. The given start position and desired goal position of each robot are located in the free vertex, and MRPP can be defined as finding a finite set of free vertices to navigate each robot between the start and goal points without interrobot collisions and deadlocks [47].

Let N_r be the total number of robots in an MRS, N_p be the total number of vertices in a path, $\mathbb{N}_r = \{1, 2, ..., N_r\}$, and $\mathbb{N}_p = \{1, 2, ..., N_p\}$. The path of robots can be denoted as

$$P = \left\{ p_j^i \right\}, i \in \mathbb{N}_r, j \in \mathbb{N}_p, \tag{1}$$

where the superscript i denotes the robot number of the MRS, and the subscript j is the vertex number of the path. We assume that it takes the same time for the robot to move from one vertex to another. Therefore, an interrobot collision will occur if $\exists\, x, y \in \mathbb{N}_r, x \neq y, j \in \mathbb{N}_p$, such that $p^x{}_j = p^y{}_j$. Moreover, a deadlock will occur if $\exists\, i \in \mathbb{N}_r$, such that $\exists\, p^i{}_{j1} = p^i{}_{j2} = ... = p^i{}_{jn}$, where $\forall\, n > 1, n \in \mathbb{N}_p$.

2.3 D* Lite

We use D* Lite as our method's underlying algorithm. D* Lite is a well-known goal-directed path planning algorithm and widely utilized for autonomous mobile robot navigation [48]. It can reuse the previous path planning information to replan a path dynamically. This characteristic can be applied to our method for dynamic collision avoidance between robots. The following notation is utilized to describe D* Lite: S is the set of vertices of the graph. $Succ(s) \subseteq S$ is the set of successors of vertex $s \in S$. $Pred(s) \subseteq S$ is the set of predecessors of vertex $s \in S$. $0 < c(s', s) < \infty$ is the cost of moving from vertex $s' \in Pred(s)$ to vertex s. D* Lite maintains two kinds of estimates of each vertex's goal distance: a g-value $g(s)$ and an rhs-value $rhs(s)$. The rhs-value always satisfies the following relationship:

$$rhs(s) = \begin{cases} 0, & if\ s = s_{goal} \\ \min_{s' \in Succ(s)} (g(s') + c(s, s')), & \text{otherwise} \end{cases}, \tag{2}$$

where a vertex is called locally consistent if $g(s) = rhs(s)$; otherwise, it is called locally inconsistent.

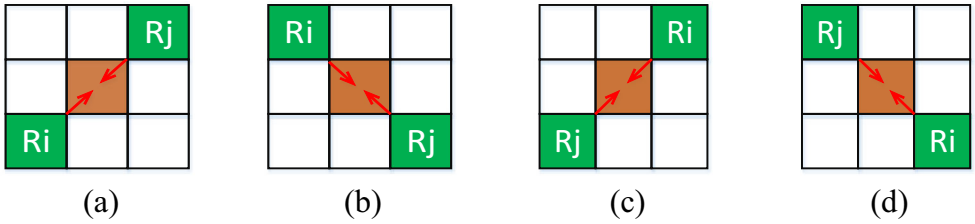


Fig. 4 CST1: Robots move diagonally and their next position is the conflict vertex (i.e., brown square)

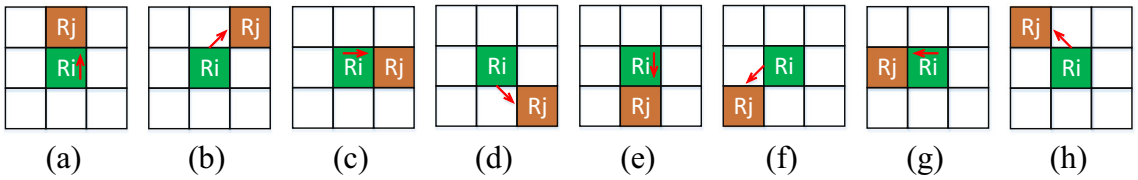


Fig. 5 CST2: The current position of Rj is the conflict vertex

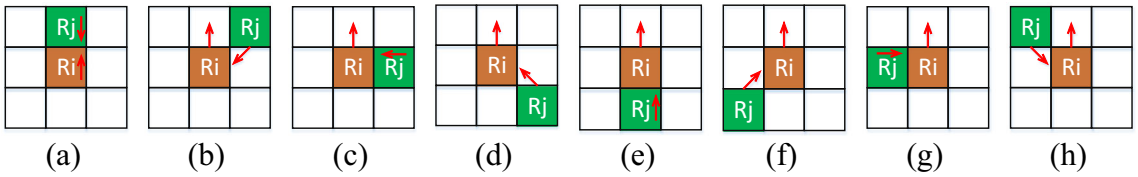


Fig. 6 CST3 when Ri moves north

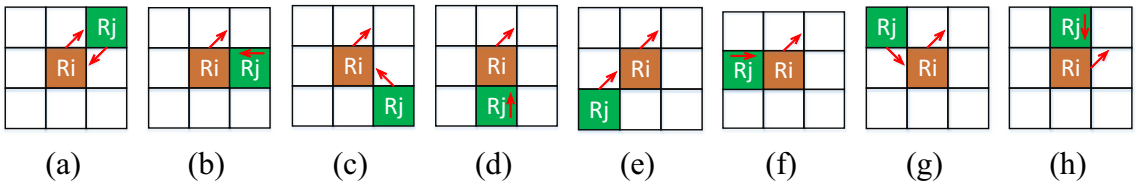


Fig. 7 CST3 when Ri moves northeast

3 Methodology

This section classifies the conflict scenarios into five types according to the vertex position where the path conflicts. Before moving to the next vertex, each robot judges whether it is in any five conflict types with other robots. Then we set the AUV for the robot in conflict and replan the path.

3.1 Conflict types

Some researchers have defined conflict types for pairs of robots [49], but some scenarios have not been discussed. In a discrete gridded environment, a robot can only move to its neighboring eight vertices, so the vertex where robots collide (i.e., conflict vertex) must be one of their current vertices and neighboring vertices. In the following, robot i (Ri) is the higher priority robot, and robot j (Rj) is the lower priority robot. According to the position of the conflict vertex, we classify the conflict scenarios into five types. Let $p^i{}_c$ and $p^j{}_c$ be the current positions of Ri and Rj, respectively. As shown in Fig. 4, if Ri and Rj move diagonally and $p^i{}_{c+1} = p^j{}_{c+1}$, the robots are in the Type 1 Conflict Scenario (CST1).

As shown in Fig. 5, if $p^i{}_{c+1} = p^j{}_c$ and $p^i{}_c \neq p^j{}_{c+1}$, the robots are in the Type 2 Conflict Scenario (CST2).

As shown in Figs. 6, 7, 8, 9, 10, 11, 12 and 13, if $p^i{}_c = p^j{}_{c+1}$, the robots are in the Type 3 Conflict Scenario (CST3).

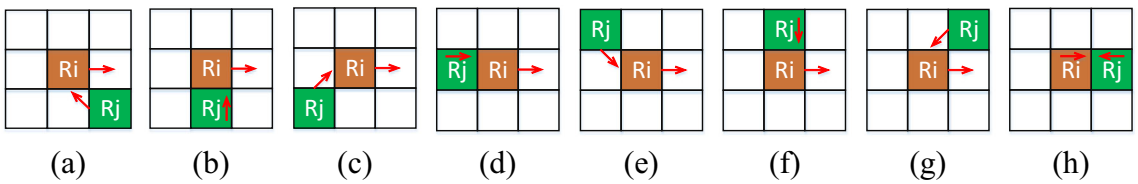


Fig. 8 CST3 when Ri moves east

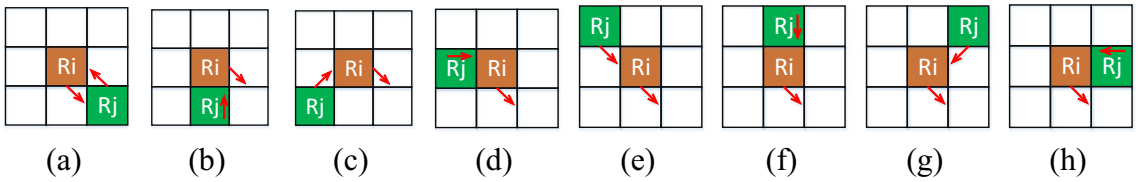


Fig. 9 CST3 when Ri moves southeast

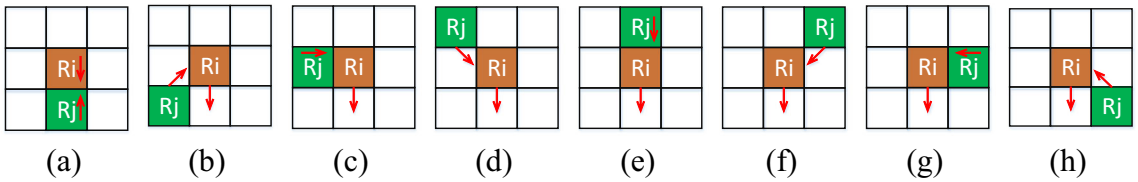


Fig. 10 CST3 when Ri moves south

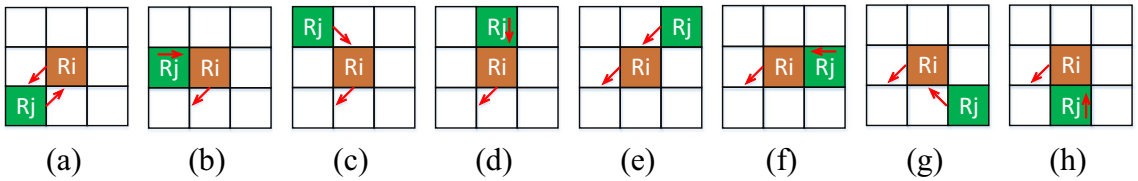


Fig. 11 CST3 when Ri moves southwest

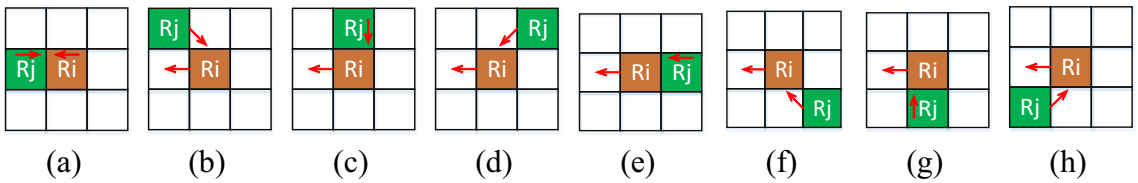


Fig. 12 CST3 when Ri moves west

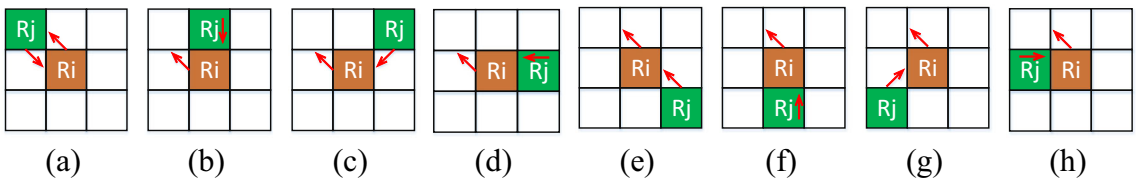


Fig. 13 CST3 when Ri moves northwest

As shown in Figs. 14, 15, 16, 17, 18, 19, 20 and 21, if Ri and Rj are adjacent to each other and $p^i{}_{c+1} = p^j{}_{c+1}$, the robots are in the Type 4 Conflict Scenario (CST4).

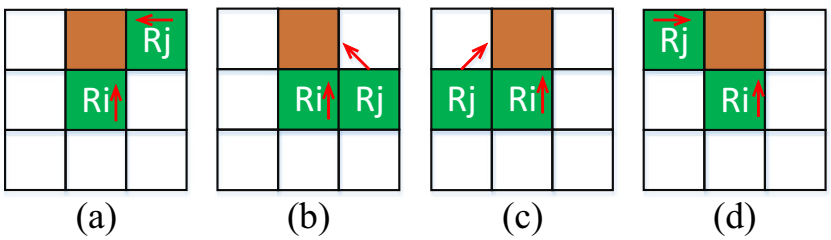


Fig. 14 CST4 when Ri moves north

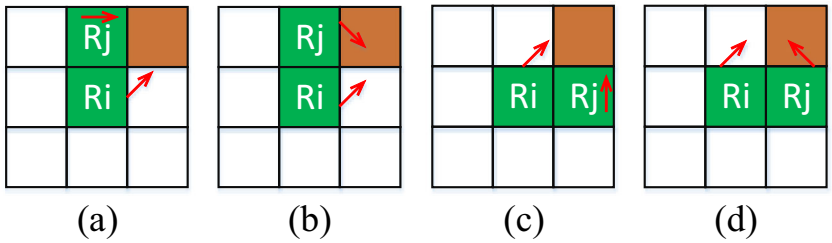


Fig. 15 CST4 when Ri moves northeast

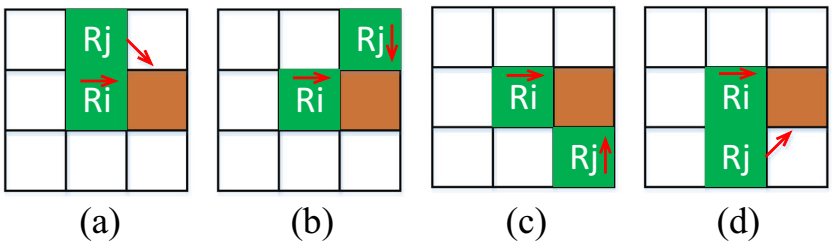


Fig. 16 CST4 when Ri moves east

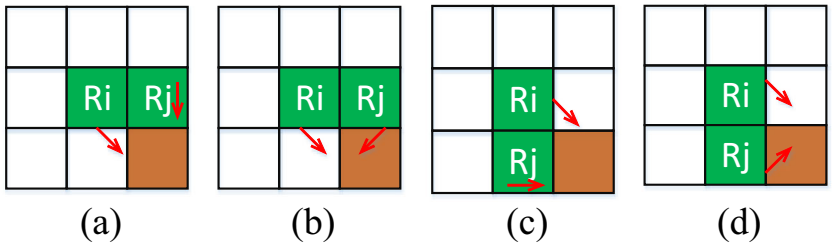


Fig. 17 CST4 when Ri moves southeast

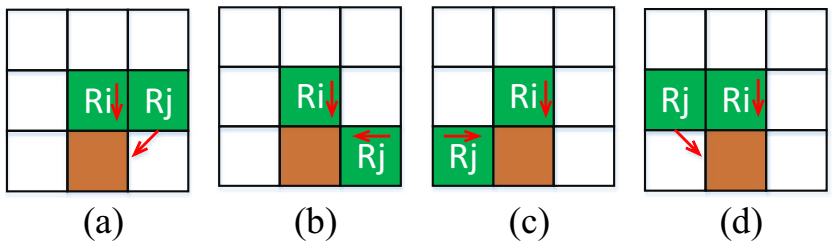


Fig. 18 CST4 when Ri moves south

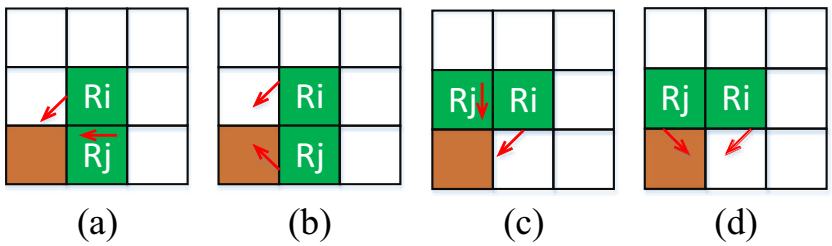


Fig. 19 CST4 when Ri moves southwest

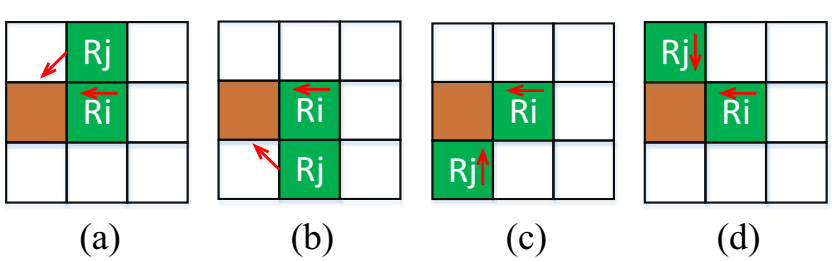


Fig. 20 CST4 when Ri moves west

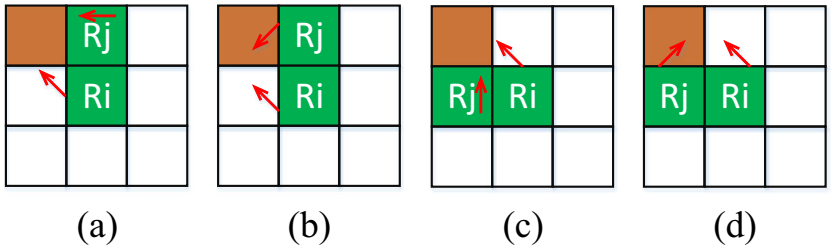


Fig. 21 CST4 when Ri moves northwest

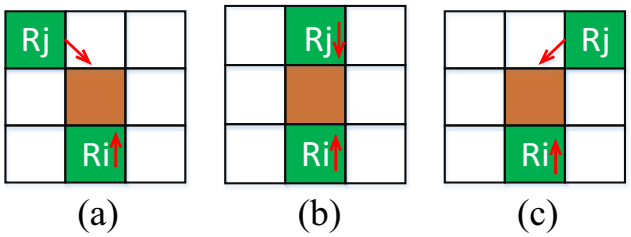


Fig. 22 CST5 when Ri moves north

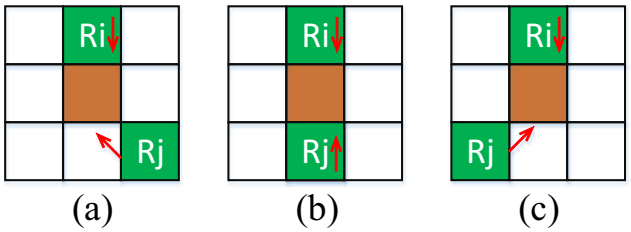


Fig. 24 CST5 when Ri moves south

As shown in Figs. 22, 23, 24, 25, 26, 27, 28 and 29, if $p^{i}{}_{c+1} = p^{j}{}_{c+1}$, Ri and Rj are not adjacent, and are not in CST1, then the robots are in the Type 5 Conflict Scenario (CST5).

3.2 Artificial untraversable vertex

The AUV is the basis for the underlying path planning algorithm to replan the path to avoid collisions and deadlocks. The number and position of the AUV can be adjusted according to needs (e.g., the number of vertices occupied by a robot). In this paper, we assume that each robot occupies a vertex. We summarized three types of AUVs during the experiment, as shown in Figs. 30, 31 and 32, which produced a minimum or slight path deviation and steering angle for collision and deadlock avoidance. In Figs. 30-32, according to the coordinate and direction of robot R, the AUV is set for the original robot (Ro), which conflicts with R. There are eight cases for each type of AUV corresponding to the eight moving directions of R. Figure 30 shows the cases for Type 1 AUV (AUVT1), which is applied to both higher priority and lower priority robots. The AUVT1 set for R_O is shown in Fig. 30 (a) when R moves northward and Figs. 30 (b)-(h) correspond to the other seven moving directions.

Figure 31 presents the eight cases for Type 2 AUV (AUVT2) applied to the lower priority robots. The AUVT2 set for R_O is shown in Fig. 31 (h) when R moves northwest, and Fig. 31 (a)-(g) correspond to the other seven moving directions.

Figure 32 shows eight cases for Type 3 AUV (AUVT3), and it is applied to the lower priority robots. The AUV set for R_O is shown in Fig. 32 (h) when R moves northwest, and Fig. 32 (a)-(g) corresponds to the other seven moving directions.

Figures 33, 34, 35, 36 and 37 list the solution for CST1–5. Robot Ri is the higher priority robot, and robot Rj is the lower priority robot. The graph is an eight-connected grid whose edge costs are initially one and change to infinity when the robot discovers that the vertices cannot be traversed [48]. We assume that the dynamic environment map range is the 48 vertices in the shadow around the robot. For CST1 in Fig. 4, we make the higher priority robot Ri continue to move along its established path, and the lower priority robot Rj bypasses the brown conflict vertex, that is, $p^{i}{}_{c+1}$ is set as the AUV for robot Rj. Taking Fig. 4 (a) as an example, we set vertex G4 as

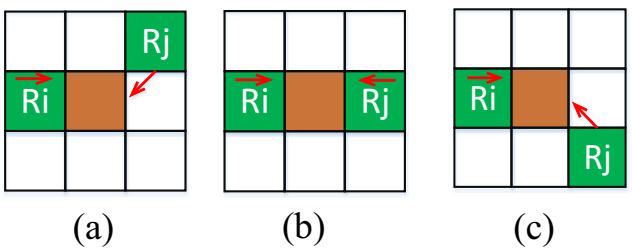


Fig. 23 CST5 when Ri moves east

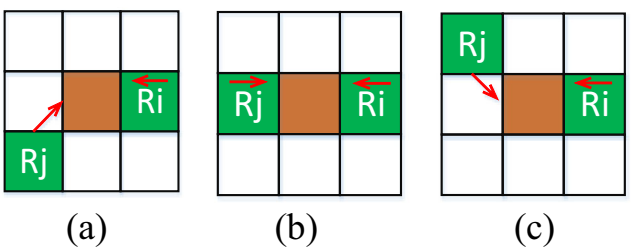


Fig. 25 CST5 when Ri moves west

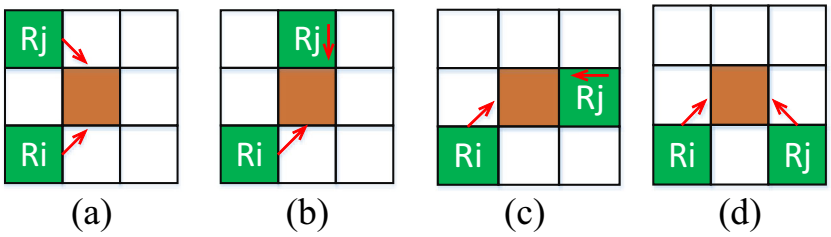


Fig. 26 CST5 when Ri moves northeast

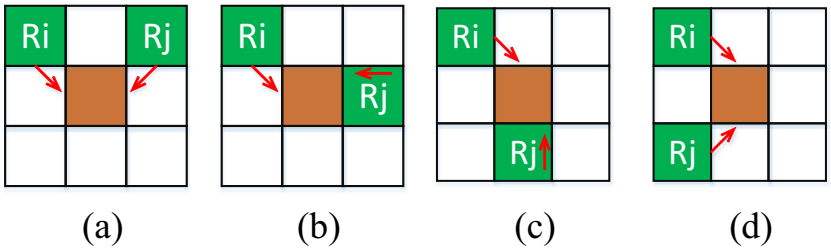


Fig. 27 CST5 when Ri moves southeast

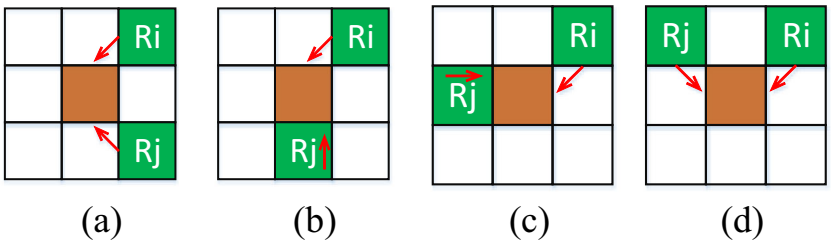


Fig. 28 CST5 when Ri moves southwest

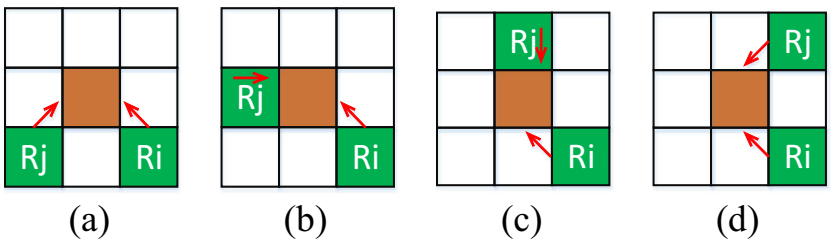


Fig. 29 CST5 when Ri moves northwest

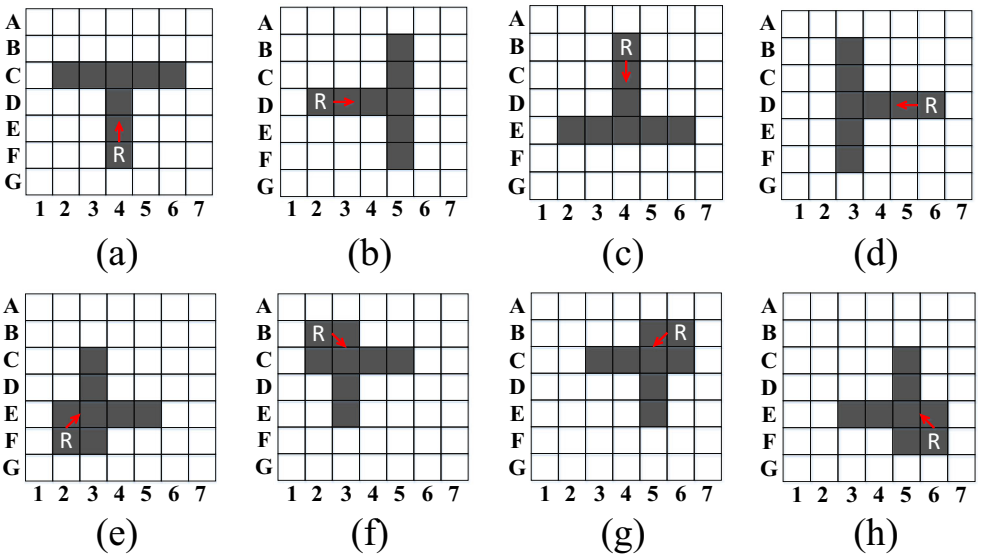


Fig. 30 Illustration of AUVT1. The vertex with the term R is the current position of R, and the arrow signifies its direction. The black vertex represents the AUV set for the robot Ro

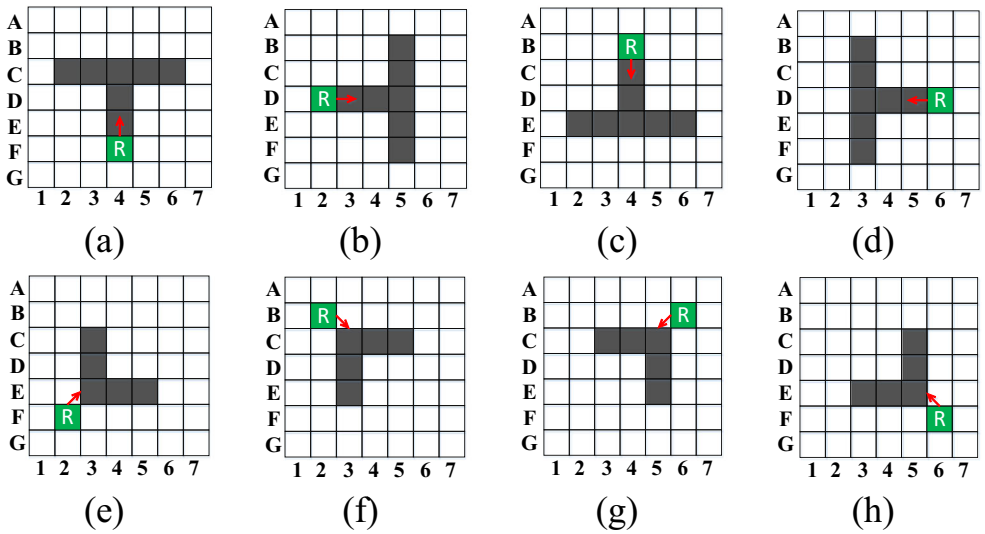


Fig. 31 Illustration of AUVT2

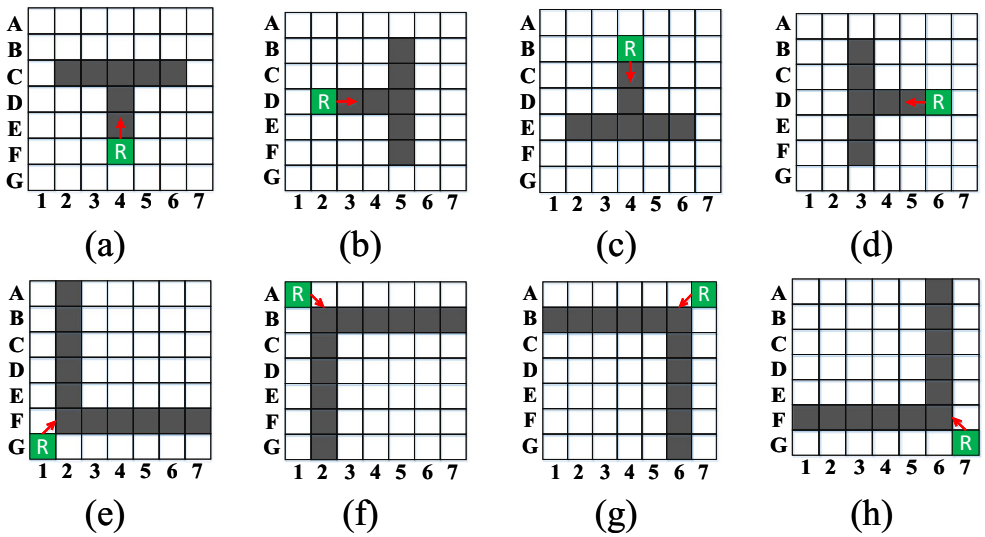


Fig. 32 Illustration of AUVT3

the AUV for Rj as shown in Fig. 33, and Rj's moving direction changes to the south.

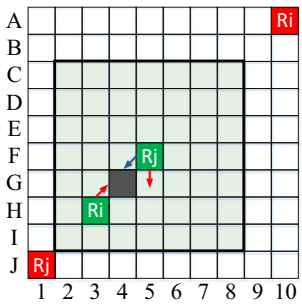


Fig. 33 Solution for CST1 (taking Fig. 4 (a) as an example). The vertex G4 is the AUV for Rj, the black square is the AUV for the robot Rj, the green and red squares are the robot and goal positions, and the blue and red arrows are the initial and modified directions

For CST2 in Fig. 5, we make the lower priority robot Rj continue to move along its established path because its direction does not conflict with Ri. To make Ri's modified path not conflict with Rj's established path, we considered both Rj's position and direction to set the AUV for Ri. Moreover, Rj is also likely to update its path due to other robots. Thus, we set the AUV for both Ri and Rj according to AUVT1. Taking Fig. 5 (a) as an example, the AUVs for Ri and Rj are set as shown in Fig. 34, and Ri's direction changes to the northwest.

For CST3 in Figs. 6-13, $p^j{}_{c+1}$ is occupied by Ri, so Rj should replan its path, and Ri needs to bypass Rj's current vertex. In MR-D* Lite, we replan the path of robots from the higher priority to the lower priority. Therefore, we set Rj's current vertex as the AUV for Ri and set the AUV for Rj according to AUVT1. Taking Fig. 6 (a) as an example, we set the AUV as shown in Fig. 35. After the replanning process,

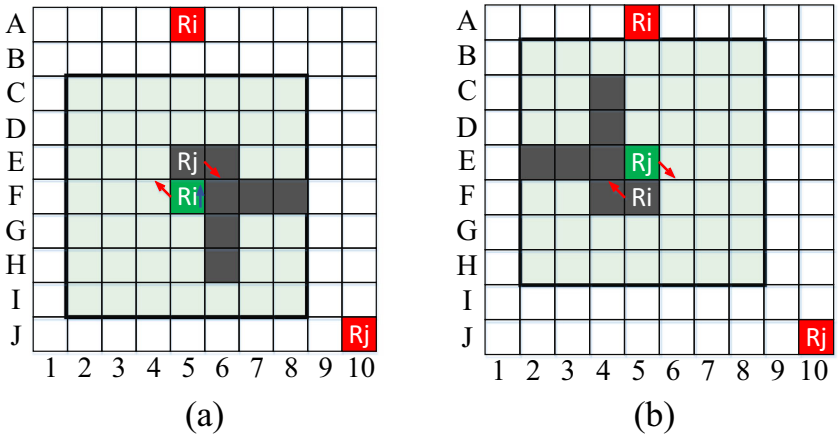


Fig. 34 Solution for CST2 (Taking Fig. 5 (a) as an example). **a** The AUV for Ri. **b** The AUV for Rj

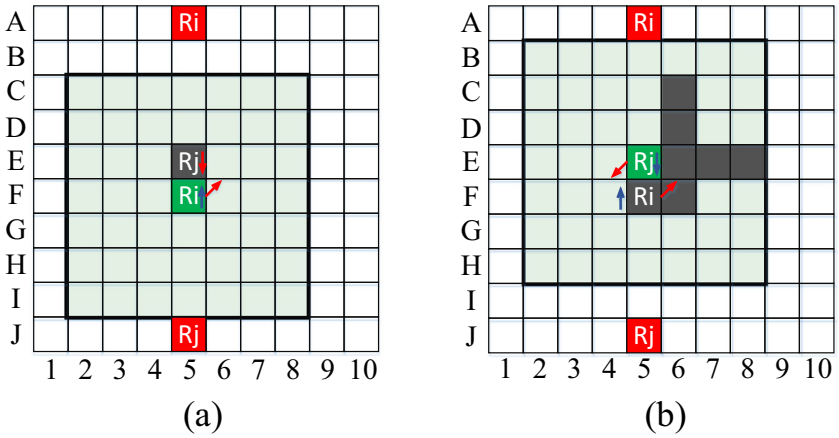


Fig. 35 Solution for CST3 (Taking Fig. 6 (a) as an example). **a** The AUV for Ri. **b** The AUV for Rj

Ri's direction changes to the northeast, and Rj's direction changes to the southwest.

For CST4 in Figs. 14-21, we make the higher priority robot Ri continue to move along its established path. However, to prevent Ri's modified path (Ri is likely to update the path due to other robots in MRS) from conflicting with Rj's current vertex, we set Rj's current vertex as the AUV for Ri. Then, we set the AUV for Rj according to AUVT1. That is, the solution for CST3 is the same as that for CST4. Taking Fig. 18 (a) as an example, the AUV is set as shown in Fig. 36, and Rj's direction changes to the northwest.

For CST5 in Fig. 22-29, we make Ri continue to move along its established path and set the AUV for Rj according to AUVT2. Taking Fig. 22 (a) as an example, corresponding

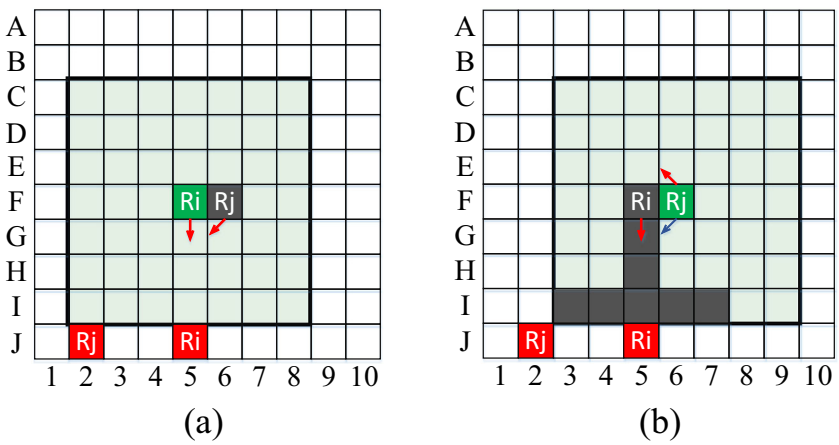


Fig. 36 Solution for CST4 (Taking Fig. 18 (a) as an example). **a** The AUV for Ri. **b** The AUV for Rj

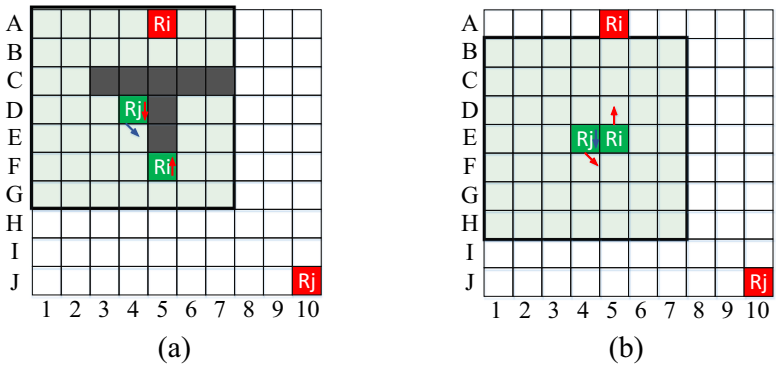


Fig. 37 Solution for CST5 (Taking Fig. 22 (a) as an example). **a** The AUV for Rj. **b** The robots after moving one step

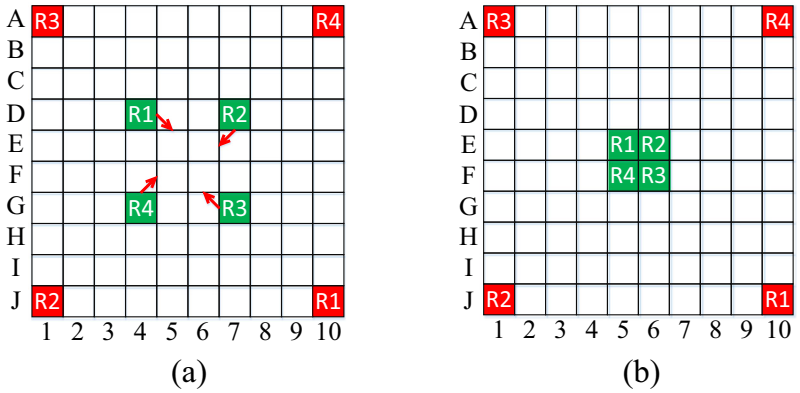


Fig. 38 A problematic situation for the MRS of three or more robots. **a** Initial positions of robots. **b** Robots crowded together

to Fig. 31 (a), the AUV for Rj is set as shown in Fig. 37 (a), and Rj's direction changes to south. Note that all AUVs are cleared after the robot moves to the next vertex. Therefore, after the robots in Fig. 37 (a) move one step and reach the position in Fig. 37 (b), all the AUVs for Rj have disappeared. Then, the path is replanned again, and Rj's direction changes to the original southeast.

The method mentioned above can solve conflicts between any two robots, but there is a problem of crowding between robots for the MRS with three or more robots. For example, if the robots in Fig. 38 (a) reach the position in Fig. 38 (b), then it is challenging to replan a collision-free path for each robot. AUVT3 is employed to address this problem, as shown in Fig. 39.

If the vertices $p^i{}_{c+1}$ and $p^j{}_{c+1}$ are adjacent to each other, we set the AUV for the lower priority robot according to AUVT3. With Fig. 38 (a) as an example, the AUV for the robots is set as shown in Fig. 39. In Fig. 38 (a), the lower priority robot R2 detects that $p^2{}_{c+1}$ is adjacent to $p^1{}_{c+1}$. Thus, corresponding to Fig. 32 (f), we set the AUV for R2 as shown in Fig. 39 (a), and its direction changes to the west after replanning the path. In Fig. 39 (a), the lower priority robot R3 detects that $p^3{}_{c+1}$ is adjacent to $p^1{}_{c+1}$. We set the AUV for R3 as shown in Fig. 39 (b), and its direction

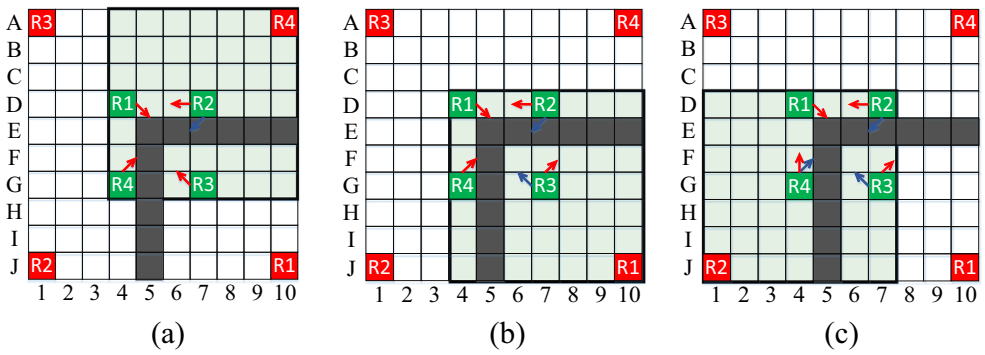


Fig. 39 Solution for crowding between robots (with Fig. 38 (a) as an example). **a** The AUV for R2. **b** The AUV for R3. (c) The AUV for R4

Algorithm – MR-D* Lite

procedure CalcKey(*i*, *s*)
{01} return [min($g(i, s)$, $rhs(i, s)$) + $h(s_{start(i)}, s)$ + $k_{m(i)}$; min($g(i, s)$, $rhs(i, s)$)];
procedure Initialize(*i*)
{02} U_i=∅;
{03} $k_{m(i)}$ = 0;
{04} for all $s \in S$, $rhs(i, s) = g(i, s) = \infty$;
{05} $rhs(i, s_{goal(i)}) = 0$;
{06} U.Insert(i, $s_{goal(i)}$, CalcKey(i, $s_{goal(i)}$));
procedure UpdateVertex(*i*, *u*)
{07} if ($u \neq s_{goal(i)}$), $rhs(i, u) = \min_{s' \in Succ(u)}(c(i, u, s') + g(i, s'))$
{08} if ($u \in U_i$), U.Remove(u);
{09} if ($g(i, u) \neq rhs(i, u)$), U.Insert(i, u, CalcKey(i, u));
procedure ComputeShortestPath(*i*)
{10} while (U.TopKey(i) < CalcKey(i, $s_{start(i)}$) OR $rhs(i, s_{start(i)})$ ⑬ $g(i, s_{start(i)})$)
{11} k_{old} = U.TopKey(i);
{12} u = U.Pop(i);
{13} if (k_{old} < CalcKey(i, u))
{14} U.Insert(i, u, CalcKey(i, u));
{15} else if ($g(u) > rhs(u)$)
{16} $g(i, u) = rhs(i, u)$;
{17} for all $s \in Pred(u)$, UpdateVertex(i, s);
{18} else
{19} $g(i, u) = \infty$;
{20} for all $s \in Pred(u) \cup \{u\}$, UpdateVertex(i, s);
procedure ScanGraph(*i*)
{21} Scan graph(i) for changed edge costs;
{22} if any edge costs changed
{23} If $k_{m(i)}$ has not changed when UpdatePath(i) is called
{24} $k_{m(i)} = k_{m(i)} + h(s_{last(i)}, s_{start(i)})$;
{25} $s_{last(i)} = s_{start(i)}$;
{26} for all directed edges (i, u, v) with changed edge costs
{27} Update the edge cost $c(i, u, v)$;
{28} UpdateVertex(i, u);
{29} ComputeShortestPath(i);
procedure UpdatePath(*i*)
{30} Clear all untraversable vertices;
{31} ScanGraph(i);
{32} if $i > 1$
{33} for $j = 1 : i - 1$
{34} $s_{next(i)} = \arg\min_{s' \in Succ(s_{start(i)})}(c(i, s_{start(i)}, s') + g(i, s'))$;
{35} $s_{next(j)} = \arg\min_{s' \in Succ(s_{start(j)})}(c(j, s_{start(j)}, s') + g(j, s'))$;
{36} if CST1(Ri, Rj)
{37} Set $s_{next(j)}$ as the AUV for Ri;
{38} else if CST2(Ri, Rj) OR CST3(Ri, Rj) OR CST4(Ri, Rj)
{39} Set the AUV for Ri according to AUVT1;
{40} else if CST5(Ri, Rj)
{41} Set the AUV for Ri according to AUVT2;
{42} else if $s_{next(i)}$ is adjacent to $s_{next(j)}$
{43} Set the AUV for Ri according to AUVT3;
{44} if $s_{start(i)}$ is adjacent to $s_{start(j)}$
{45} Set $s_{start(j)}$ as the AUV for Ri;
{46} ScanGraph(i);
{47} if $i < N$
{48} for $j = i + 1 : N$
{49} $s_{next(i)} = \arg\min_{s' \in Succ(s_{start(i)})}(c(i, s_{start(i)}, s') + g(i, s'))$
{50} $s_{next(j)} = \arg\min_{s' \in Succ(s_{start(j)})}(c(j, s_{start(j)}, s') + g(j, s'))$
{51} if CST2(Ri, Rj)
{52} Set the AUV for Ri according to AUVT1;
{53} else if $s_{start(i)}$ is adjacent to $s_{start(j)}$
{54} Set $s_{start(j)}$ as the AUV for Ri;
{55} ScanGraph(i);
procedure Main()
{56} for $i = 1 : N$
{57} $s_{last(i)} = s_{start(i)}$;
{58} Initialize(i);
{59} ComputeShortestPath(i);
{60} while ($s_{start(1)} \neq s_{goal(1)}$ OR $s_{start(2)} \neq s_{goal(2)}$ … OR $s_{start(N)} \neq s_{goal(N)}$)
{61} for $i = 1 : N$
{62} if ($s_{start(i)} \neq s_{goal(i)}$)
{63} UpdatePath(i);
{64} for $i = 1 : N$
{65} /* if ($g(s_{start(i)}) = \infty$) then there is no known path */
{66} $s_{start(i)} = \arg\min_{s' \in Succ(s_{start(i)})}(c(i, s_{start(i)}, s') + g(i, s'))$;
{67} Move to $s_{start(i)}$;

Fig. 40 The pseudocode for MR-D* Lite

changes to the northeast. In Fig. 39 (b), the lower priority robot R4 detects that $p^4{}_{c+1}$ is adjacent to t $p^1{}_{c+1}$. We establish the AUV for R4 as shown in Fig. 39 (c), and its direction changes to the north.

In summary, the method of setting the AUV for robots is as follows:

- For the CST1 in Fig. 4, we set the conflict vertex as the AUV for Rj.
- For CST2 in Fig. 5, we set the AUV for Ri and Rj according to AUVT1.
- For CST3 and CST4 in Figs. 6-21, we set $p^j{}_c$ as the AUV for Ri and set the AUV for Rj according to AUVT1.
- For CST5 in Figs. 22-29, we set the AUV for Rj according to AUVT2.
- If $p^i{}_{c+1}$ and $p^j{}_{c+1}$ are adjacent to each other, we set the AUV for Rj according to AUVT3.
- If $p^i{}_c$ and $p^j{}_c$ are adjacent to each other, we set $p^i{}_c$ and $p^j{}_c$ as the AUVs for each other.

3.3 Multi-robot D* lite

We now use D* Lite and AUV to develop our MR-D* Lite, whose pseudocode is shown in Fig. 40. The definitions of symbols used in the pseudocode are summarized in Table 1. Similar to the D * Lite algorithm, MR-D* Lite maintains two kinds of estimates of the goal distance of each vertex: a g-value $g(i, s)$ and an rhs-value $rhs(i, s)$, where i denotes the robot number of the MRS. The rhs-value always satisfies the following relationship:

$$rhs(i,s) = \begin{cases} 0, & if\ s = s_{goal} \\ \min_{s \in Succ(s)}(g(i,s) + c(i,s,s')), & \text{otherwise} \end{cases}, \tag{3}$$

Table 1 Symbol definitions for MR-D* Lite

Symbol	Meaning
N_r	Number of robots
S	Set of vertices of the graph
$s_{start(i)}$	Start vertex of Ri
$s_{goal(i)}$	Goal vertex of Ri
$h(s_{start(i)}, s)$	Distance between vertex s and $s_{start(i)}$.
$k_{m(i)}$	Key modifier of Ri
U_i	Priority queue of Ri
$g(i, s)$, $rhs(i, s)$	Estimates of the goal distance from s to $s_{goal(i)}$
$Succ(s)$	Set of successors of the vertex s
$Pred(s)$	Set of predecessors of the vertex s
$c(i, s', s)$	Cost of moving Ri from the vertex s' to $s \in Succ(s')$

where a vertex is called locally consistent for Ri if $g(i, s) = rhs(i, s)$; otherwise, it is called locally inconsistent. The distance between vertex s and $s_{start(i)}$ uses the Chebyshev distance:

$$h\left(s_{start(i)}, s\right) = \max\left(\left|x_{s_{start(i)}} - x_s\right|, \left|y_{s_{start(i)}} - y_s\right|\right), \quad (4)$$

where x_s and y_s symbolize the x-coordinate and y-coordinate of vertex s, respectively.

The functions CalcKey(), Initialize(), UpdateVertex(), and ComputeShortestPath() in D* Lite remain unchanged to plan the initial paths and modify them when potential conflicts are detected and the environmental information changes. However, the function Main() needs to be extended, and the functions UpdatePath() and ScanGraph() are introduced to address the collisions and deadlocks between different robots. The main function Main() first calls Initialize(i) to initialize the search problem for each robot {58}. (Numbers in curly brackets refer to line numbers in the pseudocode.) Initialize(i) sets the initial g-values of all vertices of Ri to infinity and sets their rhs-values according to Eq. (3) {04–05}. Note that in an actual implementation, Initialize(i) only needs to initialize a vertex when Ri encounters it during the search [50]. MR-D* Lite then computes the shortest path from the current vertex of Ri to the goal vertex {59}. After robots have calculated their initial paths {56–59}, they move forward along their respective paths. As the robot moves to the target vertex, MR-D* Lite uses the function UpdatePath(i) to monitor potential conflicts and update the path to achieve collision and deadlock avoidance {63}. UpdatePath(i) clears the AUV set for Ri previously and updates Ri's path by ScanGraph(i) {30–31}. It sets the AUV for Ri and updates the path according to the type of conflict {32–55}. A robot first monitors potential conflict with higher priority robots and executes the corresponding strategy {32–46}. Then, the robot monitors potential conflicts with lower priority robots and executes the corresponding strategy {47–55}. Finally, all robots have a collision-free and deadlock-free path, and each robot updates $s_{start(i)}$ to reflect the current vertex of the robot and makes one transition along the path {64–67}.

In the pseudocode, we have included a comment on how the robot can detect there is no path {65} but do not prescribe what it should do in this case. For the goal-directed navigation problem in unknown static terrain, if there is no path because of obstacles, the robot should stop and announce there is no path [50]. However, if a robot cannot find the path due to the AUV (e.g., robots are in a narrow passage or corner), then it should stop moving temporarily and wait for the next iteration to update the path.

Figure 41 displays a general flowchart of a robot Ri in the MRS that is executed in the implementation. The robots first plan their initial paths without considering other robots. Then, the conflict avoidance strategy between robots is implemented when the robot moves along its original path. Before robot Ri moves to the next vertex, it checks whether there is a conflict

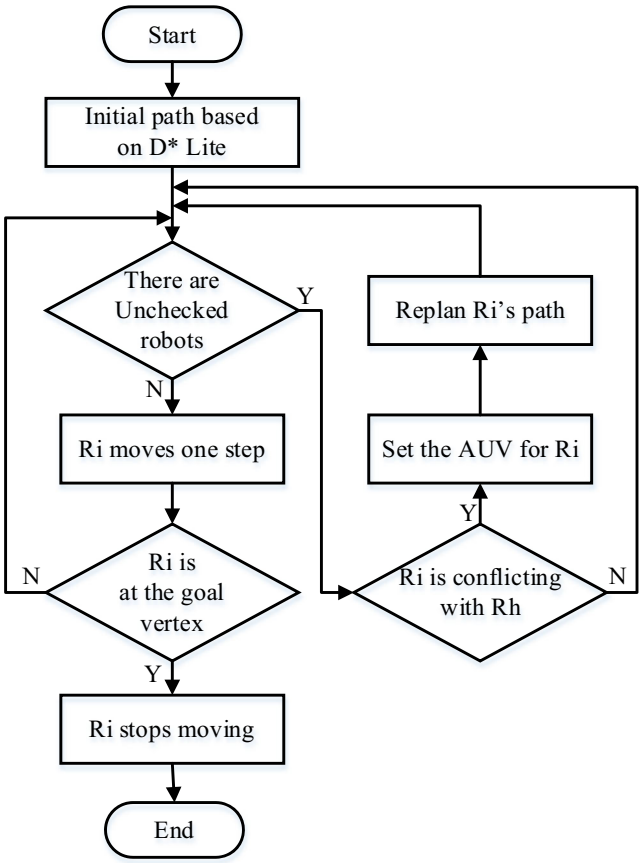
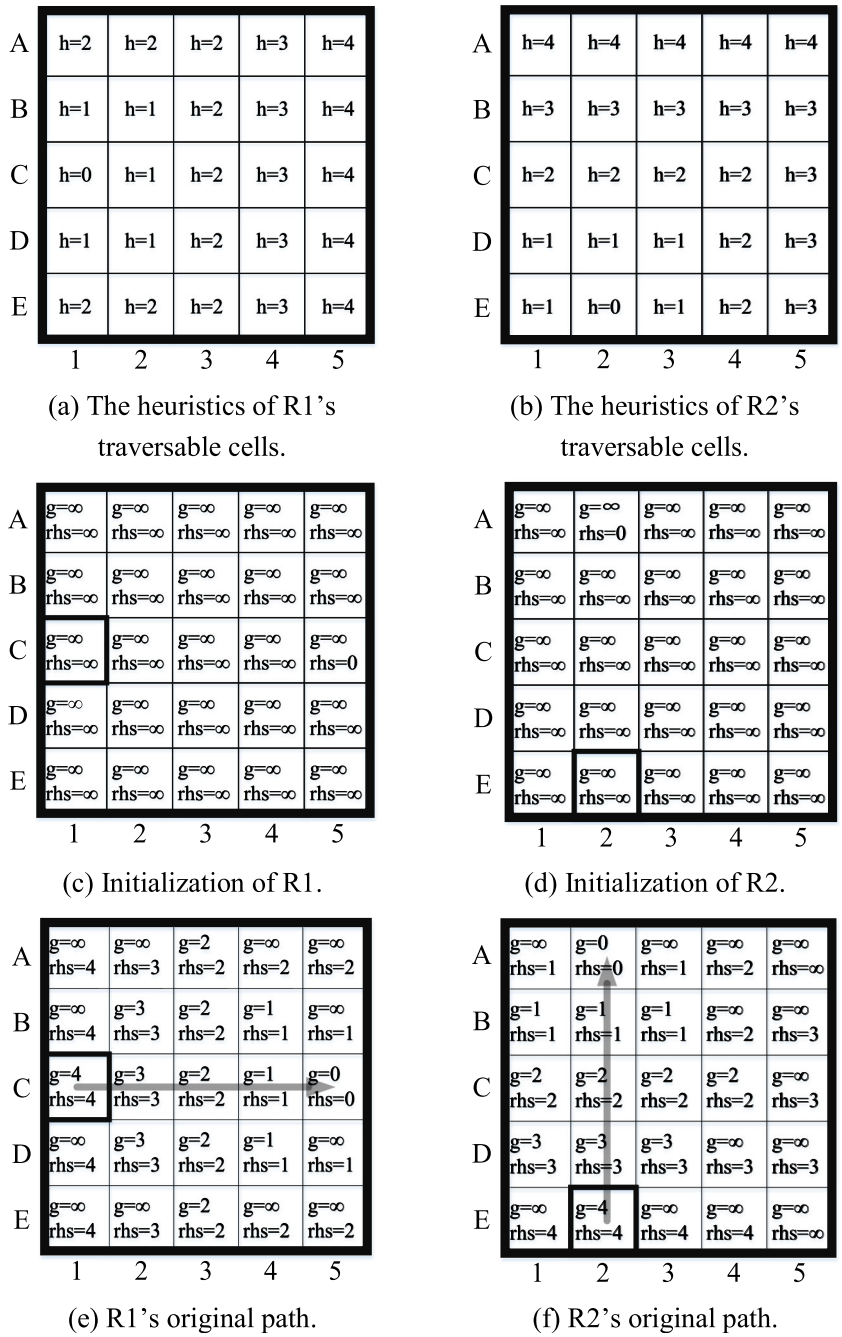


Fig. 41 Path planning of a robot (Ri) in MRS, where Rh indicates the robot with the highest priority among the unchecked robots

with the other robots in order of priority from high to low. If there is a conflict, MR-D* Lite sets the AUV for Ri and then replans its path. After all other robots are checked, robot Ri moves one step and then repeats the procedure until it eventually reaches the goal coordinates.

3.4 Example

We now step through the example of Fig. 1 to show the operation of MR-D* Lite. Figure 42 (a) and (b) show the heuristics of R1's and R2's traversable cells, which approximate the distance from a cell to the start cell with the maximum absolute differences of the x and y coordinates of both cells. Figure 42 (c)-(l) shows the g-values and rhs-values of the traversable cells. The start cells of R1 and R2 are C1 and E2, and the goal cells are C5 and A2. The cell with a bold border is the current cell of the robot. Figure 42 (c) and (d) show the values after the first call Initialize(1) and Initialize(2). Figure 42 (e) and (f) show the values after the first call ComputeShortestPath(1) and ComputeShortestPath(2). The robot follows the path from the start to the goal cell by always moving from the current vertex s, starting at the start cell, to any successor that



(a) The heuristics of R1's traversable cells.

(b) The heuristics of R2's traversable cells.

(c) Initialization of R1.

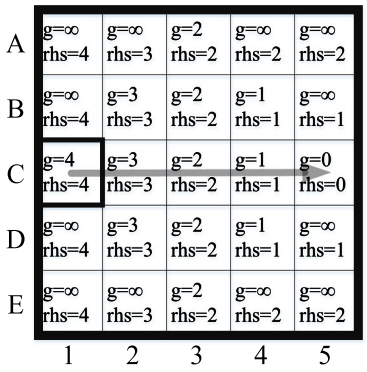
(d) Initialization of R2.

(e) R1's original path.

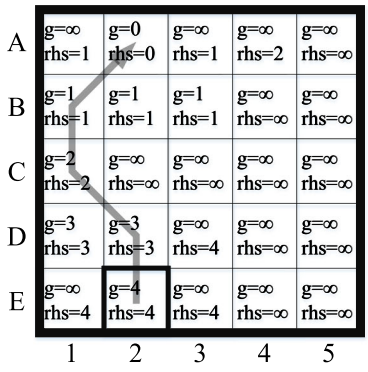
(f) R2's original path.

Fig. 42 Operation of MR-D* Lite. (**a**) The heuristics of R1's traversable cells. **b** The heuristics of R2's traversable cells. **c** Initialization of R1. **d** Initialization of R2. **e** R1's original path. **f** R2's original path. **g** R1's path after the first call UpdatePath(1). **h** R2's path after the first call UpdatePath(2). **i** R1's path after the second call UpdatePath(1). **j** R2's path after the second call UpdatePath(2). **k** R1's path after the third call UpdatePath(1). **l** R2's path after the third call UpdatePath(2)

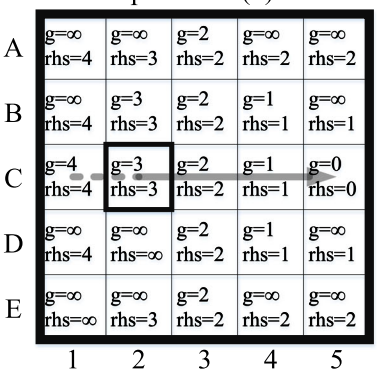
minimizes $c(s, s') + g(s')$ (i.e., the rhs-value in Fig. 42) until the goal cell is reached. Thus, as shown in Fig. 42 (e) and (f), the original path of R1 is C1 – C2 – C3 – C4 – C5, and the original path of R2 is E2 – D2 – C2 – B2 – A2. Therefore, the robots are likely to collide at C2. After the first call UpdatePath(1) and UpdatePath(2), R1's path remains unchanged, but R2's path changes, as shown in Fig. 42 (h). Each robot then follows the path from its current cell to the goal cell. Therefore, R1 moves the first step to C2, and R2 moves the first step to D2. Before the robots move the second step, MR-D* Lite calls UpdatePath(1) and UpdatePath(2) again, and the previous AUVs are cleared. The path of the robot after the first replanning no longer conflicts. However, the vertices



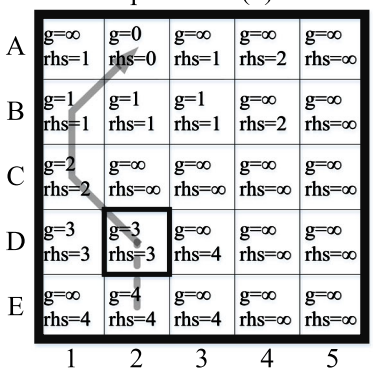
(g) R1's path after the first call UpdatePath(1).



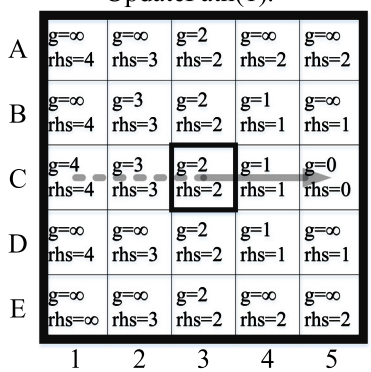
(h) R2's path after the first call UpdatePath(2).



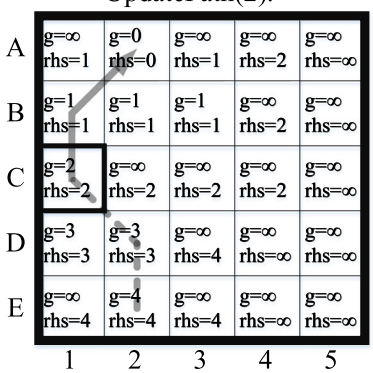
(i) R1's path after the second call UpdatePath(1).



(j) R2's path after the second call UpdatePath(2).



(k) R1's path after the third call UpdatePath(1).



(l) R2's path after the third call UpdatePath(2).

Fig. 42 (continued)

where the robots are located are set as each other's AUV, as shown in Fig. 42 (i) and (j), because they are adjacent. The paths of robots remain unchanged, and R1 and R2 move to C3 and C1, respectively. Before the robots move the third step, MR-D* Lite calls UpdatePath(1) and UpdatePath(2) for the third time. The previous AUVs are cleared, and no new AUVs are set. Each robot follows its remaining path from the current cell to the goal cell without observing other conflicts or obstacles and thus without changing the g-values and rhs-values.

4 Experimental results

We tested MR-D* Lite in experiments run on a Xeon E5 computer at 2.10 GHz with 16 Gb of RAM. The priority of the robots is R1 > R2 > R3 > R4 > R5. We assume that each robot occupies a vertex in the grid map, and it takes the same time for a robot to move from the current vertex to the next vertex. As shown in Fig. 43, the green and red squares are the start vertex and goal vertex of a robot, and the red and green lines are the trajectories of R1 and R2, respectively. The

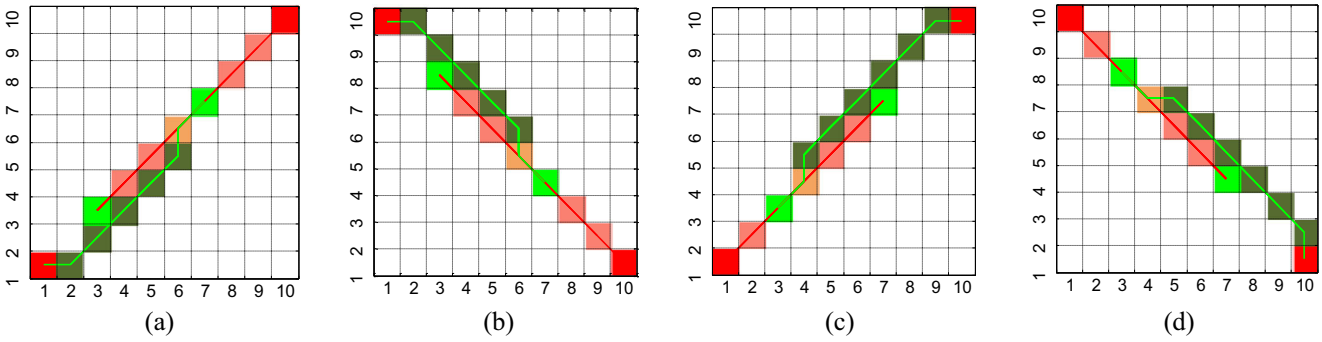


Fig. 43 Experiment results of CST1

yellow square is the vertex where the trajectories overlap. The experimental results in Fig. 43 correspond to the four cases of CST1 in Fig. 4.

The experimental results in Fig. 44 correspond to the eight cases of CST2 in Fig. 5. Note that in some cases, because MR-D* Lite gives robots the ability to deal with potential conflict scenarios in advance (taking Fig. 44 (a) as an example), the robot does not reach the position in Fig. 5 (a).

For CST3, CST4, and CST5, many scenarios correspond to the eight moving directions of the higher priority robot. Due to symmetry, we only list the experimental results when the higher priority robot R1 moves north and northeast, where the simulation results in Figs. 45 and 46 correspond to CST3 in Figs. 6 and 7, the results in Figs. 47 and 48 correspond to CST4 in Figs. 14 and 15, and the results in Figs. 49 and 50 correspond to CST5 in Figs. 22 and 26. The simulation results in Figs. 43, 44, 45, 46, 47, 48, 49 and 50 show that MR-D* Lite can successfully resolve the conflict scenario mentioned in Section 3.

Experiments are conducted with more robots to validate the flexibility of the proposed algorithm. Figure 51 shows the conflict avoidance trajectories of the three robots, that is, R1 (red line), R2 (green line), and R3 (blue line).

Figure 52 shows the conflict avoidance trajectories of four robots, that is, R1 (red line), R2 (green line), R3 (blue line), and R4 (white line), where Fig. 52 (a) is the experimental result of the scenario shown in Fig. 38 (a).

Figure 53 shows the conflict avoidance trajectories of five robots: R1 (red line), R2 (green line), R3 (blue line), R4 (white line), and R5 (magenta line). The experimental results in Figs.

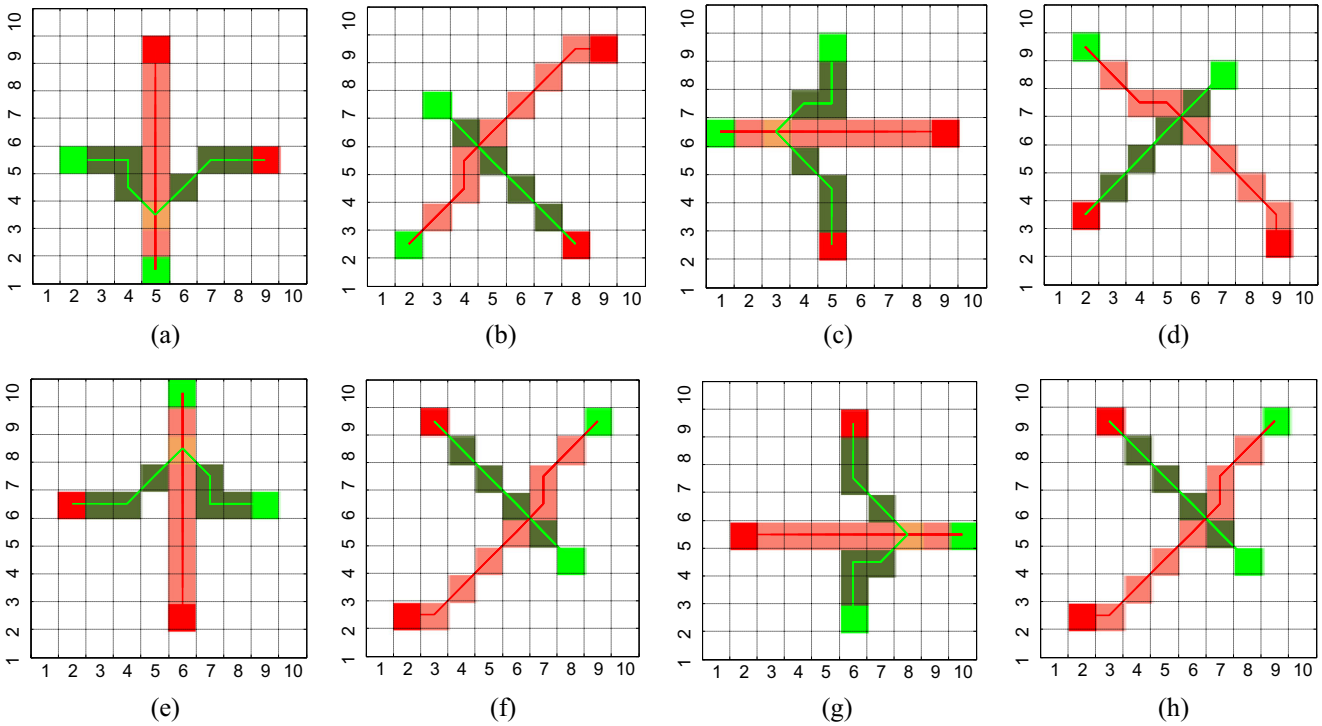


Fig. 44 Experiment results of CST2

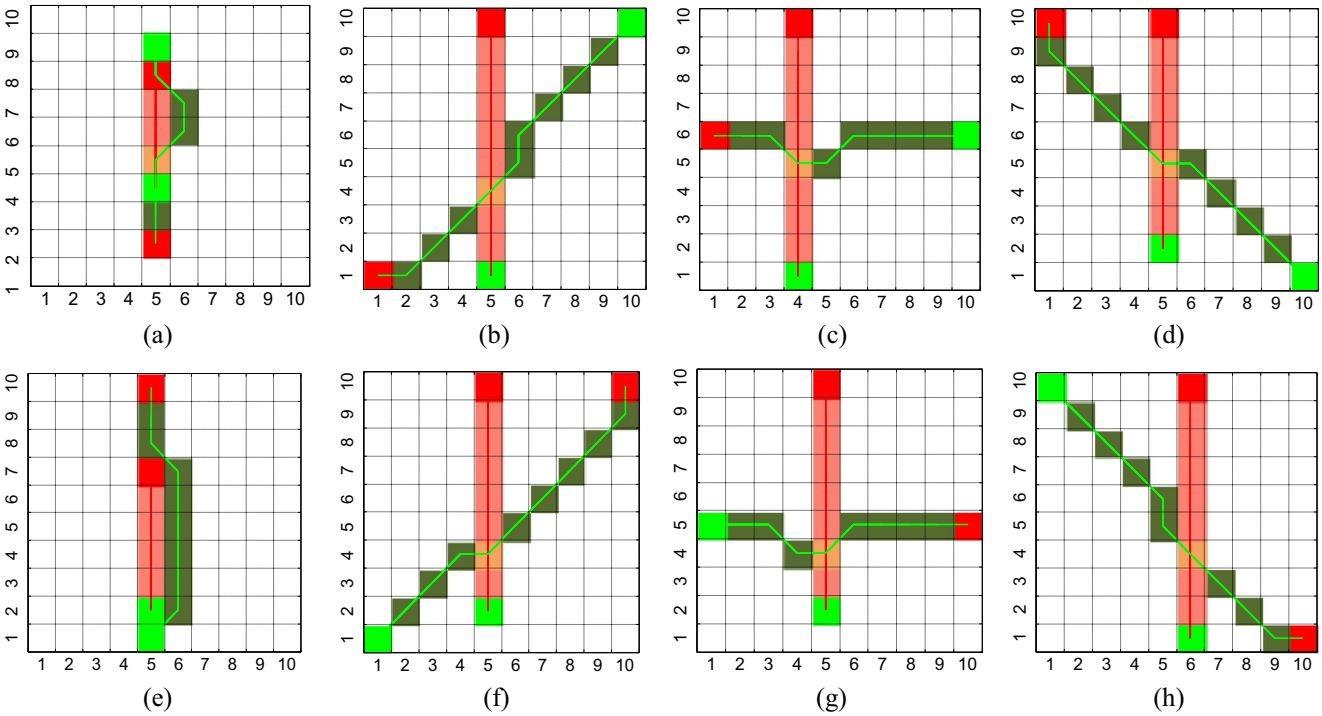


Fig. 45 Experiment results of CST3 when R1 moves north

51-53 show that MR-D* Lite can successfully avoid collisions with obstacles and other robots. Figures 51 (a), 52 (a), and 53 (a) show that the algorithm can solve the problem of many robots being concentrated at a single choke vertex.

We compare MR-D* Lite against three other MRPP algorithms: FIS, APF, and HPP [46]. As shown in Fig. 54, the experimental environment is a two-dimensional space of 100×100 rectangle, where the term Si denotes the start

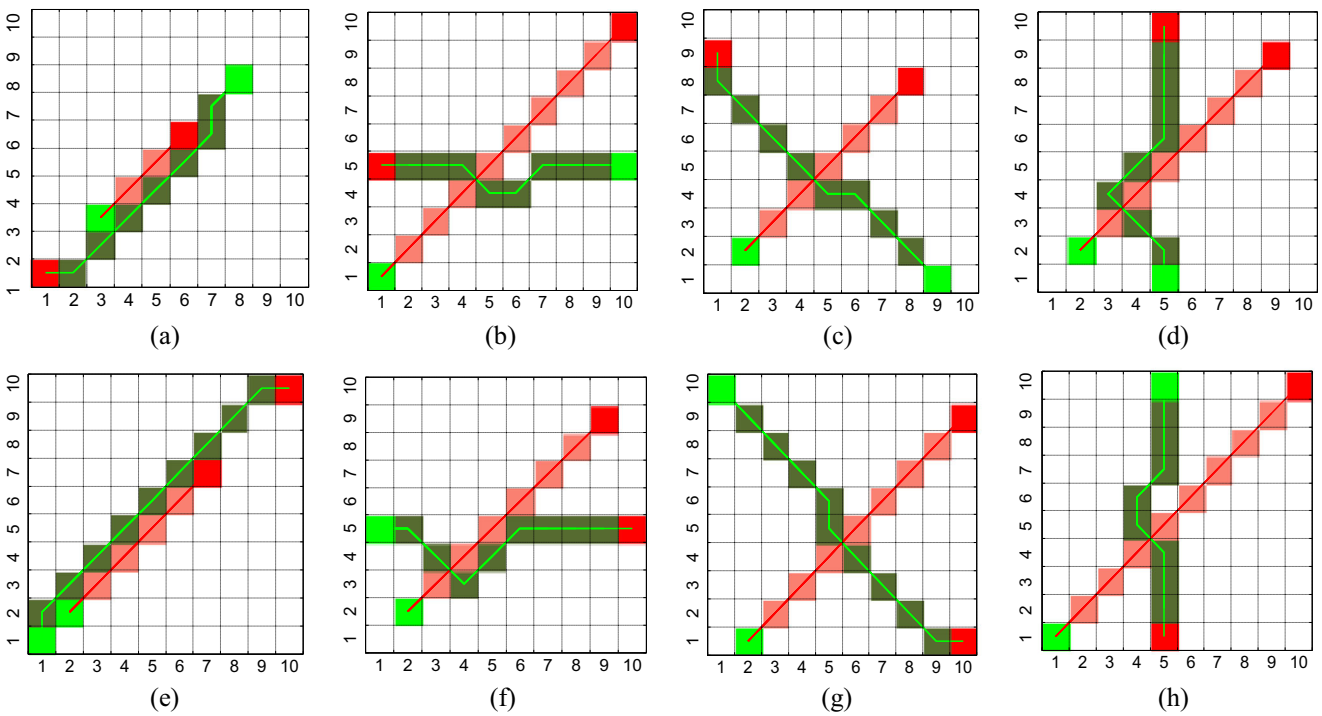


Fig. 46 Experiment results of CST3 when R1 moves northeast

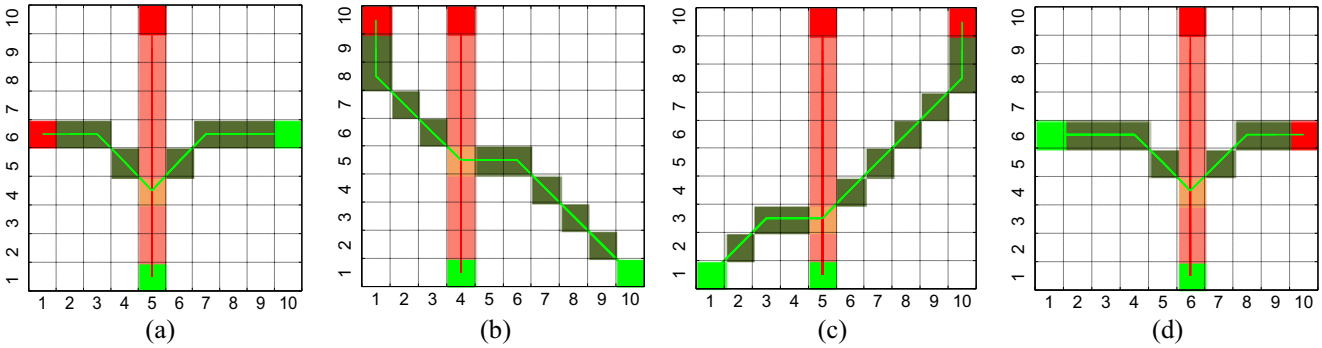


Fig. 47 Experiment results of CST4 when R1 moves north

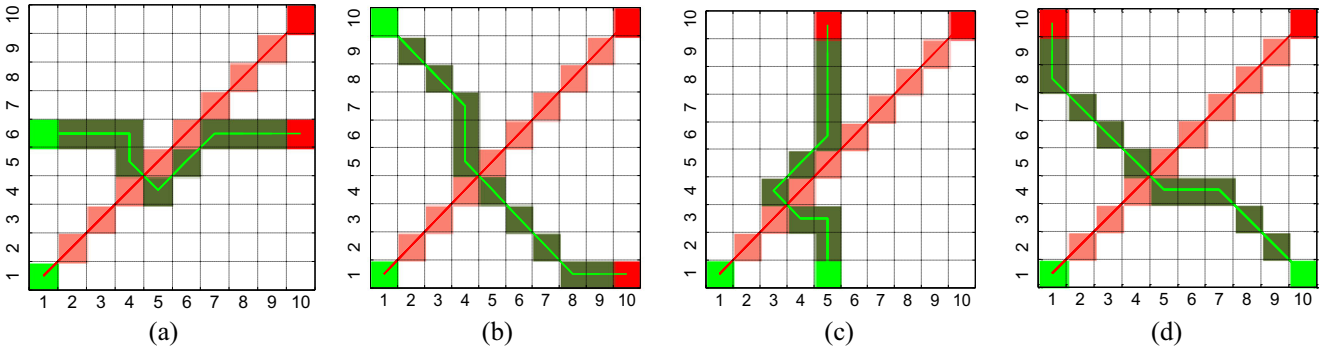
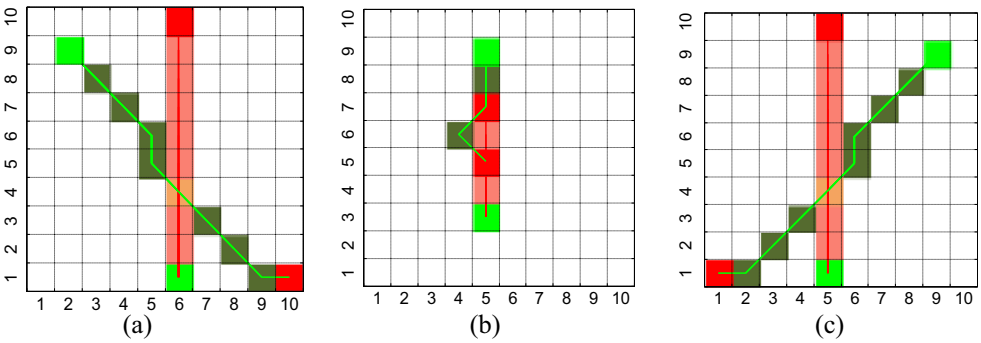


Fig. 48 Experiment results of CST4 when R1 moves northeast

Fig. 49 Experiment results of CST5 when R1 moves north



position of Ri and Gi denotes the target position. Table 2 lists the coordinates of the start position and goal position of robots.

The performance of the experimental results is first analyzed in terms of trajectory distance, by which we can minimize energy and time consumption. The MR-D* Lite

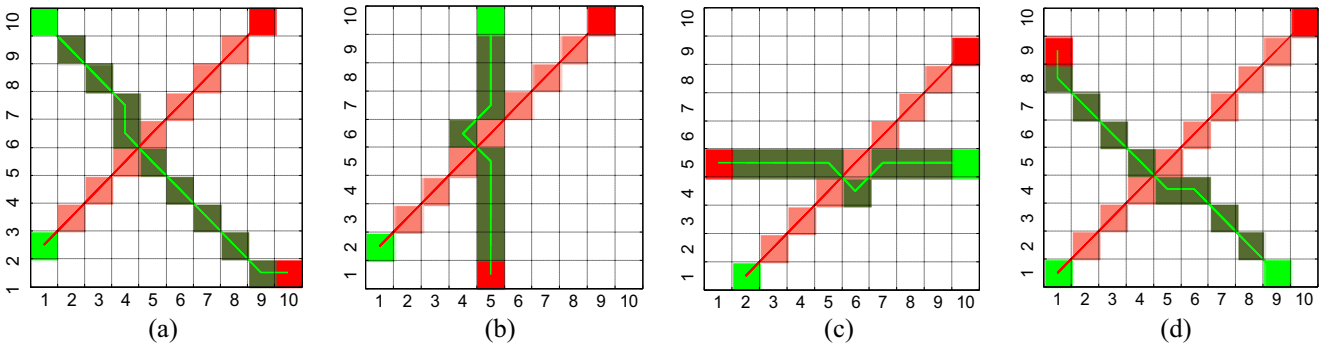


Fig. 50 Experiment results of CST5 when R1 moves northeast

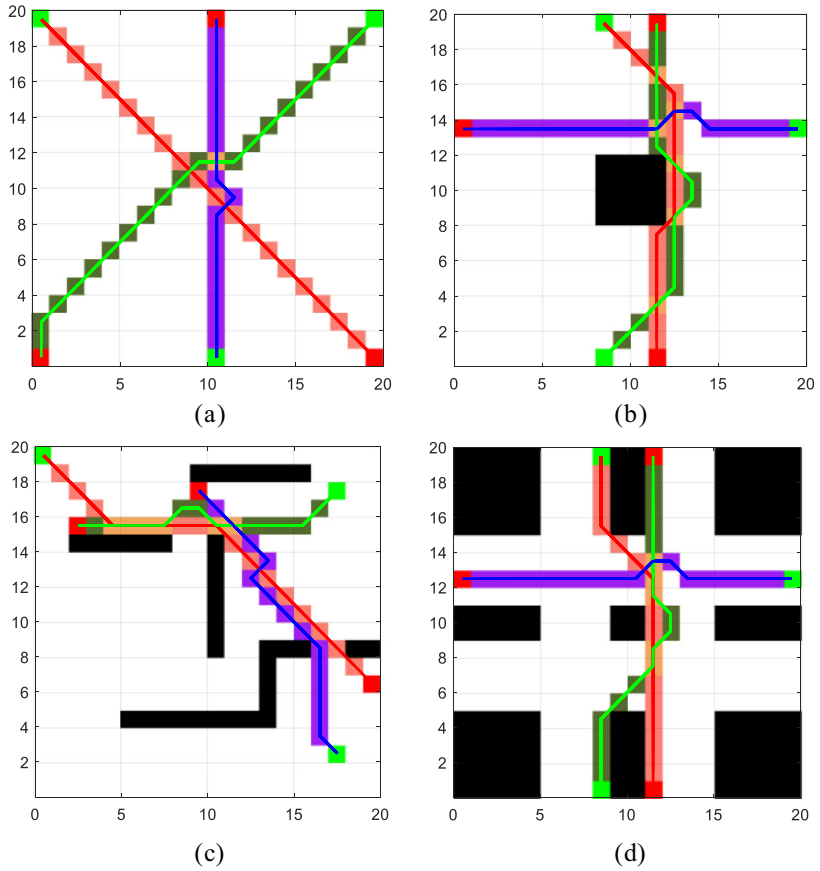


Fig. 51 Collision avoidance trajectories of three robots

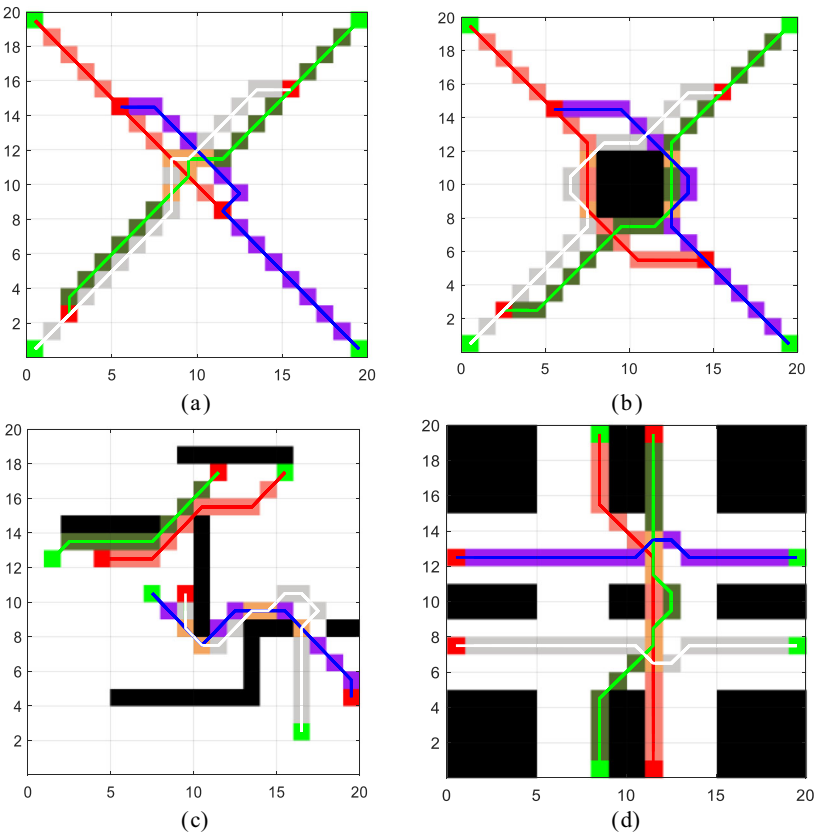


Fig. 52 Collision avoidance trajectories of four robots

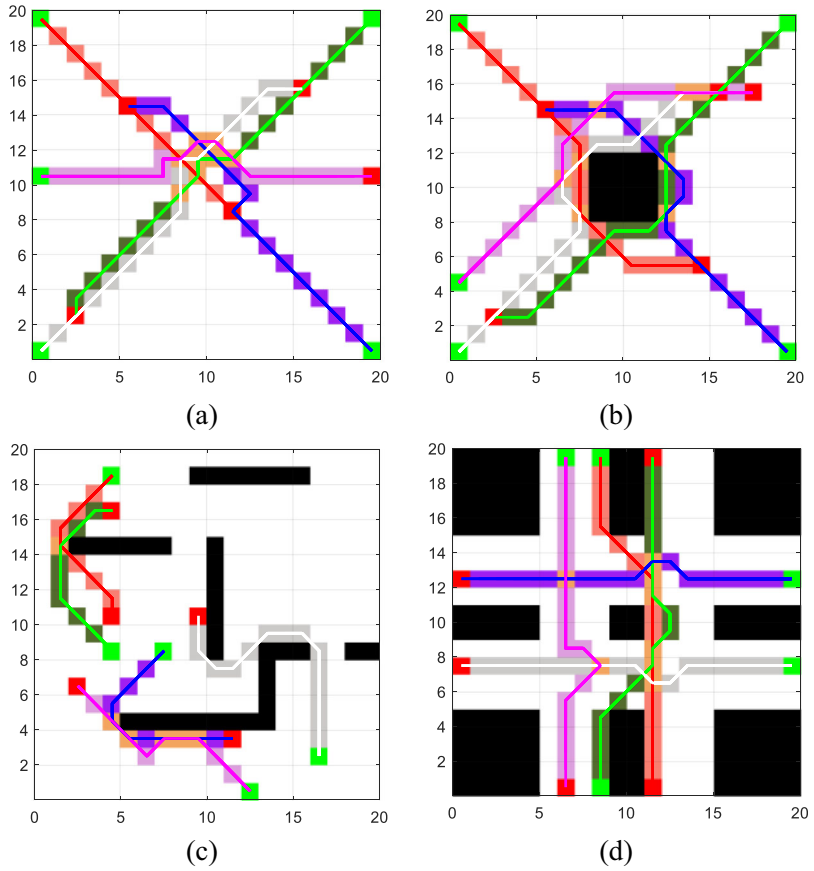


Fig. 53 Collision avoidance trajectories of five robots

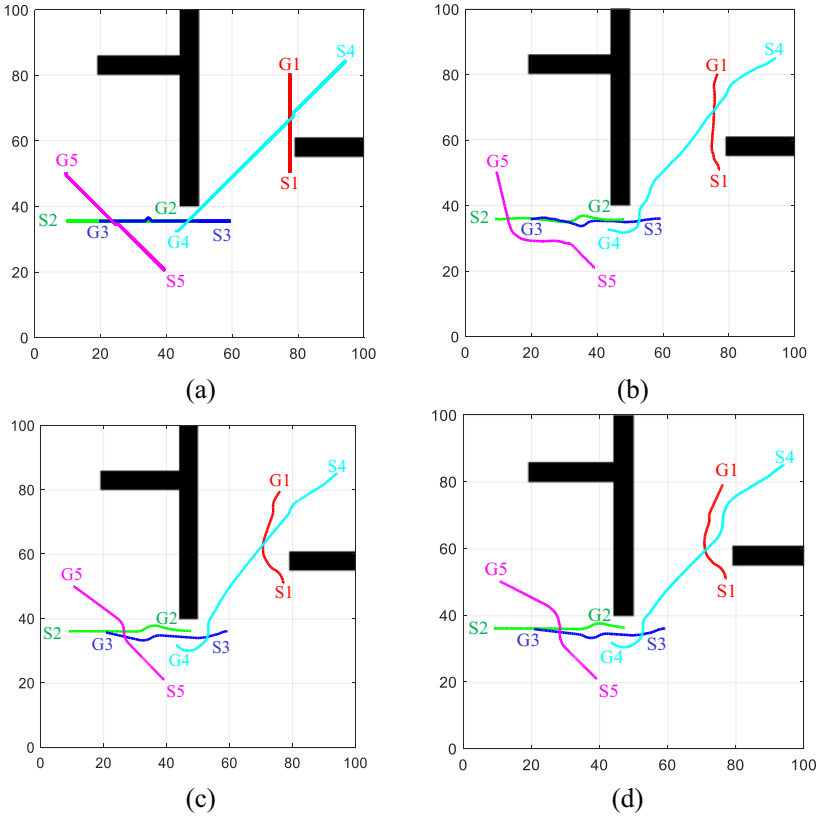


Fig. 54 Trajectories of five robots under different algorithms. **a** MR-D* Lite. **b** FIS. **c** APF. **d** HPP

Table 2 Initial and target position of robots

	R1	R2	R3	R4	R5
Initial position	(78, 50)	(10, 35)	(60, 35)	(95, 84)	(40, 20)
Target position	(78, 80)	(50, 35)	(20, 35)	(43, 32)	(10, 50)

Table 3 Trajectory distance under different algorithms

	R1	R2	R3	R4	R5
MR-D* Lite	30.0000	40.0000	40.8284	74.1249	43.0122
FIS	30.8691	40.8184	40.9980	79.5884	48.6552
APF	33.8221	40.5776	40.9603	80.9528	43.5109
HPP	33.5611	40.4810	40.8907	81.2056	44.9653

Table 4 Path deviation under different algorithms

	R1	R2	R3	R4	R5
MR-D* Lite	0.0000	0.0000	0.8284	0.5858	0.5858
FIS	0.8691	0.8184	0.9980	6.0493	6.2288
APF	3.8221	0.5776	0.9603	7.4137	1.0845
HPP	3.5611	0.4810	0.8907	7.6665	2.5389

algorithm first plans a globally optimal path for each robot and modifies the established path when a potential conflict between the robots is detected. The robots using the other three algorithms plan their paths while moving according to the surrounding environment's information. Therefore, the robots using MR-D* Lite can typically generate a shorter trajectory, which is reflected in Table 3.

One of the goals of MRPP is to ensure that any action taken to avoid a collision with another robot in real-time does not adversely affect the optimal planned route beyond the minimum extra effort required to avoid the collision [51]. Thus, the performance of the experimental results is analyzed in terms of the path deviation. Table 4 lists the robot's path deviation under different algorithms, which is the trajectory distance minus the linear distance between the initial position and the target position. In MR-D* Lite, the method to resolve conflicts between robots is replanning the path based on the AUV. The other three algorithms mainly resolve conflicts by generating a repulsive force between robots. Therefore, as shown in Table 4, the other three methods may produce more unnecessary detours.

Table 5 lists the computational time of different algorithms to solve the conflicts between robots. MR-D* Lite avoids interrobot collisions and deadlocks by bypassing the AUV, and other algorithms need to make the robot escape the predefined danger zone to address the problem. The AUV contains up to 11 vertices, and the danger zone may contain a large number of vertices. Therefore, MR-D* Lite takes less time for robots to resolve conflicts.

5 Conclusions

This paper introduces a multi-robot version of a well-known single-robot path planning algorithm D* Lite. The proposed method resolves collisions and deadlocks between multiple robots while maintaining the robustness and real-time performance of the D* Lite algorithm. Thus, MR-D* Lite can tolerate inaccuracies in a map and fast replan a conflict-free path.

Table 5 Time consumption of resolving conflicts, in seconds (s)

Algorithm	MR-D* Lite	FIS	APF	HPP
Time	0.0051	9.079612	0.3361	10.67061

The experimental results show that MR-D* Lite outperforms the remaining three algorithms in path deviation and replanning time. However, in this paper, the conflict is not detected until it almost happens, so sometimes the robot needs a large steering angle to avoid the collision. In the future, further efforts are needed to solve the problem.

Authors' contributions The study is conceived and designed by Haodong Li. The first draft of the manuscript was written by Haodong Li and Tao Zhao, and revised by Tao Zhao and Songyi Dian. All authors read and approved the final manuscript.

Funding This work is supported by the Sichuan Science and Technology Program (2020YFG0115) and Chengdu Science and Technology Program (2019-YF05-00958-SN).

Data availability All data, models generated or used during the study are available from the corresponding author by request.

Code availability All codes generated or used during the study are available from the corresponding author by request.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Chopra S, Notarstefano G, Rice M, Egerstedt M (2017) A distributed version of the hungarian method for multirobot assignment. IEEE Trans Robot 33(4):932–947
2. Feng Z, Sun C, Hu G (2016) Robust connectivity preserving rendezvous of multirobot systems under unknown dynamics and disturbances. IEEE Trans Control Netw Syst 4(4):725–735
3. Rizk Y, Awad M, Tunstel EW (2019) Cooperative heterogeneous multi-robot systems: a survey. ACM Comput Surv 52(2):1–31
4. Roldán JJ, Garcia-Aunon P, Garzón M, De León J, Del Cerro J, Barrientos A (2016) Heterogeneous multi-robot system for mapping environmental variables of greenhouses. Sensors 16(7):1018
5. Das PK, Jena PK (2020) Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators. Appl Soft Comput 106312
6. Nath A, Arun AR, Niyogi R (2019) A distributed approach for road clearance with multi-robot in urban search and rescue environment. Int J Intell Robot Appl 3(4):392–406
7. Di Nuovo A, Broz F, Wang N, Belpaeme T, Cangelosi A, Jones R, Dario P (2018) The multi-modal interface of robot-era multi-robot services tailored for the elderly. Intell Serv Robot 11(1):109–126
8. Nagavarapu SC, Vachhani L, Sinha A (2016) Multi-robot graph exploration and map building with collision avoidance: a decentralized approach. J Intell Robot Syst 83(3):503–523
9. Dai X, Jiang L, Zhao Y (2016) Cooperative exploration based on supervisory control of multi-robot systems. Appl Intell 45(1):18–29
10. Li Z, Barenji AV, Jiang J, Zhong RY, Xu G (2020) A mechanism for scheduling multi robot intelligent warehouse system face with dynamic demand. J Intell Manuf 31(2):469–480
11. Viet HH, Dang VH, Choi S, Chung TC (2015) BoB: an online coverage approach for multi-robot systems. Appl Intell 42(2):157–173
12. Liu Y, Nejat G (2016) Multirobot cooperative learning for semiautonomous control in urban search and rescue applications. J Field Robot 33(4):512–536
13. Gans NR, Rogers JG (2021) Cooperative multirobot systems for military applications. Curr Robot Rep:1–7
14. Kantaros Y, Zavlanos MM (2016) Global planning for multi-robot communication networks in complex environments. IEEE Trans Robot 32(5):1045–1061
15. Schuster MJ, Schmid K, Brand C, Beetz M (2019) Distributed stereo vision-based 6D localization and mapping for multi-robot teams. J Field Robot 36(2):305–332
16. Serpen G, Dou C (2015) Automated robotic parking systems: real-time, concurrent and multi-robot path planning in dynamic environments. Appl Intell 42(2):231–251
17. Fazlollahtabar H, Hassanli S (2018) Hybrid cost and time path planning for multiple autonomous guided vehicles. Appl Intell 48(2):482–498
18. Paden B, Čáp M, Yong SZ, Yershov D, Frazzoli E (2016) A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans Intell Vehic 1(1):33–55
19. Deplano D, Franceschelli M, Ware S, Rong S, Giua A (2020) A discrete event formulation for multi-robot collision avoidance on pre-planned trajectories. IEEE Access 8:92637–92646
20. Zhou Y, Hu H, Liu Y, Ding Z (2017) Collision and deadlock avoidance in multirobot systems: a distributed approach. IEEE Trans Syst Man Cyber Syst 47(7):1712–1726
21. Tran VP, Garratt MA, Petersen IR (2020) Switching formation strategy with the directed dynamic topology for collision avoidance of a multi-robot system in uncertain environments. IET Control Theory & Applications 14(18):2948–2959
22. Oral T, Polat F (2015) MOD* lite: an incremental path planning algorithm taking care of multiple objectives. IEEE Trans Cybern 46(1):245–257
23. Zhou Y, Hu H, Liu Y, Lin SW, Ding ZH (2020) A distributed method to avoid higher-order deadlocks in multi-robot systems. Automatica 112:108706
24. Liu F, Narayanan A (2011) Real time replanning based on a* for collision avoidance in multi-robot systems, In 2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 473–479
25. Precup RE, Voisan EI, Petriu EM, Tomescu ML, David RC, Szedlak-Stinean AI, Roman RC (2020) Grey wolf optimizer-based approaches to path planning and fuzzy logic-based tracking control for mobile robots. Int J Comput Commun Control 15(3)
26. Wei C, Hindriks KV, Jonker CM (2016) Altruistic coordination for multi-robot cooperative pathfinding. Appl Intell 44(2):269–281
27. Chen L, Zhao Y, Zhao H, Zheng B (2021) Non-communication decentralized multi-robot collision avoidance in grid map workspace with double deep Q-network. Sensors 21(3):841
28. Yu J, LaValle SM (2016) Optimal multirobot path planning on graphs: complete algorithms and effective heuristics. IEEE Trans Robot 32(5):1163–1177
29. Sharon G, Stern R, Felner A, Sturtevant NR (2015) Conflict-based search for optimal multi-agent pathfinding. Artif Intell 219:40–66
30. Sharon G, Stern R, Goldenberg M, Felner A (2013) The increasing cost tree search for optimal multi-agent pathfinding. Artif Intell 195:470–495
31. Wagner G, Choset H (2015) Subdimensional expansion for multirobot path planning. Artif Intell 219:1–24
32. Long P, Fan T, Liao X, Liu W, Zhang H, Pan J (2018) Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning, In 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 6252–6259

33. He W, Qi X, Liu L (2021) A novel hybrid particle swarm optimization for multi-UAV cooperate path planning. Appl Intell:1–15
34. Das PK, Behera HS, Jena PK, Panigrahi BK (2016) Multi-robot path planning in a dynamic environment using improved gravitational search algorithm. J Electric Syst Inform Technol 3(2):295–313
35. Hidalgo-Paniagua A, Vega-Rodríguez MA, Ferruz J, Pavón N (2017) Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach. Soft Comput 21(4):949–964
36. Precup RE, Petriu EM, Radae MB, Voisan EI, Dragan F (2015) Adaptive charged system search approach to path planning for multiple mobile robots. IFAC-PapersOnLine 48(10):294–299
37. Zhang Y, Zhnag YN, Liu XD (2019) Path planning of multiple industrial mobile robots based on ant colony algorithm, In Proceedings of 2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing, pp. 406–409
38. Contreras-Cruz MA, Lopez-Perez JJ, Ayala-Ramirez V (2017) Distributed path planning for multi-robot teams based on artificial bee colony, In Proceedings of 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 541–548
39. Jose K, Pratihar DK (2016) Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. Robot Auton Syst 80:34–42
40. Park H, Hutchinson SA (2017) Fault-tolerant rendezvous of multirobot systems. IEEE Trans Robot 33(3):565–582
41. Dewangan RK, Shukla A, Godfrey WW (2017) Survey on prioritized multi robot path planning, In 2017 IEEE international conference on smart technologies and management for computing, communication, controls, energy and materials (ICSTM), pp. 423–428)
42. Matoui F, Boussaid B, Abdelkrim MN (2019) Distributed path planning of a multi-robot system based on the neighborhood artificial potential field approach. Simulation 95(7):637–657
43. Ma X, Jiao Z, Wang Z, Panagou D (2016) Decentralized prioritized motion planning for multiple autonomous uavs in 3d polygonal obstacle environments, In 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 292–300
44. Čáp M, Novák P, Kleiner A, Selecký M (2015) Prioritized planning algorithms for trajectory coordination of multiple mobile robots. IEEE Trans Autom Sci Eng 12(3):835–849
45. Yakovlev K, Andreychuk A (2017) Any-angle pathfinding for multiple agents based on SIPP algorithm. Proceedings of the International Conference on Automated Planning and Scheduling 27(1)
46. Zhao T, Li H, Dian S (2020) Multi-robot path planning based on improved artificial potential field and fuzzy inference system. J Intell Fuzzy Syst 39(5):7621–7637
47. Nazarahari M, Khanmirza E, Doostie S (2019) Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. Expert Syst Appl 115:106–120
48. Reyes NH, Barczak AL, Susnjak T, Jordan A (2017) Fast and smooth replanning for navigation in partially unknown terrain: the hybrid fuzzy-D* lite algorithm. In: Robot intelligence technology and applications 4. Pp. 31–41
49. Liu F, Narayanan A (2014) Collision avoidance and swarm robotic group formation. International Journal of Advanced Computer Science 4(2):64–70
50. Koenig S, Likhachev M (2005) Fast replanning for navigation in unknown terrain. IEEE Trans Robot 21(3):354–363
51. Han SD, Yu J (2020) Ddm: fast near-optimal multi-robot path planning using diversified-path and optimal sub-problem solution database heuristics. IEEE Robot Autom Lett 5(2):1350–1357

Haodong Li received his B.S. degree from Zhengzhou University in 2019. Now he is pursuing the M.S. degree in control engineering from Sichuan University, China. His research interests include fuzzy control, path planning, and their applications.

Tao Zhao received his B.S. degree in mathematics and applied mathematics and Ph.D. degree in systems engineering from Southwest Jiaotong University, Chengdu, China, in 2010 and 2015, respectively. He is currently an associate professor in the College of Electrical Engineering, Sichuan University. His current research interests include type-2 fuzzy set theory and system design, rough sets, and intelligent control.

Songyi Dian received his Bachelor and MS degrees of Control Engineering from Sichuan University, China in 1996 and 2002, respectively. He received his Ph.D degree in Nanomechanics Engineering from Tohoku University, Japan in 2009. He is currently a professor in the College of Electrical Engineering, Sichuan University. His current research interests include advanced control methods and intelligent signal processing, power-electronics system and its control, motion control and robotic control.