Ann Math Artif Intell (2008) 52:335–366
DOI 10.1007/s10472-009-9128-7

Multiple UAV exploration of an unknown region

P. B. Sujit · Randy Beard

Published online: 17 March 2009

Abstract In this paper, we present an exploration system for multiple unmanned aerial vehicles (UAVs) navigating through a simulated unknown region that contains obstacles of unknown shape, size, and initial position. The UAVs have to explore and monitor the region continuously. The UAVs have limited sensor and communication ranges and kinematic constraints. The environment may have blind alleys that may cause the UAV to collide with an obstacle. Since the UAVs have limited sensor range, they cannot detect whether the alleys lead to an obstacle or not. Due to the presence of multiple agents and kinematic constraints, the UAVs have to cooperate with each other in selecting their paths, otherwise they may collide with each other. The physical and sensor constraints on the UAVs, coupled with uncertainty in the environment makes the problem of multiple UAVs exploring the unknown region a difficult problem. We developed an exploration system that uses (a) an exploration algorithm to generates safe paths for travel in narrow corridors and (b) a dynamic leader selection scheme to take the presence of multiple agents into account. We also determine the minimum communication range required to ensure no collisions occur inside the narrow corridors. Monte-Carlo simulation were carried out to analyze the effect on area coverage with changes in number of agents, sensor range, and communication range.

Keywords Path planning · Multiple UAV · Exploration · Cooperation

Mathematics Subject Classifications (2000) 90B99

P. B. Sujit (✉)
Department of Electrical and Computer Engineering,
University of Porto, Porto, Portugal
e-mail: sujit@fe.up.pt

R. Beard
Department of Electrical and Computer Engineering,
Brigham Young University, Provo, UT 84602, USA
e-mail: beard@byu.edu

1 Introduction

Unmanned aerial vehicles (UAVs) have been increasingly used for search and surveillance operations in recent years. In most of the missions, the UAVs have to explore and collect information about an unknown area. Since the region is unknown and contains obstacles of unknown shape and size, the UAVs have to explore the region cautiously. Furthermore, because the UAVs have kinematic constraints plus limited sensor and communication ranges, designing the path planning algorithms for safe exploration is an issue that needs to be addressed.

One of the primary requirements for these missions is that the region should be explored as quickly as possible. Therefore, we deploy multiple UAVs, which can cover the region quickly and also provide robustness to the mission through redundancy [1, 2]. Although using multiple UAVs has its merits, there are associated challenges that include designing cooperative mechanism for multiple UAVs, low computational requirement, scalability, etc. The UAVs should cooperate with each other for better distribution of search effort and also avoid inter-UAV collisions. Therefore, the complexity of the path planning problem increases not only from generating safe paths for the UAV exploration but also from inter-UAV collision avoidance and distribution of the search effort.

Exploring an unknown region has been a topic of interest to researchers in the robotics domain. A desirable property for multi-robot exploration is to cover the entire region. Choset [3] provides a survey of area coverage algorithms using single and multiple robots. The task of exploring a region can be performed either in continuous or discrete domain. In the discrete domain the region is split into cells and the cell values are updated as the robot explores the region. While in the continuous domain, cells of varying sizes are generated online as the robots explore the region and covers them using back and forth motion. Rekleitis et al. [4] proposed a multi-robot algorithm for effective coverage by decomposing the area into cells, and allocating cells to robots. Gabriely and Rimon [5] present a polynomial time spanning tree algorithm for a single robot to cover an area. However, using spanning trees for multi-robots is more involved [6–8]. Zheng et al. [9] present a multi-robot forest technique based on an algorithm by Even [10] that finds a tree cover of a graph. Batalin and Sukhatme [11] present coverage algorithms in which the robots spread out and move away from each other while covering the complete area. Pheromone based coverage for multi-robots has been studied by Wagner et al. [12]. Spires and Goldsmith [13] describe a multi-robot coverage algorithm based on Hilbert space filling curves. Koenig et al. [14] presents an area coverage algorithm using LRTA method. The robots leave a pheromone mark in the cell that it visits, and these markers act as the communication medium. Balch and Arkin [15] analyze the effect of communication during cooperation of multi-robot systems. Roy and Dudek [16] developed several rendezvous algorithms for heterogenous robots with limited communication range exploring an unknown region. Zolt et al. [17] designed a market-based exploration technique for multiple robots. Each robot creates a tour of goal points in the search space, and as the mission progresses these goals are negotiated for efficient exploration using auctions. Yamauchi [18] developed a scheme for exploring unknown regions based on frontier cells. The frontier cells are the boundary between known and unknown cells. It is natural that the robot should move towards frontier cells in order to explore the unknown region. However,

the author did not consider cooperation between multi-robots, which may result in ineffective exploration. Burgard et al. [19] present a multi-robot cooperation technique by choosing the target points for the robots in such a way that the exploration of the robots is distributed across the search area.

An arbitrary region of interest may contain obstacles of various shapes, sizes and locations, creating narrow corridors in the region. The corridors as shown in Fig. 1 (Regions (a) to (e)) can be within an obstacle, between obstacles or between an obstacle and the boundary of the region. Assume that robots have limited sensing range and carry out an exploration mission for the environment shown in Fig. 1 using any of the solution concepts presented in [4–19]. Then the robots can explore the environment without collisions. The exploration was possible because the robot can stop when the obstacle is in front of it, move backwards, and turn towards the open space, thus avoiding obstacles in narrow corridors. The papers presented in [4–19] address the key issue of exploration for environments with obstacles, however, UAVs have kinematic constraints hence when they use any of the strategies presented in [4–19] to travel in narrow corridors then they may collide with obstacles. These strategies will also cause robots to collide with obstacles if the robots have kinematic constraints i.e, cannot move backward, cannot stop and have turning radius constraint. For example, if we apply kinematic constraints to the robot then the robot may collide while exploring the Regions (b) and (c) due to lack of sufficient space to turn. On the other hand if we design a pessimistic planner that does not allow the robot to explore areas like Region (c) then areas like the Region (e) and Region (d) cannot be explored. Therefore, to explore regions using UAVs, the concepts presented in [4–19] cannot be applied directly, hence there is a necessity to develop new path planning algorithms.

Cooperatively searching an unknown region with multiple UAVs has been an active field of study [20–22]. In these studies, the search space is discretized, and the

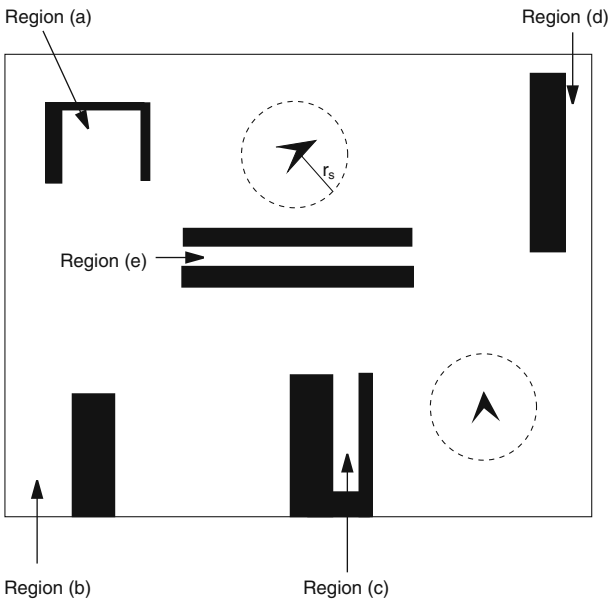

Fig. 1 Scenario of two UAVs exploring an unknown region containing obstacles of various sizes and shapes creating corridors in the region

UAVs cooperate with each other to determine a path that reduces the uncertainty of the map. These papers do not consider obstacles in the region. Beard and Mclain [23] developed cooperative search algorithms with collision avoidance under limited communication, but did not address the problem of exploring the entire region. For search and surveillance applications it is necessary to avoid obstacles and perform continuous monitoring over the region.

A number of multiple UAV path planning algorithms have been developed using market-based approaches [24], dynamic programming [25], evolutionary computation [26], probability maps [27], and MILP [28]. Salva et al. [29] presents an area coverage strategy where the UAVs distribute the areas to cover among themselves and loiter around the area. They do not consider the presence of obstacles. In the UAV domain, development of path planning algorithms to explore an entire region subject to the constraints of limited sensor and communication ranges, UAV kinematics, and obstacles of unknown nature has not been adequately addressed.

In this paper, we address the problem of exploring an unknown region in the presence of obstacles of unknown nature and subject to UAV sensor, communication and kinematic constraints. We design a mechanism for building an online map of the environment, an algorithm to determine safe paths, and a cooperative path planning algorithm for multiple UAVs to avoid inter-UAV collision and to distribute the search effort across the region.

The paper is organized as follows. In the next section we present the problem formulation and the approach we take to solve the problem. In Section 3, we describe the online map building mechanism. Using the map, the UAVs generate safe paths as described in Section 4. In Section 5, we present the cooperative path planning technique that the agents use to determine their deconflicted routes. In Section 6, we study the effects of sensor and communication ranges on the decisions made by the UAVs. Simulation results are presented in Section 7, and we conclude in Section 8.

2 Problem formulation

2.1 Scenario

Consider a bounded region with two UAVs performing a search mission as shown in Fig. 1. Each UAV also called as an *agent*, has a sensor range of r_s meters, and we assume that the communication range r_c is equal to or greater than the sensor range r_s. The world contains obstacles whose initial position, shapes, and sizes are not known a priori. The UAVs have to explore and continuously search the region while avoiding these obstacles.

As shown in Fig. 1, the obstacles can be of different shapes. There may be narrow passages adjacent to the obstacles, between obstacles and between obstacles and the boundary of the region. Region (a) in the Fig. 1 depicts an area that is sufficient for the UAV to turn around and hence cover the region in the obstacle. However, the width of the open areas in Regions (b) and (c) are insufficient for the UAV to turn, hence these regions cannot be explored. Due to limited sensor range the UAVs do not know whether a narrow passage has a barrier at the end, or if there is a free space to move out of the passage. This situation can be seen in Regions (d) and (e)

where the narrow corridors connect to open areas. A pessimistic path planner can be designed that will not allow the UAVs to enter such passages. In that case, we cannot explore Regions (d) and (e), hence cannot completely explore all the possible regions that the UAVs can visit.

The UAVs should completely explore the region and build an online map of the world. Since UAVs have limited communication range, they can only share information locally. The agents should also coordinate with their neighbors in determining their paths. The resultant paths should avoid corridors that may lead the UAV to collide with obstacles, inter-UAV collisions, and explore unknown regions.

2.2 Approach

In this paper, we address the issues that were raised with respect to the ambiguity of decision-making inherent with the UAVs during exploration. We present an exploration system that will build the map online and the agents use this map to determine safe paths that do not lead the agent to collide with obstacles. We construct a discretized version of the region and use this grid-based cell structure to map the environment and navigate in the region. The exploration system is shown in Fig. 2. Once a grid of the unknown region is generated, the process of exploring the region can be carried out in four stages that are described below.

Stage 1: In the first stage, an agent A_i present at cell C_j detects the free and obstacles cells within its sensor range. If the agent has neighbors then they share information locally.

Stage 2: The agent updates the cell value based on the information obtained from its sensors and neighbors. During the update process, if a cell C_k is sensed by multiple agents, then the value of the cell is averaged and assigned to the cell. We use an averaging mechanism as the mode of consensus that takes place when the information about a cell is different among neighboring agents. The online map building and updating mechanisms are explained in Section 3.

Stage 3: The agents use the updated map to generate paths. The agents have limited sensor range and hence use a q-step look ahead policy to generate paths of length q, where the value of q is equal to the sensor range r_s of the UAV. Assume that the set of paths generated by agent A_i be $\mathcal{P}_i$. Some of the paths $P_i \in \mathcal{P}_i$ can direct the UAV towards obstacles. The paths that intersect with the obstacles can be easily removed from $\mathcal{P}_i$, but removing paths that may steer the UAV towards a potential collision is difficult. The path length has a limited horizon, therefore the UAV cannot predict whether the path P_i is a safe path or not. In order to eliminate this ambiguity we developed an exploration algorithm (EA) that will not allow the UAVs to choose a path that may steer them towards potential collisions. The EA is based on the intuition that if an agent A_i, is present at cell C_i and if there exists a path P_i, such that this path will bring the UAV back to its current location with a heading equal to the current heading or the current heading $+ \pi$, then path P_i is a safe path. The exploration algorithm is described in Section 4.

Stage 4: Once the agents determine a set of safe paths, they need to select a path from this set for navigation. The selection of a path should be performed in

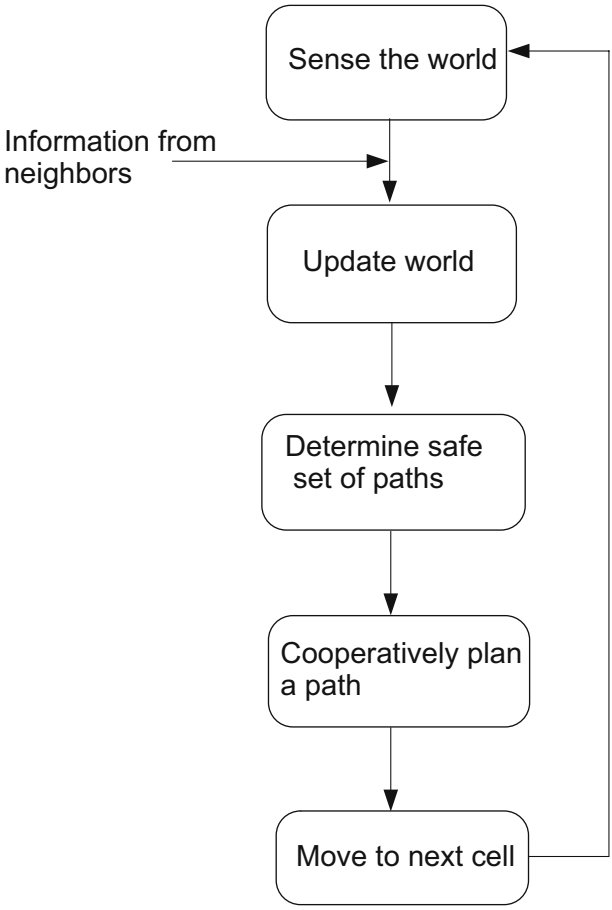

Fig. 2 Exploration system for multiple UAVs exploring unknown regions

a cooperative manner otherwise agents may collide with each other. The agents are greedy by nature and hence move towards cells that are not explored. Therefore, the path selection algorithm has two components: (1) avoid inter-UAV collision, and (2) move towards unexplored cells. The cooperation mechanism is based on a modified leader-follower approach that can handle the environmental constraints of the region and assist the agent in determining their paths. The cooperative path planning algorithm is described in Section 5. Once an agent determines its path, it will move to the next cell of that path, and the process continues until the mission is complete. In the next section we will describe the online map building and the update procedure.

3 Online map building

The agents build a map of the environment dynamically as they explore the unknown region. Initially, a grid of the unknown region is constructed and this grid is the base

of the map on which the agent updates the environment. The UAVs update the map at each time step to generate feasible paths.

3.1 World

The unknown region is assumed to be square in size with width W and length L. The region is split into a collection of cells of length l and width w, where $\frac{W}{w}$ and $\frac{L}{l}$ are integers. We assume that N agents are deployed to build the knowledge of the region through exploration. The initial map provided to all UAVs is the same and each cell C_j has a value $V_i(C_j) = 0, \forall i, j$ where $j = 1, \ldots, \frac{WL}{wl}$, and $i = 1, \ldots, N$. Each UAV is represented as A_i and its map as M_i.

The path of an agent A_i is a sequence of cells. We assume that the agents use the center of the cell as the way-points and the center of the cell C_j is represented as $C_j = \{x_j, y_j\}$. We also assume that the UAV takes a unit time to travel from one cell to another. Therefore, we need to design the l and w of the cells such that the UAV can travel in the grid taking its kinematic constraints into account. If we choose l and w to be less than the minimum turn radius of the UAV ($R_{\min}$), then the UAV will reach the waypoint earlier than unit time. If we increase l and w to be greater than $R_{\min}$ then the agent will not reach the waypoint in unit time. Hence, we use l and w to be equal to $R_{\min}$ that allows the UAVs to reach the waypoint in unit time.

We assume that each agent is equipped with omnidirectional sensors that can accurately detect obstacles in the environment. Hence they can annotate the difference between free cells and obstacle cells on the map M_i. Assume that agent A_i is located at a cell C_j and it can sense all the cells $C_k \in \mathcal{N}(C_j, r_s)$, (where $\mathcal{N}(\cdot)$ is the neighborhood function) surrounding it with a depth of r_s cells. The free cells that A_i detects at time t, are represented by the set $\mathcal{F}(C_j, t)$, while the set of obstacle cells are denoted as $\mathcal{O}(C_j, t)$. Let $\mathcal{U}$, $\mathcal{F}$ and $\mathcal{O}$ represent all the unexplored cells, free cells and obstacle cells in the world. Initially, the region is unknown, hence all the cells are unexplored and have value $V_i(C_j) = 0$. When the UAV detects an unexplored cell, its value is changed depending on whether the cell is a free cell or an obstacle cell. The free cells are assigned a value of 1 while the obstacle cells are assigned a value of -1. The general rule that a UAV uses to assign a value to a cell within its sensor is:

$$V_i(C_j) \leftarrow \begin{cases} 1 & \text{if } C_j \in \mathcal{F}(C_i, t) \\ -1 & \text{if } C_j \in \mathcal{O}(C_i, t) \end{cases} \tag{1}$$

An obstacle can be a single cell or a cluster of cells grouped together. For simplicity, we assume that the obstacles occupy complete cells. The sensor range is r_s meters and it is circular, hence it can partially sense some of the cells on the periphery. But we assume that the obstacles occupy complete cells, therefore we can accurately categorize whether the partially detected cell is a free cell or an obstacle cell. Figure 3 shows a snapshot of a typical exploration space with obstacles.

The environment contains obstacles of different shapes and sizes that create narrow corridors and traps in the region. A trap is a sequence of free cells that end with the boundary of the region or with an obstacle. The trap can be of any shape but representing a trap of general shape is difficult in discrete domain, hence we assume that there are two types of traps: (1) horizontal and (2) vertical. The geometry of these traps are shown in Fig. 4. A horizontal trap is a sequence of free cells that can end with the boundary of the region or with an obstacle cell as shown in Fig. 4 (c),

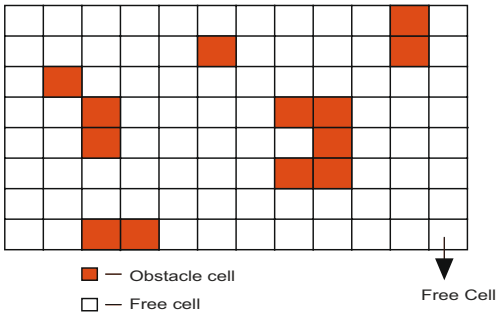

Fig. 3 The unknown region with free and obstacle cells

(d), (i) and (h). The width of the trap can be either one or two cells, while the length can be of L' cells, where $L' < L$. A horizontal trap is defined as

$$G_h = \left\{ \left(x_j^h, y_j^h\right), \ldots, \left(x_{j+d}^h, y_j^h\right), \left(x_j^h, y_{j+1}^h\right), \ldots, \left(x_{j+d}^h, y_{j+1}^h\right) \right\} \tag{2}$$

where (x_j^h, y_j^h) represents the first cell of the trap, (x_{j+d}^h, y_j^h) represents the last cell in the trap, and d represents the depth of the trap. The cells from (x_j^h, y_j^h) to (x_{j+d}^h, y_j^h) represent the first rows of the horizontal trap while the cells (x_j^h, y_{j+1}^h) to (x_{j+d}^h, y_{j+1}^h) represent the cells in the second row of G_h. The cells (x_{j+d+1}^h, y_j^h) and (x_{j+d+1}^h, y_{j+1}^h) are either the obstacle cells or the boundary of the search space. The

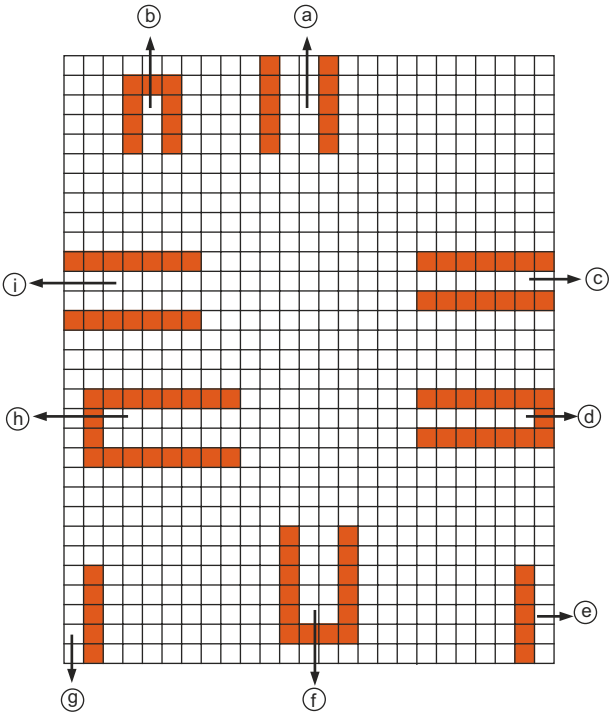

Fig. 4 *(a)* Two cell vertical trap ending with boundary *(b)* one cell that ends with obstacle. *(c)* One cell horizontal trap ending with boundary. *(d)* One cell horizontal trap ending with obstacle *(e)* Single cell vertical trap with boundary corners. *(f)* Two cell vertical trap that ends with obstacle. *(g)* Single cell vertical trap with boundaries. *(h)* Two cell horizontal trap ending with obstacle. *(i)* Two cell horizontal trap ending with boundary.

cells $\{(x^h_j, y^h_{j+2}), \ldots, (x^h_{j+d}, y^h_{j+2})\}$ represent either the obstacle cells of the boundary of the region, similarly the cells $\{(x^h_j, y^h_{j-1}), \ldots, (x^h_{j+d}, y^h_{j-1})\}$ represent the obstacle cells or the boundary.

A vertical trap is a sequence of cells that can have one cell or two cells in length and width of W' cells, where $W' < W$, and ends with an obstacle cell or the boundary. The geometrical shape of the vertical traps are shown in the Fig. 4(a), (b), (e), (f) and (g). The vertical traps are represented as

$$G_v = \left\{ \left(x^v_j, y^v_j\right), \ldots, \left(x^v_j, y^v_{j+d}\right), \left(x^v_{j+1}, y^v_j\right), \ldots, \left(x^v_{j+1}, y^v_{j+d}\right) \right\} \tag{3}$$

where (x^v_j, y^v_j) and (x^v_{j+1}, y^v_j) are the beginning cells of the vertical trap G_v, and (x^v_j, y^v_{j+d}) and (x^v_{j+1}, y^v_{j+d}) are the final cells of G_v. The cells (x^v_j, y^v_{j+d+1}) and (x^v_{j+1}, y^v_{j+d+1}) are either obstacles or the cells outside the boundary. The cells $\{(x^v_{j-1}, y^v_j), \ldots, (x^v_{j-1}, y^v_{j+d})$ and $\{(x^v_{j+2}, y^v_j), \ldots, (x^v_{j+2}, y^v_{j+d})\}$ represents either obstacle cells or the boundary of the region.

Note that if cells (x^h_{j+d+1}, y^h_j) and (x^h_{j+d+1}, y^h_{j+1}) are free cells then the trap is a corridor connecting to open space. The depth of the trap d can be greater than the sensing range r_s of the UAV. In that case, the UAV will not know whether the corridor is a trap or not. Therefore, the UAVs have to generate safe paths that will avoid traps and at the same time allow agents to traverse through narrow corridors that connect different parts of the search space. To generate a safe set of paths we use an exploration algorithm that is described in the next section.

3.2 Information sharing

When the agents are within the communication range r_c, they can share two types of information: map of the environment it has sensed, and its intended path. Sharing the map will result in faster exploration, while sharing paths enables the neighbors to chose a non-colliding routes. For map sharing three levels are possible: (1) no information (2) self information and (3) complete information. Each level has a different effect on the time of coverage.

3.2.1 No information

In this scheme, the agents do not share map information with its neighbors. This mechanism is similar to a single UAV exploring the entire region. Hence, the availability of multiple vehicles is under-utilized, yielding low performance. Since, there is no information exchange between the agents, the communication overhead is minimal. Usually the no-information-exchange scheme is not preferred in path planning and area coverage applications. However, several researchers have developed task allocation algorithms without communication [30, 31]. Although the no-information-exchange scheme gives minimal performance, sometimes it is necessary to know how much performance can be achieved without information sharing for comparison purposes.

3.2.2 Self information

Assume that agent A_i meets agent A_j at time t in the search space. When agents use the self information exchange scheme, then the agent A_i will share the information it

has obtained through its sensors prior to meeting agent A_j. The agent will not share any information it has acquired from other agents before meeting A_j. Let $\hat{\mathcal{C}}_i(0, t)$ be the set of cells that A_i has updated prior to time t. Agent A_i will share $\hat{\mathcal{C}}_i(0, t)$ with agent A_j and agent A_j will share $\hat{\mathcal{C}}_j(0, t)$ with A_i. Their locally instantiated maps are updated as described in Section 3.3.

When agent A_i meets the same agent again at time t', then both the agents will update their maps by sharing $\hat{\mathcal{C}}_i(t, t')$ and $\hat{\mathcal{C}}_j(t, t')$ respectively. As the agents receive additional information through their neighbors, the ability to accomplish the team mission is improved.

Apart from the information of the cells, the agents share the routes through which they have traversed. The route information helps in reducing the computational time to determine safe paths (described in Section 4.2). Each agent stores the route it has traversed and also the routes of the agents it has met during the exploration. The routes are stored in a set

$$\mathcal{R}_i = [R_{i1}, R_{i2}, \ldots, R_{iN}] \tag{4}$$

where R_{ij} is the route of the jth UAV and the route R_{ij} is represented as:

$$R_{ij} = [(C_{ij1}, \psi_{ij1}), (C_{ij2}, \psi_{ij2}), \ldots, (C_{ijk}, \psi_{ijk})] \tag{5}$$

where C_{ijk} is the ith agents' knowledge of the location of agent A_j at step k and ψ_{ijk} is the heading angle at cell C_{ij} at step k. When the agents A_i and A_j communicate, their routes $R_{ij} \in \mathcal{R}_i$ and $R_{ji} \in \mathcal{R}_j$ are updated.

Each UAV has a time record $E_i(A_j, t_i)$, where E_i is the record of the time steps at which A_i met A_j. The record is required to keep track of when A_i met its neighbors for information sharing, so that when these UAVs meet again at time t, they only share the information acquired between time $t_i - t$.

3.2.3 Complete information

In this scheme, each agent shares its entire map and its route information with its neighbors. Since the agents share the complete map information, the fidelity of the agent map increases that results in faster coverage than the other two schemes.

3.3 Map update

At every time step, each UAV updates its map through two steps: (1) update the map by sensing the environment and (2) update the value of the current cell location of the agent. During step (1), the agent uses (1) to classify whether the detected cells are obstacle cells or free cells. In step (2), the agent updates the value of the cell to ensure that the agent will not circle around a couple of cells. When an agent moves to a cell C_k, the value of the cell is decreased. In the cooperative search domain, this reduction in the value of a cell is called uncertainty reduction [20–22]. The reduction in value will not allow the agent to get into a local minima and is similar to the ant pheromone based area coverage, where the ants deposit a pheromone in the cell it visits. Another ant can detect the amount of pheromone deposited in the cell and choose its path to a cell where the pheromone deposit is less thereby ensuring uniform coverage [14, 32]. In this paper, we use the same strategy by reducing the value of the cell as described in the next paragraph.

At each time step, agent A_i updates its map based on the route it traveled and using the information received from its neighbors. The world map is updated in the following way:

1. If A_i is present at cell C_k at time t, then the value of the cell is reduced to $V_i(C_k) \leftarrow 0.5V_i(C_k)$. The UAV traverses to only those cells where the cell value is greater than zero.
2. When A_i receives information from agent A_j about cell C_k, then the value of C_k is updated as $V_i(C_k) \leftarrow \frac{V_i(C_k)+V_j(C_k)}{2}$.

The averaging method enables the agents to have better map fidelity that results in quicker exploration of the region. The agents use the updated maps to generate safe set of feasible paths for exploration.

4 Generating feasible paths

The UAVs have limited sensor range, hence we use a finite-horizon look-ahead policy to generate paths. The paths generated should account for the kinematic constraints of the UAV and avoid traps.

4.1 UAV kinematics

We assume that the UAVs fly at a constant and shared altitude. The UAVs have turn radius constraints and hence motion between cells depends on the current heading angle. The kinematics of the UAVs are modeled as

$$\begin{aligned} \dot{x}_i &= v_i \cos \psi_i \\ \dot{y}_i &= v_i \sin \psi_i \\ \dot{\psi}_i &= \frac{g}{v_i} \tan \phi_i \end{aligned} \tag{6}$$

where v_i is the ground speed, ψ_i is the heading angle, ϕ_i is the commanded roll angle, and g is the gravitational constant. The roll angle command is constrained as

$$-\phi_{\max} \leq \phi_i \leq \phi_{\max}. \tag{7}$$

Due to the kinematic constraints, the UAV cannot move in a straight line to the cells that are not ahead of it. In this paper, we will constrain the motion of the UAVs between cells, so that at the center of each cell, the UAVs have heading angle equal to $0, \frac{\pi}{2}, \pi, \frac{-\pi}{2}$. Let the look-ahead step length be $q = 1$, in which case the UAV can move to the cells ahead of it, as shown in Fig. 5(1). If an agent is present at cell C_i with heading $\psi_i = \frac{\pi}{2}$, then it can reach cell C_i^1 with heading angle $\psi_i = 0$, C_i^2 with heading angle $\psi_i = \frac{\pi}{2}$ and C_i^3 with heading angle $\psi_i = \pi$ respectively.

Producing feasible paths is difficult as some regions surrounding the obstacles can be covered if they are approached in the correct direction by the UAV. However, if approached from other directions, some cells may result in a collision. For instance, consider the scenario in Fig. 5(2). If the UAV travels in direction (a) then it can cover the region inside the obstacle. On the other hand, if the UAV is traveling in direction (b), then the agent will collide with the obstacle. Note that the region

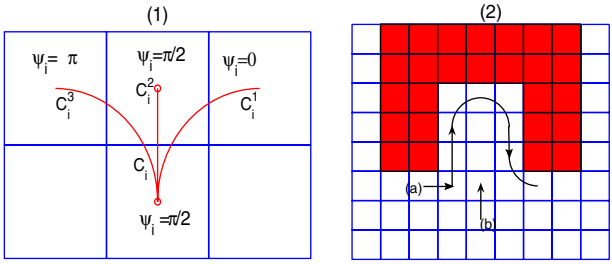

Fig. 5 (*1*) UAV with heading angle $\alpha = \pi/2$ present at cell C_i can move to cells C_i^1, C_i^2, C_i^3 with heading angles $\alpha = 0, \alpha = \pi/2, \alpha = \pi$. (*2*) *(a)* Direction in which the region inside the obstacle can be covered. *(b)* Direction in which the region cannot be covered

inside the obstacle which is of 3 cells width and 3 cells depth and hence this region is not a vertical trap G_v, but the collision occurs due to the selection of the path. Therefore, one of the main problems involved in path planning is to ensure that the UAV generates safe paths that does not allow the UAV towards potential collision with an obstacle. To address this problem we developed an exploration algorithm.

4.2 Exploration algorithm

The agent A_i, present at cell C_i, generates a set of paths $\mathcal{P}_i$. Each path is a sequence of tuples, where each tuple consists of a cell and the heading angle in the cell. The jth path $P_i^j \in \mathcal{P}_i$ is given as $P_i^j = [(C_i^{j,0}, \psi_i^{j,0}), (C_i^{j,1}, \psi_i^{j,1}), \ldots, (C_i^{j,q}, \psi_i^{j,q})]$, where $(C_i^{j,k}, \psi_i^{j,k})$ is the agent location and its heading angle at time step k.

The set $\mathcal{P}_i$ can be decomposed into two sets, $\mathcal{P}_i = \mathcal{P}_i^f \bigcup \mathcal{P}_i^o$ where $\mathcal{P}_i^f$ is the set of paths that are collision free and $\mathcal{P}_i^o$ are the set of paths that intersect obstacles, and $\mathcal{P}_i^f \bigcap \mathcal{P}_i^o = \emptyset$. Not all of the paths in $\mathcal{P}_i^f$ can be used for exploration since some of them may lead the UAV into traps. To generate safe paths we use an exploration algorithm.

In the exploration algorithm we assume that if A_i is present at cell $C_i^{j,0}$ with heading $\psi_i^{j,0}$, then the path it took to arrive at $(C_i^{j,0}, \psi_i^{j,0})$ must have been a collision free path. Assume that there exists a path P from $C_i^{j,q} \in P_i^j$ with heading angle $\psi_i^{j,q}$ to $C_i^{j,0}$, where $P_i^j \in \mathcal{P}_i^f$. Let the path $P_i^j = [(C_i^{j,q}, \psi_i^{j,q}), (C_i^{j,q+1}, \psi_i^{j,q+1}), \ldots, (C_i^{j,q+m}, \psi_i^{j,q+m})]$, where $C_i^{j,q+m} = C_i^{j,0}$, and $\psi_i^{j,q+m}$ is the corresponding arrival angle at cell $C_i^{j,q+m}$.

The classification of whether a path P_i^j is a safe path or not, depends on the arrival angle $\psi_i^{j,q+m}$ at cell $C_i^{j,0}$. The path P_i^j is a safe path if the heading angle $\psi_i^{j,q+m}$ at $C_i^{j,0}$ is equal to $\psi_i^{j,0}$ or $\psi_i^{j,0} + \pi$, otherwise it is not considered to be a safe path. If $\psi_i^{j,q+m}$ is equal to $\psi_i^{j,0}$, then the path traverses back to the current location of the agent. Hence, the agent can choose a path in a different direction implying that the path is safe. If $\psi_i^{j,q+m}$ is equal to $\psi_i^{j,0} + \pi$ then the heading is opposite to the current heading angle $\psi_i^{j,0}$ implying that the agent can travel back through the path that it used to arrive at $C_i^{j,0}$, and the path is safe. If $\psi_i^{j,q+m}$ is equal to $\psi_i^{j,0} \pm \pi/2$, then there may be a possibility that the cells ahead of $C_i^{j,0}$ may lead towards an obstacle or a trap resulting in a collision. Hence, we do not classify these paths as safe paths. This concept is similar to the obstacle avoidance techniques for small UAVs described in [33].

Let the set of all the last cells $C_i^{j,q}$ of each path $P_i^j \in \mathcal{P}_i^f$ be represented as $\mathcal{C}_i^q$:

$$\mathcal{C}_i^q = \bigcup_{j=1}^{|\mathcal{P}_i^f|} C_i^{j,q} \quad \text{where } C_i^{j,q} \in P_i^j, \text{ and } P_i^j \in \mathcal{P}_i^f. \tag{8}$$

The number of elements in set $\mathcal{C}_i^q$ depends on the length of the look ahead step. Now we will determine the safe paths using a recursive search approach.

Let $\mathcal{N}(C_i^{j,q}, \psi_i^{j,q})$ be the neighborhood function that will determine the neighboring cells and the angle at which the UAV may arrive to these cells by considering the kinematic constraints of the UAV. In the recursive search approach, we take a cell $C_i^{j,q} \in \mathcal{C}_i^q$ with its angle $\psi_i^{j,q}$ and determine the next cells and angles. The new cells determined by the neighborhood function are examined to find if any of the cells are equal to $C_i^{j,0}$ and $\psi_i^{j,0}$ is equal to $\psi_i^{j,q}$ or $\psi_i^{j,q} + \pi$. If this condition is satisfied, then the path P_i^j is considered a safe path. Otherwise, recursively explore the end points until $C_i^{j,0}$ is reached. The set of safe paths are represented as $\mathcal{P}_i^s \subseteq \mathcal{P}_i^f$. Algorithm 1 formalizes the process of determining $\mathcal{P}_i^s$.

Algorithm 1 Exploration algorithm using recursive search

```
1: P_i^s = ∅;
2: for j=1 to |C_i^q| do
3:    flag ← 0, cells ← {C_i^{j,q}}, angles ← {ψ_i^{j,q}};
4:    while flag == 0 do
5:       [cells, angles] ← nextCellsAngles(cells, angles);
6:       if C_k == C_i^{j,0}, and ψ_k = ψ_i^{j,0} or ψ_k = π + ψ_i^{j,0}, ∀ C_k ∈ cells and ψ_k ∈ angles
         then
7:          flag ←1, P_i^s ← [P_i^s  P_i^j];
8:       else
9:          if cells == [] then
10:            flag ← 1;
11:         end if
12:      end if
13:   end while
14: end for
```

```
1: Function [newCells, newAngles] = nextCellsAngles(cells, angles)
2: newCells = ∅;newAngles = ∅
3: for Each C_k ∈ cells do
4:    [nextCells, nextAngles]← determineNextCells(C_k, ψ_k);
5:    newCells ← [newCells nextCells];
6:    newAngles ← [newAngles nextAngles];
7: end for
```

The function *nextCellsAngles* represents the neighborhood function $\mathcal{N}(C_i^{j,q}, \psi_i^{j,q})$ and returns the next cells and angles that the UAV can move to, given its current cells and angles. The exploration algorithm with recursive search procedure produces all the safe paths.

Theorem 1 *The Algorithm* 1 *will generate a safe path that will not allow the UAV to steer towards traps.*

Proof Assume that agent A_i is located at cell C_i at time t and uses Algorithm 1 to determine safe paths. Consider a path $P_i^j \in \mathcal{P}_i^f$, Algorithm 1 classifies this path $P_i^j \in \mathcal{P}_i^s$, if and only if, C_k equals $C_i^{j,0}$ with ψ_k equals $\psi_i^{j,0}$ or $\psi_i^{j,0} + \pi$, otherwise $P_i^j \notin \mathcal{P}_i^s$. If $P_i^j \notin \mathcal{P}_i^s$ implies that the *nextCellsAngles* function returned cells$= \emptyset$. The *nextCellsAngles* function can return a null vector, if and only if, each cell $C'_k \in \mathcal{O}$ or $C'_k \in \mathcal{U}$, where $C'_k \in$ cells. Since each cell $C'_k \in$ cells, leads the agent towards obstacles or unexplored cells, hence the path $P_i^j \notin \mathcal{P}_i^s$. Therefore Algorithm 1 correctly classifies whether a path $P_i^j \in \mathcal{P}_i^s$ or $P_i^j \notin \mathcal{P}_i^s$ and hence do not lead the agent into traps. □

The agent A_i utilizes Algorithm 1 to generates safe paths for all of the paths $P_i^j \in \mathcal{P}_i^f$. Since, Theorem 1 guarantees that Algorithm 1 generates safe paths, and the agent A_i uses Algorithm 1 for each of its path $P_i^j \in \mathcal{P}_i^f$, hence the set of paths $P_i^j \in \mathcal{P}_i^s$ are safe set of paths.

Theorem 2 *Algorithm* 1 *generates* $\mathcal{P}_i^s \subseteq \mathcal{P}_i^f$ *and each path* $P_i^j \in \mathcal{P}_i^s$ *is a safe path.*

Proof Assume that a path P_i^j was wrongly classified as a safe path ($P_i^j \in \mathcal{P}_i^s$) by Algorithm 1 and the path P_i^j will lead the agent towards a horizontal trap G_h of depth d. This implies that $C_i^{j,q} \in G_h$ and $\psi_i^{j,q} = 0$, where $C_i^{j,q}$ is the q^{th} cell of the path P_i^j. When agent A_i uses Algorithm 1 for P_i^j, then the *nextCellsAngles* function returns a null vector when it encounters the obstacle cells (x_{j+d+1}^h, y_j^h) or (x_{j+d+1}^h, y_{j+1}^h), or the unexplored cells $(x_{j+d'}^h, y_j^h)$, $(x_{j+d'}^h, y_{j+1}^h)$ where d' represents some cell between (x_{j+2}^h, y_j^h) to (x_{j+d}^h, y_j^h) or (x_{j+2}^h, y_{j+1}^h) to (x_{j+d}^h, y_{j+1}^h) in G_h. Since cells $(x_{j+d+1}^h, y_j^h) \in G_h$ and $(x_{j+d+1}^h, y_{j+1}^h) \in G_h$, the *nextCellsAngles* function will return a null vector and the Algorithm 1 will classify $P_i^j \notin \mathcal{P}_i^s$. Therefore, all the paths generated by Algorithm 1 are safe paths. □

At every time step, each UAV generates a set of safe paths using Algorithm 1. When there is a large obstacle, the number of steps required recursively would be large, and if this process is carried out for every time step then the computational time accumulates. Hence, we develop a heuristics based on other agents path knowledge to reduce the computational time of the Algorithm 1.

Using path knowledge: Each UAV has information about the route it has traveled and also about the routes of the other UAVs that it has met during exploration. As described in Section 3.2, the information that an agent has depends on the kind of information sharing scheme adopted.

Instead of searching for a path from $C_i^{j,q}$ to $C_i^{j,0}$, search for a path from $C_i^{j,q}$ to any cell $C_i^n \in \hat{\mathcal{C}}_i$, where $\hat{\mathcal{C}}_i = \bigcup_{m=1}^{N} R_{im} \bigcup C_i^{j,0}$, is the set of cells and angles from all the routes in $\mathcal{R}_i$. The heuristic algorithm is given in Algorithm 2.

Algorithm 2 Exploration algorithm using recursive search with heuristics

```
1: P_i^s ← ∅;
2: for j=1 to |C_i^q| do
3:    flag ← 0, cells ← C_i^{j,q} ∈ C_i^q, angles←ψ_i^q;
4:    while flag == 0 do
5:       [cells, angles] ← nextCellsAngles(cells, angles);
6:       if C_k == C_i^n, and ψ_k == ψ_i^n or ψ_k == π + ψ_i^n, forall C_k ∈ cells and ψ_k ∈
         angles, C_i^n ∈ Ĉ_i then
7:          flag ←1, P_i^s ← [P_i^s P_i^j];
8:       else
9:          if cells == [] then
10:            flag ← 1;
11:         end if
12:      end if
13:   end while
14: end for
```

Theorem 3 *Algorithm* 2 *also generates safe set of paths* $\mathcal{P}_i^s \subseteq \mathcal{P}_i^f$.

Proof Each cell $C_i^n \in \hat{\mathcal{C}}_i$ is a part of the route of some agent A_j. That is, some agent A_j traveled through C_i^n. If agent A_j used C_i^n implies that the route through C_i^n was classified as a safe path because A_j uses Algorithm 1 to generate a safe path and Theorem 1 proves that Algorithm 1 guarantees to generate safe paths. Thus, when agent A_i uses the cell C_i^n with angle ψ_k or $\psi_k + \pi$ as termination criterion then the path P_i^j can reach cell C_i^n and use the route of A_j. Therefore the path P_i^j is a safe path. The agent uses Algorithm 2 to determine if each path $P_i^j \in \mathcal{P}_i^f$ is a safe path, hence the set $\mathcal{P}_i^s \subseteq \mathcal{P}_i^f$ represents the safe set of paths generated using Algorithm 2 that do not lead the agents to traps. □

Using the route information enables the UAVs to determine a set of safe paths quickly. Initially, when the region is unknown, Algorithm 2 may not yield a significant reduction in computational time. However, as the exploration continues, routes get accumulated and the Algorithm 2 yields better reduction of computational time. Once a set of safe paths are generated, the agent has to select a path. Due to the presence of multiple UAVs in the shared altitude, the agents have to cooperate with each other in selecting paths so that collisions between UAVs do not occur.

5 Path planning

The selection of appropriate paths affects the performance of the exploration. The agents choose paths that can give high yield by moving towards an unexplored region or high cell values. As multiple agents are present in the space, all the agents may not be able to move towards the unexplored territory due to kinematic constraints of the UAV. Therefore we need a conflict resolution scheme that will efficiently allocate the UAVs to explore the unknown region and also avoid inter-UAV collision. Collision

between UAVs can occur if more than one UAV is present at the same cell at the same time or when they crossover from one cell to another during transition. The development of a conflict resolution scheme is relatively easy for cases with global communication, but difficult for agents with limited communication.

There are different ways in which the conflicts can be resolved when the agents are subjected to limited communication. These schemes are based on (1) leader-follower and (2) negotiation-based schemes. In a leader-follower scheme, the agent that has the highest token number is elected as a leader, while the other agents are the followers. The leader makes the first decision and the followers take leader's decision into account and produce de-conflicting paths for themselves. In [23], the lead UAV evaluates its path without taking team members action into account. The followers evaluate their paths by taking all their leaders decisions into account. However in this paper, the leader-follower approach cannot be used directly. For example, consider Fig. 6a where A_2 and A_3 have two feasible next cells, while A_1 has only one possible next cell. Each agent has a unique token number denoted as T_i and the token numbers are in the form $T_1 > T_2 > T_3$. Hence A_1 is the leader and announces its path, A_2 then generates its path based upon this information and announces it as shown in Fig. 6b. Since A_1 and A_2 have selected their next cells without regard for A_3, they have inadvertently removed all possible de-conflicted paths from A_3. On the other hand, if the token were ordered as $T_1 > T_3 > T_2$, then the paths would have been selected as in Fig. 6c resulting in de-conflicted trajectories.

Negotiation-based schemes can be used for the resolution but usually these schemes take more time to arrive at an agreement. Hence, we do not explore the possibility of using these schemes. To resolve the conflict for path planning, we developed a dynamic leader election algorithm based on priority.

In our dynamic leader selection algorithm, the leader is elected dynamically based on the number of free $C_i^{j,1}$ cells. The token number of an agent is incremented

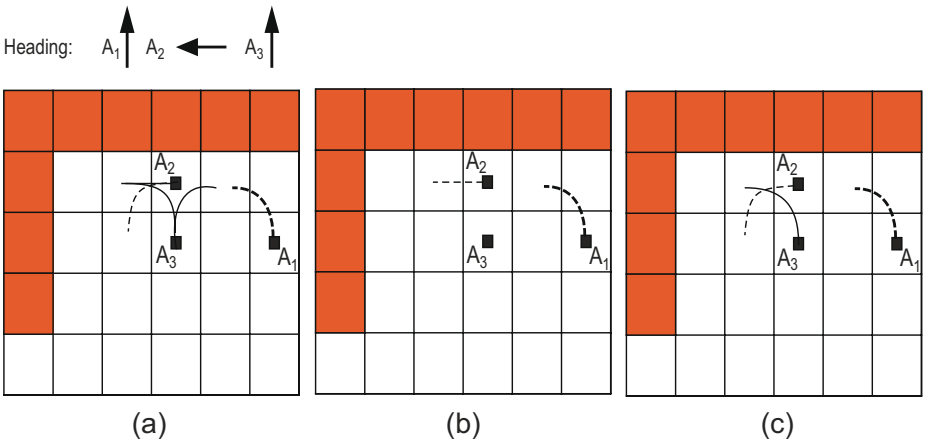

Fig. 6 *(a)* Possible directions in which UAVs A_1, A_2 and A_3 can move. *(b)* Conflict occurs as A_3 does not have any paths to follow. *(c)* Conflict resolution achieved due to priority based token number generation

depending on the number of free $C_i^{j,1}$ cells available for generating feasible paths. The new token number T_i of agent A_i is given by

$$T_i = rand * N + (3 - F) * B, \quad B > N \tag{9}$$

where 3 represents the maximum number of next cells ($C_i^{j,1}$) that any agent can have, F represents the number of next cells available through which feasible paths can be generated, while B is a positive constant integer greater than N (the total number of UAVs). In this method of token number generation, the UAV that has minimum number of feasible next cells would get higher priority than the UAV that has higher number of feasible next cells. Once the leader announces its decision, the followers who have the same number of feasible next cells would regenerate their token numbers. Due to a leaders' decision, a follower may not have the same number of de-conflicted next cells.

In Fig. 6a A_1 has one feasible next cell while agents A_2 and A_3 have two feasible next cells, hence A_1 is the leader and announces its path. After A_1 announces its path, A_2 and A_3 regenerate token numbers and A_3 becomes the next leader. After A_3 chooses its path, A_2 chooses its path. This method of conflict resolution avoids inter-UAV collisions. The paths taken by the UAVs are shown in Fig. 6c.

5.1 *Evaluating paths*

Let $\mathcal{P}_i^s$ be the set of safe paths determined after executing Algorithm 1. The leader evaluates its paths without considering the actions of other UAVs, according to

$$J\left(P_i^j\right) = \sum_{k=1}^{q} V_i\left(C_i^{j,k}\right),\ j = 1, \ldots, |\mathcal{P}_i^s| \tag{10}$$

and chooses a path P_i^s using a greedy strategy

$$s = \arg\max_j J\left(P_i^j\right). \tag{11}$$

The follower UAV has to consider the paths announced by all the UAVs preceding its turn. Assume agent A_i has D_i leaders and let $\mathcal{D}_i$ represent the set of leaders of A_i and let the path selected by each leader agent $A_d \in \mathcal{D}_i$ be $\bar{P}_d^s$. The agent A_i evaluates its paths as

$$J\left(P_i^j\right) = \sum_{k=1}^{q} \bar{V}_i\left(C_i^{j,k}\right) \tag{12}$$

where $\bar{V}_i(C_i^{j,k})$ is given by

$$\bar{V}_i\left(C_i^{j,k}\right) = \begin{cases} V_i\left(C_i^{j,k}\right) & \text{if } C_i^{j,k} \neq C_d^{s,k},\ C_i^{j,k} \neq C_d^{s,k+1},\ \text{and } C_i^{j,k+1} \neq C_d^{s,k},\ \text{for all } d \\ -K & \text{Otherwise} \end{cases} \tag{13}$$

where $K > q$ is a positive number greater than the length of look ahead step q. The condition $C_i^{j,k} \neq C_d^{s,k}$ implies that the cell $C_i^{j,k}$ is in conflict with one of the leader agent paths at the k time step, while conditions $C_i^{j,k} \neq C_d^{s,k+1}$ and $C_i^{j,k+1} \neq C_d^{s,k}$ check

if the paths of the agent are crossing over any of the leader paths. Since, crossing leaders path is a collision hence we assign $-K$ to that path. Once $J(P_i^j)$ is determined, we split the set J as J^+ and J^-, where $J = J^+ \bigcup J^-$, J^+ represents the positive cost paths, while J^- represents the negative cost paths. The negative cost paths represent those paths that are in conflict with some leader $D_i \in \mathcal{D}_i$. The agents use (14) to select a path P_i^s

$$s = \arg\max_j J^+. \tag{14}$$

Theorem 4 *If A_i is a leader then* (10) *and* (11) *generate collision free paths. If A_i is a follower then* (12), (13) *and* (14) *produce a collision free path.*

Proof Agent A_i can be either a leader or a follower. We first prove that A_i generates collision free paths when it is a leader and then prove when A_i is a follower.

Assume A_i is the leader and $\mathcal{P}_i^s \neq \emptyset$. The cost of the path P_i^j is evaluated using (10). Since, all the paths in $\mathcal{P}_i^s$ are collision free paths and A_i is the first agent to make the decision, the costs of the paths will be positive. The agent A_i uses (11) to determine a path that has the maximum benefit. Therefore the selected path $P_i^s \in \mathcal{P}_i^s$ is a safe and collision free path.

Assume that agent A_i is a follower, $\mathcal{P}_i^s \neq \emptyset$ and has $|\mathcal{D}_i|$ number of leaders. The agents use (12) and (13) to determine the cost of the paths by taking the same cell and cross over collisions with other agents into account. If $J^+ \neq \emptyset$, then there exists at least one path $P_i^j \in \mathcal{P}_i^s$ which is not in collision with its leader agents. Since, (14) only selects a path from the collision free set in J^+, the selected path P_i^s is a collision free path.

However, if $J^+ = \emptyset$ implies that the agent is in collision course with at least one of the leaders for each path $P_i^j \in \mathcal{P}_i^s$. Hence, the agent will not have paths to move. Therefore, (12), (13) and (14) will generate a collision free path if there exists one such path. □

Note that the world is discretized and the agents have limited maneuvering capability using the grid (only three directions). If the number of agents are large within a region compared to the number of free paths then some of the agents may not have collision free paths. In that case, there may be inter-UAV collisions. This problem is due to the limited sensor range and communication ranges. The path planning algorithm can only select a path that is not in conflict with other UAVs from the set of safe paths if one such path exists. Once the agent selects a path, it broadcasts the path to other agents. The agents who have not yet made their decision will use this information to evaluate paths. This process is carried out until the mission is completed. The area that can be covered depends on the sensor ranges, which we will study in the next section.

6 Effect of sensor, communication and kinematic constraints

As discussed in Section 4.1, the UAVs have kinematic constraints that may prevent them from covering all of the cells. The UAVs also have limited sensor and communication ranges that limit the ability to inform other agents about paths while

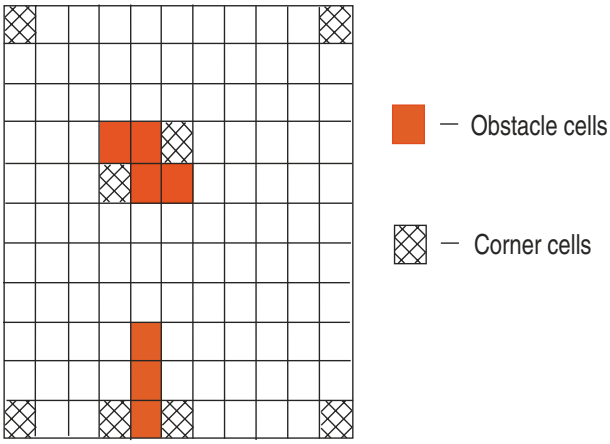

Fig. 7 Cells that cannot be covered by UAVs

traveling along narrow corridors. Due to this limitation collisions may occur. In this section we will describe the situations under which the collisions can happen and the necessary conditions needed to eliminate any possibility of collisions between UAVs.

6.1 Unreachable cells

Figure 7 shows the cells that cannot be reached. The cells in the corners and the traps cannot be reached. The corner cells are defined as: If a cell C_j with its center (x_j, y_j) has a pair of obstacles or a combination of obstacles and the boundary on the orthogonal angles, then that cell cannot be covered. For example, consider the location of cell C_j, if a pair of obstacles are located at $\frac{\pi}{2}$ and 0 angles to (x_j, y_j), then this cell is not covered. Although this cell is a free cell, due to kinematic constraints of the UAVs this type of cells cannot be covered. Therefore the set of reachable cells is represented as $\mathcal{R} = \mathcal{F} \setminus \mathcal{E}$, where $\mathcal{E}$ represents the set of corner cells in the environment. We can say that the UAVs have explored the region if all the cells in $\mathcal{R}$ are visited.

6.2 Sensor and communication range limitations

The agents have limited sensor ranges and hence some of the narrow corridors cannot be covered completely. If the depth of the narrow corridor d is greater than twice the sensor range r_s of the agent, then those corridors will never be covered. For example consider the scenario shown in Fig. 8a, where a corridor of six cells exists and an agent with $r_s = 3$ detects some part of the narrow corridor during exploration. After some time the agent detects the other half of the corridor as shown in Fig. 8b. Thus the agent has sensed the complete passage and can cover it through exploration. On the hand if the sensor range is decreased to $r_s = 2$, then from Fig. 8c we can see that some of the cells in the narrow passage are not detected. When Algorithm 1 is used to generates safe paths, it encounters unexplored cells in the corridor and no paths through the corridor are generated. Hence this corridor will never be covered.

Theorem 5 *If the sensor range $r_s \geq \frac{d}{2}$ then a corridor of depth d will be covered.*

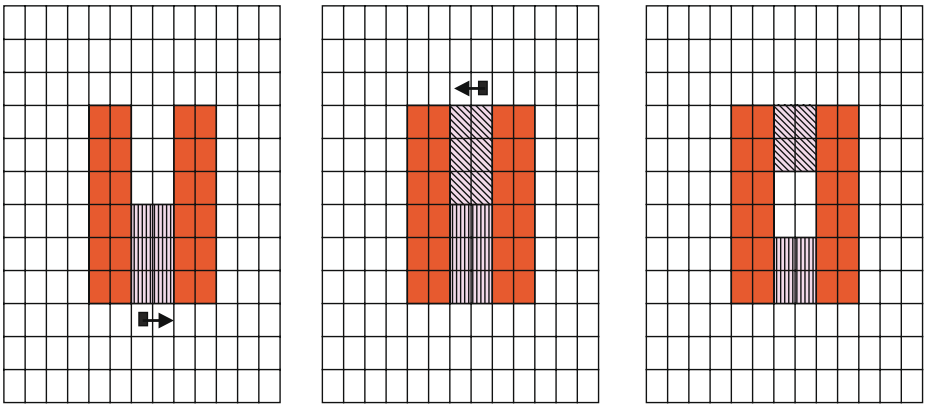

Fig. 8 *(a–c)* Example to show that if the depth of a corridor is greater than twice the sensor range of the agent then that corridor cannot be covered

Proof Consider a corridor G_h as defined in (2) with cell $(x^h_{j+d+1}, y^h_j) \in \mathcal{F}$, i.e., the trap leads the agent towards open space. Assume that the depth of G_h is d and $d > 2r_s$. When an agent A_i is present at cell (x^h_{j-1}, y^j_h) with heading $\frac{\pi}{2}$ or $\frac{3\pi}{2}$, then it can detect cells from (x^h_j, y^h_j) to $(x^h_{j+r_c}, y^h_j)$, and when A_i is present at cell (x^h_{j+d+1}, y^h_j) with heading $\frac{\pi}{2}$ or $\frac{3\pi}{2}$, then it can detect cells from (x^h_{j+d}, y^h_j) to $(x^h_{j+d-r_c}, y^h_j)$ in G_h. Since, $d > 2r_c$ there exists cells $(x^h_{j+r_c+1}, y^h_j), (x^h_{j+r_c+2}, y^h_j), \ldots, (x^h_{j+d-r_c-1}, y^h_j) \in \mathcal{U}$. The Algorithm 1 encounters these undetected cells and does not generate path through G_h. Therefore, to cover G_h the sensor range $r_s \geq \frac{d}{2}$. □

The agents also have limited communication range, which implies that different agents can enter narrow corridors from opposite directions without knowing that they are on a collision course. From Theorem 5 any corridor that has depth less than or equal to $2r_s$ can be covered and to avoid collisions in these corridors the agent should communicate about their movement before entering these corridors. Therefore, the agents should have minimum communication range to ensure that there are no collisions inside the corridors. The minimum communication range also depends on the arrival angle of the agent before entering the passage. For example consider a situation as shown in Fig. 9a where the agents A_1 and A_2 have zero degree heading and these two agents can enter the passage with 90 and 270 heading angles respectively. In order to avoid the agents entering the passage we need communication range of $2r_s + 1$. Now, consider the worst case where the agents are already on the collision course as shown in Fig. 9b. In this scenario, the agents do not have any other paths and will collide. To avoid this situation the agents should have communicated their paths one step earlier. Hence, to avoid collision inside the corridor the agents need to have communication range of $2r_s + 2$.

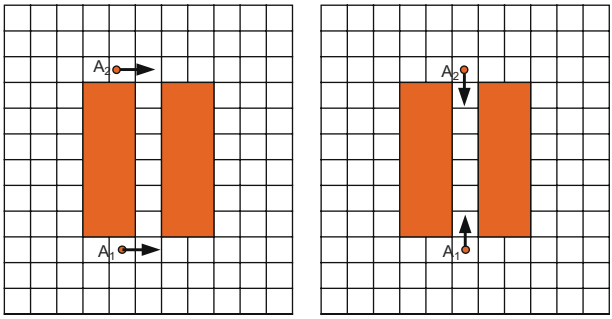

Fig. 9 *a*, *b* Example to show that if the depth of a corridor is greater than twice the sensor range of the agent then that corridor cannot be covered

Theorem 6 *The minimum communication range r_c required to ensure no collisions in narrow corridors is $2r_s + 2$, where $r_s \geq \frac{d}{2}$.*

Proof We consider the worst case scenario of a trap G_h of one cell width and two agents situated at cells $C^h_{j-1} = (x^h_{j-1}, y^h_j)$ and $C^h_{j+d+1} = (x^h_{j+d+1}, y^h_j)$ with heading angles of 0 and π degrees respectively. We assume that $d \leq 2r_s$, and all the cells in G_h have been detected. Since the agents are heading into G_h in opposite directions they will collide. In order to communicate between the agents the communication range is $2r_s + 1$, but to ensure there was no collision the agents should have communicated at least one step earlier. Hence the minimum communication range required to ensure no collision occur in narrow corridors is $2r_s + 2$. □

As described in Section 5, the agents use a leader follower strategy, where the leader decides first and the follower agents have to take all its preceding leaders decision into account while making its decision. Assume that there exists a trap G_h of depth $d = 2r_s$ and agents A_i and A_{i+1} are located at cells (x^h_{j-2}, y^h_j) and (x^h_{j+d+2}, y^h_j) with heading angles 0 and π respectively. We also assume that $r_c \geq 2r_s + 2$. Let A_i be the leader and A_{i+1} be the follower, therefore A_i decides its path and let this path be $P_i = \{(x^h_{j-1}, y^h_j), (x^h_j, y^h_j), \ldots, (x^h_{j+r_s-1}, y^h_j)\}$. Now, the follower agent takes the leader path into account and assume that it decides on a path $P_{i+1} = \{(x^h_{j+d+1}, y^h_j), (x^h_{j+d}, y^h_j), \ldots, (x^h_{j+d-r_s-1}, y^h_j)\}$. The two paths are non-intersecting as d is equal to $2r_s$ and both the paths are leading the agent towards the same corridor. Because the agents plan only for r_s steps and the depth of G_h is $2r_s$, this may result in a collision and to avoid this situation we need to modify the algorithm of evaluating the paths.

In order to re-evaluate the paths, agents must know when their paths will lead the UAV into corridors. Assume A_i is located at cell C_j and let the number of paths emanating through cell C^1_j be $|\mathcal{P}^{s,1}_i|$, cell C^2_j be $|\mathcal{P}^{s,2}_i|$, and cell C^3_j be $|\mathcal{P}^{s,3}_i|$. If $|\mathcal{P}^{s,k}_i| < 3$ then all the paths emanating through cell C^k_j will enter the corridor. When this situation is detected, the follower agent can virtually extend those paths from $q = r_c$ to $q = r_c + 2$ to check if the collision occurs. If there is a collision then those paths are not considered as described in Algorithm 3.

Note that because of the leader follower strategy, only one agent can enter G_h, while the follower agent will extend its path length to check for a collision in G_h.

Algorithm 3 Re-evaluate paths

1: Determine $|\mathcal{P}_i^{s,1}|$, $|\mathcal{P}_i^{s,2}|$, $|\mathcal{P}_i^{s,3}|$;
2: **for** $\lambda = 1$ to 3 **do**
3: **if** $|\mathcal{P}_i^{f,\lambda}| < 3$ **then**
4:

$$J(P_i^j) = \sum_{k=1}^{q+2} \bar{V}_i(C_i^{j,k}) \tag{15}$$

where $\bar{V}_i(C_i^{j,k})$ is given by

$$\bar{V}_i(C_i^{j,k}) = \begin{cases} \text{Use (13)} & \text{if } k \leq q \\ -K & \text{if } C_i^{j,k+1} == C_d^{s,k'}, \text{ for some } d, k' \\ 0 & \text{Otherwise.} \end{cases} \tag{16}$$

5: **end if**
6: **end for**

Once the agent decides on its path then it will move towards the next cell in the path. The agents use the exploratory system at every step to explore the region. The effectiveness of the exploration is analyzed using simulations.

7 Results

The agents use the proposed exploration algorithm to survey the region consisting with obstacles of unknown size and location. The validation of the proposed algorithm is carried out using Monte Carlo simulations. Initially, we will demonstrate the functioning of the algorithm using a sample scenario, where the agents avoid traps and pass through safe corridors. The time required to cover the entire region depends on (a) the kind of information exchanging mechanism adopted by the agents (b) the number agents in the mission and (c) the sensor ranges. In this section, we will study the effect of these parameters on the coverage times.

7.1 Sample scenario

We demonstrate the functioning of the proposed technique using a simulated 20×20 square grid with obstacles. We use five UAVs with sensor range $r_s = 4$ and communication range $r_c = 10$. The agents use the complete information exchange scheme during exploration.

Figure 10a shows the initial positions of the agents with the corridors C1 and C2 in rectangles, and the traps in ellipse. The corridor C1 has a depth of 6 and C2 has a depth of $d = 7$. Initially, the entire region is unknown and hence the cells are not colored. Once the agents detect the free cells, the cells are colored orange. The agents must avoid the traps and explore the corridors. Figure 10b shows the routes of the agents after two steps. From the figure we can see that agent A_1 avoids trap T1 indicated in ellipse as the exploration technique encounters unexplored cells in the trap. The agent A_2 detects some of the cells in the corridor C1. Since the corridor is

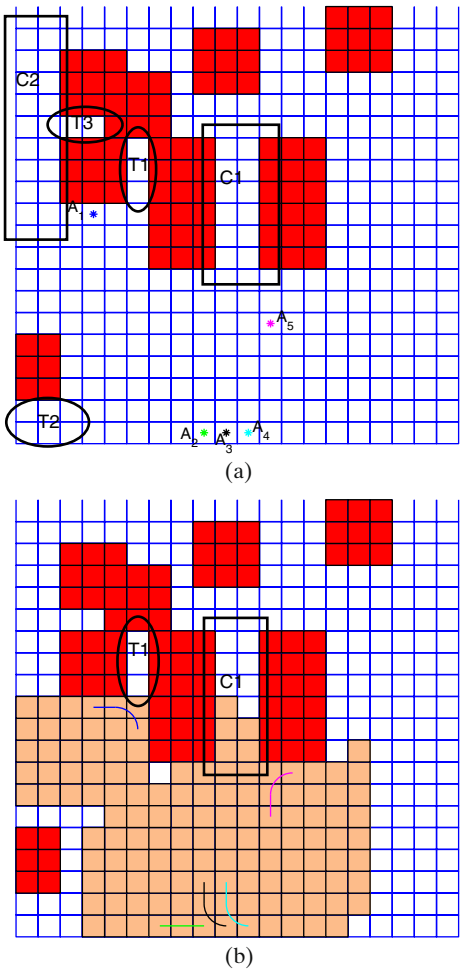

Fig. 10 *(a)* Initial region of interest to explore with traps and corridors. *(b)* The routes of the agents after two steps. A_1 avoids trap T1 and A_5 detects few cells in the corridor C1

not known completely the agent does not travel through it. The free cells detected for the two steps by all the agents are colored orange.

After seven exploration steps, the cells in C1 are known up to three steps as shown in Fig. 11a. In the figure we can also see that A_2 avoids the trap indicated with an ellipse. Figure 11b shows the routes of the agents after 42 steps. In the figure we can see that A_5 detects all the cells in C1 except for one cell. Hence it can use the corridor to explore. In the figure we can see that agents A_1 to A_4 are moving towards the unexplored region on the right side of the search space after completely exploring the left side of the search space. Figure 12a shows one part of the corridor being explored by A_5. The complete region was explored in 117 steps by all the agents and the explored region is shown in Fig. 12b.

From Figs. 10a and 12b we can see that the corridor C2 is not explored because it is not completely detected by the agents. Hence, given a region the agents can cover all the cells that are feasible taking its limitation on the sensor range into account. In order to cover C2 we may have to increase the sensor range. Next, through

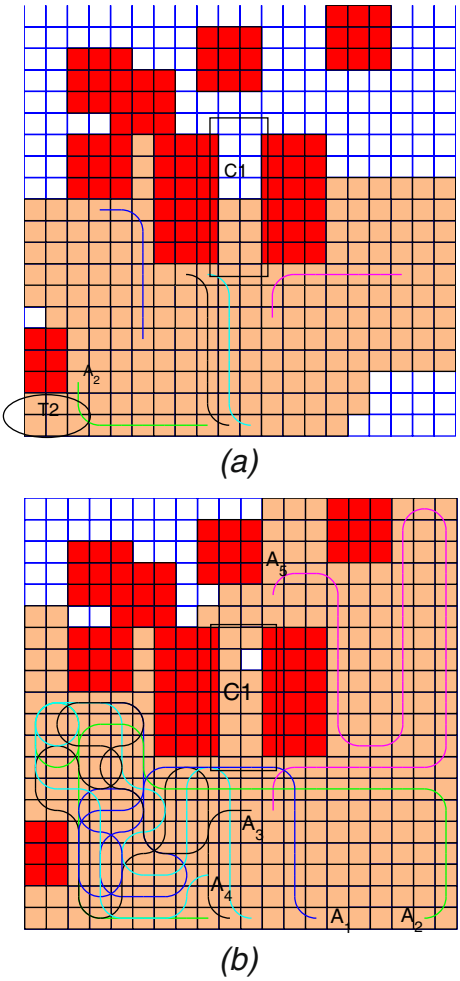

Fig. 11 *(a)* The routes of the agents after seven steps. A_2 avoids a trap T2, and the cells in corridor C1 are sensed up to three steps in depth. *(b)* A_5 detects the remaining undetected cells of C1 and hence the entire corridor is now known and the agent can use this corridor to explore

Monte-Carlo simulation we study the effect of sensor range and information exchange on the percentage area covered.

7.2 Effect of increase in sensor range

With increase in sensor range, the UAVs can detect larger regions of the environment. So, the agents can plan better paths. Naturally, with increase in sensor range the coverage time decreases. As we assumed that the communication range can be greater than or equal to the sensor range, with increase in sensor range the communication range also increases. Hence, the information sharing is better and results in faster area coverage. We conducted simulations for sensor range $r_s = 2, 3$ and 4 with different numbers of agents. The simulations were carried out for 20 maps of 20×20 cells with the position and heading of the agents selected randomly.

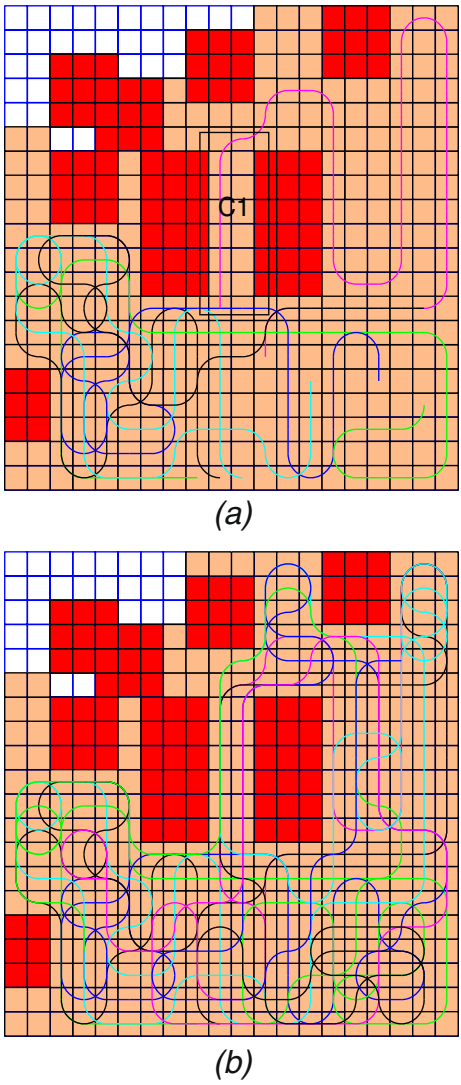

Fig. 12 *(a)* Agent A_5 passes through the corridor. *(b)* The region is covered in 117 steps

Figure 13 shows the results obtained from the simulations. For the two agent case, we can see that 100% area coverage is not achieved for agents with sensor ranges $r_s = 2$ and $r_s = 3$. However for sensor range $r_s = 4$, the agents are able to cover the entire region. Note that for smaller sensor range, the narrow corridors or the regions inside the obstacles are never completely detected and hence cannot be covered. For example, consider the coverage achieved using five agents and sensor range $r_s = 4$ as shown in Fig. 12. But, with sensor range $r_s = 2$ and 3, the corridor C1 cannot be covered as shown in Fig. 14.

During exploration all the agents successfully avoided traps for all the maps. However, the percentage number of corridors that were safe were not completely explored due to sensor range limitations. With sensor range r_s=2, only 22.45% of

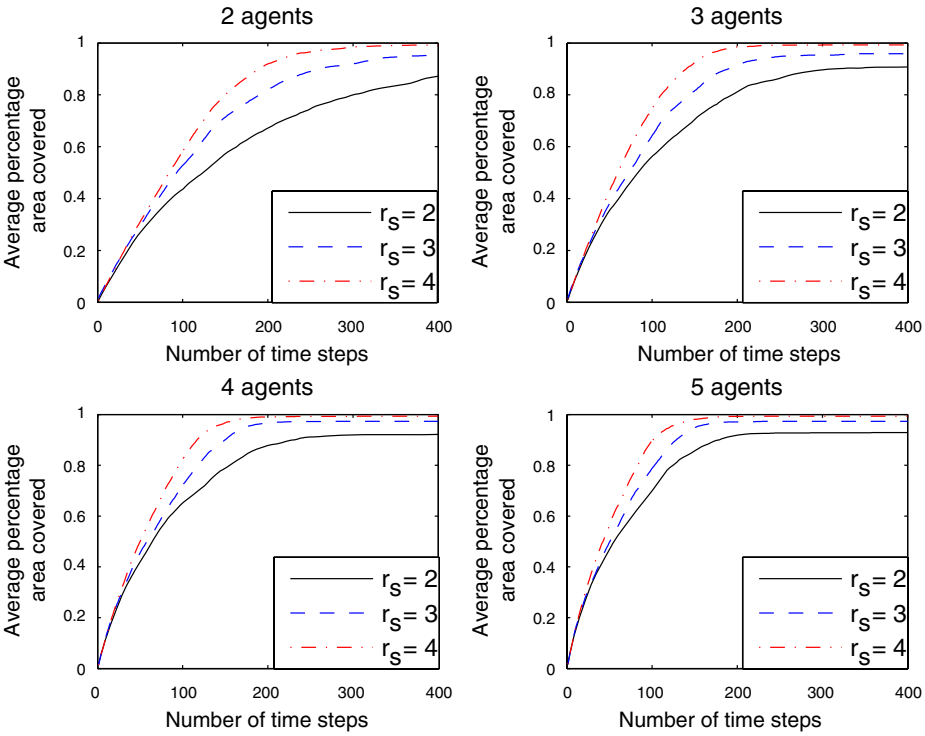

Fig. 13 Mean percentage area covered for a given number of agents and varying sensor ranges

the available narrow corridors were explored. Since the percentage narrow corridors explored is less this resulted in lower area coverage as shown in Fig. 13. With increase in sensor ranges to $r_s = 3$ and $r_s = 4$ the percentage narrow corridors covered increased to 42.86% and 83.67% respectively as shown in Fig. 13.

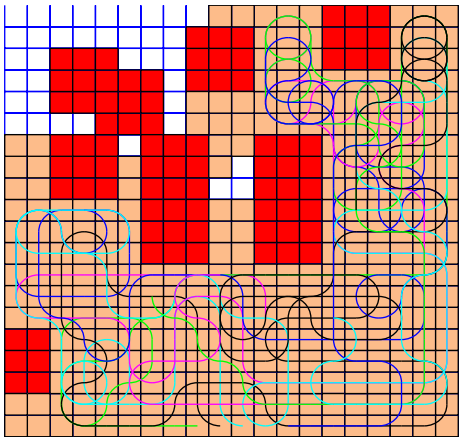

Fig. 14 Area covered by five agents with sensor range $q = 2$ and 3

7.3 Effect of information exchange

Information exchange plays an important role in determining the coverage time. In Section 3.2, we proposed three types of information exchange schemes that the agents can use to enhance the performance in terms of decreasing the time to cover a region. Intuitively, the performance of coverage with no information will be worst, while the complete information exchange scheme the best and the self information exchange scheme will perform in between the no information and the complete information schemes. We conducted Monte Carlo simulations to study the effect of information on coverage time with a fixed number of agents and sensor range.

The average percentage area covered for 20 random maps with obstacles is shown in Fig. 15. For the simulations we assumed the sensor and the communication range to be $r_s = 4$. As expected, from the Fig. 15a we can see that the performance of complete and self information schemes is better than the no information exchange scheme. However, in Fig. 15b, c and d the performance of self information and complete information exchange schemes is similar. When agents meet, they share information, but if the time interval between meetings is large then the disparity between their maps increases. If the agents meet often then the fidelity of their maps is higher. Thus one possible reason that agents using self information scheme

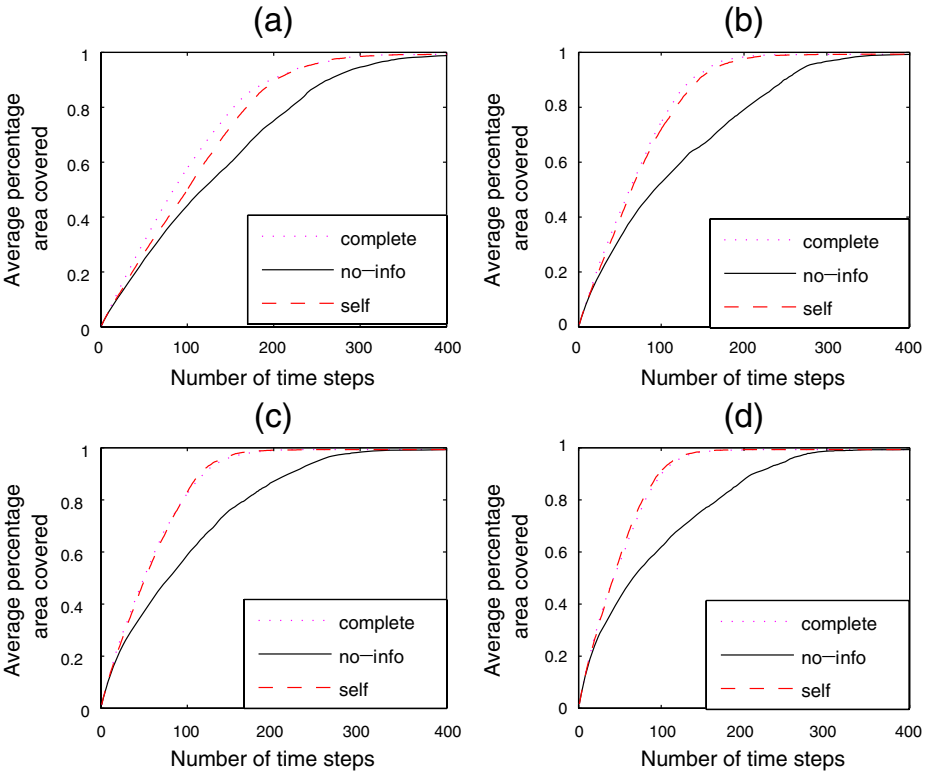

Fig. 15 *(a)* Percentage area covered with two agents. *(b)* Percentage area covered with three agents. *(c)* Percentage area covered with four agents. *(d)* Percentage area covered with five agents

performing similar to complete information scheme is due to frequent meetings between the agents.

7.4 Effect of increase in number of agents

Monte-Carlo simulations were carried out for 20 maps with random obstacle and UAV positions. The time taken to cover a region depends on the number of agents exploring the region. Hence, each set of simulations was carried out for different number of UAVs (two to five UAVs). Increasing the number of UAVs causes the area to be covered in less time. The performance curves in Fig. 16 show that the time taken for five agents to cover the region is less that for two agents. In the figure we can see that with increase in the number of agents, the coverage time decreases. However, with increase in number of agents greater than 3, the benefit obtained in coverage time is marginal for this size of the map.

We carried out another set of experiments on a large search space (50×50 grid) with ten and 20 agents, the sensor range was $r_s = 4$ and communication range $r_c = 10$. Figure 17 shows the performance achieved by the agents. From the figure we can see that the percentage area covered with large number of agents is quicker than using low number of agents. Also we can see that with increase in sensor range the performance also increases. During the simulations there were no reported conflicts.

7.5 Comparison between optimal path planning and heuristic path planning

The proposed path planning technique (Algorithm 3) does not yield optimal solutions. Hence, we conducted an experiment to determine the closeness of the solution to that of optimal. In this experiment, we consider 20 different maps with randomly placed obstacles and five agents.

From the Fig. 18 we can see that the optimal path planner performed better than the heuristic path planner. However, the performance of both the strategies are similar until 70% of the area has been covered, but after that there is a significant

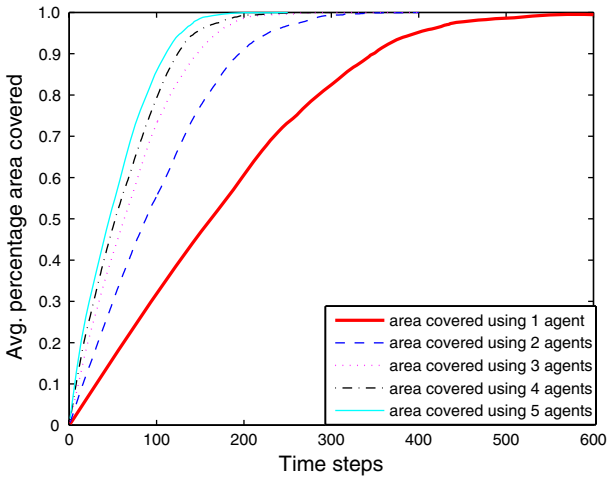

Fig. 16 Percentage area covered for different number of agents

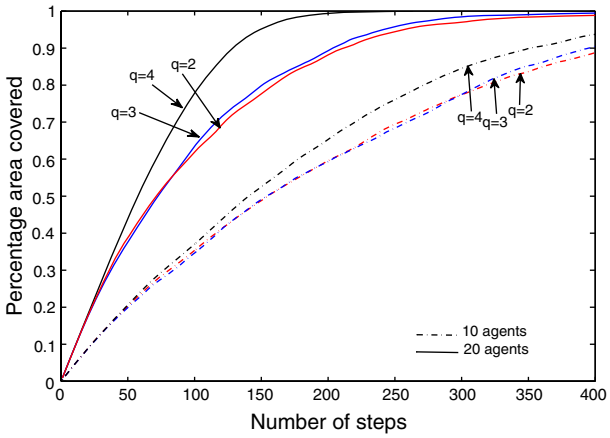

Fig. 17 Percentage area covered for large search space

difference in performance. This is because the agents using the optimal path planner efficiently cover the region.

Even though the optimal strategy covers the region quickly, the computational time required to compute a decision takes 14.37 s. While the heuristic path planner taken 0.13 s to compute a decision. Hence, there is a trade off between computation time and the performance achieved.

7.6 Time taken by the exploration technique

As described in Section 4.2, the exploration technique may take longer time to determine safe paths. To reduce the search time for the technique we proposed a path knowledge based heuristic. An experiment with five agents and a sensor range of $r_s = 4$ for 20 maps was conducted to determine the time taken by the exploration technique with and without heuristics. Figure 19 shows the average time taken by the exploration technique. From the figure we can see that the time taken by the exploration technique *without heuristic* is almost 4 times *more* than with

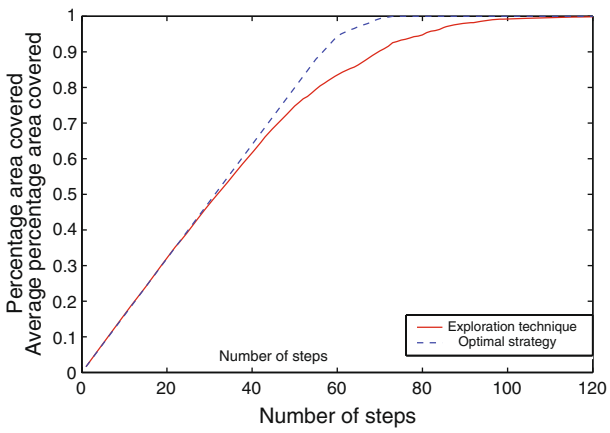

Fig. 18 Performance of optimal and heuristic path planners

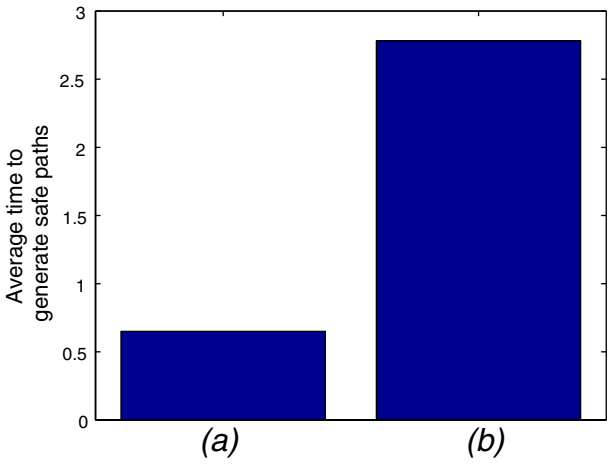

Fig. 19 *(a)* Average time taken for the exploration technique with heuristics to determine safe set of paths. *(b)* Average time taken for the exploration technique without heuristics to determine safe paths

heuristic. Therefore, storing the routes of the agents and using them for safe path determination reduces the computational time for safe path generation.

8 Conclusions

In this paper, we proposed a technique for multiple UAVs exploring an unknown region. The terrain contains obstacles of unknown shape, size and position. The agents build the map of the region online by sensing the environment and receiving information from other agents. From the map, a safe set of paths are generated using an exploration technique that ensures the paths do not direct the UAVs into traps. A heuristic based on path knowledge was developed to reduce the search time for safe paths. We also developed a cooperative path planning technique for the UAVs to select a path from a safe set in cooperation with neighboring agents, while avoiding inter-UAV collisions. Using simulations we have shown that the UAVs cover the feasible region, and that the time taken to cover the region decreases with an increase in the number of agents. We also showed that with increase in sensor range, the time to cover the region decreases. The time taken for the exploration technique with heuristic is almost 4 times less than that taken without using the heuristic.

Acknowledgements This work was partially funded by the National Science Foundation under Information Technology Research Grant CCR-0313056 and by NASA under STTR contract number NNA04AA19C to Scientific Systems Inc, and Brigham Young University and by the Air Force Office of Scientific Research award No. FA9550-04-0209.

References

1. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: A taxonomy for multiagent robotics. Auton. Robots **3**(4), 375–397 (1996)
2. Guzzoni, D., Cheyyer, A., Julia, L., Konolige, K.: Many robots make work short. AI Mag. **18**(1), 55–64 (1997)

3. Choset, H.: Coverage for robotics-A survey of recent results. Ann. Math. Artif. Intell. **31**, 113–126 (2001)
4. Rekleitis, I., Lee-Shue, V., New, A.P., Choset, H.: Limited communication, multi-robot team based coverage. In: Proc. of the IEEE Internation Conference on Robotics and Automation, pp. 3462–3468. New Orleans, LA (2004)
5. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. Ann. Math. Artif. Intell. **31**(1–4), 77–98 (2001)
6. Hazon, N., Kaminka, G.A.: Redundancy, efficiency and robustness in multi-robot coverage. In: Proc. of the IEEE International Conference on Robotics and Automation (2005)
7. Agmon, N., Hazon, N., Kaminka, G.A.: Constructing spanning trees for efficient multi-robot coverage. In: Proc. of IEEE International Conference on Robotics and Automation (2006)
8. Hazon, N., Mieli, F., Kaminka, G.A.: Towards robust on-line multi-robot coverage. In: Proc. of IEEE International Conference on Robotics and Automation (2006)
9. Zheng, X., Jain, S., Koenig, S., Kempe, D.: Multi robot forest coverage. In: Proc. of the IEEE/RJS Interational Conference on Intelligent Robots and Systems, pp. 3852–3857. Edmonton, Alberta, Canada (2005)
10. Even, G., Garg, N., Konemann, J., Ravi, R., Sinha, A.: Min-max tree covers of graphs. Oper. Res. Lett. **32**, 309.315 (2004)
11. Batalin, M.A., Sukhatme, G.S.: Spreading out: a local approach to multi-robot coverage. In: Proc. of Distributed Autonomous Robotic Systems, pp. 373–382. Fukuoka, Japan (2002)
12. Wagner, I.A., Lindenbaum, M., Bruckstein, A.M.: Distributed covering by ant-robots using evaporating traces. IEEE Trans. Robot. Autom. **15**(5), 918–933 (1999)
13. Spires, S.V., Goldsmith, S.Y.: Exhaustive geographic search with mobile robots along space-filling curves. Lect. Notes Comput. Sci. **1456**, 1–12 (1998)
14. Koenig, S., Szymanski, B., Liu, Y.: Efficient and inefficient ant coverage methods. Ann. Math. Artif. Intell. **31**, 41–76 (2001)
15. Balch, T., Arkin, R.C.: Communication in reactive multiagent robotic systems. Auton. Robots **1**(1), 27–54 (1994)
16. Roy, N., Dudek, G.: Collaborative robot exploration and rendezvous: algorithms, performance bounds and observations. Auton. Robots **11**(2), 117–136 (2001)
17. Zolt, R.M., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: Proc. of the IEEE Conference on Robotics and Automation, pp. 3016–3023. (2002)
18. Yamauchi, B.: Fronteir-based exploration using multiple robots. In: Proc. of the International Conference on Autonomous Agents, pp. 47–53. Minneapolis, Minnesota (1998)
19. Burgard, W., Moors, M., Stachniss, C., Schneider, F.: Coordinated multi-robot exploration. IEEE Trans. Robot. **21**(2), 376–386 (2005)
20. Polycarpou, M., Yang, Y., Passino, K.: A cooperative search framework for distributed agents. In: Proc. of the IEEE Symposium on Intelligent Control, pp. 1–6. Mexico City, Mexico (2001)
21. Flint, M., Gaucherand, E.F.A., Polycarpou, M.: cooperative control of UAVs searching risky environments for targets. In: Proc. of the IEEE Conference on Decision and Control, pp. 3568–3572. Maui, Hawaii (2003)
22. Sujit, P.B., Ghose, D.: Search using multiple UAVs with flight time constraints. IEEE Trans. Aerosp. Electron. Syst. **40**(2), 491–509 (2004)
23. Beard, R.W., McLain, T.W.: Multiple UAV cooperative search under collision aviodance and limited range communication constraints. In: Proc. of the IEEE Conference on Decision and Control, pp. 2543–2548. Maui, Hawaii (2003)
24. Pongpunwattana, A., Rysdyk, R.: Real-time planning for multiple autonomous vehicles in dynamic uncertain environments. J. Aero. Comput. Inform. Comm. **1**, 580–604 (2005)
25. De Mot, J., Kulkarni, V., Gentry, S., Gavrilets, V., Feron, E.: Coordinated path planning for a UAV cluster. AINS Symposium. Los Angeles, California (2002)
26. Rubio, J.C., Vagners, J., Rysdyk, R.: Adaptive path planning for autonomous UAV oceanic search missions. In: Proc. of the AIAA Intelligent Systems Technical Conference (2004)
27. Bertuccelli, L.F., How, J.P.: Search for dynamic targets with uncertain probability maps. In: Proc. of the American Control Conference. Minneapolis, Minnesota (2006)
28. Schouwenaars, T., How, J.P., Feron, E.: Multi-vehicle path planning for non-line of sight communication. In: Proc. of the American Control Conference. Minneapolis, Minnesota (2006)
29. Salva, K., Bullo, F., Frazzoli, E.: The coverage problem for loitering dubins vehicles. In: Proc. of the IEEE Conference on Decsion and Control (2007)
30. Sujit, P.B., Sinha, A., Ghose, D.: Multiple UAV task allocation using team theory. In: Proc. of the IEEE Conference on Decision and Control, pp. 1497–1502. Seville, Spain (2005)

31. Arsie, A., Frazzoli, E.: Efficient routing of multiple vehicles with no explicit communciations. Int. J. Robust Nonlinear Control **18**(2), 154–164 (2007)
32. Wagner, I.A., Lindenbaum, M., Bruckstein, A.M.: Distributed covering by ant-robots using evaporating traces. IEEE Trans. Robot. Autom. **15**(5), 918–933 (1999)
33. Saunders, J., Beard, R.: Obstacle avoidance using circular paths. In: Prof of the AIAA Guidance, Navigation and Control Conference and Exhibit, Hilton Head Island, South Carolina, AIAA-2007-6604 (2007)