Motion Coordination of Swarm Robots for Mobile Target Search

Hai Liu, Shujin Ye, Chris Y. T. Ma, Yue Wang, *Member, IEEE*, and Tse-Tin Chan, *Member, IEEE*

***Abstract*—With the rapid advancement of robotics technologies, a group of robots are able to communicate with one another by wireless transmissions and form a robot swarm. Robot swarm has many applications and a typical one is target search in which swarm robots are sent to places that might be dangerous for human workers, and they coordinate with one another to search for targets such as survivors in a disaster. However, motion coordination of swarm robots for target search has received little attention especially when the targets are mobile. In this work, we develop a motion coordination algorithm for swarm robots to search for targets in an unknown area. Our basic idea is to divide the search area into grids and build a gray-scale map in which each grid is associated with a gray scale indicating the efficiency of searching targets in this grid. The Voronoi diagram is adopted to coordinate swarm robots to search different portions of the search area for maximizing search efficiency. By theoretical analysis, our motion coordination algorithm is validated to ensure that all static targets are guaranteed to be found. We derive an upper-bound on the total time for robots to traverse all the grids in the search area. Extensive simulations are conducted and the results show that the proposed motion coordination algorithm outperforms the state-of-the-art and achieves a success rate of over 90% in finding all mobile targets with low search latency.**

***Note to Practitioners*—This paper was motivated by the problem of target search in an unknown area (e.g., the search for Malaysia Airlines MH370). With advanced robotics technologies, swarm robots could be sent to the area to perform a search mission. This work suggests a motion coordination algorithm for swarm robots to search for both static and mobile targets in an unknown area with possible obstacles. The proposed algorithm divides the search area into grids and each grid is associated with a gray scale indicating the efficiency of searching targets in this grid. This gray-scale map guides the robots in target search. In real applications, robots are normally powered by batteries and are prone to fail, requiring efficient coordination of the robots to maximize search efficiency. A Voronoi mask is further introduced to coordinate swarm robots to search different portions of the area. This mask divides the space based on the robots' current positions, assigning each robot a specific region that is closer to it than to others, leading to efficient and coordinated operation. We mathematically derive an upper-bound on the total time for robots to traverse all the grids in the search area. We then show by simulations that the proposed motion coordination algorithm outperforms the state-of-the-art and coordination of robots is critical to improving success rates and minimizing search latency. The proposed algorithm has not yet been tested in a robot testbed which is our future work.**

(Corresponding author: Shujin Ye.)

Hai Liu and Yu Tak Ma are with the Department of Computer Science, The Hang Seng University of Hong Kong, Hong Kong SAR, China (e-mail: hliu@hsu.edu.hk; chrisma@hsu.edu.hk).

Shujin Ye is with the School of Data Science and Engineering, Guangdong Polytechnic Normal University, Guangzhou, Guangdong, China (e-mail: yeshujin@gpnu.edu.cn).

Yue Wang and Tse-Tin Chan are with the Department of Mathematics and Information Technology, The Education University of Hong Kong, Hong Kong SAR, China (e-mail: wyue@eduhk.hk; tsetinchan@eduhk.hk).

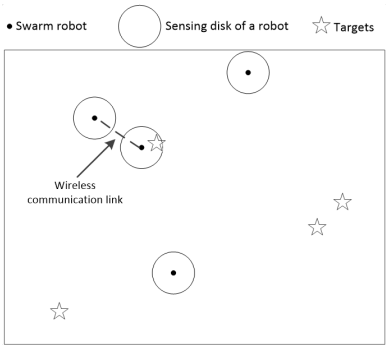


Fig. 1. Target search using swarm robots.

***Index Terms*—Swarm robots, target search, motion control, motion coordination, mobility control.**

I. Introduction

FROM small toasters to huge industrial machines, robots are already an indispensable part of our daily lives. With robotics technologies advancing rapidly, a group of robots are able to communicate with one another and form a robot swarm to realize new applications. Examples of swarm robots include various types of drones, unmanned vehicles (e.g., Google driverless cars), UAVs (unmanned aerial vehicles), AUVs/UUVs (autonomous underwater vehicle/unmanned underwater vehicles), and unmanned ships. These robots are usually battery powered. Each robot is equipped with at least one wireless transmitter to exchange information with its neighbors within the communication range. The robots can carry sensors such as accelerometers, infrared detectors, microphones and cameras. With the capabilities of moving, sensing and communicating, swarm robots have been used in many applications including surveillance, search and rescue, payload delivery and military missions.

Target search, in which a collection of robots are sent to search for targets in a specific area (see Fig. 1), is an important application of swarm robots. Swarm robots can be effectively utilized for target search in a variety of practical applications, including the following three examples.

- In robot-assisted exploration (e.g., mine exploration), sending human miners into drill holes of a mine is dangerous because there is a risk of collapses. Mobile mining robots, equipped with different sensors, are more suitable to explore mining holes than human miners. These robots can go into an underground tunnel and apply the motion coordination algorithm for mine exploration. In this way, the human miners can be safely above

ground and collaborate with the mining robots to get useful information without going into dangerous areas. The targets are normally static in this application.
- In a disaster rescue (e.g., earthquake and tsunami), swarm robots can be sent to places that are too dangerous for human workers. Swarm robots can coordinate with one another to collect data and search for targets in a wide area [1]–[3]. In this application, the targets (e.g., survivors) could be either static or mobile.
- In security and military missions [4], swarm robots (e.g., UAVs/UUVs) can be programmed to patrol a specific area for intruders, or follow the order to search for enemies (e.g., submarines) in the area [5], [6]. In this application, the targets (e.g., intruders and enemy submarines) are normally mobile.

Target search is a challenging problem due to the following reasons. 1) Search area is wide while the lifetime of robots is limited as robots are normally powered by batteries. It requires swarm robots to cooperate for an effective search; 2) Targets can be moving. A target could move into an area that is just searched before; and 3) Robots are prone to fail. During the search, a robot could fail due to power depletion, hardware failure, or malicious attack. Existing works have been devoted to target search. Some of these works consider static targets only such as [7]–[13], while some others consider mobile targets [14]–[19]. A review of these works is given in Section II.

In this paper, we developed a motion coordination algorithm for swarm robots to search for targets in an unknown area. We consider both static targets (e.g., mine) and mobile targets (e.g., intruders). With the proposed algorithm, swarm robots are able to perform effective target search by coordinating with one another. The major contributions of this paper can be summarized as follows.

- We propose novel concepts of gray-scale map and Voronoi mask. The gray-scale map divides the search area into grids and a gray scale of each grid demonstrates search efficiency of the grid. The Voronoi mask is designed to coordinate robots to search in different areas so as to maximize search efficiency.
- Based on the gray-scale map and the Voronoi mask, we propose a motion coordination algorithm for searching targets in an unknown area. The proposed algorithm enables each robot to identify an efficient search path and it is applicable to both static and mobile targets.
- We perform theoretical analysis on the proposed algorithm. By theoretical analysis, our motion coordination algorithm is validated to ensure that all static targets are guaranteed to be found if coefficient of the Voronoi mask is equal to zero and the side length of the grid is equal to the length of the search path. We derive an upper-bound on the total time for robots to traverse all the grids in the search area.
- We conduct extensive simulations to evaluate performance of the proposed motion coordination algorithm. Simulation results show that our algorithm outperforms the state-of-the-art and achieves a success rate of over 90% in finding all mobile targets, and the communication among robots is crucial to improve the search efficiency in terms of the search latency and the success rate.

The rest of the paper is organized as follows. Related works are reviewed in Section 2. System model is described in Section 3. We present a motion coordination algorithm in Section 4 and theoretical analysis in Section 5. Simulation results are presented in Section 6. We conclude our work in Section 7.

II. Related Works

The relevant literature on target search can mainly be divided into two categories: searching for static targets and searching for mobile targets.

A. Searching for Static Targets

Several studies have examined searches for a single static target. Dimidov et al. [8] analyzed the efficiency of two classes of random walk search strategies, namely the correlated random walk and Lévy walk, for discovering a static target in either a bounded or open space. The search strategies were tested through simulations that used up to 30 real robots (Kilobots). During the experiment, robots communicate to share information about the discovery of target. Sakthivelmurugan et al. [9] introduced a number of searching strategies for the detection and retrieval of a static target, including the straight-line, parallel-line, divider, expanding square, and parallel sweep approaches. Experiments were conducted with up to four robots in an environment with known boundaries and they showed that a parallel sweep with the divider approach was the most efficient strategy. Wang et al. [10] formulated a partially observable Markov decision process (POMDP) model for the indoor environment object search problem. A belief state in the POMDP can be a room or an unknown place. They use Gaussian mixture model (GMM) to model the distribution of belief states. To reduce the number of belief states, they use a graph structure called belief road map to represent the explored area. Czyzowicz et al. [28] proposed deterministic algorithms for robots to coordinate and search for a target in an area. Their algorithms are able to handle accidental and malicious faults of the robots, and shown to be asymptotically optimal. However, the deterministic nature of their algorithms makes them inapplicable to search for an intelligent target which tries to avoid detection.

Searching for multiple static targets was also studied. Tang et al. [11] proposed a particle swarm optimization (PSO) algorithm to search for multiple targets. The algorithm uses a constriction-factors-based grouping strategy to group robots into sub-groups. They assumed that each target emits a signal that can be detected by robots, e.g., light and temperature, and the signal strength decreases as the distance from the target increases. Wi-Fi is used for communication between robots. Rango et al. [12] considered the mine detection problem using a modified ant colony algorithm. Given multiple mines that were randomly distributed in an unexplored area, the problem involved moving robots to explore the area and detect all of the mines using the minimum amount of time.

TABLE I
DIFFERENCES BETWEEN THE PROPOSED WORK AND EXISTING SOLUTIONS IN TARGET SEARCH

	Consider single / multiple targets	Static / mobile target(s)	Consider message exchange for coordination of robots	Require information of motion model of target(s)	Consider obstacles	Other requirements
This work	Multiple	Mobile	Yes	No	Yes	No
[20]	Multiple	Mobile	Yes	No	No	No
[21]	Multiple	Static	Yes	No	No	No
[22]	Multiple	Static	Yes	No	Yes	No
[14]	Multiple	Mobile	Yes	No	Yes	There is a requirement on the minimum number of robots
[16]	Single	Mobile	Yes	No	Yes	Robots must be able to release pheromone
[17]	Single	Mobile	Yes	Yes	Yes	Robots know search maps, e.g., the positions of obstacles
[18]	Single	Mobile	No	Yes	Yes	The original target position is known approximately in advance
[19]	Single	Mobile	Yes	Yes	No	Robots know search maps
[23]	Single	Mobile	No	No	Yes	Robots know search maps
[24]	Single	Static	Yes	No	No	No
[25]	Multiple	Static	Yes	No	No	No
[26]	Multiple	Mobile	Yes	No	No	No
[27]	Multiple	Mobile	Yes	No	No	No

During the search process, the robots lay repelling anti-pheromones on the explored area. When choosing their next movements, the robots perceive pheromone information from their surroundings and travel to undetected regions with the least pheromone intensity. Once one or more robots discover a mine, attractive pheromones are deposited to recruit other robots. After the required number of robots are attracted to the mine location, they work cooperatively to disarm the mine. Zhang et al. [13] presented a prior knowledge based approach to search dynamic objects in a home environment. They consider that the targets are dynamic, e.g., a laptop may change its location often, but they are static during the search. The target location and relationship between targets are used as a prior knowledge to guide robots for searching. Al Amin et al. [25] proposed a Gravitational-Search-Algorithm-based swarm coordination model for multiple UAVs detecting unknown static targets in unknown environments. Their approach models UAVs as masses in a gravitational field, where gravitational forces guide movement toward potential target locations, evaluated by fitness functions. Zhao et al. [22] proposed a distributed model predictive control (DMPC)-based method for multi-target search in unknown environments. The algorithm incorporates a hierarchical grid map and switching prediction mechanisms to plan effective paths under limited signal coverage and communication ranges. Notably, it includes an obstacle avoidance mechanism based on a virtual force model, allowing robots to detect and avoid both static obstacles and neighboring robots. Furthermore, an obstacle-following strategy is implemented to navigate around obstacles when avoidance is difficult, ensuring robust performance in cluttered environments.

B. Searching for Mobile Targets

The works [14]–[20], [23] focused on finding mobile targets. Durham et al. [14] studied the pursuit-evasion problem where multiple robots coordinate to detect evaders in an unknown environment. They proposed a frontier-based distributed algorithm that can guarantee that moving evaders are detected as long as there are a sufficient number of robots. Their proposed algorithm requires robots to maintain complete coverage of the frontier to avoid recontamination. Coogle et al. [15] studied the iceberg observation problem, and defined it based on an existing robotic observation problem known as the cooperative multi-robot observation of multiple moving targets (CMOMMT). In this model, robots are generally fixed in limited positions and are reallocated only occasionally. They provided a GMM-based probabilistic model for the target sources to guide robots in reallocation if necessary. Tang et al. [16] assumed that the target is always signaling, and the robots sense the distance to the target through the signal strength of the target. They proposed an implicit-communication-based algorithm for target search, i.e., information transfer by releasing/reading pheromones. The path planning algorithms of [17]–[19] are based on the target motion model. Asfora et al. [17] assumed that the search environment is given in advance. They proposed three mixed integer linear programming models to path planning. Mou et al. [18] applied a particle-based method to estimate the position of the target. A particle represents a possible position of the target. They proposed an entropy-based method to generate paths for robots. Acevedo et al. [19] also used a particle-based method to predict the target's position. They divided the search map into multiple cells, and the probability of the target being in a cell is determined by counting the number of particles in that cell. Then the coverage search algorithm is proposed to generate the search path. Table I summarizes the differences between the proposed work and current literature on finding targets. Saadaoui et al. [20] introduced the Local PSO (LoPSO) algorithm, which leverages the concept of local sub-swarms of robots to enhance the efficiency of cooperative multirobot search tasks. The algorithm dynamically forms sub-swarms based on the target detection status. This approach mitigates communication and sensing limitations by enabling robots within the same sub-swarm to exchange information more effectively. Compared to LoPSO [20], which forms local sub-swarms to enhance coordination and reduce communication

overhead, Zhao et al. [22] explicitly address obstacle avoidance by integrating a virtual force model and an obstacle-following mechanism. While LoPSO emphasizes rapid convergence and efficient division of labor in open environments, [22] demonstrates stronger adaptability in complex environments with unknown static obstacles. Both methods support inter-robot communication and coordination, but differ significantly in their assumptions and handling of environmental complexity. Guo et al. [23] proposed DRL-Searcher, an algorithm based on distributional reinforcement learning, aimed at solving the multirobot efficient search problem for a moving target. DRL-Searcher estimates the probability distribution of target capture time and adjusts the search strategy according to different objectives, such as minimizing capture time or maximizing detection probability, to enhance the efficiency of multirobot search. Jia et al. [26] proposed an asynchronous Bayesian updating strategy for multi-AUV dynamic target search under unstable underwater communication. By leveraging opportunistic optical links for high-capacity data exchange, AUVs asynchronously update local information maps and optimize paths via predictive control. Li et al. [27] introduced a MARL-based persistent coverage framework for random targets without prior knowledge. Vehicles estimate a global "knowability map" via distributed max-consensus and adaptive area partitioning, enabling scalable cooperation with limited observations.

Recent studies have leveraged Voronoi-based methods for multi-robot coordination in dynamic environments. Hu et al. [29] proposed a Voronoi partition approach combined with deep reinforcement learning (DRL) for autonomous exploration, enabling robots to handle sudden obstacles through policy learning from human demonstrations. Similarly, Wang et al. [30] integrated Voronoi graphs with channel-spatial attention mechanisms (CSAM) in convolutional neural networks to optimize UAV coverage paths. While these DRL-based methods excel in adaptability to complex environments, they require substantial training data and computational resources, posing challenges for real-time deployment in resource-constrained scenarios. In contrast, our Voronoi-based motion coordination algorithm relies on predefined mathematical models and implicit communication (via gray-scale maps), eliminating the need for training or prior knowledge of target dynamics. This offers higher computational efficiency and predictability in those applications where training data may not be available in advance.

Although many existing works have addressed multi-robot search tasks, they often suffer from several limitations. For instance, some approaches focus on static or known targets [8]–[13], [18], [22], [25], [28], which limits their applicability in real-world scenarios involving dynamic targets. Others require prior knowledge of the targets' motion models [15], [17]–[19], [27], which is often unavailable. Certain methods impose hardware constraints (e.g., pheromone release capability [16]) or minimum robot threshold [14]. Moreover, DRL-based techniques [29], [30] demand substantial computational resources for training, challenging real-time operation. Critically, many algorithms do not consider inter-robot communication or coordination [18], [23], leading to inefficient search due to redundant exploration. In environments with unknown obstacles, their adaptability and robustness are also restricted [19]–[21], [24]–[27].

To overcome the aforementioned limitations, we propose a search method that integrates gray-scale maps, Voronoi masks, and inter-robot motion coordination. Specifically:

- A gray-scale map that allows robots to revisit and search previously explored areas, which is critical for locating moving targets.
- A Voronoi mask that partitions the search space among robots, reducing redundant searches and improving overall search efficiency.
- Robot motion coordination is achieved through inter-robot communication, which enables information sharing about explored areas. This shared knowledge allows the robots to better plan their search paths, avoid redundant exploration, and improve overall coverage efficiency.

Compared with previous approaches, the proposed method features:

- More generalized algorithm (e.g., addressing multiple moving targets).
- Applicable to harsh environments (e.g., the search area with unknown obstacles).
- Less demanding on resources (e.g., do not require information of motion model of targets, and no special requirements requested by existing approaches).
- Better search efficiency (e.g., simulation results show our proposed algorithm outperforms the state-of-the-art by 125% in terms of success rate (100% vs. 44.4%)).

III. System Model

We assume that the search area is a rectangle with arbitrarily shaped obstacles. This assumption is valid in practice because we can use the smallest rectangle to cover an irregular search area. All obstacles are assumed to be unpredictable in our problem. That is, robots have no prior knowledge of presence of obstacles and their information such as shapes and locations. The size of the rectangle is assumed to be $a \times b$. The main notations used in this section are summarized in Table II.

There are T targets distributed in the search area and their positions are unknown. A target may move or be static at any time. There are N robots, denoted by r_1, r_2, ..., r_N, in the target search task. These robots could be UAVs and unmanned vehicles/ships. We assume that the robots have unique IDs, numbered from 1 to N. The robots are powered by batteries and have limited lifetime specified by l_i, $i = 1, 2, \cdots, N$. Each robot is equipped with a wireless transceiver with a communication range of R_c and is aware of its own position by a GPS device. Two robots are able to detect each other's presence and exchange information if they are within each other's communication range. Each robot is equipped with sensors with a sensing/searching range of R_s to explore and identify the targets. A target is said to be found by a robot as long as the target is within R_s of this robot. The robot is able to move, and its velocity is adjustable in the range of $[V_{min}, V_{max}]$. Note that $V_{min} = V_{max}$ if a robot moves with a

TABLE II
SUMMARY OF KEY PARAMETERS AND VARIABLES

N	number of robots
T	number of targets
L	path length
θ_i	path angle of robot r_i
R_i	set of neighboring robots of robot r_i
R_C	communication range
R_S	sensing range
l_i	lifetime of robot r_i
g_i	gray-scale map of robot r_i
V_i	speed of robot r_i
P_i	position of robot r_i
P_j^T	position of target j
$a \times b$	size of the rectangular search area (number of grids)
$g_i(j)$	gray scale value of grid j in robot r_i's map
$m_i(j)$	Voronoi mask for grid j by robot r_i
α	mask coefficient in $[0, 1]$ for Voronoi mask
$dist(r_i, j)$	Euclidean distance from r_i to center of grid j

constant velocity. Robots may fail due to various reasons such as energy depletion, hardware errors, or malicious attacks. In our model, we adopt a decentralized system structure, i.e., there is no central coordinator and each robot independently determines its own operation.

The problem of our concern is to design a motion coordination algorithm for robots to search for targets. The objectives are 1) to maximize the number of successfully-found targets before all robots deplete their energy; and 2) minimize the average search latency which is critical to search and rescue applications.

Our model aims to address a general scenario where no prior knowledge is available about the target distribution. Therefore, in our model, finding one target does not provide useful information about the likelihood of discovering other targets nearby. This modeling choice ensures broad applicability, even when targets are sparsely or irregularly distributed. In contrast, if multiple targets are known to be densely clustered within a search area, inter-robot information sharing can be beneficial to enable nearby robots to efficiently search for remaining targets in the area.

IV. MOTION COORDINATION ALGORITHM

Our basic idea is to divide the search area into grids and the size of each grid is designed to be fully covered by a robot with the sensing range. This ensures that any target within a grid can be detected when a robot visits it. Each grid is associated with a gray scale indicating the efficiency of searching targets in this grid. The darker the grid (i.e., the higher the gray scale), the higher the efficiency of searching targets in this grid. All the grids associated with gray scales form a "gray-scale map". Each robot keeps its own gray-scale map that is computed based on the knowledge of this robot.

Let $g_i(j) \in [0, 1]$ denote the gray scale of grid j on the gray-scale map of robot r_i, $i = 1, 2, \cdots, N$ and $j = 1, 2, \cdots, a \times b$. Fig. 2 shows an illustrative example of a gray-scale map of robot r_i. We have $g_i(A) < g_i(B) < g_i(C) < g_i(D)$. That is, D is the grid with the highest efficiency for target search while A is the grid with the lowest efficiency from the perspective of robot r_i.

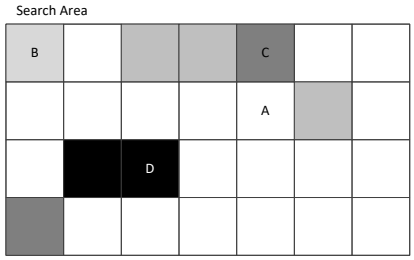


Fig. 2. An illustrative example of a gray-scale map.

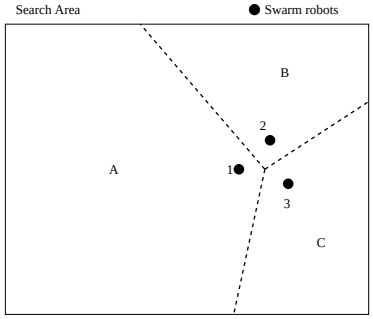


Fig. 3. Voronoi diagram of three neighboring robots for search coordination.

A. Gray-Scale Map and Voronoi Mask

In this section, we introduce the basic idea for computing the gray-scale map of robots. Notice that gray scale of a grid indicates the efficiency of searching targets in this grid. The search efficiency is dependent on two factors, namely, visit to the grid and coordination of neighboring robots.

Firstly, if a grid j has just been searched by robot r_i or other robots, it is not efficient for r_i to search targets in this grid again. Thus, its corresponding gray scale $g_i(j)$ should be low. That is, $g_i(j)$ depends on the elapsed time after the last visit of grid j. The longer the time elapsed from the last visit, the higher the gray scale (i.e., the darker) it should be. There are two ways to obtain information of the last visit of a grid. One is from its own record as the robot visited the grid and the other is from the information shared by neighboring robots. A grid can be visited by the same robot for several times during the search task. The time of the last visit of this grid is updated each time the robot visits this grid. Two robots can also exchange information of their gray-scale maps if they are within the communication range R_c of each other. Hence, each robot can update the time of the last visit of a grid either by its own visit or by the information shared by neighboring robots. The robot accordingly updates its gray-scale map and computes its search path which maximizes the search efficiency. However, notice that after exchanging information among neighboring robots, the robots may generate similar gray-scale maps and thus may compute similar search paths which consequently decrease the search efficiency (Robots are preferred to search different areas to maximize the search efficiency. The search efficiency is low if the robots compute similar search paths). Therefore, coordination of neighboring robots is necessary.

Secondly, the search efficiency also depends on coordination of robots. The robots are expected to search different portions of the search area to increase the search efficiency. In our algorithm, after neighboring robots exchange information, they

are coordinated to partition the search area and give their search preferences to different partitions. Our basic idea is to adopt the concept of the Voronoi diagram. An illustrative example is given in Fig. 3, where three neighboring robots, numbered from 1-3, meet and exchange information. The three robots partition the search area into Voronoi cells A, B and C, where their locations serve as the seed points. Robots 1, 2 and 3 are expected to take care of Voronoi cells A, B and C, respectively. However, it may not be desirable to limit a robot to search in its Voronoi cell only. For example, if all the grids in Voronoi cell B are with light gray scales (i.e., low efficiencies of searching targets), robot 2 is preferred to search Voronoi cells A or C, together with robots 1 or 3, respectively. To address this problem, we define a *Voronoi mask* that is an additional gray scale applied to all the grids in the corresponding Voronoi cell, such that the robot is motivated, but not limited, to search the Voronoi cell. In the above case, robot 2 will add a Voronoi mask to all the grids in Voronoi cell B. The Voronoi mask of grid j in robot r_i's gray-scale map can be calculated as follows

$$m_i(j) = \begin{cases} \alpha, & dist(r_i, j) < dist(r_k, j), \forall k \in N_i \\ 0, & \text{otherwise} \end{cases}, \quad (1)$$

where α is the mask coefficient in $[0, 1]$, $dist(r_i, j)$ is a function that returns the Euclidean distance between the position of robot r_i and the center of grid j, and N_i is the set of neighboring robots of robot r_i. If robot r_i does not meet any other robots, there is no Voronoi mask and $m_i(j)$ is zero.

Finally, the efficiency of searching targets in grid j by robot r_i is determined by the time of the latest visit and the Voronoi mask of grid j imposed by the robot. Hence, the gray scale of grid j in robot r_i's gray-scale map is calculated as follows

$$g_i(j) = \min\left\{1, 1 - \frac{t_i(j)}{t_{curr}} + m_i(j)\right\}, \quad (2)$$

where t_{curr} is the current time and $t_i(j)$ the time of the latest visit to grid j known by robot r_i. Because $t_i(j) \leq t_{curr}$ and $0 \leq m_i(j) \leq 1$, we have $0 \leq g_i(j) \leq 1$. Notice that the gray-scale maps are updated in a distributed manner and two robots can share information only if they meet and are within the communication range R_c of each other. It may happen that a certain grid has just been explored by robot r_i, but this grid is marked as unexplored (i.e., the gray scale is one) in robot r_j, $i \neq j$.

B. Information Exchange

In practice, the robots are moving and a contact period of two robots is normally short. It may not be feasible to exchange the whole gray-scale map during the short contact period because the search area is typically wide (e.g., the search area of Malaysia Airlines Flight MH370). We assume that a robot has a limited bandwidth and each robot can exchange information of up to K, $K < a \times b$, grids with its neighboring robots. The K grids are the most recently visited ones, either by the robot itself or by other robots, because more recent information is more accurate and thus more valuable. A sample exchanged information is illustrated in Table III.

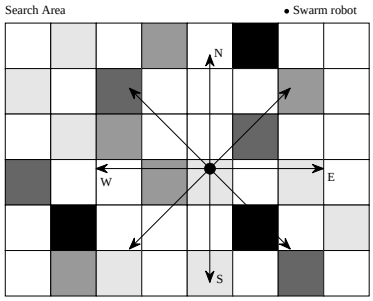


Fig. 4. Eight candidates of search paths of a robot.

TABLE III
EXCHANGE INFORMATION BETWEEN NEIGHBORING ROBOTS

Grid Coordinates	(5, 268)	(547, 217)	(3869, 114)	···
Time of last visit	19:23:43	18:04:30	12:00:24	···

In Table III, information of each grid includes two attributes. One is the grid coordinates and the other is the time of the latest visit of this grid. For example, (5, 268) denotes the grid that is located in the 5th row and the 268th column of the search area, and 19:23:43 is the time of the most recent visit to the grid with coordinates (5, 268). Notice that there could be other forms of coordinates and time (e.g., including dates) which depend on the applications.

Due to the limited bandwidth, a robot cannot send information of the entire gray-scale map to its neighboring robots in the message exchange process. Instead, once two robots meet (i.e., within the communication range of each other), they exchange information of up to K grids, where K is a parameter dependent on the available bandwidth and contact duration of the robots. Upon receiving the exchange information, a robot updates its gray-scale map by examining the time of the last visit of those grids in the table. If the received time is more recent than that of its gray-scale map, r_i updates its g_i using equation (2) to obtain a lower gray scale (i.e., a lighter color).

C. Search Path Planning

With the gray-scale map, a robot can compute a search path that maximizes search efficiency. Notice that the darker a grid is (i.e., the longer time that this grid has not been visited), the higher the efficiency of searching targets in this grid. Intuitively, targets are more likely located in darker grids. Therefore, a search path should traverse darker grids to maximize search efficiency. We assume that the length of the search path is L. We consider Q candidates of search paths that are originated from the current position of the robot. The Q candidate paths equally divide the plane (i.e., the angle between two neighboring paths is the same).

Fig. 4 shows an example of eight candidate paths (i.e., $Q = 8$). It is clear that the candidate path heading to the northeast direction is better than that heading to the south direction, because the former passes through more dark and gray grids. In order to quantitatively evaluate search efficiency of candidate paths, we define the search efficiency of a

Algorithm 1: *Path_planning* Algorithm

Input: robot r_i's current position, robot r_i's gray-scale map, path length L;
Output: the optimal search path P_{opt};
// Suppose there are Q candidate paths that are originated from the current position of r_i.
Generate Q candidate paths such that the angle between any two adjacent paths is equal to $2\pi/Q$;
Let $e_{max} = 0$;
for $q = 1$ *to* Q **do**
  $e_q = eff(P_q)$; // compute the search efficiency of P_q based on (3).
  if $e_q > e_{max}$ **then**
    $e_{max} = e_q$;
    $P_{opt} = P_q$;
  end
end

candidate path $P_q, q = 1, 2, ..., Q$, computed by robot r_i as follows

$$eff(P_q) = \frac{\sum_{j=1}^{ab} O(P_q, j) \times g_i(j)}{\text{valid length of path } P_q}, \tag{3}$$

where $O(P_q, j)$ represents the length of the overlapping part between candidate path P_q and grid j. Intuitively, the search efficiency of a candidate path is higher if it has more overlap with darker grids. The valid length of path P_q is the length of the path segment within the search area.

Algorithm 1 gives details of the proposed *Path_planning* algorithm. The input of the algorithm includes robot r_i's current position, robot r_i's gray-scale map, and path length L. For each candidate path, we calculate the search efficiency of the path based on equation (3). The algorithm outputs the path with the highest search efficiency. When robot r_i is close to boundary of the search area, some candidate paths may exceed the boundary. In this case, the intersection point of the path and the boundary is treated as the destination, and only the valid length of the path within the search area is calculated in equation (3).

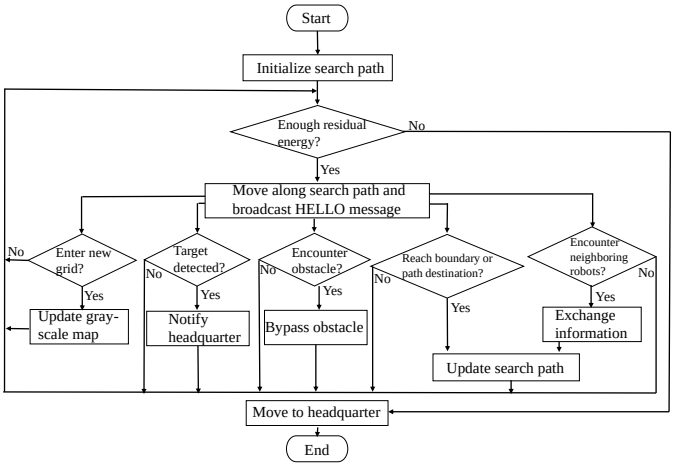


Fig. 5. Decision flowchart of the robot's motion coordination process.

Algorithm 2: *Motion_coordination* Algorithm

Input: robot r_i's current position, path length L, mask coefficient α;
Initialize a search path P_i of L in length;
while *robot* r_i *has enough residual energy* **do**
  // Each robot should reserve enough energy allowing the robot to return to the headquarter.
  Move along P_i while broadcasting a HELLO message periodically;
  if *robot* r_i *moves to a new grid* j **then**
    Update gray-scale map $g_i(j)$ based on (2);
  end
  if *robot* r_i *encounters neighboring robots* **then**
    Exchange information with neighboring robots;
    Compute the Voronoi mask of grids based on (1);
    Update gray-scale map $g_i(j)$ based on (2);
    Call *Path_planning* algorithm to update P_i;
  end
  if *robot* r_i *reaches the boundary of the search area or the destination of path* P_i **then**
    Call *Path_planning* algorithm to update P_i;
  end
  if *robot* r_i *encounters an obstacle* **then**
    Call *Obstacle_avoidance* algorithm [31] to bypass the obstacle;
  end
  if *robot* r_i *identifies a target* **then**
    Notify the headquarter;
  end
end
Move to the headquarter for battery charging or replacement;

D. Motion Coordination

The motion coordination algorithm for robot r_i is formally presented in Algorithm 2 and is sketched as follows. A decision flowchart illustrating the overall process is shown in Fig. 5. The algorithm is executed at each robot in a distributed manner. Each robot maintains a gray-scale map and initializes a search path (e.g., a randomly selected path of length L) when the algorithm starts. The robot continuously moves along the search path while checking the following conditions.

- If the residual energy is only enough for the robot to return to the headquarter, the robot stops searching and moves to the headquarter for battery charging or replacement.
- If the robot moves to a new grid, it updates its gray-scale map according to equation (2).
- If the robot encounters neighboring robots, it exchanges information with the neighboring robots and computes the Voronoi mask of the grids according to equation (1). Then, the robot updates its gray-scale map according to equation (2) and calls the *Path_planning* algorithm to update the search path. The robot moves along this new search path.
- If the robot reaches the boundary of the search area or the destination of the search path, it calls the *Path_planning* algorithm to update the search path. The robot moves along this new search path.
- If the robot encounters any obstacles during the movement, it calls the *Obstacle_avoidance* algorithm [31]

(using either the right-hand rule or the left-hand rule) to bypass the obstacle. In this method, the robot keeps its virtual right-hand/left-hand in constant contact with the boundary of the obstacle and moves until it either returns to its planned path or reaches the extension of the path beyond the obstacle (in cases where the goal lies within the obstacle).
- If the robot identifies a target, the robot notifies the headquarter (e.g., by moving back or via 5G/satellite communications) and continues the search.

The above process is repeated until the robot runs out of its energy.

V. Theoretical Analysis

In this section, we analyze theoretical performance of the proposed algorithm. To simplify analysis, we set the mask coefficient α to zero and the side length of the grid to L, where L is the length of the search path. That is, impact of robot coordination is removed and each robot only determines its search path based on the time elapsed after the last visit of grids. Despite this simplification, the theoretical results that will be derived in this section remain meaningful because the derived theorems rigorously establish performance bounds for our proposed algorithm without coordination. The simulation results (presented in the next section) show that our algorithm performance improves when coordination between neighboring robots is enabled. Therefore, these theoretical results are still valid when robot when robot coordination is enabled. Because the length of the search path is equal to the side length of the grid, each robot can move to one of the eight neighboring grids in each move. We have the following lemma.

Lemma 1. *If a grid has been visited K times, all its neighboring grids have been visited at least $\lfloor K/8 \rfloor$ times.*

Proof. See the illustrative figure in Fig. 5. After each visit to grid A, one of A's eight neighboring grids should be visited. According to the *Motion_coordination* algorithm, the grid with the highest gray scale is selected for search. Thus, if there exist neighboring grids that have not been visited before (i.e., with the highest gray scale), one of these grids must be visited after each visit of grid A. That is, if grid A has been visited eight times, all its eight neighboring grids must have been visited at least once. By analogy, we can conclude that every eight visits on grid A result in at least one visit to all the eight neighbors. The lemma follows. □

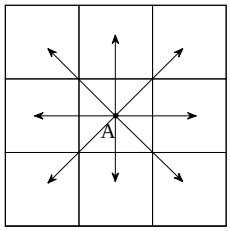


Fig. 6. Eight possible movement directions of the robot at grid A.

We assume that it takes one unit of time to move from a grid to a neighboring grid. We have the following theorem.

Theorem 1. *The proposed Motion_coordination Algorithm ensures that all static targets are guaranteed to be found. The total time it takes to visit all grids is no greater than $8^{\max(a,b)}$, where a and b are the numbers of grids on the length and width of the search area, respectively.*

Proof. Notice that impact of robot coordination is removed (i.e., α=0). If the theorem is true for one robot, the theorem still holds for multiple robots. We consider only one robot in the search area and assume that the robot starts with grid A. Notice that robots may visit a same grid multiple times during the search process. According to Lemma 1, it takes at most 8^i for the robot to visit all the grids that are i hops away from grid A. Notice that a and b are the numbers of grids on the length and width of the search area, respectively. It takes at most $8^{\max(a,b)}$ until all the grids in the search area have been visited. Because targets are static, the proposed *Motion_coordination* Algorithm ensures that all static targets are guaranteed to be found. □

Theorem 1 gives a loose upper-bound on the total time it takes to find all static targets. We derive a tighter upper-bound in the following.

To facilitate the analysis, we assume that there is only one robot in the search area. We number the grids according to the order in which they are first visited by the robot, where grid K represents the K-th grid visited by the robot for the first time. We suppose that the robot visits grid A first and then visits grid K. That is, we have $K > A$. We define the following variables.

- t_K: The time when the robot first visited grid K.
- $t^i_{A,K}$: The time when the robot visited grid A for the i-th time during time interval $[t_K, t_{K+1}]$. Notice that grid A is visited by the robot before grid K.
- $B_{i,K}$: The grid from which the robot entered grid A during its i-th visit to grid A. Obviously, $B_{i,K}$ is a neighboring grid of A.
- $T_{[t_1,t_K]}$: A set of grids that the robot visited during time interval $[t_1, t_K]$.

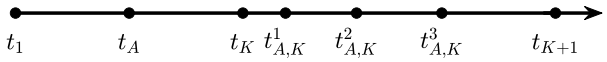


Fig. 7. Temporal relationships between variables on the timeline.

Fig. 7 illustrates an example of the temporal relationships between variables on the timeline.

Lemma 2. *If the robot visits a grid, say A, m times during time interval $[t_K, t_{K+1}]$, then the robot must have visited at least one grid belonging to $T_{[t_1,t_K]} - T_{[t_K,t^i_{A,K}]}$ during time interval $[t^i_{A,K}, t^{i+1}_{A,K}]$ where $1 \le i < m$.*

Proof. We prove this by contradiction: assume that all grids visited by the robot belong to $T_{[t_K,t^i_{A,K}]}$ during the time interval $[t^i_{A,K}, t^{i+1}_{A,K}]$. For simplicity, unless otherwise specified, the time intervals discussed in the following analysis will refer to $[t^i_{A,K}, t^{i+1}_{A,K}]$.

First, we examine the conditions under which the robot would revisit grid A. According to the *Motion_coordination*

algorithm, at time $t_{B_{i+1,K}}$, all neighbors of grid $B_{i+1,K}$ must have been visited by the robot (note that the robot has already visited grid A at time $t_{B_{i,K}}$). If any neighbor of grid $B_{i+1,K}$ has not been visited by time $t_{B_{i+1,K}}$, it must have a higher gray scale than grid A. In this case, the robot would move from grid $B_{i+1,K}$ to the neighbor with the highest gray scale, rather than returning to grid A. This contradicts the assumption that the robot moves from grid $B_{i+1,K}$ back to grid A.

Therefore, for *any grid* A that has been visited after time t_K, if the robot visits this grid again, it must have a neighbor such that all of its neighbors, except grid A, have been visited during the robot's two visits to grid A. Thus, if all grids visited by the robot belong to $T_{[t_K,t^i_{A,K}]}$, then all neighbors of any grid in $T_{[t_K,t^i_{A,K}]}$ must also belong to $T_{[t_K,t^i_{A,K}]}$.

To illustrate this conclusion, consider the following example. Suppose the robot moves from grid A to its neighbor, grid C. Then, all of grid A's neighbors must have been visited during the robot's two visits to grid C. This process continues until the robot returns to grid A. In this example, the robot's visits to grid A and its neighbors lead to a visit to grid C, and the robot's visits to grid C and its neighbors lead to a visit to grid A. Intuitively, the robot must visit grid C before visiting its neighbor, grid A, and similarly, it must visit grid A before revisiting grid C.

Obviously, there are some grids in $T_{[t_K,t^i_{A,K}]}$ whose neighbors are not in $T_{[t_K,t^i_{A,K}]}$. Therefore, the robot must visit at least one grid belonging to $T_{[t_1,t_K]} - T_{[t_K,t^i_{A,K}]}$ during time interval $[t_K, t_{K+1}]$. □

Lemma 3. *We suppose that grid j is the last grid that the robot visits before grid $K+1$. We have grid $j \in [1, K]$ and grid j is visited by the robot exactly once during time interval $[t_K,\ t_{K+1}]$.*

Proof. According to the grid numbering rule, grid $K+1$ is the next grid of K that is visited by the robot for the first time. All the grids visited by the robot during time interval $[t_K,\ t_{K+1}]$ should have been visited before during time interval $[t_1,\ t_K]$. Thus, we have grid $j \in [1, K]$. Notice that grid $K+1$ is an unvisited neighbor of grid j before time t_{K+1}. That is, grid j has at least one unvisited neighbor. According to the *Motion_coordination* algorithm, if the robot visits grid j during time interval $[t_K,\ t_{K+1}]$, then it must visit one of its unvisited neighbors which is grid $K+1$. Therefore, grid j is visited by the robot exactly once during time interval $[t_K,\ t_{K+1}]$. □

Theorem 2. *The total time taken by the proposed Motion_coordination Algorithm to traverse all the grids is no greater than $\frac{2(ab)^3-3(ab)^2+ab}{6}$, where a and b are the numbers of grids on the length and width of the search area, respectively.*

Proof. We consider the special case where there is only one robot in the search area. Notice that impact of robot coordination is removed (i.e., $\alpha = 0$). If the theorem is true for one robot, the theorem still holds for multiple robots.

We first analyze the upper bound of time interval $[t_K, t_{K+1}]$. According to Lemma 2, during this interval, between two visits to the same grid, at least one grid is visited for the first time after t_K. Furthermore, since the robot has visited at most K different grids during time interval $[t_K, t_{K+1})$, if the time interval between two visits to the same grid exceeds K, then there must exist at least one other grid that the robot visits at least twice in between. Therefore, starting from any time point during time interval $[t_K, t_{K+1}]$, until the robot visits a grid for the first time after t_K, the total number of grids visited by the robot will not exceed K.

According to Lemma 3, grid $K+1$ must be a neighbor of some grid i, where $i \in \{1, 2, \ldots, K\}$. Thus, we can conclude that the upper bound of time interval $[t_K, t_{K+1}]$ is K^2.

Therefore, the time required for the robot to visit all grids does not exceed:

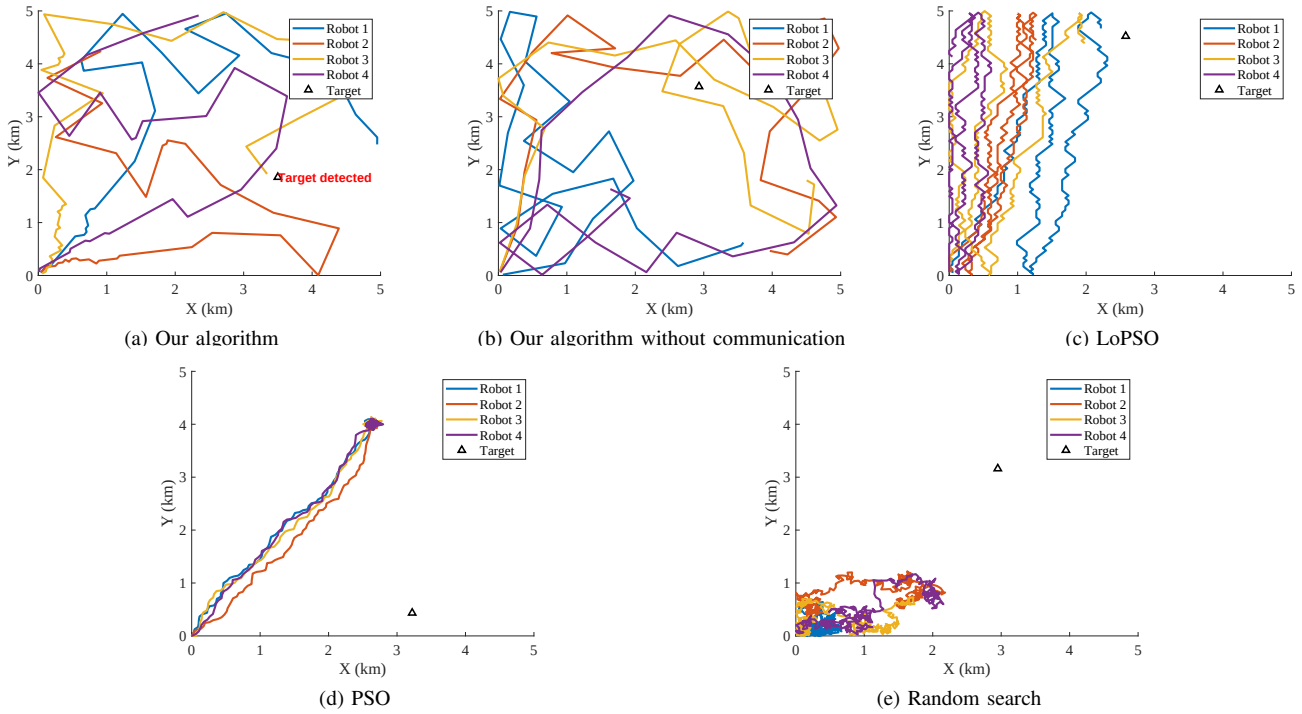
$$1^2 + 2^2 + \cdots + (ab-1)^2 = \frac{2(ab)^3 - 3(ab)^2 + ab}{6}. \quad (4)$$

□

Theorem 1 and Theorem 2 guarantee the theoretical performance of the proposed *Motion_coordination* Algorithm when the mask coefficient α is zero (i.e., there is no coordination of the robots) and the targets are static. The performance of the proposed algorithm in cases with $\alpha > 0$ (i.e., robot coordination is incorporated) and mobile targets will be validated through simulations in the next section.

Note that the theoretical guarantees provided in Theorems 1 and 2 are derived under several idealized assumptions, such as static targets, the absence of robot coordination (i.e., $\alpha = 0$), and uniform grid-based environments without obstacles. These assumptions were made to facilitate tractable analysis and derive provable bounds. However, in real-world applications, robot coordination, dynamic target behaviors, irregular terrain, and obstacle presence may affect the actual performance of the algorithm. Relaxing these assumptions to obtain theoretical guarantees under more realistic conditions remains a challenging open problem. Our current simulations (in Section VI) aim to bridge this gap by empirically validating the effectiveness of the algorithm in complex, dynamic environments with $\alpha > 0$ and moving targets.

We now estimate the running time of the *Motion_coordination* algorithm. This algorithm repeatedly invokes the *Path_planning* procedure (Algorithm 1) under several conditions, such as when a robot reaches the end of its current path, encounters nearby robots, or reaches the boundary of the search area. Let H denote the total number of such replanning events during execution. In the absence of interactions, the frequency of path replanning can be estimated based on a robot's energy budget l_i, speed V_i, and the path length L. Specifically, a robot would replan approximately $H = l_i V_i / L$ times. Each time *Path_planning* is called, it generates Q candidate paths of length L, with evenly spaced directions. For each candidate path P_q, a search efficiency score e_q is estimated using a Monte Carlo method. This process involves examining the grid cells that intersect with the path. Since the grid resolution is determined by the sensor range R_S, the number of intersecting grid cells is at least L/R_S. As each cell can be processed in constant time (i.e., for lookup, distance computation, and score accumulation),



(a) Our algorithm (b) Our algorithm without communication (c) LoPSO

(d) PSO (e) Random search

Fig. 8. Trajectories of different algorithms in a search area of 5km×5km without obstacles. There are four robots and one static target which is randomly deployed. Search duration is 50 minutes. The probability of robot failure is 0.2. The probability of successful target detection is 0.8. The probability of successful communication is 0.5. Our algorithm successfully finds the target.

TABLE IV
SIMULATION PARAMETER SETTINGS

Range of search area	10km × 10km
Number of robots	10
Number of targets	5
Robot speed	400m/min
Target's maximum speed	25m/min
Detection range	200m
Communication range	500m
Robot lifetime	120min
Number of candidate paths	8
Length of a time slot	10s
Mask coefficient	0.1
Broadcast packet size	1000bytes
Probability of robot failure	0.2
Probability of successful target detection	0.8
Probability of successful communication	0.5

the computation for each candidate path is $\mathcal{O}(L/R_S)$. Given that there are Q candidate paths per replanning event and H such events over the execution, the total time spent on path planning is approximately $\mathcal{O}(H \cdot Q \cdot L/R_S)$. Other operations within *Motion_coordination*, such as information exchange, grayscale updates, and Voronoi mask computation, are either constant-time or linear with respect to the robot's local state and thus do not dominate the overall running time.

VI. SIMULATION

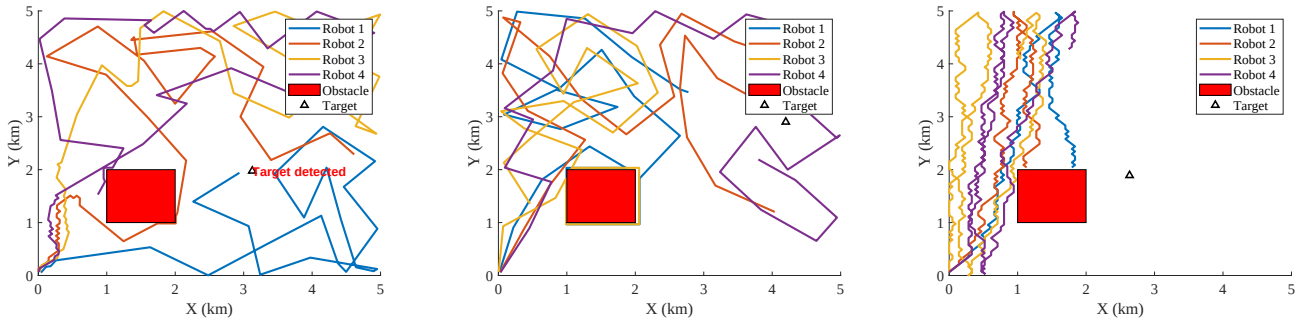
A. Evaluation Metrics and Parameter Settings

We conduct simulations on MATLAB 2018b in a Windows 10 desktop. We compare our proposed method with PSO [21], LoPSO [20], and random search. All three methods are population-based search algorithms, which makes them suitable for comparison under the same problem setting. PSO and LoPSO are recently proposed algorithms that have demonstrated effectiveness in previous studies, making them relevant choices for benchmarking. These algorithms share similar features with our proposed method, such as population-based search strategies, inter-robot communication within a limited communication range for information sharing, and the use of a detection range for target detection. These similarities ensure a fair and meaningful comparison. Random search is included as a baseline method to emphasize the advantages of swarm intelligence approaches over unguided exploration.

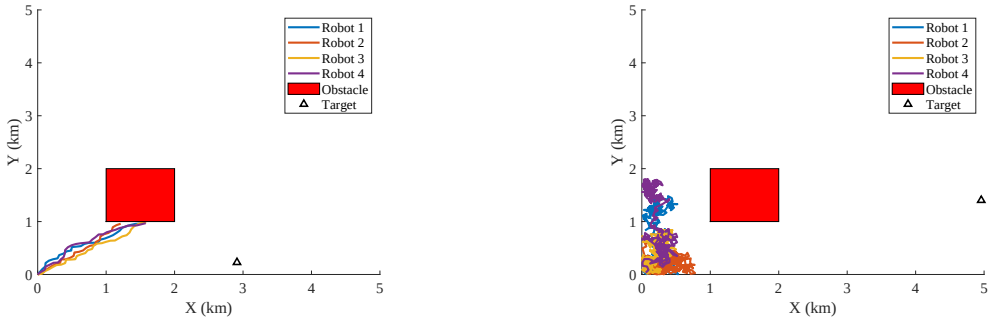
In summary, we compare five algorithms, i.e., our algorithm (i.e., *Motion_coordination* in Algorithm 2), our algorithm without communication (i.e., without the Voronoi mask and information exchange with neighboring robots), the PSO [21], the LoPSO [20] and the random search. The following criteria are adopted to evaluate performance of the algorithms.

- *Success rate:* The ratio of the number of targets successfully detected to the total number of targets during the search mission.
- *Average search time*: The mean time required from the start of the search mission to the end of the search mission. There are two cases when the search mission ends: one is that all targets are detected, and the other is that all robots run out of energy.
- *First success time:* The time required to find the first target from the start of the search mission (e.g., we hope to find the people to be rescued as soon as possible.).

The search area is assumed to be a 10km × 10km plane. We assume that all the robots start search from the bottom



(a) Our algorithm (b) Our algorithm without communication (c) LoPSO



(d) PSO (e) Random search

Fig. 9. Trajectories of different algorithms in a search area of 5km×5km with an obstacle. There are four robots and one static target which is randomly deployed. Search duration is 50 minutes. The probability of robot failure is 0.2. The probability of successful target detection is 0.8. The probability of successful communication is 0.5. Our algorithm successfully finds the target.

left position. For each robot, we set the communication range to be 500m, as specified in [32], and the detection/sensing range to be 200m based on [31]. The robot lifetime is set to 120min [33]. The robot speed is set to 400m/min [34]. The probability of robot failure is set to 0.2, as specified in [35]. The probability of successful detection when a target is within the sensing range is set to 0.8, following [24]. The probability of successful communication between neighboring robots is set to 0.5, based on [36]. In our simulation framework, time is discretized into time slots, and each time slot represents 10 seconds of simulated time. All parameters are listed in Table IV.

Terrain variations can impact algorithm performance. Varying terrain includes different topographic features and variations in landforms. Such topographic variations may include mountains, hills, plains, deserts, swamps and many other terrain types, each with its own unique characteristics and challenges.

- For robotic vehicles (e.g., self-driving cars), varying terrain might greatly affect their moving speeds, communication/sensing ranges, and other operational parameters. Given the complexity of varying terrain for robot operation, robot testbeds are preferred over simulation experiments to test the algorithms, which will be our future work.
- For flying robots (e.g., UAVs and drones), varying terrain has limited impact because these flying robots operate in the air. Our current simulation is mainly for this case.

Notice that the robots will move to the headquarter for battery charging or replacement after they deplete their energy reserves in our algorithm. In the simulation, the robots that run out of batteries are removed to eliminate the impact of energy reserves (i.e., no battery charging or battery replacement), so that we can accurately evaluate performance of the five algorithms.

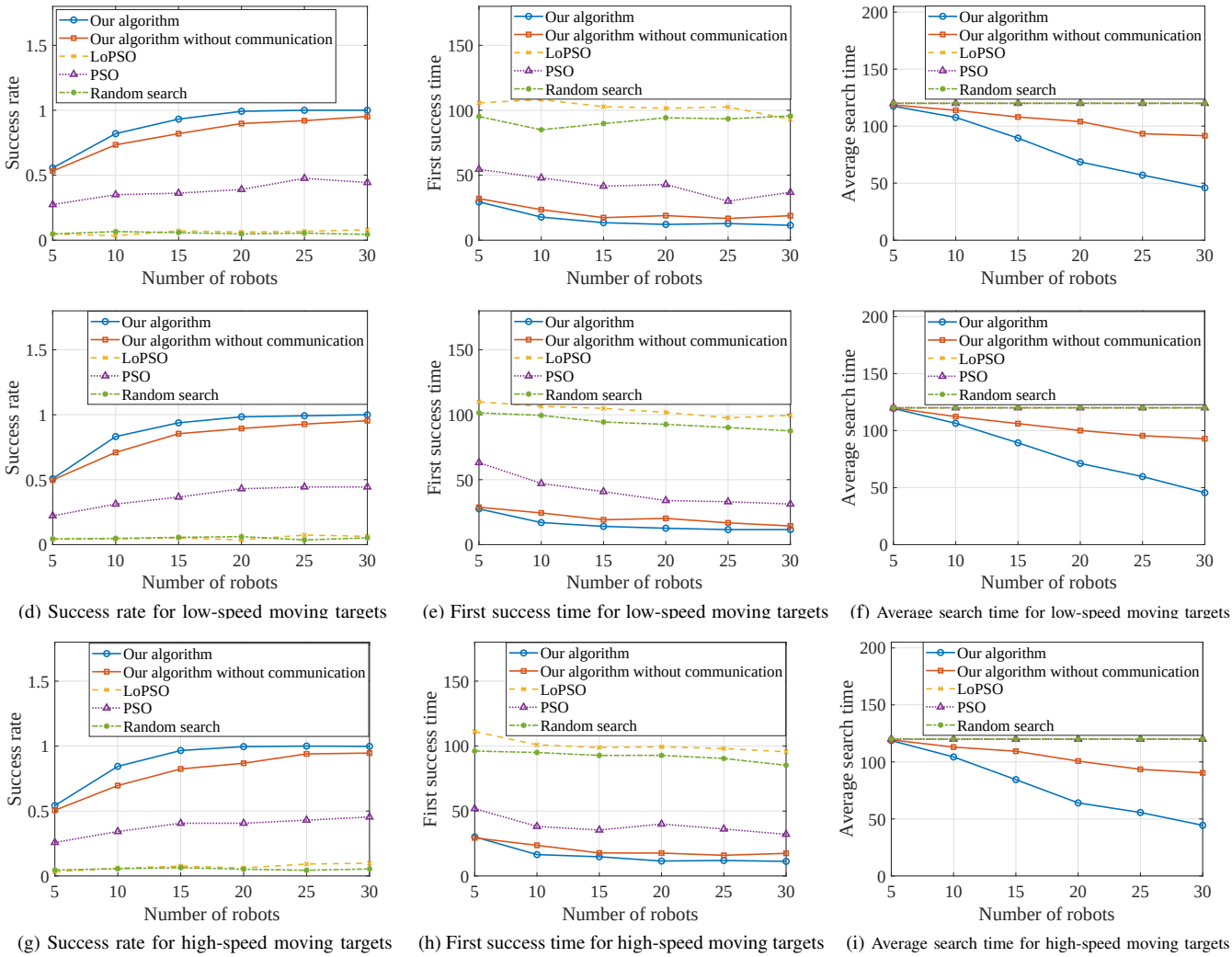
To evaluate algorithm performance under different conditions, we consider the following three scenarios:

- *Scenario 1:* Static targets, such as stationary objects or injured people in need of rescue.
- *Scenario 2*: Low-speed moving targets, such as people drifting with water currents in a water environment. Referring to the speed of water current, we set the maximum speed of the targets to 25m/min.
- *Scenario 3:* High-speed moving targets, such as intruders in battlefields. Referring to typical human running speed, the maximum target speed is set to 200m/min.

B. Comparison of Search Trajectories in the Search Area Without Obstacles

Fig. 8 illustrates typical search trajectories of robots in the five algorithms when there is no obstacle in the search area. To effectively display the search trajectories, we place 4 robots in a search area of 5km×5km and set a search duration to 50 minutes.

The figure demonstrates that the overall coverage of the robots in different algorithms are ranked as follows: our algorithm $>$ our algorithm without communication $>$ the LoPSO $>$ the PSO $>$ the random search. We observe a similar ranking of the five algorithms' performance in the later experiments such as the success rate and the first success time.



(d) Success rate for low-speed moving targets (e) First success time for low-speed moving targets (f) Average search time for low-speed moving targets

(g) Success rate for high-speed moving targets (h) First success time for high-speed moving targets (i) Average search time for high-speed moving targets

Fig. 10. Varying number of robots under static, low-speed moving, and high-speed moving target conditions, in a search area of 10km×10km without obstacles. There are five targets which are randomly deployed. Search duration is 120 minutes. The probability of robot failure is 0.2. The probability of successful target detection is 0.8. The probability of successful communication is 0.5.

Search trajectories of our algorithm is shown in Fig. 8(a) where the four robots almost visit the entire search area. Comparing Fig. 8(a) and Fig. 8(b), we can see that communication between the robots is effective to enhance the search efficiency. For instance, Robot 1 and Robot 3 searched the upper left part of the search area, while Robot 2 and Robot 4 searched the lower right part in Fig. 8(a). The search trajectories of the four robots are sufficiently separated to enhance the search efficiency in our algorithm. In contrast, the search trajectories of the four robots mixed together without communication of the robots. Fig. 8(b) demonstrates there are still some unsearched areas in the middle and the bottom right corner of the search area.

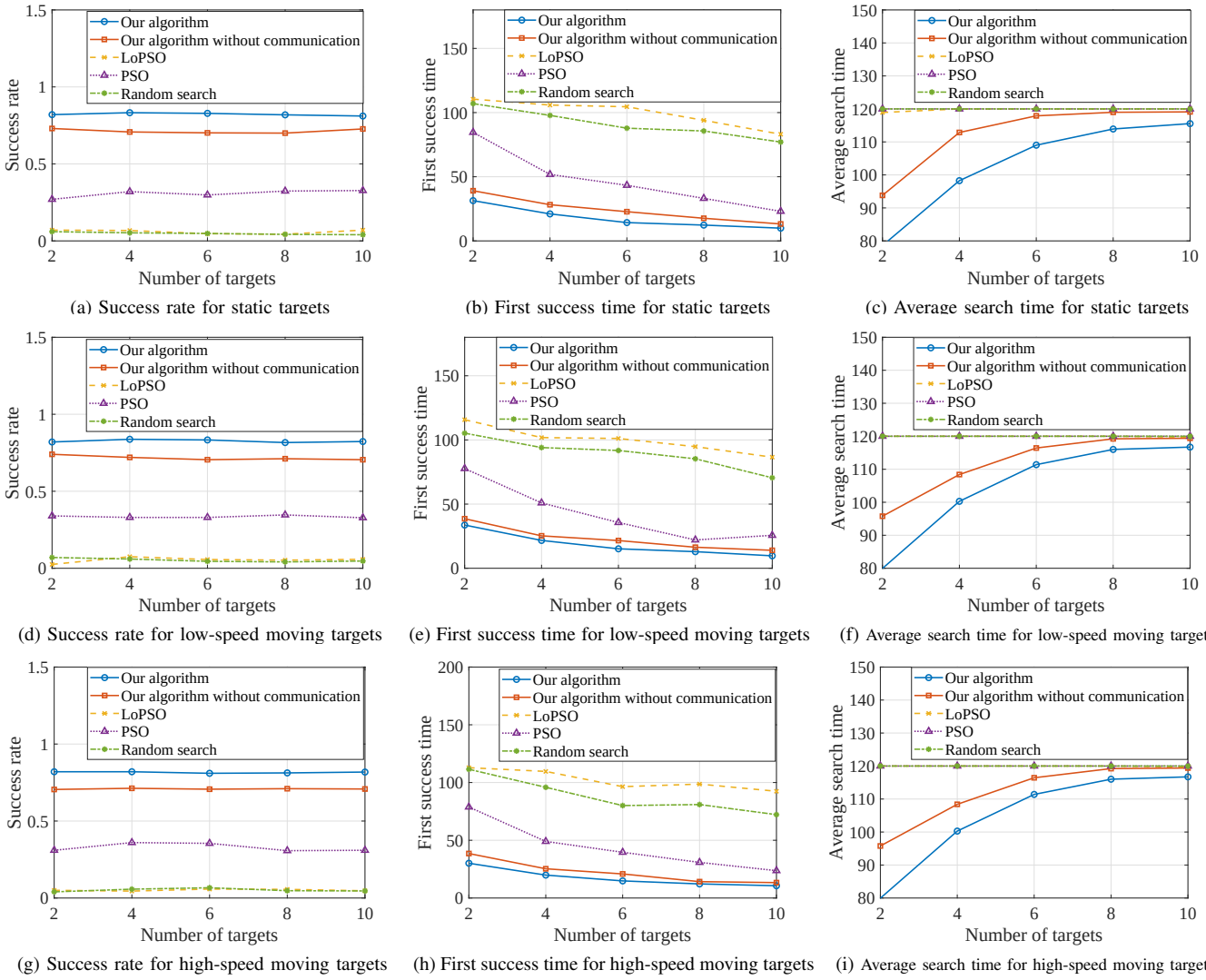
Fig. 8(c) shows that search trajectories of the robots in the LoPSO roughly cover half of the search area. This is because the LoPSO relies on signals emitted by the targets to estimate their approximate locations. When the robots are unable to capture the targets' signals, the algorithm cannot efficiently plan the robots' search paths. Fig. 8(d) and Fig. 8(e) demonstrate that the robots using the PSO and the random search only explored a small portion of the search area.

C. Comparison of Search Trajectories in the Search Area With Obstacles

Fig. 9 illustrates typical search trajectories of robots in the five algorithms when there is an obstacle in the search area. The obstacle is a square with a side length of 1 km, centered at the coordinates (1.5km, 1.5km). Similar to that in Section 5.2, we place 4 robots in a search area of 5km×5km and set a search duration to 50 minutes.

Fig. 9(a) demonstrates that our algorithm well tackles the obstacle problem and the search trajectories of the robots are evenly separated in the search area. Our algorithm without communication also enables the robots to cover most of the search area in Fig. 9(b). Fig. 9(c)-(d) illustrate that the robots using the LoPSO, the PSO and the random search are unable to bypass the obstacle because they were not designed for this purpose.

For the sake of fairness, we remove obstacles in all the other experiments in this work.



(a) Success rate for static targets (b) First success time for static targets (c) Average search time for static targets

(d) Success rate for low-speed moving targets (e) First success time for low-speed moving targets (f) Average search time for low-speed moving targets

(g) Success rate for high-speed moving targets (h) First success time for high-speed moving targets (i) Average search time for high-speed moving targets

Fig. 11. Varying number of targets under static, low-speed moving, and high-speed moving target conditions, in a search area of 10km×10km without obstacles. There are ten robots. Targets are randomly deployed. Search duration is 120 minutes. The probability of robot failure is 0.2. The probability of successful target detection is 0.8. The probability of successful communication is 0.5.

D. Impact of Number of Robots

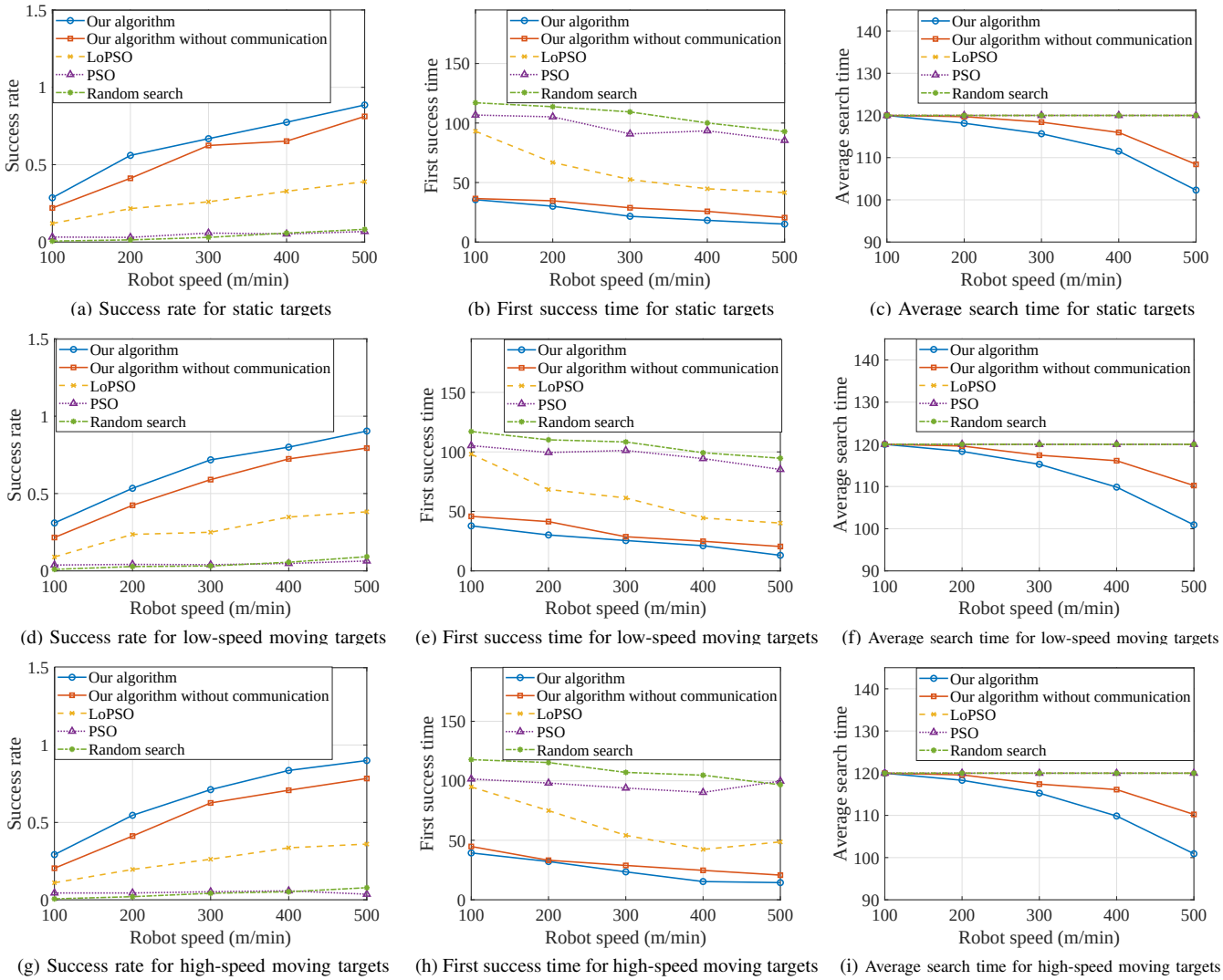
Fig. 10 illustrates the impact of varying the number of robots from 5 to 30 on the success rate, the first success time, and the average search time across three scenarios: static targets, low-speed moving targets, and high-speed moving targets. Fig. 10(a), (d) and (g) show that the success rates of the five algorithms improve as the number of robots increases. Notably, both versions of our algorithm achieve almost 100% success rates (i.e., all targets are found) when the number of robots is greater than 20. Performance of the LoPSO is ranked in the middle of the five algorithms. The PSO and the random search give a low success rate at around 7% .

Fig. 10(b) (e) and (h) reveal that the first success time decreases for all the algorithms as the number of robots increases. Both versions of our algorithm significantly outperform the others. The results also indicate very close performance of our algorithm and our algorithm without communication. This is because most of the search area is unexplored at the beginning of the search when communication of the robots offers limited benefit.

Fig. 10(c) (f) and (i) show that our algorithm consistently outperforms the others in terms of average search time. As the number of robots increases, the average search time for both versions of our algorithm decreases, while the LoPSO, the PSO and the random search's average search time stabilizes at 120 minutes. This is because these three algorithms are unable to find all the targets and all robots eventually deplete their energy.

E. Impact of Number of Targets

Fig. 11 illustrates the impact of varying the number of targets from 2 to 10 on the success rate, the first success time, and the average search time across three scenarios: static targets, low-speed moving targets, and high-speed moving targets. Since the experimental trends across the three scenarios are similar, we take the static target case (i.e., subfigures (a)-(c)) as a representative example for discussion. Fig. 11(a) shows that the success rates of the five algorithms remain stable as



(a) Success rate for static targets (b) First success time for static targets (c) Average search time for static targets

(d) Success rate for low-speed moving targets (e) First success time for low-speed moving targets (f) Average search time for low-speed moving targets

(g) Success rate for high-speed moving targets (h) First success time for high-speed moving targets (i) Average search time for high-speed moving targets

Fig. 12. Varying speed of robots under static, low-speed moving, and high-speed moving target conditions, in a search area of 10km×10km without obstacles. There are ten robots and five targets which are randomly deployed. Search duration is 120 minutes. The probability of robot failure is 0.2. The probability of successful target detection is 0.8. The probability of successful communication is 0.5.

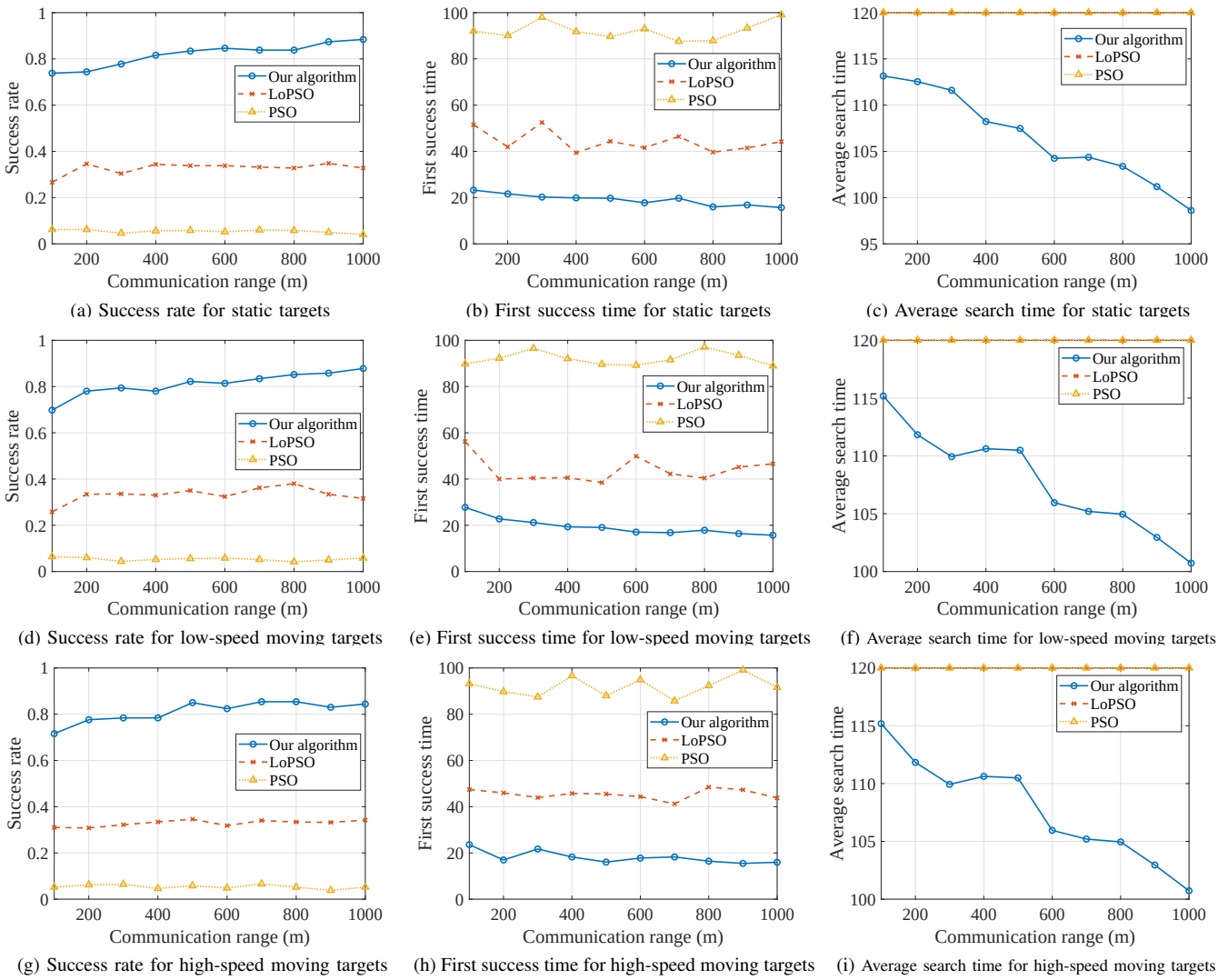
the number of targets increases. Specifically, the success rate is kept at 0.82 for our algorithm, 0.70 for our algorithm without communication, 0.30 for the LoPSO, and 0.06 for both the PSO and the random search. When the number of targets increases, it becomes easier for robots to encounter targets, but meanwhile it becomes harder to find all the targets. Thus, the success rate remains almost unchanged with these two factors canceling each other out.

Fig. 11(b) shows that the first success time decreases as the number of targets increases. Both versions of our algorithm achieve significantly shorter first success times than the others. For example, with two targets, our algorithm achieves a first success time of 31 minutes, which decreases to 10 minutes with ten targets. The LoPSO (ranked third in the figure), however, starts at 85 minutes and reduces to 23 minutes as the number of targets increases to ten.

Fig. 11(c) demonstrates that our algorithm has the best performance among all the five algorithms. The average search time of our algorithm increases from 78.6 minutes with two targets to 115.5 minutes with ten targets. In contrast, the LoPSO, the PSO, and the random search all reach the maximum search time limit of 120 minutes because the search mission ends with energy depletion of the robots before all the targets are found.

F. Impact of Robot Speed

In Fig. 12, we vary the speed of robots from 100m/min to 500m/min across three scenarios: static targets, low-speed moving targets, and high-speed moving targets. Since the experimental trends across the three scenarios are similar, we take the static target case (i.e., subfigures (a)-(c)) as a representative example for discussion. It is observed that as the speed of robots increases, the performance of all five algorithms improves, i.e., with larger success rates, lower first success time and average search time. It is evident that a higher speed of the robots facilitates search of the targets. Once again, our algorithm outperforms all the other algorithms across all metrics. For instance, with the highest speed (500m/min) in Fig. 12(a), our algorithm achieves a success rate of 0.89,



(a) Success rate for static targets (b) First success time for static targets (c) Average search time for static targets

(d) Success rate for low-speed moving targets (e) First success time for low-speed moving targets (f) Average search time for low-speed moving targets

(g) Success rate for high-speed moving targets (h) First success time for high-speed moving targets (i) Average search time for high-speed moving targets

Fig. 13. Varying communication range under static, low-speed moving, and high-speed moving target conditions, in a search area of 10km×10km without obstacles. There are ten robots and five targets which are randomly deployed. Search duration is 120 minutes. The probability of robot failure is 0.2. The probability of successful target detection is 0.8. The probability of successful communication is 0.5.

compared to 0.81 and 0.39 for our algorithm without communication and the LoPSO, respectively.

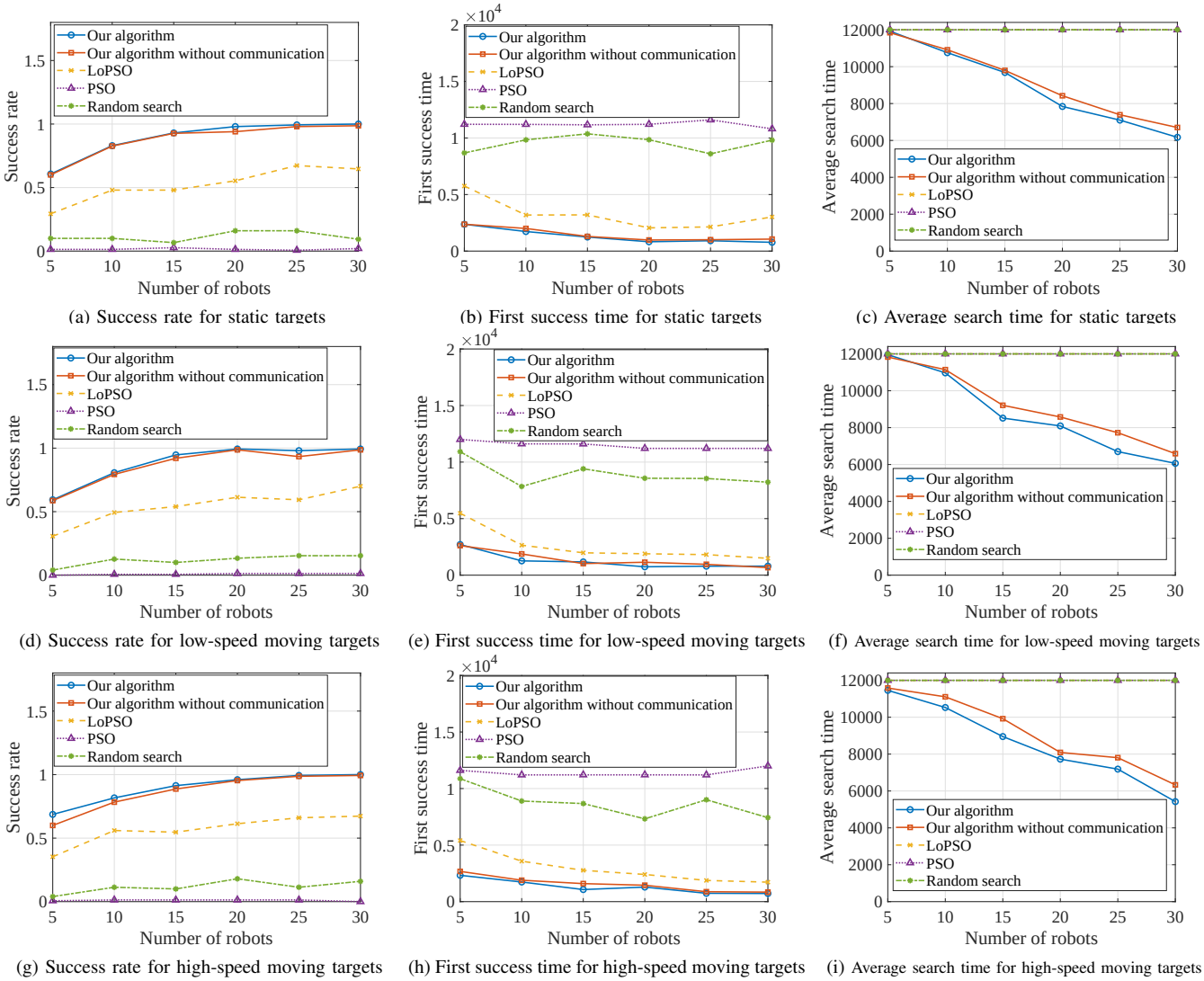
G. Impact of Communication Range

Robotic networks is a type of dynamic networks where connectivity of robots changes over time due to movement of the robots. In our algorithm, if a robot encounters another robot during the movement, they build a tentative connection followed by message exchange. Once the two robots move apart and exceed the communication range, their connection link is broken. Therefore, the impact of intermittent connectivity has been implemented and tested in our study.

Notice that the LoPSO and the PSO both considered communications of robots in their approaches. We compare our algorithm with the LoPSO and the PSO by varying the communication range of robots from 0m to 1000m in Fig. 13. Since the experimental results across the three scenarios of target movement are similar, we take the static target case (i.e., subfigures (a)-(c)) as a representative example for discussion. The figure demonstrates that our algorithm outperforms the others across all performance metrics. As the communication range increases, it facilitates connecting neighboring robots and consequent information exchange among those robots, which enables more effective planning of their search paths. In Fig. 13(b), with the increase of the communication range, the first success time of our algorithm and the LoPSO both decrease while that of the PSO increases. This is because, in the PSO, when a robot detects a target (i.e., a source in [21]), it shares this information with nearby robots who will move to the shared position to locate the target. As a result, the robots converge into clusters which prevents them from exploring other areas for new targets.

H. Impact of Search Area Scale

Fig. 14 illustrates the impact of search area scale by varying the number of robots from 5 to 30 in a large-scale search area of 100km×100km. To ensure the robots can cover the entire

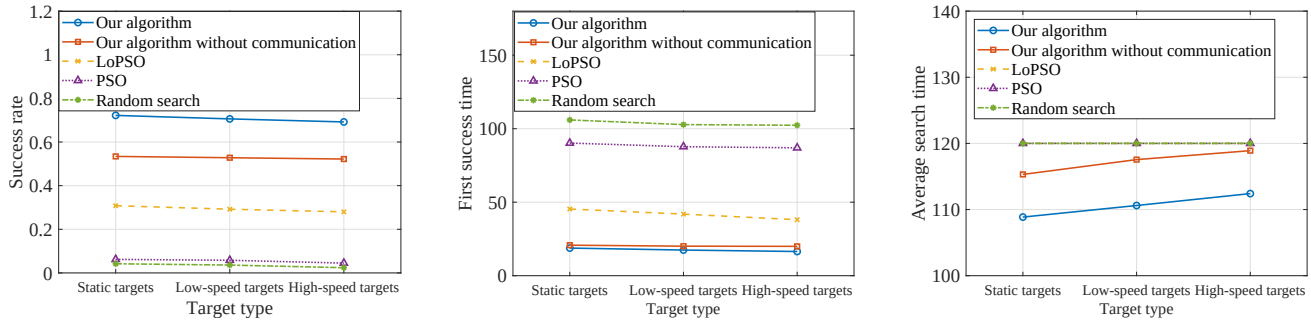


(a) Success rate for static targets (b) First success time for static targets (c) Average search time for static targets

(d) Success rate for low-speed moving targets (e) First success time for low-speed moving targets (f) Average search time for low-speed moving targets

(g) Success rate for high-speed moving targets (h) First success time for high-speed moving targets (i) Average search time for high-speed moving targets

Fig. 14. Large-scale scenarios with a varying number of robots under static, low-speed moving, and high-speed moving target conditions, in a search area of 100km×100km without obstacles. There are five targets which are randomly deployed. Search duration is 200 hours. The probability of robot failure is 0.2. The probability of successful target detection is 0.8. The probability of successful communication is 0.5.



(a) Success rate (b) First success time (c) Average search time

Fig. 15. Heterogeneous robots under static, low-speed moving, and high-speed moving target conditions, in a search area of 10km×10km without obstacles. There are ten robots with energy reserves randomly set between 50% and 100%, and five targets which are randomly deployed. Search duration is 120 minutes. The probability of robot failure is 0.2. The probability of successful target detection is 0.8. The probability of successful communication is 0.5. The curves of LoPSO, PSO and random search in Fig. 15(c) completely overlap.

search area, their lifetime was increased to 200 hours (100×), while other settings remained unchanged.

As shown in Fig. 14(a) and Fig. 14(d), in the static and low-speed scenarios, our algorithm achieved nearly 100% success rate with 20 robots. For high-speed targets (Fig. 14(g)), 25 robots were required to reach a similar level due to increased detection difficulty. Across all scenarios, Fig. 14 demonstrates that our algorithm consistently outperformed our algorithm

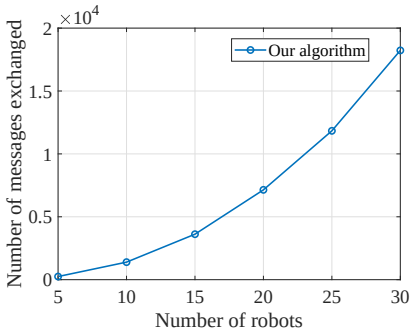


Fig. 16. Communication overhead of our algorithm in a search area of 10km×10km without obstacles. There are five targets which are randomly deployed. Search duration is 120 minutes. The probability of robot failure is 0.2. The probability of successful target detection is 0.8. The probability of successful communication is 0.5.

without communication, which in turn outperformed LoPSO, followed by random reach and PSO. In terms of average search time (Fig. 14(c), (f), and (i)), our method showed a clear advantage over all baselines. Notably, unlike earlier results, Fig. 14(a), (b), (d), (e), (g), and (h) show that random search outperformed PSO in success rate and first success time. This is because PSO is prone to premature convergence, limiting early coverage.

I. Impact of Heterogeneous Robots

Fig. 15 illustrates the impact of heterogeneous robots under three scenarios (static, low-speed, and high-speed targets), where each robot's energy reserve was randomly set between 50% and 100%. Other experimental settings remained unchanged. The figure presents results for success rate, average search time, and first success time.

Across all scenarios, our algorithm consistently outperformed our algorithm without communication, followed by LoPSO, PSO and random search. Particularly in terms of success rate and average search time (as shown in Fig. 15(a) and Fig. 15(c)), our algorithm demonstrated a significant advantage. As expected, the success rate and average search time decreased with increasing target speed, while the first success time increased.

J. Communication Overhead

In Fig. 16, the communication overhead of the proposed algorithm is evaluated by measuring the total number of messages exchanged among robots during the search process. The number of robots is increased from 5 to 30. As the team size grows, the communication overhead rises from 248 to 18226, showing a clear upward trend. This increase is mainly due to the decentralized coordination mechanism, where more robots lead to more frequent information exchange. Despite the growth, the overhead remains within a reasonable range and does not compromise the scalability of the system.

VII. Conclusion

In this paper, we developed a motion coordination algorithm that enables swarm robots to search for targets in an unknown area with obstacles. By theoretical analysis, our motion coordination algorithm was validated to ensure that all static targets are guaranteed to be found if the mask coefficient α is zero. We derived an upper-bound on the total time for the robots to traverse all the grids in the search area. The performance of our algorithm was further validated via extensive simulations in general cases that α is greater than zero and the targets are mobile. Simulation results showed that our algorithm outperformed the state-of-the-art, achieving a success rate of over 90% in finding all mobile targets.

Acknowledgments

This work was supported in part by the Faculty Development Scheme (UGC/FDS14/E02/21 and UGC/FDS14/E08/23) and the Big Data Intelligence Centre at Hang Seng University of Hong Kong.

References

[1] D. Stormont, "Autonomous rescue robot swarms for first responders," in *CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety, 2005.*, 2005, pp. 151–157.

[2] R. R. Murphy, J. Kravitz, S. L. Stover, and R. Shoureshi, "Mobile robots in mine rescue and recovery," *IEEE Robotics Automation Magazine*, vol. 16, no. 2, pp. 91–103, 2009.

[3] M. Bakhshipour, M. Jabbari Ghadi, and F. Namdari, "Swarm robotics search & rescue: A novel artificial intelligence-inspired optimization approach," *Applied Soft Computing*, vol. 57, pp. 708–726, 2017.

[4] "DARPA Announces "Gremlins" UAS Program," http://www.unmannedsystemstechnology.com/2015/09/darpa-announces-gremlins-uas-program/, 2015.

[5] M. A. Ma'sum, G. Jati, M. K. Arrofi, A. Wibowo, P. Mursanto, and W. Jatmiko, "Autonomous quadcopter swarm robots for object localization and tracking," in *MHS2013*, 2013, pp. 1–6.

[6] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörenyi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3866–3873.

[7] H. Çelikkanat and E. Şahin, "Steering self-organized robot flocks through externally guided individuals," *Neural Computing and Applications*, vol. 19, no. 6, pp. 849–865, 2010.

[8] C. Dimidov, G. Oriolo, and V. Trianni, "Random walks in swarm robotics: an experiment with kilobots," in *International Conference on Swarm Intelligence*. Springer, 2016, pp. 185–196.

[9] E. Sakthivelmurugan, G. Senthilkumar, K. Prithiviraj, and K. T. Devraj, "Foraging behavior analysis of swarm robotics system," in *MATEC Web of Conferences*, vol. 144. EDP Sciences, 2018, p. 01013.

[10] C. Wang, J. Cheng, J. Wang, X. Li, and M. Q.-H. Meng, "Efficient object search with belief road map using mobile robot," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3081–3088, 2018.

[11] Q. Tang, L. Ding, F. Yu, Y. Zhang, Y. Li, and H. Tu, "Swarm robots search for multiple targets based on an improved grouping strategy," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 15, no. 6, pp. 1943–1950, 2017.

[12] F. De Rango, N. Palmieri, X.-S. Yang, and S. Marano, "Swarm robotics in wireless distributed protocol design for coordinating robots involved in cooperative tasks," *Soft Computing*, vol. 22, no. 13, pp. 4251–4266, 2018.

[13] Y. Zhang, G. Tian, J. Lu, M. Zhang, and S. Zhang, "Efficient dynamic object search in home environment by mobile robot: A priori knowledge-based approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 9466–9477, 2019.

[14] J. W. Durham, A. Franchi, and F. Bullo, "Distributed pursuit-evasion without mapping or global localization via local frontiers," *Autonomous Robots*, vol. 32, no. 1, pp. 81–95, 2012.

[15] R. A. Coogle and A. M. Howard, "The iceberg observation problem: Using multiple agents to monitor and observe ablating target sources," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2013, pp. 1660–1665.

[16] Q. Tang, F. Yu, Y. Zhang, and J. Zhang, "A stigmergetic method based on vector pheromone for target search with swarm robots," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 32, no. 3, pp. 533–555, 2020.

[17] B. A. Asfora, J. Banfi, and M. Campbell, "Mixed-integer linear programming models for multi-robot non-adversarial search," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6805–6812, 2020.

[18] J. Mou, T. Hu, P. Chen, and L. Chen, "Cooperative mass path planning for marine man overboard search," *Ocean Engineering*, vol. 235, p. 109376, 2021.

[19] J. J. Acevedo, J. Messias, J. Capitán, R. Ventura, L. Merino, and P. U. Lima, "A dynamic weighted area assignment based on a particle filter for active cooperative perception," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 736–743, 2020.

[20] H. Saadaoui, F. E. Bouanani, and E. Illi, "Information sharing based on local pso for uavs cooperative search of moved targets," *IEEE Access*, vol. 9, pp. 134 998–135 011, 2021.

[21] J. Zhang, Y. Lu, Y. Wu, C. Wang, D. Zang, A. Abusorrah, and M. Zhou, "Pso-based sparse source location in large-scale environments with a uav swarm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 5249–5258, 2023.

[22] L. Zhao, R. Li, J. Han, and J. Zhang, "A distributed model predictive control-based method for multidifferent-target search in unknown environments," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 1, pp. 111–125, 2023.

[23] H. Guo, Q. Peng, Z. Cao, and Y. Jin, "Drl-searcher: A unified approach to multirobot efficient search for a moving target," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 3, pp. 3215–3228, 2024.

[24] H. Saadaoui and F. El Bouanani, "A local pso-based algorithm for cooperative multi-uav pollution source localization," *IEEE Access*, vol. 10, pp. 106 436–106 450, 2022.

[25] A. Al Amin, I. Azam, M. Masuduzzaman, A. Qayyum, M. S. Sarwar, S. Khan, and S. Y. Shin, "Gravitational search algorithm swarm-based uav reconnaissance for multiple targets detection in unknown environment," *IEEE Access*, vol. 13, pp. 36 897–36 908, 2025.

[26] Q. Jia, L. Zhang, L. Huang, H. Xu, H. Sun, W. Kong, and X. Feng, "Multi-auv cooperative search method based on high-capacity chance communication," *IEEE Sensors Journal*, vol. 25, no. 3, pp. 5603–5614, 2025.

[27] Z. Li, G. Li, A. Sadeghi, J. Sun, G. Wang, and J. Wang, "Multi-vehicle cooperative persistent coverage for random target search," *IEEE Robotics and Automation Letters*, vol. 10, no. 7, pp. 6680–6687, 2025.

[28] J. Czyzowicz, M. Godon, E. Kranakis, and A. Labourel, "Group search of the plane with faulty robots," *Theoretical Computer Science*, vol. 792, pp. 69–84, 2019, special issue in honor of the 70th birthday of Prof. Wojciech Rytter. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397518306005

[29] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 413–14 423, 2020.

[30] J. Wang and R. Wang, "Multi-uav area coverage track planning based on the voronoi graph and attention mechanism," *Applied Sciences*, vol. 14, no. 17, p. 7844, 2024.

[31] H. Liu, X. Chu, Y. Leung, and R. Du, "Simple movement control algorithm for bi-connectivity in robotic sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 7, pp. 994–1005, 2010.

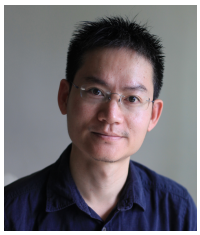
[32] Q. Shen, J. Peng, W. Xu, Y. Sun, W. Liang, L. Chen, Q. Zhao, and X. Jia, "Fair communications in uav networks for rescue applications," *IEEE Internet of Things Journal*, vol. 10, no. 23, pp. 21 013–21 025, 2023.

[33] G. Xu, X. Chen, B. Wang, K. Li, J. Wang, and X. Wei, "A search strategy of uav's automatic landing on ship in all weather," in *2011 International Conference on Electrical and Control Engineering*, 2011, pp. 2857–2860.

[34] S. Bao, J. Lai, Z. Chen, P. Lyu, and W. Chen, "Aerodynamic model/ins/gps failure-tolerant navigation method for multirotor uavs based on federated kalman filter," in *2017 Chinese Automation Congress (CAC)*, 2017, pp. 1121–1125.

[35] M. Ishat-E-Rabban and P. Tokekar, "Failure-resilient coverage maximization with multiple robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3894–3901, 2021.

[36] X. Yu, B. Su, Z. Wang, J. Zhang, and C. Zhang, "Cognitive robotics: Enhancing multirobot target search in unknown environments through adaptive communication strategies," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 55, no. 5, pp. 3449–3463, 2025.



Hai Liu is Professor with Department of Computer Science, The Hang Seng University of Hong Kong (HSUHK). Before joining HSUHK, he held several academic posts at University of Ottawa and Hong Kong Baptist University. Dr Liu received PhD in Computer Science at City University of Hong Kong, and received MSc and BSc in Applied Mathematics at South China University of Technology. His research interest includes wireless networking, cloud computing, artificial intelligence and algorithm design and analysis.

Shujin Ye received the BE degree in computer science and technology from South China Normal University, the MS degree in software engineering from South China University of Technology, and the PhD degree in computer science from Hong Kong Baptist University. He is currently a lecturer at the School of Data Science and Engineering, Guangdong Polytechnic Normal University (GPNU). Before joining GPNU, he worked with the Department of Computer Science at The Hang Seng University of Hong Kong. His research interests include cloud computing and swarm intelligence.

Chris Y. T. Ma is an assistant professor at the Hang Seng University of Hong Kong, where he joined in 2016. He received his BEng in Computer Engineering from The Chinese University of Hong Kong; MPhil in Computer Science and Engineering from The Chinese University of Hong Kong; and PhD in Computer Science from Purdue University, West Lafayette, IN, USA. His research interests include performance, privacy, and security study of networks and cyber-physical systems.

Yue Wang received the B.S. degree in Electronics and M.E. in Information Science from Peking University, Beijing China, and the Ph.D. degree in Industrial Engineering from The Hong Kong University of Science and Technology. He is an Associate Professor in the Department of Mathematics and Information Technology, The Education University of Hong Kong. His research interests include machine learning, natural language processing, education analytics, healthcare informatics, and operations management.

Tse-Tin Chan (Member, IEEE) received the B.Eng. (First Class Hons.) and Ph.D. degrees in Information Engineering from The Chinese University of Hong Kong (CUHK), Hong Kong SAR, China, in 2014 and 2020, respectively. He is currently an Assistant Professor with the Department of Mathematics and Information Technology, The Education University of Hong Kong (EdUHK), Hong Kong SAR, China. From 2020 to 2022, he was an Assistant Professor with the Department of Computer Science, The Hang Seng University of Hong Kong (HSUHK), Hong Kong SAR, China. His research interests include wireless communications and networking, Internet of Things (IoT), age of information (AoI), and artificial intelligence (AI)-native wireless communications.