A Distributed Search Algorithm for Multiple Robots with Collision Avoidance

Xiaotian Yang
School of Electrical Engineering and Telecommunications
The University of New South Wales
UNSW, Sydney, NSW 2052, Australia
xiaotian.yang@unsw.edu.au

Abstract

This paper proposes a collision-free search algorithm for multiple robots in an unknown area. The algorithm is decentralized and robots only have limited sensing and communication ranges. A grid pattern is used to represent the search area and robots move between vertices along the edges to visit all accessible area. Robots choose their future steps based on a random algorithm with repulsive force when there are unvisited neighbor vertices and use the breadth-first search algorithm to find the path to one of the nearest unvisited vertices when they are surrounded by visited vertices. The convergence of the algorithm is proved and it is compared with seven other algorithms from related works to show the effectiveness.

1 Introduction

The advancement in electromechanics, material engineering and 3D printing in recent years allows robots to be smaller and cheaper. Thus, there is an increasing number of applications with multiple robots which attract researchers' interests. Those applications include fire detection [Marjovi *et al.*, 2009; Casbeer *et al.*, 2005], data harvest [Gu *et al.*, 2006] border patrol [Matveev *et al.*, 2012; 2011], intruder detection [Nazarzehi and Baranzadeh, 2015b; Yang, 2017a; 2017e], and mine clearance [Hameed, 2016]. The advantage of multiple robots over a single robot is that it provides efficiency, robustness, reliability, flexibility, and scalability as shown in [Yang, 2016; 2017d; 2017b; 2017c].

There are two kinds of search tasks which are categorized by whether the number of targets is known or not. If it is known (situation I), the search task will be finding a certain number of targets by using multiple mobile robots. In situation II, with an unknown number of targets, the mission becomes driving robots to detect each point of the area. This can be thought as a dynamic method to achieve blanket coverage [Yang, 2017a; 2017e]. Other relative coverage types include barrier coverage [Barr *et al.*, 2011; Cheng and Savkin, 2009] and sweep coverage [Semakova *et al.*, 2017; Cheng and Savkin, 2011].

To represent the search area, the topological map and the geometrical map can be used. A geometrical map describes every point in the area and needs to deal with complex data fusion. The topological maps can simplify the area to a grid thus only some points are recorded which decreases the calculation load. Some algorithms used Voronoi partition [Mohseni *et al.*, 2015; Aurenhammer, 1991] to build a topological map while others used uniform tessellation [Baranzadeh, 2013; Baranzadeh and Savkin, 2017; Yang, 2016; 2017d; 2017b; 2017a]. The later one is easier to generate and all the cells of the grid will be the same. Some random algorithms inspired by animals such as the random walk (RW) algorithm and the Lévy flight (LF) algorithm [Sutantyo *et al.*, 2010] do not need a map. They allow robots to walk to any place in the area which ensures the whole area can be detected.

When utilizing a group of robots, the control and communication methods can be classified into centralized methods and decentralized methods. Centralized algorithms need a leader robot with better communication and computation ability than followers to collect information and distribute the commands. Thus, the leader is expensive and the system will not work if the leader fails. However, in decentralized control, all the robots have equal functions and only need local communication and sensing; see e.g. [Ren and Beard, 2008; Savkin, 2004]. Therefore, robots can use cheaper devices and is easier to achieve. Moreover, the system can still work after some robots are broken.

Another significant topic for a team of robots is collision avoidance. In [Baranzadeh, 2013; Savkin *et*

This work was supported by the Australian Research Council (Grant DP170103750).

al., 2012; Nazarzehi and Baranzadeh, 2015b; 2015a; Nazarzehi *et al.*, 2015], robots were assumed as particles and collisions were allowed which is impractical. [Baranzadeh and Savkin, 2017] claimed that the collision avoidance algorithm in ARIA library of Pioneer 3-DX robots is used, however, it did not describe how it worked and did not state the sensing range for obstacles. In fact, that algorithm is the repulsive force algorithm which was also used in [Sutantyo *et al.*, 2010] so, using that algorithm, a robot may not stay at vertices which disagrees with its assumptions and the design scheme. However, no solutions were provided for this situation. Other collision avoidance in [Savkin and Wang, 2013] required global information which is not suitable for the unknown area.

Some grid-based algorithms [Baranzadeh, 2013; Yang, 2017d; 2017b; 2016; Nazarzehi *et al.*, 2015] had many repeated areas. Thus, the method to decrease the repetition should be researched to improve the algorithm. In [Nazarzehi and Baranzadeh, 2015b; 2015a], the LF algorithm is used to have fewer repetitions than the random choice (R) algorithm in [Baranzadeh, 2013]. However, the LF algorithm of a robot only ensures this robot has less repetition on its own route than using the R algorithm, different robots may still repeat the visited points of other robots as robots are independent with no communication on the information about the visited area. The algorithms in [Nazarzehi and Baranzadeh, 2015b; 2015a] also had another problem as they allowed collisions to happen. The algorithm in [Baranzadeh and Savkin, 2017] asked robots to choose the nearest unvisited vertex and to go there straightly without following vertices and edges. The term 'nearest' means the minimum Euclidean distance. However, if there is an obstacle in the path, the nearest vertex may not lead to the shortest path and continuous collision avoidance need to be considered which was ambiguous in that algorithm.

If the robot uses the explored map to find an unvisited vertex and moves through edges of the grid, the shortest path means the route with the fewest vertices. There are three common graph search methods, namely, the breadth-first search algorithm, the greedy best-first search algorithm and the A* Algorithm. The breadth-first search calculates the distance to the start point in each direction evenly which results in a large explored area but the shortest path while the other two algorithms focus on one target vertex only. The greedy best-first algorithm estimates the distance to that target and only needs to explore a small area. A* algorithm estimates the sum of the estimated values of the two algorithms above for one target, so it results in both small explored area and shortest path. However, in this paper, the target vertex is unknown because there may be many unvisited vertices around the visited area. So the algorithm should firstly find one target vertex which leads to the fewest steps and then plan the route. If the A* algorithm is used, it needs to be executed for each unvisited vertex at the boundary of the explored area which may have many repeated explored parts. Then all paths need to be compared to find the one with the fewest steps. Therefore, the breadth-first search algorithm should be used as it is applied in each direction without repeated area.

This paper presents a new search method using multiple robots in an unknown area. The algorithm is decentralized to allow each robot to be cheap with limited sensing and communication ranges. The search area is represented by an equilateral triangular (T) grid, which is a kind of topological map, as it provides 100% complete search coverage based on [Yang, 2017d]. Robots move between vertices for one edge length in each step. Based on the grid, a robot and its neighbors in the communication range will select the next vertex in sequence to avoid collisions caused by choosing the same vertex, and robots are synchronized to move simultaneously to avoid collisions while moving. If there are unvisited neighbor vertices, robots use the random algorithm. The repulsive force is applied if there are also neighbor robots. If robots are surrounded by visited vertices, the breadth-first search algorithm will be used to decrease repeated visitation by finding the path with the smallest number of steps to an unvisited vertex. In this paper, both situations of the search tasks will be discussed and compared with other algorithms.

The rest of the paper is organized as follows. Next part is the problem statement followed by the algorithm described in the third section. Section four compares the simulation result of the proposed algorithm with other related algorithms. Finally, a conclusion is drawn in section five.

2 Problem Statement

This algorithm is improved based on the algorithms in [Yang, 2016; 2017d] to solve the search problem so this paper takes advantage of the method of definition in those two papers.

The search area is $A \subset \mathbb{R}^2$ which has a limited number of static obstacles O_1, O_2,..., O_l. There are m robots labeled as rob_1 to rob_m to search static targets t_1 to t_n. The position of rob_i at time k is $p_i(k)$. An arbitrary known T grid pattern with a side length of a is used for robots to move in each step with the following definitions and assumptions. All these initial settings can be seen in Figure 1.

Assumption 2.1. The area A is a bounded, connected and Lebesgue measurable set. The obstacles O_i are non-overlapping, closed, bounded and linearly connected sets for any $i \geq 0$. Robots have no knowledge about the area before exploration.

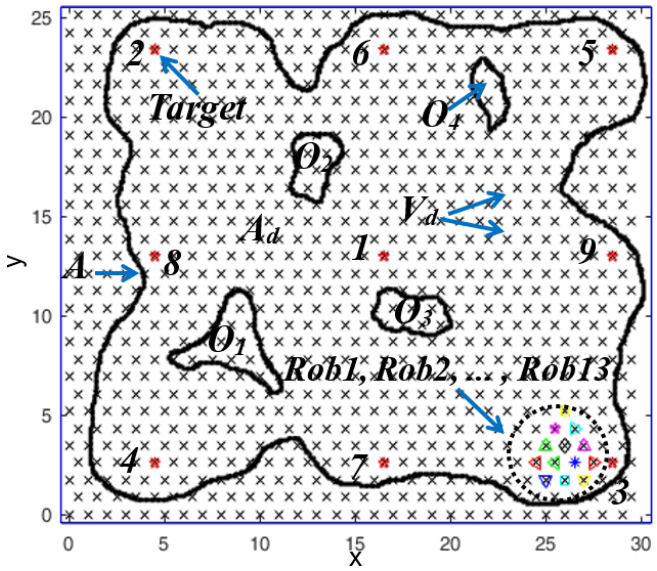

Figure 1: The area and initial settings.

Definition 2.1. Let $O := \bigcup O_i$ for all $i > 0$. Then $A_d := A \backslash O$ represents the area that needs to be detected.

Assumption 2.2. The initial condition is that all robots use the same T grid pattern and the same coordinate system. They start from different vertices near a corner of the area.

Assumption 2.3. All the ranges and radius are circular.

Definition 2.2. To avoid collisions with other things before move, a safe distance r_{safe} is set which includes both r_{rob} and e. So $r_{safe} \geq r_{rob} + e$.

To guarantee the area can be completely searched and each part can be detected, the following relations about the width of passage W_{pass}, the sensing range for target r_{st}, the sensing range for obstacle r_{so} and the curvature should be assumed. Note that W_{pass} and r_{so} was not defined in previous grid-based search algorithms in [Baranzadeh, 2013; Baranzadeh and Savkin, 2017; Nazarzehi and Baranzadeh, 2015b; 2015a; Nazarzehi *et al.*, 2015]

Assumption 2.4. The W_{pass} between obstacles or between the obstacle and the boundary is wider than $a + 2 * r_{safe}$. So that there will always be accessible vertices in the passage.

Assumption 2.5. $r_{so} \geq a + r_{safe}$.

Assumption 2.6. $r_{st} \geq a + r_{safe}$.

Assumption 2.7. $r_c \geq 2 * a + e$.

Assumption 2.8. The curvature of the convex part of the boundary should be smaller than or equal to $1/r_{st}$.

Let V_a be the set of all the accessible vertices. $N_{s,i}(k)$ represents the set of sensing neighbors around rob_i which can be visited in the next step. In $N_{s,i}(k)$, $V_{v,i}(k)$ and $V_{u,i}(k)$ denote the two sets for visited vertices and unvisited vertices. Then the choices from these two sets are c_v and c_u respectively.

The following definition is about variables in calculating the repulsive force which is from [Yang, 2017d].

Definition 2.3. In loop k, $p_{i,ave}(k)$ is defined as the average position of choices made by robots which chose earlier and positions of communication neighbors which have not chosen the next step yet. Then $d_{c_u,p_{i,ave}(k)}$ is the distance from $p_{i,ave}(k)$ to one unvisited vertex. Then the maximum values is $max(d_{c_u,p_{i,ave}(k)})$ and the corresponding set of vertices which result in the maximum distance is represented as $V_{umax,i}(k)$.

If the breadth-first search algorithm is used in step k for rob_i, a flag $f_i(k)$ is set. The time for the 'choose' loop is set as t_c which is small but enough for only a few check loops and the path generation. So the robot could move in current step k if it is near the unvisited vertex but it would not increase the total time for a step too much. The corresponding time for path generation is represented as t_p. The route is represented as a set of vertices $V_{r,i}(k)$ for rob_i at time k. If $|\bullet|$ is used to represent the number of elements in set $\bullet$, $|V_{r,i}(k)|$ is the total number of steps in that route. Let $v_{r,i,j}(k)$ be the jth step in the route and $c_{r,i}$ be the choice from the route.

Definition 2.4. If the route has not been fully generated, $|V_{r,i}(k)| = 0$.

3 Algorithm

The proposed algorithm drives robots to go through the area via a grid pattern. The general procedure of the proposed algorithm is similar to the one in [Yang, 2017d] which will be briefly described. The major difference is on how to choose the next vertex in step 5 which will be discussed in detail. The whole procedure is in Figure 2 for situation I and in Figure 3 for situation II. 'comm' in flowcharts stands for communication. As the differences between the two situations are small, the explanation is based on situation I but special features of situation II will be mentioned specially.

3.1 Preparations for the choice

Initially, rob_i sends its position and $f_i(k)$ to neighbors in r_c. Based on the relative positions, robot obstacles are identified, and $N_{s,i}(k)$ and $N_{c,i}(k)$ are generated. To decide the selection sequence, rob_i at $p_i(k)$ with $f_i(k) = 1$ has higher priority than its communication neighbor rob_j at $p_j(k)$ with $f_j(k) = 0$. Because if the planned path for rob_i is blocked by rob_j, rob_i will waste the step k+1 in waiting or calculating a new route. If rob_j could choose another unvisited vertex, both robots will make

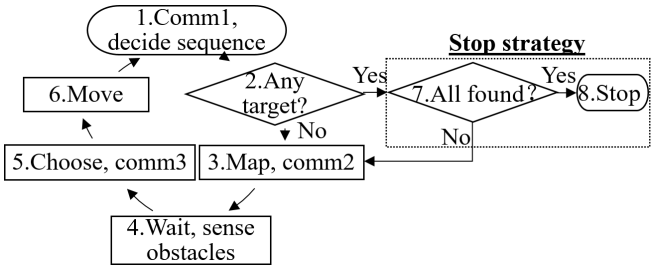

Figure 2: The flowchart of the algorithm in situation I.

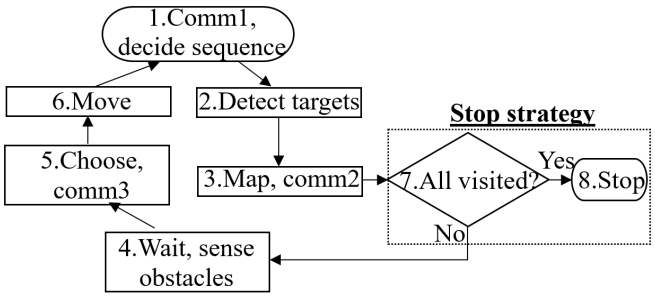

Figure 3: The flowchart of the algorithm in situation II.

step k+1 more useful. If they have the same priority based on flags, then if $p_{jx} > p_{ix} \mid (p_{jy} > p_{iy} \& p_{jx} = p_{ix})$, rob_j will have the higher priority in making the next choice.

In the second step, robots detect targets within r_{st}. In situation I, if a target is detected, the robot will go to step 7. Otherwise, it moves to step 3. In situation II, robots only go to step 3.

In step 3, the robot shares its explored map which is from its detection and received positions in step 1 to update its map again. In situation II, whether the sensing neighbors of vertices are visited or not will also be sent to update the visitation states of all vertices and step 7 will use this information to judge the stop condition. If the condition is not satisfied, the robot goes to step 4. Note that if a sensing neighbor is blocked, it is also recorded as visited.

In the fourth step, rob_i needs to wait for the choices from robots with higher priorities if there is any because the chosen vertices will be visited points in the next step and may be obstacles for rob_i. If all robots are connected, the last one in the choosing sequence should wait for every other robot and the processing time for the whole selection process should be set to the value in this condition to synchronize all robots in each loop. When it is the turn of rob_i, it will detect obstacles which block its ways to its sensing neighbors. The blocked vertex $n_{s,i,j}(k)$ will be removed from $N_{s,i}(k)$. If no objects are sensed, this robot will still wait for the same amount of processing time to get synchronized.

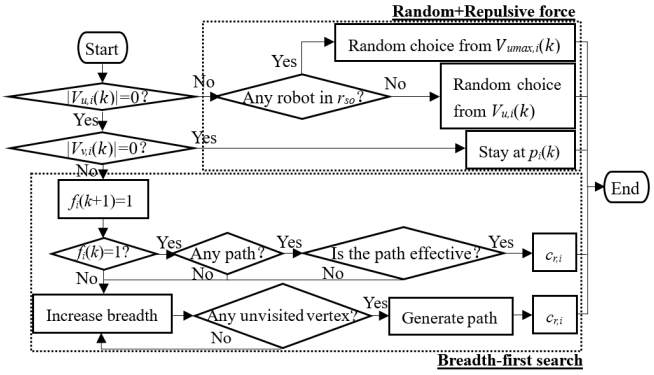

Figure 4: The flowchart for the step 'choose'.

3.2 Choose and Comm3

The 'choose' step for the proposed algorithm can be seen in Figure 4. Previous work [Yang, 2016] only considered random algorithm and chose the unvisited vertex first. Reference [Yang, 2017d] added repulsive force algorithm on [Yang, 2016] for nearby robots. The proposed algorithm improves [Yang, 2017d] by applying breadth-first search algorithm for robots to escape a visited area which is the bottom half of Figure 4. When rob_i has unvisited neighbors, it will check whether there is any robot obstacle. If there is not any, it chooses one from $V_{u,i}(k)$ randomly. Otherwise, rob_i calculates the $d_{c_u,p_{i,ave}(k)}$ for each vertex in $V_{u,i}(k)$ to find the $max(d_{c_u,p_{i,ave}(k)})$ and $V_{umax,i}(k)$. Then it randomly chooses a vertex from $V_{umax,i}(k)$. When there are no accessible vertices to choose, rob_i could only stay at its current place.

The breadth-first search algorithm will be applied when all the sensing neighbors are visited. Firstly, rob_i sets the $f_i(k+1)$ to have a high priority in next step in case it needs to execute the planned path. Then if $f_i(k) = 0$, this step will generate a new path. But when $f_i(k) = 1$, the path may still have not been generated in the previous step as the path may be too long. Even if it is generated, the target unvisited vertex may become a visited one based on the updated explored map in step 3. In these two situations, rob_i still needs to generate the path to the unvisited vertex using the breadth-first search algorithm.

In this algorithm, breadth means the steps from $p_i(k)$ to the vertex to be checked. As the sensing neighbors are visited, the breadth is increased to 2 and rob_i checks the visitation states of the sensing neighbors of the checked vertices. Those check vertices are called the father nodes of the related sensing neighbors which will be recorded while checking. As each vertex only needs to be checked once, each vertex has one father node. If no unvisited vertices are found, the breadth will be increased again until an unvisited vertex is found. Then based on the father nodes, a path $V_{r,i}(k)$ from the unvisited vertex to

$p_i(k)$ is found. The breadth-first search algorithm guarantees that the target unvisited vertex has the smallest breadth from $p_i(k)$, namely the path has the fewest steps. When there is an available path $V_{r,i}(k)$, rob_i chooses the vertex $v_{r,i,j}(k)$ if this is the jth step in the path.

When checking the visitation state of each vertex in the breadth-first search, the current time in making choice for rob_i will be compared with $t_c - t_p$, if the current time is greater, rob_i has no time to finish the algorithm in this step and will choose to stay at $p_i(k)$ at the end of t_c. Then in the movement step it will continue its calculation until the next loop $k+1$. After t_c, the choice will be sent to the communication neighbors of rob_i as the information is needed in their choices. The above choose algorithm can be mathematically represented as:

$$p_i(k+1) = \begin{cases} c_u \text{ with prob. } 1/|V_{u,i}(k)|, \text{ if} \\ \quad (|V_{u,i}(k)| \neq 0) \& (V_{umax,i}(k) = 0), \\ c_u \text{ with prob. } 1/|V_{umax,i}(k)|, \text{ if} \\ \quad (|V_{u,i}(k)| \neq 0) \& (V_{umax,i}(k) \neq 0), \\ c_{r,i}, \text{ if } (|V_{u,i}(k)| = 0) \& \\ \quad ((|V_{v,i}(k)| \neq 0) \& (|V_{r,i}(k)| \neq 0)), \\ p_i(k), \text{ if } (|N_{s,i}(k)| = 0) \mid ((|V_{u,i}(k)| = 0) \& \\ \quad ((|V_{v,i}(k)| \neq 0) \& (|V_{r,i}(k)| = 0))). \end{cases} \quad (1)$$

The random choice is used when there are equal choices because if a deterministic choice like the first available choice is used, the trend of the movement will follow the direction of the first choice instead of searching all directions evenly. Thus, more repeated vertices will be caused which will lead to longer search time.

3.3 Move

All robots move in the same pattern at the same time based on realistic constraints. Each move consists of two phases namely the rotation and the translation. The rotation aims the heading of the robot at the target vertex. Then rob_i can go along the chosen edge in translation where trapezoidal velocity profiles are used based on parameters of the used robots.

3.4 Stop Strategies

For situation I, rob_i broadcasts the detected targets and should stop when the number of targets that it finds equals to the number of targets in the area. Namely, for any robot,

$$p_i(k+1) = p_i(k), \text{ if } |T_{ki}| = |T| \quad (2)$$

In situation II, the visitation states of explored vertices need to be checked. If all visited vertices have no unvisited accessible sensing neighbors, the robot should stop. Namely,

$$p_i(k+1) = p_i(k), \text{ if } \forall v_i \in V_a,\ |V_{u,i}(k)| = 0 \quad (3)$$

This paper focuses on search tasks only which is looking for the coordinates of the targets. In situation I, when the judgment condition of rob_i is satisfied, all robots stop as locations of targets are broadcast when found. However, in situation II, if rob_i satisfies the stop condition, it needs to broadcast its knowledge of targets and a stop signal to inform all other robots. Then all robots will broadcast the positions of targets that they know and terminate the search.

Theorem 3.1. *Suppose that all the above assumptions hold and the search algorithm* (1) *and a stop algorithm* (2) *or* (3) *are used for situation I or situation II. Then, for any number of robots, there is such a time* $k_0 > 0$ *that all targets are detected with probability 1.*

Proof. The algorithm (1) with a judgment method forms an absorbing Markov chain. It includes both transient states and absorbing states. The transient states are all the approachable vertices of the grid that robots visit but do not stop forever. Absorbing states are the vertices that the robots stop forever. Applying algorithm (1), robots select the unvisited neighbor vertices to go if there is any. If all sensing neighbors are visited before finding all targets, robots will look for the shortest route to the nearest unvisited vertex and only stop when all targets are found, or all approachable vertices are visited. For (2), an absorbing state is reached when robots know $|T_{ki}| = |T|$. For (3), the absorbing state is reached until $|V_{u,i}(k)| = 0$ for any vertex. The assumption about W_{pass} guarantees that all the accessible vertices can be visited so absorbing states can be achieved from any initial states, namely with probability 1. This completes the proof of Theorem 3.1 □

4 Simulations

4.1 Routes with the Strict Requirement

Situation I of the proposed algorithm is simulated to verify the algorithm and to show the routes. Nine targets which are averagely allocated in the area are tested. The search results of three robots are in Figure 5 for robot rob_1 and in Figure 6 for all the three robots. Initial setting can be found in Figure 1. There are 30*30 vertices in total in the graph and the three robots are at the bottom right corner which can be thought as the only known entrance to allocate robots. The boundaries and obstacles are shown by the bold black lines formed by discrete dots which obey Assumption 2.4. Parameters used in the simulation are based on the parameters of the Pioneer 3-DX robot as in [Yang, 2017d]. The figures demonstrate

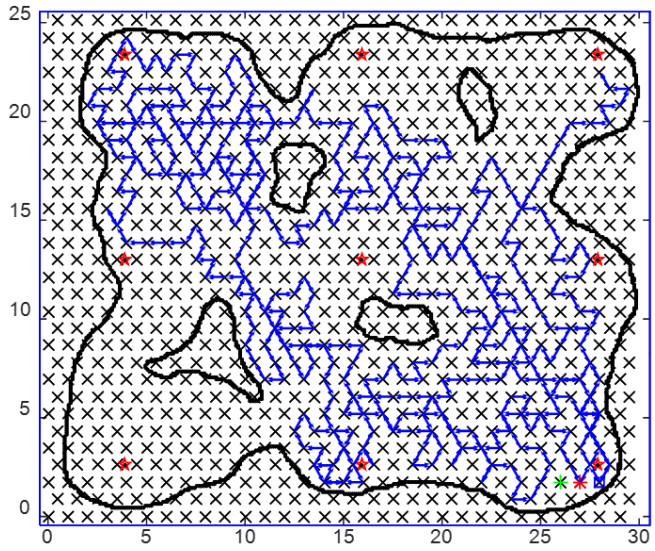

Figure 5: The route of robot 1 in a simulation with 3 robots.

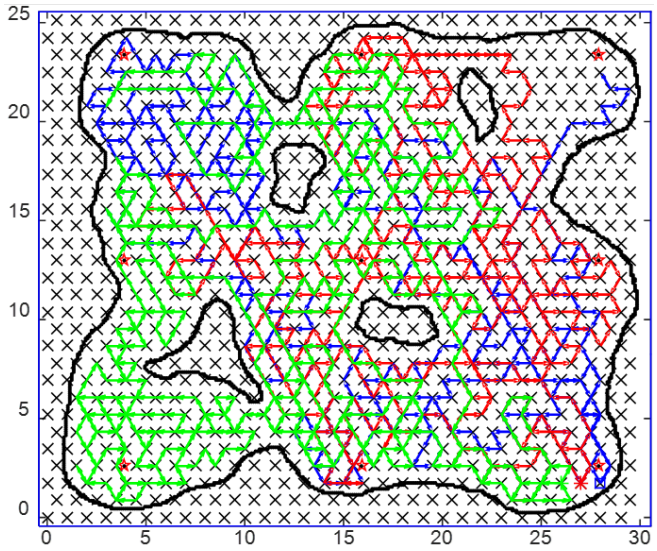

Figure 6: The combined routes in a simulation with 3 robots.

that each target was within r_{st} of a visited vertex, so the search task was finished successfully. Based on the recorded traces, robots had no collisions with each other so it was a collision-free algorithm. Also, the routes in Figure 6 had less repetition compared to [Yang, 2017d]. The route of each robot occupied a larger percentage of the area than [Yang, 2017d] which benefited from the breadth-first search algorithm.

4.2 Comparison

In this section, the proposed algorithm (represented as S3) is compared with its base algorithms in [Yang, 2016] and [Yang, 2017d] which are labeled as S1 and S2 in the graph. It is also compared to the R algorithm [Baranzadeh, 2013] and the random choice with unvisited vertices first (RU) algorithm [Baranzadeh and Savkin, 2017] which used the grid pattern but did not consider collision avoidance clearly. Algorithms in [Sutantyo *et al.*, 2010] including the RW algorithm, the LF algorithm, and the Lévy flight with potential field (LFP) algorithm are compared as well because they are also random algorithms in unknown areas. Algorithm LF and LFP use Lévy distribution to generate step lengths and uniform distribution to generate the moving directions. Algorithm LFP uses repulsive force to avoid collisions between robots. But there is no collision avoidance scheme in algorithm R, RU, RW, and LF. To compare all algorithms with the same requirement of collision avoidance. One loop is added to the total number of loops for each collision. In reality, one step may not be enough considering consecutive collisions. So this is a reasonable way to calculate the total search time.

To compare all methods in the same environment, all the eight algorithms are simulated in the area shown in

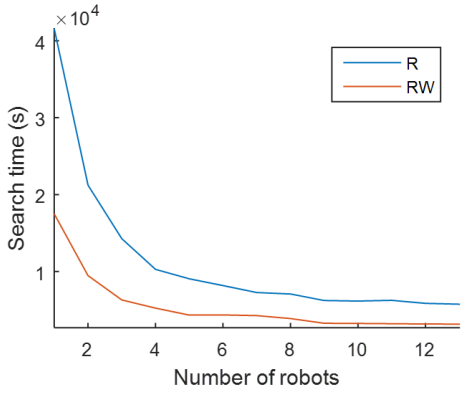

Figure 7: Search time for 9 targets using algorithms R and RW.

Figure 1 with one to thirteen robots. The results in the figures are the average result of at least 100 simulations. To control the communication ability to be the same, equation 2 is used as the stop judgment, the ad-hoc mode with r_c is used for communication and the broadcasting mode is chosen for sending positions of found targets. [Sutantyo *et al.*, 2010] claimed that the LF algorithm and the LFP algorithm would perform better when targets are sparsely and randomly distributed. Therefore, the nine evenly separated targets are used to fully utilize the potential of those two algorithms. Thus, and the effectiveness of the proposed algorithm will be more convincible. Two slowest algorithms R and RW were picked up to be illustrated in Figure 7. So the small differences in other seven algorithms are compared in Figure 8. The Figures 7 and 8 illustrate that the proposed algorithm (1) is always the best under the strict requirement in situation I. The general ranking of search time of the eight

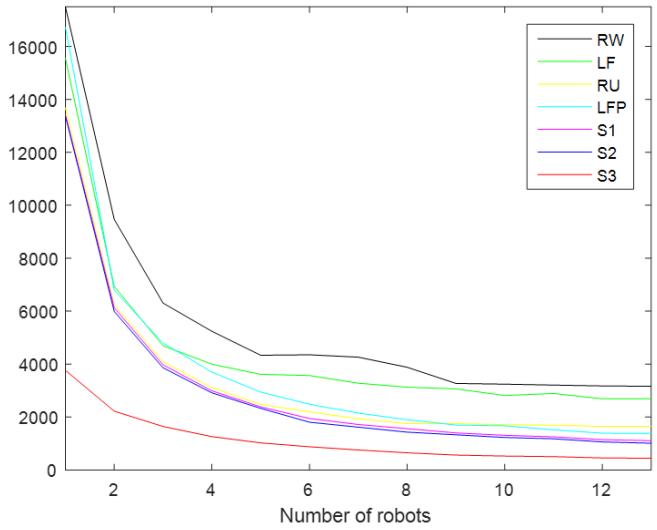

Figure 8: All algorithms except R for 1-13 robots.

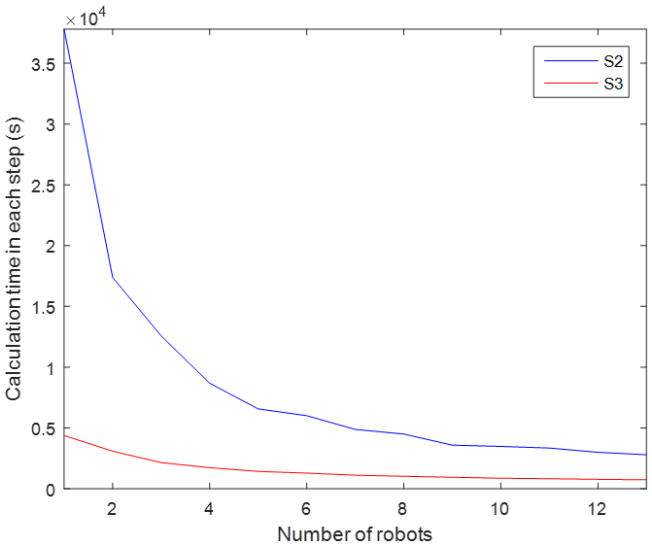

Figure 9: Search time for search algorithm 2 and 3.

algorithms is S3<S2<S1< LFP<RU<LF<RW<R. The proposed algorithm largely decreased the search time to less than half of that in algorithm S2 due to the effect of the breadth-first search algorithm in reducing repeated visitations. So the proposed algorithm is effective in Situation I. Generally speaking, as the number of robots increased, the search time in all algorithms decreased drastically at the beginning but slowly later as more time was spent on other things instead of exploring new vertices only. Specifically, robots in the algorithms S1 S2 R RU and RW need to spend time on visiting repeated vertices. For the R, RU, RW and LF algorithms, robots are set to deal with extra steps caused by collision avoidance. Algorithm S3 requires robots to spend time on finding the route to leave the area of repeated vertices. In the LF, the RW, and the LFP algorithm, robots will meet others more frequently which will break the generated route, thus, decrease the effect of the algorithm.

In Situation II, only algorithm S2 is compared with algorithm S3 in Figure 9 as Algorithm S2 is the best in the seven algorithms to compare. It is clear that Algorithm S3 decreased the search time significantly and it is the best search algorithm.

The proposed algorithm needs a considerable amount of calculation especially when the known map is huge. However, as a small t_c is set, the time-consuming calculation will not affect the normal calculation of other robots and the increased calculation time is not significant comparing to the decrease time caused by decreased steps. t_c can even be set as 0. But a relatively larger t_c could allow more robots to finish path planning and move earlier. This algorithm outweighs all other algorithms and needs to be used on robots with high computation ability and large memory.

5 Conclusion

This paper presented a decentralized random search algorithm with the breadth-first search algorithm in unknown 2D areas. The movements of robots are synchronized and based on a grid pattern to avoid collisions. When robots have unvisited sensing neighbors but no robot obstacles, they choose one from those neighbors randomly. If there are robot obstacles, the repulsive force is applied to separate robots to search different sections. When there are only visited neighbors, the robot will use the breadth-first search algorithm to find the shortest route to one nearest unvisited vertex using its explored map. It is proved that the algorithm converges with probability 1, and the proposed algorithm was compared with seven other algorithms in simulations. The result showed that the proposed algorithm is always the best in situations I and II and it can decrease the search time significantly. Future work may include searching moving targets and avoiding moving obstacles [Hoy *et al.*, 2015; Matveev *et al.*, 2012; Savkin and Wang, 2014].

References

[Aurenhammer, 1991] Franz Aurenhammer. Voronoi diagramsa survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[Baranzadeh and Savkin, 2017] Ahmad Baranzadeh and Andrey V Savkin. A distributed control algorithm for area search by a multi-robot team. *Robotica*, 35(6):1452–1472, 2017.

[Baranzadeh, 2013] Ahmad Baranzadeh. A decentralized control algorithm for target search by a multi-

robot team. In *Australasian Conference on Robotics and Automation*, 2013.

[Barr *et al.*, 2011] Stanley Barr, Jie Wang, and Benyuan Liu. An efficient method for constructing underwater sensor barriers. *Journal of Communications*, 6(5):370–383, 2011.

[Casbeer *et al.*, 2005] David W Casbeer, Randal W Beard, Timothy W McLain, Sai-Ming Li, and Raman K Mehra. Forest fire monitoring with multiple small uavs. In *American Control Conference, 2005. Proceedings of the 2005*, pages 3530–3535. IEEE, 2005.

[Cheng and Savkin, 2009] Teddy M Cheng and Andrey V Savkin. A distributed self-deployment algorithm for the coverage of mobile wireless sensor networks. *IEEE Communications Letters*, 13(11):877–879, 2009.

[Cheng and Savkin, 2011] Teddy M. Cheng and Andrey V. Savkin. Decentralized control for mobile robotic sensor network self-deployment: Barrier and sweep coverage problems. *Robotica*, 29(2):283–294, 2011.

[Gu *et al.*, 2006] Yaoyao Gu, Doruk Bozda, Robert W. Brewer, and Eylem Ekici. Data harvesting with mobile elements in wireless sensor networks. *Computer Networks*, 50(17):3449–3465, 2006.

[Hameed, 2016] Ibrahim A Hameed. Motion planning for autonomous landmine detection and clearance robots. In *International Workshop on Recent Advances in Robotics and Sensor Technology for Humanitarian Demining and Counter-IEDs (RST)*, pages 1–5. IEEE, 2016.

[Hoy *et al.*, 2015] Michael Hoy, Alexey S Matveev, and Andrey V Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33(03):463–497, 2015.

[Marjovi *et al.*, 2009] Ali Marjovi, João Gonçalo Nunes, Lino Marques, and Aníbal de Almeida. Multi-robot exploration and fire searching. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1929–1934. IEEE, 2009.

[Matveev *et al.*, 2011] Alexey S. Matveev, Hamid Teimoori, and Andrey V. Savkin. A method for guidance and control of an autonomous vehicle in problems of border patrolling and obstacle avoidance. *Automatica*, 47(3):515–524, 2011.

[Matveev *et al.*, 2012] Alexey S. Matveev, Chao Wang, and Andrey V. Savkin. Real-time navigation of mobile robots in problems of border patrolling and avoiding collisions with moving and deforming obstacles. *Robotics and Autonomous Systems*, 60(6):769–788, 2012.

[Mohseni *et al.*, 2015] F. Mohseni, A. Doustmohammadi, and M. B. Menhaj. Distributed model predictive coverage control for decoupled mobile robots. *Robotica*, 35(4):922–941, 2015.

[Nazarzehi and Baranzadeh, 2015a] Vali Nazarzehi and Ahmad Baranzadeh. A decentralized grid-based random search algorithm for locating targets in three dimensional environments by a mobile robotic network. In *Australasian Conference on Robotics and Automation. ACRA2015*, 2015.

[Nazarzehi and Baranzadeh, 2015b] Vali Nazarzehi and Ahmad Baranzadeh. A distributed bio-inspired algorithm for search of moving targets in three dimensional spaces. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2507–2512. IEEE, 2015.

[Nazarzehi *et al.*, 2015] V. Nazarzehi, A. V. Savkin, and A. Baranzadeh. Distributed 3D dynamic search coverage for mobile wireless sensor networks. *IEEE Communications Letters*, 19(4):633–636, 2015.

[Ren and Beard, 2008] Wei Ren and Randal W Beard. *Distributed consensus in multi-vehicle cooperative control*. Springer, 2008.

[Savkin and Wang, 2013] Andrey V. Savkin and Chao Wang. A simple biologically inspired algorithm for collision-free navigation of a unicycle-like robot in dynamic environments with moving obstacles. *Robotica*, 31(6):993–1001, 2013.

[Savkin and Wang, 2014] Andrey V Savkin and Chao Wang. Seeking a path through the crowd: Robot navigation in unknown dynamic environments with moving obstacles based on an integrated environment representation. *Robotics and Autonomous Systems*, 62(10):1568–1580, 2014.

[Savkin *et al.*, 2012] Andrey V Savkin, Faizan Javed, and Alexey S Matveev. Optimal distributed blanket coverage self-deployment of mobile wireless sensor networks. *IEEE Communications Letters*, 16(6):949–951, 2012.

[Savkin, 2004] Andrey V Savkin. Coordinated collective motion of groups of autonomous mobile robots: analysis of Vicsek's model. *IEEE Transactions on Automatic Control*, 49(6):981–983, 2004.

[Semakova *et al.*, 2017] Anna A. Semakova, Kirill S. Ovchinnikov, and Alexey S. Matveev. Self-deployment of mobile robotic networks: an algorithm for decentralized sweep boundary coverage. *Robotica*, 35(9):1816–1844, 2017.

[Sutantyo *et al.*, 2010] Donny K Sutantyo, Serge Kernbach, Paul Levi, and Valentin A Nepomnyashchikh. Multi-robot searching algorithm using Lévy flight and

artificial potential field. In *2010 IEEE International Workshop on Safety Security and Rescue Robotics (SSRR)*, pages 1–6. IEEE, 2010.

[Yang, 2016] Xiaotian Yang. A decentralized algorithm for collision free navigation of multiple robots in search tasks. In *2016 35th Chinese Control Conference (CCC)*, pages 8096–8101, 2016.

[Yang, 2017a] Xiaotian Yang. A collision-free self-deployment of mobile robotic sensors for three-dimensional distributed blanket coverage control. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2017. in press.

[Yang, 2017b] Xiaotian Yang. A decentralized algorithm for multiple robots in collision-free search tasks. unpublished, 2017.

[Yang, 2017c] Xiaotian Yang. A decentralized collision-free search algorithm in 3D area for multiple robots. In *2017 36th Chinese Control Conference (CCC)*, pages 8107–8112, 2017.

[Yang, 2017d] Xiaotian Yang. Grid based 2D navigation by a decentralized robot system with collison avoidance. In *2017 36th Chinese Control Conference (CCC)*, pages 8473–8478, 2017.

[Yang, 2017e] Xiaotian Yang. Optimal distributed self-deployment of multiple mobile robotic sensors with collision avoidance for blanket coverage. unpublished, 2017.