Proceedings of the 2003 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
Las Vegas, Nevada · October 2003

Toward Finding an Universal Search Algorithm for Swarm Robots

Daniel J. Pack
Department of Electrical Engineering
United States Air Force Academy
USAF Academy, Colorado 80840-6236
USA
daniel.pack@usafa.edu

Barry E. Mullins
Department of Electrical Engineering
United States Air Force Academy
USAF Academy, Colorado 80840-6236
USA
barry.mullins@usafa.edu

***Abstract*— We present a novel cooperative search algorithm for distributed, independent swarm robots. The focus of this paper is three fold: (1) identify fundamental issues associated with searching an area by multiple robots; (2) design a metric to measure the cooperation among multiple robots to complete a search; and (3) propose an effective search algorithm for swarm robots.**

I. INTRODUCTION

Over the past decade, researchers in the robotics, biology, and AI communities have shown considerable interests in swarm robot research [1], [2], [3]. Such development is natural; compared to a single sophisticated robot, a set of swarm robots can complete a task faster, provide redundancy, and offer alternatives. In this paper we propose an effective swarm robot search algorithm, making a small step toward developing an universal swarm robot search algorithm.

II. PROBLEM DESCRIPTION

Our focus for the paper is to develop a cooperative search algorithm for multiple robots in a two dimensional search area. We initially discretize the search area by equally-distanced nodes, forming a grid. The search task then becomes obtaining a path for each robot such that the combined paths cover all the nodes on the grid. We assume that all swarm robots have the same capability; we attempt to develop a distributed algorithm rather than a centralized algorithm with a leader.

For a distributed search algorithm, we must answer the following fundamental questions: (a) Does the search algorithm for an individual robot vary based on the starting position or the current position of the robot in the search space? (b) What information must be transmitted amongst robots? (c) How to keep track of nodes that have been and have not been visited? (d) How to measure the cooperativeness of multiple robots?

Obviously, we want the answer to the initial question to be 'No' to make the algorithm universal, not environment dependant. Since each robot works as an independent entity in a distributed system, a current map that shows which node has been and which node has not been visited by any robot must be kept by each robot. Thus, each robot must broadcast its x,y position within the environment to all other robots. The record of the current map should then be updated by each robot using a proper data structure.

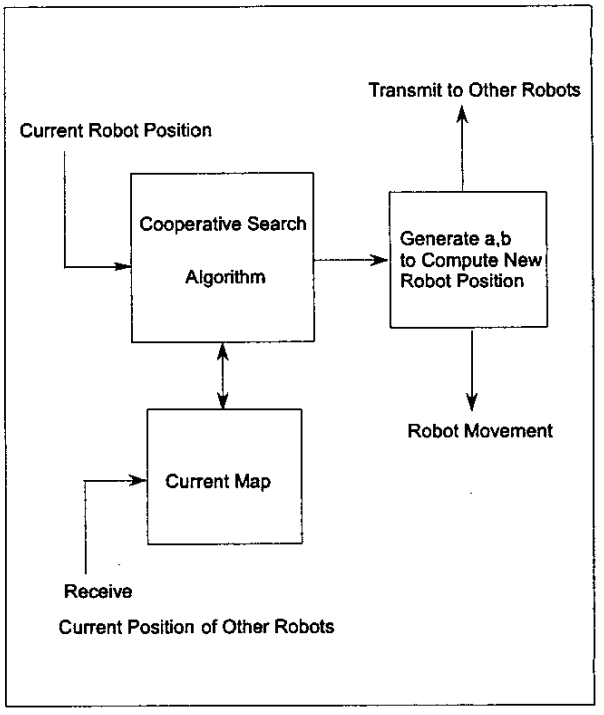


Fig. 1. A block diagram describing the input and output of the search algorithm

We transform the task of searching an area as computing variables a,b in the following equation, where $[x_i,y_i]^T$ is the current location of a robot and $[x_{i+1},y_{i+1}]^T$ is the location where the robot needs to go.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} a + \begin{bmatrix} 0 \\ 1 \end{bmatrix} b \quad (1)$$

Thus, our task then is creating an architecture shown in Figure 1 for each robot. To measure the level of cooperativeness among robots, we propose the following mechanism, cooperation score (cs).

$$cs = |\sum_{i=1}^{n} D^i_{ideal} - D^i_{actual}|/n \quad (2)$$

A small *cs* value represents high cooperation among robots while a large value *cs* indicates poor cooperation among robots. The subscript and the superscript of the sum operator denote robot 1 and robot n, where n is the number of robots in a group. For each robot, symbol D^i_{ideal} denotes the ideal minimum distance[1] that must be travelled by the *i*th robot, where the sum of ideal distance values equal to the ideal minimum distance that must be travelled by all robots to search the area. The ideal minimum distance is specified as (# of nodes in the search area - # robots)/ # of robots. If this ideal minimum distance is not available, in such cases of searching an unknown area, we assign zero to symbol D^i_{ideal}. Similarly, symbol D^i_{actual} equals the actual distance travelled by the *i*th robot, sum of symbols a and b in Equation (1). The dividing factor *n* normalizes the cooperation score for *n* robots. In short, the cooperative measurement encourages robots to travel an equal distance to complete the overall search; the cooperative measure is adversely affected whenever an overactive searcher or a passive searcher exists.

III. COMMUNICATION

A major concern is the scalability of proposed communication networks and associated protocols. Since we are dealing with a lossy environment and robots that roam, we are not guaranteed that all distributed robots have direct communication paths to all others. Depending on the transmission band, signal strength may vary by 30 dB within one meter. Furthermore, we assume the probability of communication failure for any one robot is the same (i.e., all robots have the same chance of failing).

Wireless Local Area Networks provide a unique capability to users in the form of dynamic network topologies. We selected an *ad hoc* peer-to-peer network setup to provide the most freedom to our protocol in the presence of the dynamic and uncertain environment.

Our proposed medium access control (MAC) is a variation of the now ubiquitous IEEE 802.11 wherein the essence of the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) scheme is maintained. However, acknowledgements are not generated by the receiving robots and retransmissions are not attempted, since the data being transmitted are perishable (guaranteed frame delivery is not a requirement of the swarm system).

Frames are comprised of the sending robot's MAC address (robot number), its current position (X and Y coordinates), and a frame check sequence comprised of a 32 bit cyclic redundancy code. Since each robot is broadcasting its frames, a destination address (robot number) or a frame ID number are not required. The source robot number and coordinates are encoded using 16 bits each. Therefore, each frame consists of 80 bits.

Given this frame structure and a 9600 bps data rate, we now analyze how often a robot can broadcast its data. The time to transmit one frame is 8.46ms. This includes the actual time to transmit (80 bits / 9600 bps = 8.33ms) and the DIFS time (128 us). Therefore, the best frame rate we could hope for is 118.19 (1/8.46ms) frames per second. This implies only 118 robots will be allowed to transmit within a one second window assuming each robot only transmits once in the window. Naturally, this is the best case scenario and not realistic. Robots will experience backoff and there will be periods when the medium is not being used (besides the DIFS time), but for now we keep the idealistic assumption for our preliminary analysis.

What is the upper limit on the speed of the robots under these conditions? The answer depends on how many nodes a robot is willing to traverse before it is ready to update its peers regarding where it has been. For now, we assume the robot transmits its location every time it enters a new cell. If 118 robots are in the swarm, each robot is limited to one transmission per second. This translates into a maximum robot speed of (inter-node distance/1 second). We can generalized this by stating the maximum robot speed as

$$S = \frac{D * R}{n * (b + R * 128us)} \tag{3}$$

where S is the maximum robot speed, D is the inter-node distance, R is the data rate in bits per second, *n* is the number of robots, and b is the number of bits in each frame. For example, if R is 9,600 bps, n is 118 robots, and b is 80 bits and assuming the distance between adjacent nodes is one unit; this yields a maximum speed of approximately one distance unit per second, which makes sense given the assumptions we made.

Now let the number of history cells transmitted vary. That is, instead of requiring each robot to transmit as soon as it enters a new node, allow the robot to accumulate a history of nodes and transmit this list after every Q nodes are visited. How is the number of robots in the swarm now affected by these parameters? Assuming all robots travel at the same speed, the equation for the number of robots now is

$$n = \frac{D * Q * R}{S * ((80 + (Q - 1) * 32) + R * 128us)} \tag{4}$$

Our goal is to investigate the impact of increasing the number of robots exchanging frames at various data rates and varying levels of history via simulation and building a hardware swarm system. The simulations will naturally account for the lossy channel and frame collisions thereby rendering a realistic model of how many robots can communicate effectively at a given transmission data rate and robot speed.

[1]The ideal minimum distance occurs when the initial robot locations are such that robots can cover a search area without crisscrossing the paths of other robots.

IV. Search Algorithm

We describe our search algorithm using several search scenarios. We start with the scenario shown in Figure 2.

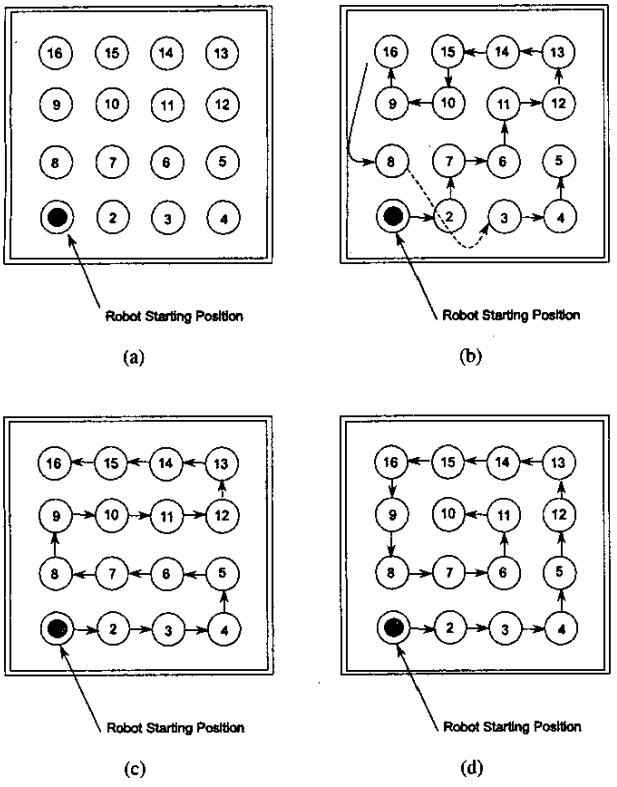


Fig. 2. Searching a rectangular area with a single robot

A single robot is placed at a corner of a rectangular search area with 16 nodes, as shown in frame (a). We found that the robot chooses a shortest total distance (15 units) to visit all nodes if its search algorithm contains the following rules: (1) Given a current robot position, identify the closest (in Euclidean distance sense) node; (2) If multiple nodes are equally close to the current node, pick the one that lies along the same direction the robot just travelled; and (3) If multiple nodes satisfy conditions 1 and 2, pick one randomly.

Frame (b) shows a scenario where only rule 1 is used (18 distance units travelled). Frame (c) shows the paths generated as a result of using a sweeping technique[2]; Frame (d) shows the robot paths generated when rules 1, 2, and 3 are used. Frames (c) and (d) show that the two methods result in the same number of distance units travelled by the robot. The trouble with the sweeping strategy is that if more than one robot is involved in searching, the search area *a priori* must be partitioned and assigned to robots, a task not feasible for distributed, independent robots.

We now expand our example to two search robots. Suppose that the two robots start at locations shown in Figure 3 frame (a). We assume that *robot A* moves initially followed by *robot B*. Frame (b) shows one of possible paths generated by the robots, when only rules 1 and 3 are used. Note that a solid line indicates the path of *robot A* while a dotted line depicts the path generated by *robot B*. Assuming that robots can only turn at 90 degree angles,[3], the length of the robot paths for the case shown in frame (b) is 19 units.

From the examples shown in Figure 3, we see that the three rules we proposed for one robot search scenario are not enough, causing the robots to crisscross unnecessarily each other's path to cover the search area. To remedy the situation, we add one more rule, shown as the second rule in the following rule list.

1) Given a current node position, identify the closest (in Euclidean distance sense) node as the next node to visit.
2) If multiple nodes are equally close to the current node, select the one that lies farthest from the node location of the other robot.
3) If multiple nodes are equally far from the other robot, pick the one that lies along the same direction the robot just travelled.
4) If multiple nodes satisfy conditions 1, 2, and 3, pick one randomly.

If rules 1 and 3 are used only, the robots generate paths shown in frame (c). The total traversed path is 17 units. Frame (d) shows a case where rules 1, 2, and 4 are used. The generated path length corresponds to the shortest distance, 14 units. Frame (e) illustrates the case when all four rules are used. The total path length generated again matches with the minimum distance of 14 units. From this example, one may obtain a false impression that rule 3 is redundant and does not play any important role. Figure 4 shows a scenario where rule 3 improves the overall search scheme for two robots. Frame (a) shows the paths obtained using all rules except rule 3. The total number of distance units for this frame is 16 (we count two units of distance if a robot must move diagonally.). When rule 3 is added, paths in frame (b) are generated with the total distance unit of 14, the ideal total distance.

The obvious question is how to expand these rules for a case with n robots. No alteration is necessary for rules 1 and 3. Rules 2 and 4 do not change for cases when $n \geq 2$ if we can identify the other robot as the one that is closest to the current robot. By doing so, we are discouraging

[2]A sweeping technique seeks to visit all nodes by covering all nodes in one direction until all options are exhausted, similar to well known breadth search technique.

[3]This requirement is solely for the purpose of our discussion.

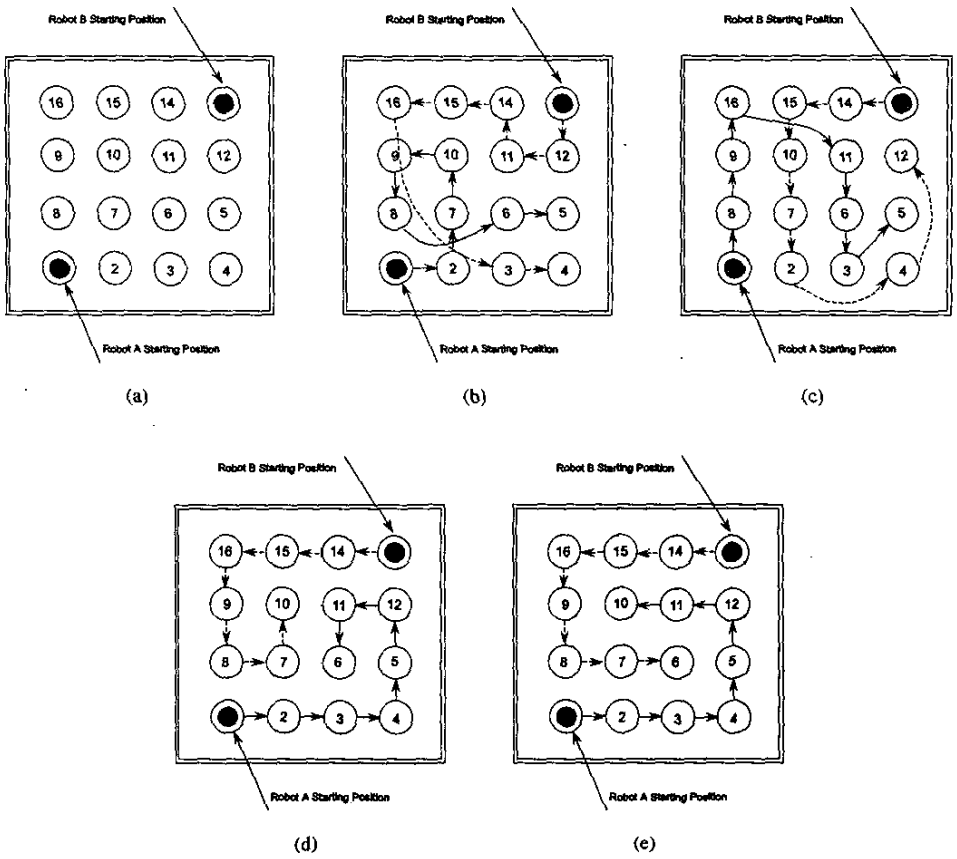


Fig. 3. Searching a rectangular area with two robots

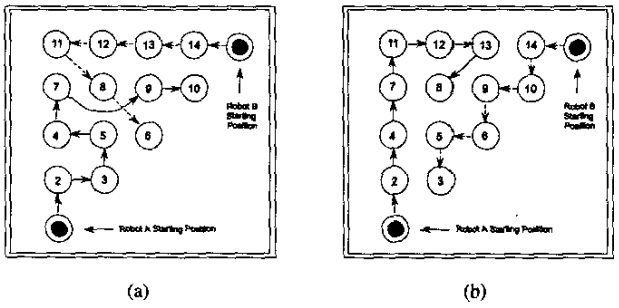


Fig. 4. Comparison of using all four rules versus rules 1, 2 and 4 only

robots from clustering together; such effort enhances the effective coverage of the search area by the participating robots.

V. Experimental Results and Discussion

Our goal is to quantitatively identify the role of each rule we proposed in forcing the robots to cooperate. We start with two robots in a 4 by 4 grid search space. Once we illustrate the workings of the search algorithm using two robots, we expand the algorithm to n robots working together. We show simulation results for the $n = 100$ case.

For our discussion, we call the two robots *robot A* and *robot B*. Since each robot can start from any one of the 16 locations, there are 240 different combinations of starting nodes for *robot A* and *robot B*. We ran four different experiments with two robots starting at 240 different combinations of initial nodes: (1) search with rule 4 only; (2) search with rules 1 and 4; (3) search with rules 1, 2 and 4; and (4) search with rules 1, 2, 3, and 4.

For each experiment, we repeated the process ten times to accommodate the randomness embedded in the algorithm via rule 4 and the starting locations of the robots. Using the experimental results, we computed the average distance travelled by each robot, the average turns made by each robot, and the average cooperation score, shown in Equation (2). Table I shows the experimental results for the four experiments.

The far left column of the table shows the names of

TABLE I

EXPERIMENTAL RESULTS FOR SEARCHING 4 BY 4 SPACE WITH TWO ROBOTS:(1)EXPERIMENT ONE(RULE 4 ONLY), (2) EXPERIMENT TWO (RULES 1 AND 4), (3) EXPERIMENT THREE (RULES 1, 2, AND 4), AND (4) EXPERIMENT FOUR (ALL FOUR RULES).

measured/computed items	experiment 1	experiment 2	experiment 3	experiment 4
robot A ad	13.41(3218)	7.97(1913.2)	7.17(1721)	7.16(1718.6)
robot B ad	13.52(3245.5)	8.42(2020.6)	7.28(1748)	7.25(1740.6)
combined average turn	26.22(6292.6)	14.89(3572.6)	13.56(3254.9)	13.13(3151.6)
average cooperative score	6.465(1551.55)	1.195(286.9)	0.227(54.5)	0.207(49.6)

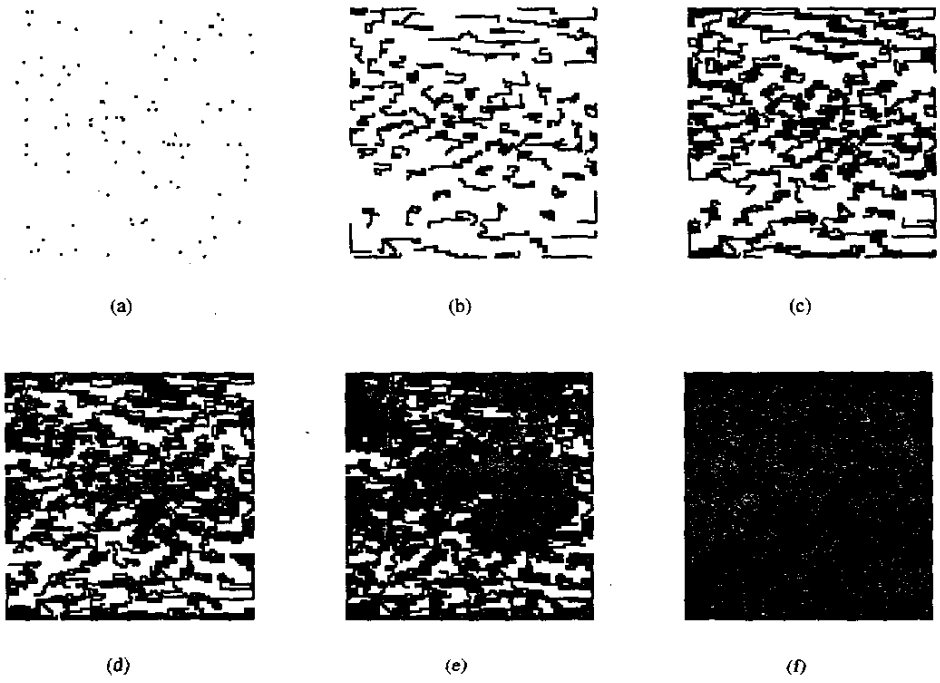


Fig. 5. Progression of searching 100 by 100 area by 100 robots. Frame (a) shows the initial locations of 100 robots. The initial locations, the row and the column numbers, were randomly selected using a random number generator with uniform distribution between 1 and 100. Frame (b) shows the search area after each robot has made 20 moves. Notice that each robot tries to minimize the number of turns by moving in a straight line in the horizontal direction. The particular direction is arbitrary since by default each robot's initial direction was horizontal. Notice clusters of black areas are starting form. Such event matches with our intuition that once an area begins to be crowded by visited spots, robots must make multiple turns to identify locations that have not visited, resulting the quick coverage of the areas. Frames (c), (d), and show the search area after each robot has made 40, 60, and 80 moves. The last frame shows the search area just before the last remaining location, (row=98, column=32), has been visited.

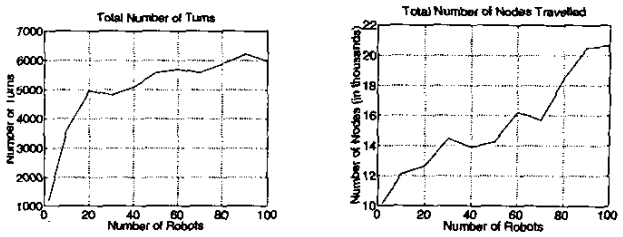
the measured and computed items for the experiment. Names *robot A ad* and *robot B ad* represent the average distance travelled by *robot A* and *robot B*, respectively. The distance is the average value of 2400 different cases (240 different starting locations and 10 separate experiments). The number within the parenthesis in the table is the average sum distance of the 10 separate experiments. Similarly, the *combined average turn* represents the average of turns made by two robots to cover the search area. The numbers within the parenthesis were obtained using the same method as in the case of the average distance value.

We see clearly how the addition of each rule improves the search task in all measured values. For example, the addition of rule 1 considerably improved the search effectiveness with approximately a 41% reduction in *robot A*'s average distance, a 43% reduction on the average number of turns, and an 82% reduction in the cooperation score. The addition of rules 2 and 3 also improved the search substantially, although not as much as rule 1. The addition of rule 2 improved the average distance travelled by *robot A* by approximately 10%, the average distance travelled by robot B by approximately 14%, the average number of turns by approximately 9%, and the cooperation score by roughly 81%. The usage of all four rules improved the overall performance once again: *the robot A* average distance is reduced by approximately 0.1%, the *robot B*

TABLE II

EXPERIMENTAL RESULTS FOR SEARCHING 100 BY 100 SPACE WITH 100 ROBOTS

Run #	Total Distance	Total Turns	CS
1	15260	5782	53.6
2	23322	6159	134.22
3	16797	6212	68.97
4	20657	5993	107.57
5	18756	5948	88.56
6	15394	5796	54.94
7	17298	5809	73.98
8	16375	6200	64.75
9	16006	5876	61.06
10	19199	5523	92.99



(a) (b)

Fig. 6. The two graphs shows the change on the total number of turns (a) and the total distance travelled (b) by robots as the number of robots vary.

average distance value is reduced by approximately 0.4%, the average number of turns has improved by approximately 3.4%, and the cooperation score has improved by approximately 9%. The total average distance travelled by each robot, when all four rules are used, approaches the ideal number, seven, for the particular environment, which shows the effectiveness of the search algorithm.

Another important observation is that the robot who started initially, (*robot A*) for our case, always travelled a shorter distance and had a fewer number of turns to make. The results match with our intuition; the initial robot that occupies a desirable node, in the distance sense, forces the other one to look for a less desirable node.

We now show an experimental run with 100 robots in a 100 by 100 grid environment. Each robot uses the four rules to move about the search space. The goal of the robots, again, is to cover all search areas while minimizing the total distance travelled and the total number of turns made by robots. Figure 5 illustrates the search activity of the 100 robots in an example case. Black surfaces represent locations that have been visited by robots. The initial robot locations were generated using a random number generator function. The frames show snapshots of the search area after 20, 40, 60, etc. moves are made by each robot until the entire area has been covered. We performed ten additional experiments shown in Table II. The table shows the total number of distance units travelled (column 2) and the total number of turns made (column 3) by 100 robots. The last column of the table contains the corresponding cooperation score for the each run. Ideally, each robot should travel 99 locations before covering the entire search area. The average distance travelled and the number of turns made by each robot in the ten experiments are 179.06 units and 59.30, respectively. The average cooperative score is 80.06, meaning that on average each robot made 80 additional moves to complete the search. Figure 6 shows how the total number of turns made and the total number of distance travelled by robots vary as we change the number of robots involved in the search of the 100 x 100 area. It shows that the number of turns has a steep increase at the start and a gradual increase as more robots are added. The distance travelled, on the other hand, increased steadily up to 100 robots. Our experiment demonstrates the feasibility of the search algorithm for a high number of participants in a somewhat large search area. The experiment shows a good argument for the scalability of the proposed algorithm.

VI. CONCLUSION

In this paper, we presented a novel search algorithm for swarm robots. The algorithm is based on four simple rules that govern movements of each robot based on geometrical constraints. The proposed algorithm does not require any central control unit, making it suitable for distributed control system with minimal interactions with other robots. To measure the effectiveness of the algorithm, we introduced a new measure to evaluate the robot cooperation in a search task. We also presented the technique used by robots to communicate among themselves essential information on their positions within search areas. We demonstrate the effectiveness of our algorithm using two cases: (1) two robots in a 4 by 4 space and (2) 100 robots in a 100 by 100 space.

VII. REFERENCES

[1] E Bonabeau, M. Dorogo and G. Theraulaz, *Swarm Intelligence: From Natural to Artifcial Systems,* Oxford University Press, 1999.

[2] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings of 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA 2000.

[3] J. Bay, "Design of the army-ant cooperative lifting robot", *IEEE Robotics and Automation Magazine*, vol. 2, 1995, pp. 36-43.