Multimodal Central Place Foraging

Nicholas Dolan-Stern, Kevin Scrivnor, and Jason T. Isaacs

Abstract— **Implementations of central place foraging using multi-robot systems need efficient mechanisms for searching for resources and gathering them in a central home-nest location. We propose an approach that partitions the search space and assigns agents multiple behavioral roles inspired by honey bee colonies as a way of organizing the search and gather operation. Through simulation we demonstrate that this approach minimizes spatio-temporal congestion that results from many robots sharing a common space near the home-nest. We compare our role based algorithm to the Distributed Deterministic Spiral Search Algorithm (DDSA) in a high fidelity simulation environment built using ROS and Gazebo.**

I. Introduction

As the cost of robotic components such as sensors, actuators, memory, computer processors, and communication modules decrease in price, the complexity of robotic applications has continued to increase. This shift has caused researchers to be able to shift from a single robot being controlled by many operators to many robots being controlled by a single operator. In addition the capacity for completely autonomous behavior in robotic systems has greatly increased. The research area of swarm robotics takes this concept and distributes it among many agents, using simple autonomous individual behaviors to create robust and complex group behaviors [1], [2]. In many applications such as planetary exploration [3], manufacturing [4], and land mine detection [5], a large swarm of inexpensive robots can outperform a single expensive robot with the additional benefits of robustness, flexibility, and scalability [6].

The National Aeronautics and Space Administration (NASA) is particularly interested in using swarms of rovers to explore other planets such as Mars in search of ice [3]. If enough ice is gathered into a central processing area it can be separated into oxygen and hydrogen to be used for fuel. This mission equates to the problem of central place foraging (CPF).

In CPF resources are distributed around a resource depository in the center of the environment that acts as a "home-nest". Agents are assigned to find each of the resources, gather them, and bring them back to the home-nest. Traditionally, researchers have employed *homogeneous foraging* where each robot searches for resources and independently transports them to the home-nest [7]–[9]. Multiple agents having to perform independent tasks within this shared space leads to many problems with congestion [10]. The traffic that occurs near the home-nest as well as collisions between the incoming and outgoing agents represent some of the largest bottlenecks within this problem domain. In fact, early work in multi-agent foraging showed that performance actually decreased with swarm size due to congestion [11]. In order to forage efficiently, agents need to share what they have found with others, let the other agents know what has already been gathered, and effectively avoid each other. To allow scaling to large swarms of agents, communication needs to be minimized, agent behavior needs to be simple, and access to the home-nest needs to be managed.

Our approach to central place foraging is heavily inspired by the way bee colonies divide the task of foraging for nectar into multiple behavioral roles. Similarly our algorithm utilizes similar roles in order organize search and gather behavior. Bees do not simply randomly happen upon nectar and bring it back to the nest. Instead they have developed a surprisingly complex system of searching and gathering [12]. Scout bees are first sent out to survey the area around the hive, and upon completing their route, they are able to report back on the concentration of nectar found by means of a "waggle dance". The collector bees then are allocated appropriately to gather nectar in the most concentrated area [13], [14]. Surveying the environment before beginning collection allows for much more efficient and informed gathering routing.

The main contribution of this work is an algorithm which utilizes multiple roles for organizing search and gather behavior, dividing the search space between the agents so as to reduce collisions, and sharing search and gathering progress amongst the agents. We compare our results to an implementation of the Distributed Deterministic Spiral Search Algorithm (DDSA) [9]. We chose the DDSA as the baseline for comparison due to its simple implementation, consistent (deterministic) results, and its improved performance over biologically inspired random algorithms such as the Central Place Foraging Algorithm (CPFA) [8]. Similar to DDSA, we use a spiral search pattern due to its complete coverage, scalability, and minimal overlap of sensor coverage [9], [15], [16]. We deviate from DDSA in that we do not collect any resources until the search space has been completely covered and the location of all resources on the search path are determined. Once the search area is covered, we assign agents to collect resources closest to them in areas separate from one another.

By decoupling the foraging task into two distinct categories of search and collection, we are able to retain the benefits of a deterministic pattern while using information gathered in the search phase to address the problem of

N. Dolan-Stern, K. Scrivnor, J.T. Isaacs are with the Department of Computer Science, California State University, Channel Islands, nicholas.dolan-stern630@myci.csuci.edu, kevin.scrivnor988@csuci.edu, and jason.isaacs@csuci.edu

congestion near the home-nest while collecting resources. While we utilize a deterministic spiral search pattern for our work, it is conceivable that by decoupling the foraging task other search algorithms could be deployed [17]. The collection portion of the foraging task can be thought of as a vehicle routing problem with simultaneous pickup and delivery where each rover has a carrying capacity of one [18]. If rovers were capable of carrying more resources, potential improvements could be gained by pairing our approach with the one provided by Ai and Kachitvichyanukul [18]. An additional subproblem within CPF is having multiple collection depots, which drastically changes the congestion problem [19].

The remainder of this paper is organized as follows. The problem formulation is presented in Section 2, followed in Section 3 by a discussion of the proposed multimodal distributed foraging algorithm. Section 4 describes the simulation tests used to compare the results of the proposed algorithm with that of the DDSA algorithm. The results of these tests are discussed in Section 5. Section 6 provides some final conclusions and directions for future work.

II. Problem Statement

The objective of any central place foraging (CPF) algorithm is to minimize the time required for a group of robots to search an area and deliver every resource in the environment back to a home-nest. We assume that the area of interest is a square region in the $2D$ plane with an area $\mathcal{A} = \mathcal{L}^2$, where $\mathcal{L}$ is the length of the region. Located at the center of the region is the home-nest which also acts as the origin of a local coordinate system.While we assume a square environment, adapting to different shapes and layouts would be as simple as generating new partitions and way points for the agents to follow on their search path. It is even conceivable to dynamically generate the partitions and paths for the agents to follow.

The objectives of CPF require coordinated pathing of a group of N differential drive skid-steer robots. The state of a skid-steer vehicle x can be represented by the triplet $(x_d, y_d, \theta) \in SE(2)$, where $(x_d, y_d) \in \mathbb{R}^2$ describe the position of the vehicle center of gravity and $\theta \in \mathbb{S}^1$ represents the orientation. Vehicle control input $u \in \mathbb{R}^2$ is modeled as $(v, \omega) \in \mathbb{R}^2$, where v is the forward velocity of the vehicle center of gravity and ω is the rate of change in vehicle orientation. Motion of the vehicle evolves according to the following kinematic equation:

$$\dot{x} = \begin{bmatrix} \dot{x_d} \\ \dot{y_d} \\ \dot{\theta} \end{bmatrix} = f(x,\ u) = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix}. \tag{1}$$

For more details on skid-steering kinematics, the authors refer you to [20].

We then assume that a control algorithm exists that, in the absence of obstacles, will guarantee rover traversal to the goal waypoint [20]. This controller is modeled by:

$$u = g(x_c,\ x_g) \tag{2}$$

where $x_c \in SE(2)$ describes the rover's current pose, and $x_g \in SE(2)$ describes the rover's goal pose. During the deterministic search phase it is necessary to follow paths with sharp corners at each waypoint to ensure that the entire search area is covered. To guarantee that each rover follows the desired search path, a rotation pivot controller is executed at each waypoint to orient the robot in the direction of the desired next waypoint before executing the waypoint controller (2).

Each robot is equipped with a sensing device that is capable of detecting and locating individual resources when they come within some limited field of view, $\mathcal{F}$. Additionally, each robot has the capability to collect and carry exactly one resource at a time. Upon returning to the home-nest, the robot can deposit the collected resource and return to the field to collect other resources.

Obstacle avoidance is a critical capability of any robot operating in large groups. Robot to robot encounters most commonly occur when multiple agents are trying to pick up resources from nearby locations, as well as when several agents are trying to drop a resource off near the single home-nest. Obstacles are detected by a set of range sensors mounted in the front of each agent. We assume that each sensor has a maximum range, and the collection of sensors can determine if the object is located in the front-right, front-center, or front-left of the robot. Upon one or more of these rays being interrupted, an Obstacle Detection event is generated. Upon handling of the event, the agent saves the position that caused the event onto a stack, then attempts to move to an alternate nearby location to the side of the detected object. This process continues until Obstacle Detection events are no longer being received. Once the agent is able to attempt to resume Collection, the agent's goal location is set to the oldest position from its saved waypoint stack.

III. Multimodal Distributed Foraging Algorithm

A. Overview

In our approach to the CPF, Multimodal Distributed Foraging Algorithm (MMDF), agents take on the role of either a *Searcher* or a *Collector*. Each agent begins as a Searcher, but upon completing its search path, transitions to *Collector*. By searching before collecting, agents are able to maximize the number of targets found while minimizing the distance traveled during this phase. The agents are then able to use the data gathered during the searching phase to inform the collection phase. We use this information to assign the closest available target to an agent while at the same time keeping the agents apart from one another, preventing collisions. This data could also be used for a variety of other purposes including fitting a pattern to the target distribution as well as more complex path planning with additional pickup capacity.

B. Target Handling

The state of each target in the environment is shared amongst each agent in the system. Each time the target

sensor detects one or more targets within its field of view, it processes the detection and publishes various messages when a target changes its state. Each target is initialized to be in the *Unknown* state, which means that its position cannot yet be determined. Regardless of the role of the agent, all targets that have not been detected previously need to be reported as *Detected*, which means sending a message consisting of the target's unique identifier as well as the position at which the target was seen. Upon receipt of the detection message each agent updates their local *Target State* list so as to have that target's position stored and state changed to *Detected*. The goal of each agent in *Search* mode is to detect as many of the *Unknown* targets as possible, and *Search* agents do not have the ability to transition a target from any other state than from *Unknown* to *Discovered*. For *Collector* agents the action that needs to be taken depends on whether the *Collector* is carrying a target or has claimed a target.

Figure 1 shows the various states and transitions for each target and Algorithm 1 provides a more detailed description of the target processing algorithm:

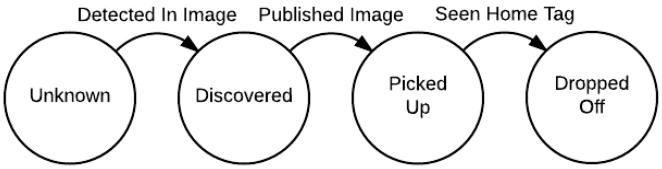

Fig. 1: *State Diagram for targets.*

Algorithm 1 *Target Handler*: Determines what actions to take upon seeing target or home tags.

```
for <target in targets_in_view> do
    if isUnknownTarget(target) then
        reportDetected(target);
    if role == COLLECTOR then
        if capacity == CARRYING then
            if isHomeTag(target) then
                isHomeSeen = true;
                dropOff(claimed);
        else
            if capacity == CLAIMED then
                if claimed! = target then
                    unclaim(claimed);
            isTargetSeen = true;
            pickup(target);
```

C. Searcher

The goal of agents in the *Searcher* role is to determine the location of as many targets as possible by driving over them in such a way that every target falls within the sensor's field of view, $\mathcal{F}$. Each *Searcher* follows a set of way-points guaranteed to have complete coverage of the environment by following a spiral pattern inspired by the DDSA algorithm. As agents search the area, they broadcast the location and a unique identifier for each target that comes within their sensor's vision. Upon receiving these broadcasts, each agent stores a local copy of a data structure that contains all known outstanding targets.

The *Searchers* first seek to divide the area of interest into regions of equal area and assign each agent to search a particular region in isolation. In this way, agents are less likely to physically encounter each other even in imperfect navigation conditions. Let N be the number of searching agents in a region of area $\mathcal{A}$ with a side of length, $\mathcal{L}$. We define two new variables,

$$F = \frac{\mathcal{L}}{2}$$
$$L = \frac{\mathcal{F}}{2}.$$

We then may calculate the length of each partition side with the function:

$$l(x) = \sqrt{(L \cdot 2)^2 + l(k-1)^2}$$

where $k \in \{1, 2, ..., N\}$, and the distance between each partition as:

$$h(k) = \frac{l(k) - l(k-1)}{2}$$

where $k \in \{1, 2, ..., N\}$ & $h(0) = 0$, $h(N+1) = 0$.

With these functions, we may now define a function to determine the total distance to the partition with:

$$H(k) = \sum_{i=k}^{N} h(i)$$

Figure 2 displays a possible partitioning of the arena where $N = 3$.

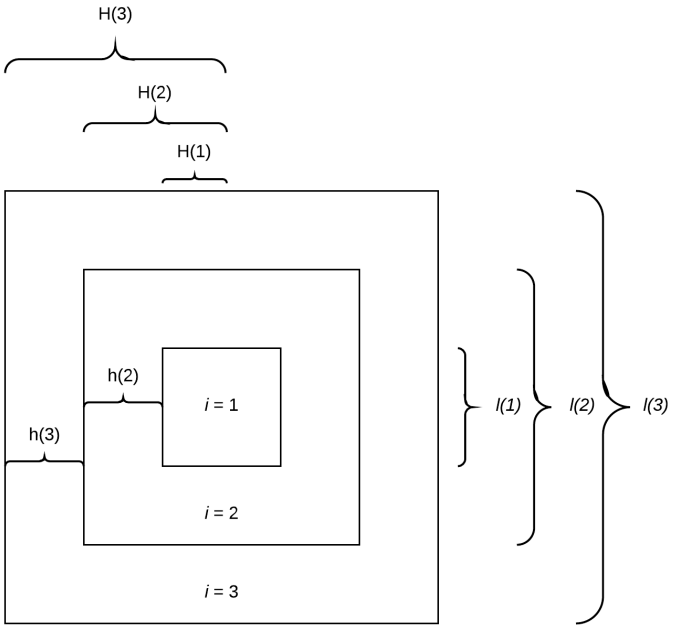

Fig. 2: *Example of a partition scheme for* $N = 3$.

Setting the origin to the center of the arena, we can now calculate x, y waypoint coordinates with the following:

$$W_{i,n} = \begin{cases} n \equiv 0(4) & (L - H(i+1) - nF), \\ & (L - H(i+1) - (n+1)F) \\ n \equiv 1(4) & (-L + H(i+1) + (n+1)F), \\ & (L - H(i+1) - (n+1)F) \\ n \equiv 2(4) & (-L + H(i+1) + (n+1)F), \\ & (-L + H(i+1) + (n+1)F) \\ n \equiv 3(4) & (L - H(i+1) - (n+1)F), \\ & (-L + H(i+1) + (n+1)F) \end{cases}$$

where i is the partition number, $i \in \{1, 2, ..., N\}$ and n is the waypoint index. There is a special case when $n = 0$, the x value is $(L - H(i+1) - F)$. Waypoints are generated counter-clockwise starting from the upper left hand corner. Figure 3 shows an example of this for partition $i = 1$.

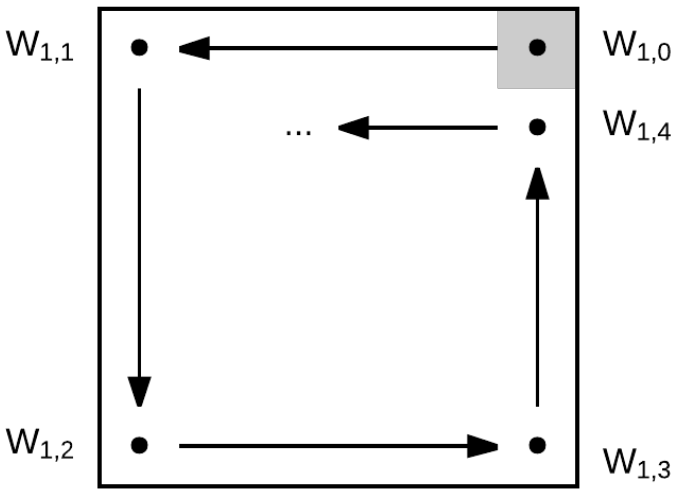

Fig. 3: *Example of waypoints on* $i = 1$.

The spiral paths described above retain the minimum search time property of the DDSA [9], but by partitioning the search region, the agents are no longer searching on adjacent paths. For each agent the search path is stored on a stack, and as each goal location is reached, it is popped from the stack to allow the agent to move to the next waypoint. Unlike DDSA, in which discovered targets are immediately brought to home, when a MMDF *Searcher* encounters a target, it does not take it back to home, it broadcasts its location and identifier and continues on its search path. After an agent has finished searching their region, which means that its set of waypoints has been exhausted, they switch modes to that of a *Collector*.

D. Collector

Now that the locations of the targets have been determined by the *Searcher* agents, the Collectors return to each of the target locations in order to retrieve it and bring it to the home-nest. Each *Collector* follows a state machine, described in Algorithm 2, that carries it through the travel, pickup, and return process. This process continues until all known targets have been collected.

Algorithm 2 *Collector State Machine*: Determines which goal and state to transition to.

```
1:  if obstacle_encountered == true then
2:      goal = getAlternativeLocation();
3:      previous_state = current_state;
4:      obstacle_encountered = false;
5:      current_state = OBSTACLE;
6:  if current_state == GO_TO_TARGET then
7:      if isTargetSeen then
8:          goal = getHomeLocation();
9:          current_state = GO_TO_HOME;
10:     else if isGoalReached() then
11:         goal = getRandomNearbyGoal();
12: else if current_state == GO_TO_HOME then
13:     if isHomeSeen() then
14:         goal = getNextTargetGoal();
15:         current_state = GO_TO_TARGET;
16:     else if isGoalReached() then
17:         goal = getRandomNearbyGoal();
18: else if current_state == OBSTACLE then
19:     if isGoalReached() then
20:         goal = getCurrentLocation();
21:         current_state = previous_state;
22: goTowardsGoal();
```

When a *Collector* agent is in need of a target, it will search its list of *Target States* for the closest *Detected* target and then claim it. Once a target is *Claimed* by a *Collector*, a sector of the search space in the direction of the claimed target is off limits to other agents. In Figure 4, an agent has claimed the target marked by a c in the large cluster of targets in the upper right of the figure, making the sector formed by the angle β temporarily inaccessible to other agents. This is an important step in the MMDF algorithm as it aides in reducing congestion on the traffic lanes between large clusters of targets and the home-nest, prevents two agents from going after the same target, and reducing collisions between agents due to low proximity.

On route to the expected location of the claimed target, any *Unknown* or *Discovered* target encountered will be picked up instead. If the *Collector* has arrived at the expected location of a target and nothing has been detected, the *Collector* will perform a randomized search in an attempt to find the target. Once a target has been located the *Collector* will then broadcast a Pick-Up message containing the target's unique id. Each receiving agent updates their target states so that no additional attempts to collect that target are made. Once this message is published the *Collector* agent now travels back to the home-nest.

Once the *Collector* has reached the location of the home-nest, the *Collector* looks for visual cues of the home-nest. If the home-nest is not visible, the *Collector* will perform

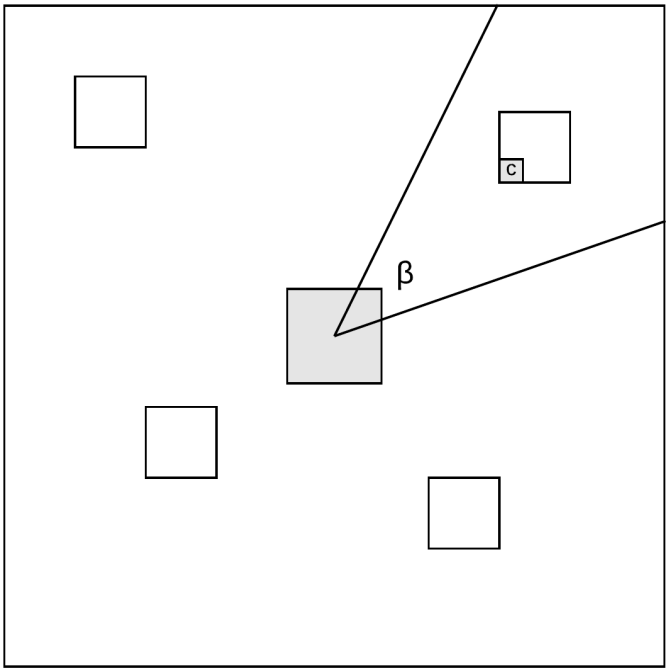

Fig. 4: *Sector of angle β has been claimed by an agent therefore other agents must claim targets outside this sector.*

a randomized search about that location. Once home-nest is visible, the *Collector* drops off the target and broadcasts a message signaling to the other agents that this target has been collected. The sector claimed by that agent is released when the corresponding *Dropoff* message has been received by the other agents. The collection process for the agent dropping off the target is then repeated.

IV. Simulation Setup

The proposed MMDF algorithm was developed and validated on the NASA Swarmathon 2017 [3] swarm robotics platform. Our algorithms are implemented in C++ using the Robot Operating System (ROS) [21] framework and the Gazebo plug-in for the Graphical User Interface (GUI).

A. Simulation Setup

The search region is a 15 by 15 meters square with walls at the boundaries. There were 256 targets in this environment that were distributed in two ways, uniform and clustered. If the environment is in the uniform configuration, the targets are randomly distributed around the map in such a way that every location is equally likely to contain a target. If the environment is in the clustered configuration, every target is contained within four large square clusters of targets, each containing 64 targets. For these simulation experiments $N = 3$ agents were used and all agents were initialized to be *Searchers*.

The ROS environment allows for easy message passing of any type of data between agents. Messages are able to be sent by an agent through publishing on a specific "topic", similar to a subject line of an email. Messages are received by subscribers that constantly listen for a message to be published on one of these topics. Once a message is received it is parsed and able to be stored within an agent's state. This is the primary means of an agent to communicate with any other agent. Any type of communication activity will therefore need to pass through this messaging system.

In this simulation environment, targets are represented by AprilTag patches that are placed on the ground and the target detection sensor is a webcam and AprilTag detection software. All agents share a list of target states. Each target's state consists of its unique identifier, the location of the target (initialized to unknown), and whether it has been seen, picked up, or has been dropped off at home. Target state is able to be changed by any agent seeing a previously undiscovered target, or picking one up, or dropping it off at home. When any of these events occur the state of that target is updated by all of the agents. Targets are detected or "picked up" when the agent's camera senses an AprilTag on the ground. The image is processed and the target's unique identifier is received by the agent.

V. Results

A total of 25 simulations were performed for both a uniform target distribution and a clustered distribution. For each run the DDSA and MMDF algorithms were each applied to same target distribution. The time at which each target was delivered to the home-nest was recorded.

Table I shows a comparison of the percentage of total targets collected after $15, 30, 45,$ and 60 minutes for both uniform and clustered target distributions. At the 15 minute point DDSA is clearly collecting a high percentage of the targets than MMDF as expected since MMDF searches the entire region before collecting any targets. By the 30 minute point MMDF has collected as many targets as DDSA, and by the 60 minute point MMDF has collected 17.4% and 11.2% more targets for the uniform and clustered distributions respectively.

Figures 5 and 6 contain the search trajectories for each algorithm for the first 400 seconds of the operation. In Figure 5, it can be seen that for MMDF hardly any resources have been collected, but a complete search of the region is nearly complete. In Figure 6, it can be seen that for DDSA several resources have been collected near the home-nest, but only a small portion of the region has been searched.

In Figures 7 and 8 it is clear to see that MMDF initially pays a time penalty, as no resources are collected while agents are searching, but quickly after the initial search phase is completed the MMDF overtakes the DDSA algorithm and collects resources at a faster rate. The main explanation for this is that after the initial search phase all resource locations are known, so collectors can optimize their collection sequence to reduce congestion on the lanes to and from the home-nest.

VI. Conclusion

We have proposed a multimodal foraging algorithm, separating the tasks of searching and collecting. One of the benefits of our approach is being able to improve each mode

		% Collected			
		15 minutes	30 minutes	45 minutes	60 minutes
Uniform	MMDF	23.1 ± 0.9	51.8 ± 1.9	76.0 ± 2.0	96.3 ± 2.1
	DDSA	26.1 ± 0.6	45.8 ± 0.6	63.7 ± 0.6	78.9 ± 0.7
Clustered	MMDF	19.0 ± 1.1	49.6 ± 2.3	75.7 ± 2.9	95.2 ± 2.3
	DDSA	25.5 ± 2.4	48.5 ± 3.7	67.4 ± 3.7	84.0 ± 3.6

TABLE I: *Percentage of Total Targets Collected after 15, 30, 45, and 60 minutes for both Clustered and Uniform target distributions. Each entry shown with corresponding* 95% *confidence interval.*

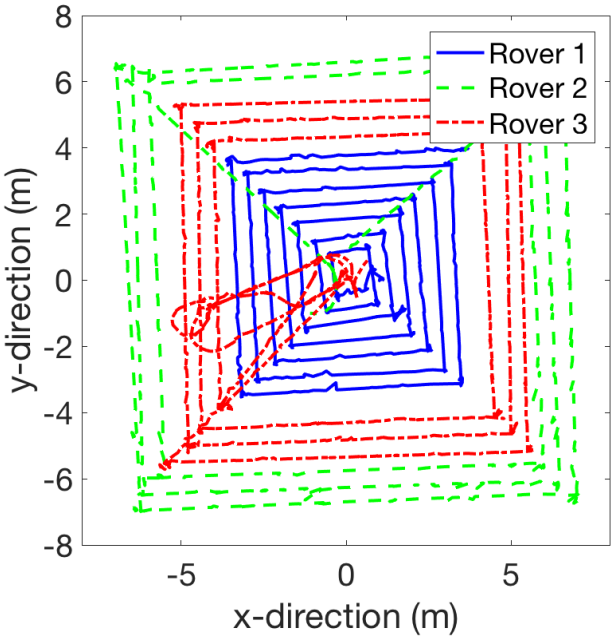

Fig. 5: *Search Trajectory of MMDF for first* 400 *seconds of search.*

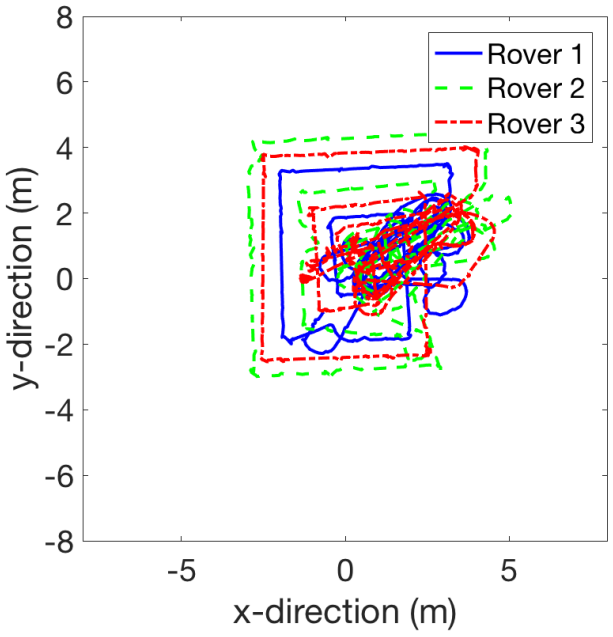

Fig. 6: *Search Trajectory of DDSA for first* 400 *seconds of search.*

independently of the others. The search algorithm covers the environment as fast as possible with minimal oversampling and locates close to all the resources, even under imperfect sensor readings. In the future we will need to continue to optimize search mode to ensure more complete detection of resources in presence of realistic error. The collection algorithm, knowing the location of most of the resources, collects them in near linear time, scaling with the distance of the resources from home. Optimizing collection with location of all targets known to prevent agent collision is also of high priority. We are also interested in experimenting with increased resource capacity of agents, as well as multiple collection depots.

In addition we have proposed a system of locking directions for agents to collect in. This method reduces the number of collisions by having the agents going to and from home in different directions. Despite this progress, one of the main factors of speed reductions continues to be obstacle avoidance. Agents continue to occasionally collide, although this is now less frequent.

We compare the results to the DDSA algorithm proposed by Fricke and show a significant increase in performance in both uniform and clustered resource configurations. While we do initially pay a performance penalty by not collecting while searching, our rate of pickup after search allows us to quickly catch up to and surpass the performance of DDSA.

References

[1] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[2] A. Reina, T. Bose, V. Trianni, and J. A. Marshall, "Effects of spatiality on value-sensitive decisions made by robot swarms," in *Proceedings of 13th International Symposium on Distributed Autonomous Robotic Systems (DARS 2016). In press. Video available online: https://www. youtube. com/watch*, 2016.

[3] "Learn More NASA Swarmathon," 2017, Retrieved on 10/19/2017 from http://nasaswarmathon.com/about. [Online]. Available: http://nasaswarmathon.com/about

[4] K. L. Doty and R. E. Van Aken, "Swarm robot materials handling paradigm for a manufacturing workcell," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 778–782.

[5] R. Cassinis, G. Bianco, A. Cavagnini, and P. Ransenigo, "Strategies for navigation of robot swarms to be used in landmines detection," in *Advanced Mobile Robots, 1999.(Eurobot'99) 1999 Third European Workshop on*. IEEE, 1999, pp. 211–218.

[6] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *International workshop on swarm robotics*. Springer, 2004, pp. 10–20.

[7] D. Levin, J. P. Hecker, M. E. Moses, S. Forrest, G. M. Fricke, S. R. Black, J. L. Cannon, F. Asperti-Boursin, K. Stolleis, B. Swenson *et al.*, "Volatility and spatial distribution of resources determine ant foraging strategies," in *Proceedings of the European Conference on Artificial Life*, 2015, pp. 256–263.

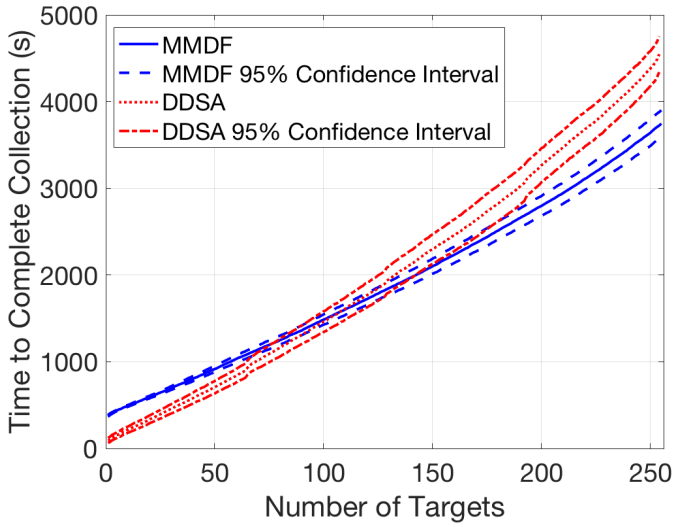

Fig. 7: *Clustered Distribution: Time to Complete Collections vs. Number of Targets.*

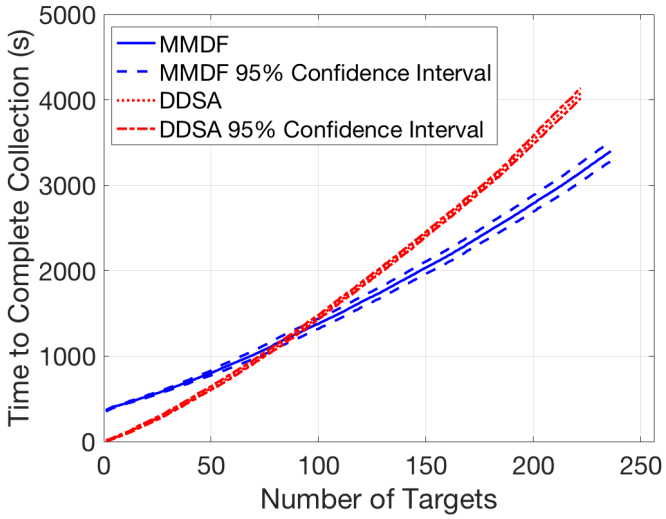

Fig. 8: *Uniform Distribution: Time to Complete Collections vs. Number of Targets.*

[8] J. P. Hecker and M. E. Moses, "Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms," *Swarm Intelligence*, vol. 9, no. 1, pp. 43–70, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11721-015-0104-z

[9] G. M. Fricke, J. P. Hecker, A. D. Griego, L. T. Tran, and M. E. Moses, "A distributed deterministic spiral search algorithm for swarms," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4430–4436.

[10] D. A. Shell and M. J. Mataric, "On foraging strategies for large-scale multi-robot systems," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 2717–2723.

[11] M. Schneider-Fontan and M. J. Mataric, "Territorial multi-robot task division," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 5, pp. 815–822, 1998.

[12] V. Tereshko and T. Lee, "How information-mapping patterns determine foraging behaviour of a honey bee colony," *Open Systems & Information Dynamics*, vol. 9, no. 2, pp. 181–193, Jun 2002. [Online]. Available: https://doi.org/10.1023/A:1015652810815

[13] S. Cazamine and J. Sneyd, "A model of collective nectar source selection by honey bees," *Journal of Theoretical Biology*, vol. 149, no. 4, pp. 547–571, 1991.

[14] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[15] S. Burlington and G. Dudek, "Spiral search as an efficient mobile robotic search technique," in *Proceedings of the 16th National Conf. on AI, Orlando Fl*, 1999.

[16] E. Tunstel, G. Anderson, and E. Wilson, "Motion trajectories for wide-area surveying with a rover-based distributed spectrometer," in *Automation Congress, 2006. WAC'06. World*. IEEE, 2006, pp. 1–8.

[17] G. Mathew and I. Mezic, "Spectral multiscale coverage: A uniform coverage algorithm for mobile sensor networks," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, Dec 2009, pp. 7872–7877.

[18] T. J. Ai and V. Kachitvichyanukul, "A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery," *Computers & Operations Research*, vol. 36, no. 5, pp. 1693–1702, 2009.

[19] Q. Lu, M. E. Moses, and J. P. Hecker, "A scalable and adaptable multiple-place foraging algorithm for ant-inspired robot swarms," *CoRR*, vol. abs/1612.00480, 2016. [Online]. Available: http://arxiv.org/abs/1612.00480

[20] E. Maalouf, M. Saad, and H. Saliah, "A higher level path tracking controller for a four-wheel differentially steered mobile robot," *Robotics and Autonomous Systems*, vol. 54, no. 1, pp. 23 – 33, 2006.

[21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.