University of Texas Rio Grande Valley

ScholarWorks @ UTRGV

Computer Science Faculty Publications and Presentations

College of Engineering and Computer Science

2023

Adaptive Multiple Distributed Bidirectional Spiral Path Planning for Foraging Robot Swarms

Qi Lu

Ryan Luna

Follow this and additional works at: https://scholarworks.utrgv.edu/cs_fac

 Part of the Computer Sciences Commons

Adaptive Multiple Distributed Bidirectional Spiral Path Planning for Foraging Robot Swarms

Qi Lu
Department of Computer Science
The University of Texas Rio Grande Valley
Edinburg, USA
qi.lu@utrgv.edu

Ryan Luna
Department of Computer Science
The University of Texas Rio Grande Valley
Edinburg, USA
ryan.luna01@utrgv.edu

***Abstract*—The Distributed Deterministic Spiral Algorithm (DDSA) has shown great foraging efficiency in robot swarms. However, when the number of robots in the swarm increases, scalability becomes a significant bottleneck due to increased collisions among robots, making it challenging to deploy them in the search space (e.g., 20 robots). To address this issue, we propose an adaptive Multiple-Distributed Bidirectional Spiral Algorithm (MDBSA) that enhances scalability. Our proposed algorithm partitions the squared search arena into multiple identical squared regions and assigns robots to regions dynamically based on the number of regions. In each region, a bidirectional spiral search path is planned, and when a robot completes its search, it is assigned to either an unassigned region or a region with one robot. The two robots will then travel along the path from the starting and ending points of the spiral path. We evaluated the performance of robot swarms using the MDBSA algorithm in the ARGoS robot simulator. Our experimental results show that the proposed MDBSA algorithm outperforms DDSA. When robots deliver collected resources to regions instead of the center, it reduces collisions and significantly improves the scalability of the robot swarm. Our findings suggest that a multiple-distributed search strategy is an efficient solution for foraging robot swarms.**

***Index Terms*—Path Planning, Task Partitioning, Distributed Spiral Search, Foraging Robot Swarms**

I. Introduction

Foraging involves the search for and collection of objects, and it is a challenging task for many species to retrieve resources (e.g., seeds, nectar, or water) and bring them back to their nests, especially when searching for multiple objects in an unfamiliar environment. Foraging is not only a complex task for many living organisms but also a metaphor for various real-world robotics applications, such as search and rescue, humanitarian de-mining, intrusion tracking, agriculture harvesting, infrastructure inspection, and planetary exploration [1]–[9].

Foraging swarm robotics involves using a large group of simple robots to search for and retrieve multiple targets or resources (e.g., minerals, hazardous waste, or survivors) and transport them to designated locations (e.g., warehouses, hospitals, or military bases) in order to maximize the collection of resources within a limited time period. Achieving this goal requires integrating various autonomous capabilities, including localization, navigation, image processing, object detection, retrieval, and communication, to create a seamless and effective robot system.

The focus of this work is on designing a search strategy for robot swarms, which can be classified into two main types: stochastic search and deterministic search. In stochastic search, robots use probability-based methods to explore the search space and find the optimal solution. The Central Place Foraging Algorithm (CPFA [10]) is an efficient stochastic search algorithm in swarm robotics. In this algorithm, robots search for resources randomly and share the discovered information in the environment to improve the effectiveness of foraging. Performance is optimized using a genetic algorithm. In deterministic search, robots follow predefined rules or planned paths to reach a specific goal. Unlike stochastic search, deterministic search algorithms do not involve randomness or probability-based decision-making. They are typically more predictable and can be more efficient in situations where the search space is well-defined and structured. For example, in the Distributed Deterministic Spiral Search Algorithm (DDSA), robots search for resources on planned squared spiral paths, and all areas in the search space will be visited once (minimal oversampling) when robots travel on the planned paths.

The work in [11] compares the stochastic search algorithm CPFA to the deterministic search algorithm DDSA. The CPFA takes advantage of sharing detected resource clusters with other robots, which recruits additional robots to the detected resource clusters and increases the foraging rate. However, the results demonstrate that the DDSA performs at least as well as the CPFA with a small number of robots. In the DDSA, robots collect the closest targets first and can complete the collection of the search space. However, the DDSA faces two major scalability issues. First, as the number of robots increases, congestion around the central collection zone increases rapidly. Second, spiral paths cannot be planned for a large number of robots.

We propose an adaptive path-planning algorithm called MDBSA (Multiple Distributed Bidirectional Spiral Algorithm) that can allocate robots to multiple distributed regions for efficient resource search. The algorithm uses planned squared spiral paths in the regions for the robots to search for resources. We partition the search arena into equal size regions based on the number of robots, and each robot is assigned to a region to search for multiple resources on the spiral path in their respective region. This approach minimizes

interventions in the swarm. However, since the number of randomly distributed resources varies in each region, robots complete their search at different times. Some robots may be idle, while others keep working. To utilize idle robots and further improve performance, the algorithm is adaptive. If a robot completes its search earlier, it can be assigned to a new unassigned region or one that has already been assigned. In the latter case, the remaining spiral path in the region will be shared by the two robots, with one starting from the beginning point of the spiral path and searching clockwise, and the other starting from the endpoint and searching counterclockwise.

We implement the MDBSA algorithm in the ARGoS multi-physics robot simulator [12] for a robot swarm and design two versions of the MDBSA: $\text{MDBSA}_{Centralized}$ and $\text{MDBSA}_{Distributed}$. In $\text{MDBSA}_{Centralized}$, robots collect resources and deliver them to the central collection zone, while in $\text{MDBSA}_{Distributed}$, robots deliver collected resources to their respective regions. We expect to see how delivering to the center affects foraging performance. We assume that resources can be transported to the center in a single high-capacity vehicle in a consequent delivering task. Therefore, the $\text{MDBSA}_{Distributed}$ algorithm is designed for optimal performance and serves as an upper bound for comparison.

We will compare the performance of the two MDBSA algorithms to the DDSA algorithm using three sets of experiments. The experiments will vary the number of regions, the number of robots, and the number of distributed resources to demonstrate the efficiency, adaptation, and scalability of the algorithms. Our evaluation metrics will be the foraging performance of the robot swarms, measured as the number of resources collected per robot, and the collision time, which measures the total time spent per robot on avoiding collisions with each other.

The remainder of this paper is organized as follows. In Section II, we summarize related work in spiral search algorithms. In Section III, we describe the background work in the deterministic spiral search algorithm. In Section IV, we describe the design of our proposed adaptive MDBSA algorithm. In Section V, we present the experimental setup. In Section VI, we evaluate the results of our experiments. Finally, we conclude in Section VII with a summary of our contributions and future work.

II. Related work

The spiral search behavior, first observed in desert ants (*Cataglyphis fortis*) [13], is a method for searching for a target within a designated area using a spiral trajectory. Each circle of the spiral represents the detection diameter of the agent. This search method has been extensively studied and has desirable optimality features [14]–[18], such as full coverage of the search area, detection of the nearest targets first, complete coverage of the area within the spiral, and minimal oversampling. Detecting the nearest targets first is particularly important, as it minimizes the travel time to the search point. The spiral search algorithm can be a useful tool for target search, especially in scenarios involving stationary targets.

An approach for conducting square search patterns using a single helicopter has been proposed in a study [19]. Additionally, a centralized multi-agent spiral search algorithm, where agents begin at a central point, has been described in another study [20]. Furthermore, parallel search methods utilizing spiral trajectories have also been developed, where each agent performs an individual spiral search independently of the other members of the swarm [21].

The spiral search algorithm has gained popularity as a benchmark and is being used in various real-world applications, as demonstrated by recent works. For instance, [22] presented a multi-actor-attention-critic (MAAC) reinforcement learning method for foraging robot swarms, comparing the foraging performance of their approach with the DDSA algorithm in simulation. Similarly, [23] implemented DDSA on multiple physical ground robots and tested it in a large outdoor environment. Furthermore, [24] developed a multi-agent target search method for moving and invisible objects using autonomous underwater vehicles (AUVs), comparing it to the DDSA algorithm. In their approach, they divided the original square area into four small squares, with each AUV performing a spiral search on its small square. In addition, [25] designed a lawnmower and the DDSA survey algorithm to map the CO_2 of a region of interest using flocking UAVs. Finally, the theoretical analysis of DDSA's performance was conducted in [26], which revealed that it outperformed the deterministic rotating-spoke and the random-ballistic algorithms.

III. The DDSA algorithm

The distributed deterministic spiral search algorithm (DDSA) [11] algorithm differs from the stochastic search algorithm central place foraging algorithm (CPFA) [10] by leveraging the optimality of spiral search for single agents and extending it to a swarm of robots [27]. Unlike CPFA, DDSA is a geometric approach that employs a planned path of interlocking square spirals for robots to search for targets, starting from the central collection zone. When executed without error, noise, or collisions, the DDSA ensures that robots discover the nearest targets first, reducing transportation costs. Moreover, the algorithm provides complete area coverage while minimizing repeated searches of the same location.

To calculate the path for each robot, we consider the number of robots r, the robot's ID i, the c-th circle (one revolution of the spiral), and the interlocking spiral distance g, which depends on the robot's target detection range. We generate the points along the spiral path in the north, east, south, and west directions in the order of their occurrence in each circle. Once we generate all four directions' points in the current circle, we increase c, and the generated path enables robots to travel clockwise. Equations 1 and 2 calculate the number of steps (F) of each spiral path in the north (N) and south (S) directions for the i-th robot on circle c. The number of steps to the east is the same as the number of steps to the north, and the number of steps to the west is the same as the number of steps to the south.

The number of steps to the east (E) and west (W) directions is calculated similarly, using a recurrence relation given in [11], [23]. Solving this relation, we can simplify the DDSA formulation to the following two equations:

$$F_{c,i}^{N} = F_{c,i}^{E} = \begin{cases} i & \text{if } c = 1 \\ (2c-3)r + 2i & \text{if } c > 1 \end{cases} \quad (1)$$

$$F_{c,i}^{S} = F_{c,i}^{W} = \begin{cases} 2i & \text{if } c = 1 \\ F_{c,i}^{N} + r & \text{if } c > 1 \end{cases} \quad (2)$$

Fig. 1 displays the spiral trajectories of five robots in an ARGoS simulation. The spiral search pattern commences at the center, with resources presented as black dots arranged in different sizes of clusters. The robots with green LED lights depict the resource-searching and resource-delivery process. The large red circle represents the center. The colored lines indicate the paths of the five robots, with only the squared spiral paths presented here. We do not display the paths returning to and from the center. The robots will eventually explore all areas within the search arena.

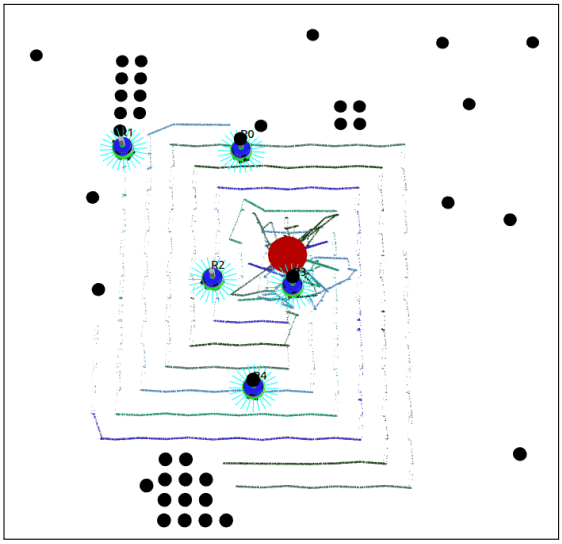

Fig. 1: The DDSA squared spiral trajectories of 5 robots in ARGoS, overhead view.

Fig. 2 illustrates how an individual robot transitions through a sequence of states during its foraging process using the DDSA algorithm. Initially, the robots are dispersed around the center, and the center calculates the spiral paths using the equation provided above. Subsequently, it assigns the path to each robot sequentially. Once this assignment is complete, each robot moves to its designated path and initiates the search for resources along that path. If a robot discovers a resource during its search, it retrieves and transports it directly to the center. Upon the robot's return to the center, it completes one foraging trip.

During subsequent foraging trips, the robot returns to the last location where it discovered a target (assuming that the robot can recall the location from the previous foraging trip) and resumes its spiral search from that point. If a robot completes the search along its entire spiral path without finding a target, it returns to the center and awaits further instructions until the completion of the foraging task.

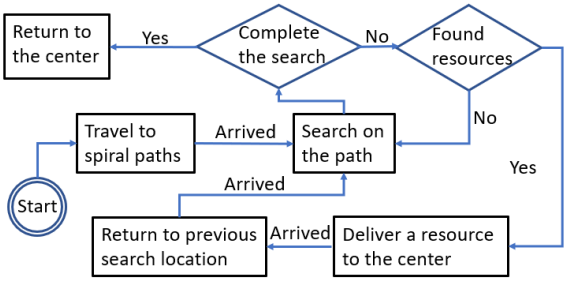

Fig. 2: Robots' states in the DDSA

IV. The Design of MDBSA

We divide the square arena into smaller square regions of equal size and plan spiral paths that fit within each region. The center serves as a communication hub and interacts with robots only when they are in the center. Communication between the center and robots only takes place within short distances. Robots make independent decisions when they leave the center, and long-range communication is not possible. A robot can communicate with another robot in the same region. The center shares information with the robots about the allocation of spiral paths and the search progress of other robots. The robot chooses to either move to an unassigned region or a region with the longest spiral path that has not been completed by an assigned robot. The longest spiral path indicates that the region was recently assigned to a robot or that many resources are present, and the robot is busy collecting them, resulting in minimal movement on the spiral path.

We assume that there are $N_R \in 4^k$ regions, where $k \in \mathbb{N}$. The size of the search arena is $L \times L$ meters, where L is the length and width of the arena. The center of the search arena is located at $(0,0)$. The center of a region (x_i, y_j) at row i and column j can be calculated using the following equation:

$$(x_i, y_j) = \left(\frac{L}{2}\left(1 - \frac{2i+1}{\sqrt{N_R}}\right), \frac{L}{2}\left(1 - \frac{2j+1}{\sqrt{N_R}}\right)\right) \quad (3)$$

In each region, there is only one spiral path. We can generate a single spiral path in each region based on the equations 1 and 2 described above when the number of robots is 1. Alternatively, we can calculate it using the derived equations below.

$$F_{c}^{N} = F_{c}^{E} = \begin{cases} 1 & \text{if } c = 1 \\ (2c-3)r + 2 & \text{if } c > 1 \end{cases} \quad (4)$$

$$F_{c}^{S} = F_{c}^{W} = \begin{cases} 2 & \text{if } c = 1 \\ F_{c}^{N} + r & \text{if } c > 1 \end{cases} \quad (5)$$

In this work, we make the assumption that a robot can detect a resource if the distance between the center of the robot and the center of the resource is less than or equal to 0.13 m. To account for simulated Gaussian noise on the wheel encoder, we set the distance g between adjacent spiral circles to 0.18 m.

Fig. 3 illustrates the state transitions of each robot in the MDBSA algorithm. The robots begin at the center and are allocated to regions based on the number of regions and robots. The MDBSA algorithm adapts to the number of robots and regions, and two possible cases are presented below.

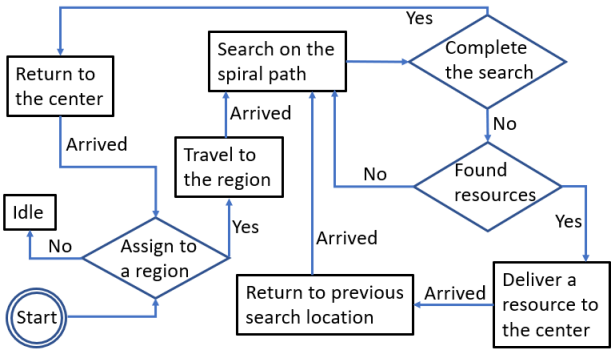

Fig. 3: Robots' states in the MDBSA

When the number of robots is less than or equal to the number of regions, the central server allocates each robot to a region based on its ID. Once a robot completes its search in a region, it returns to the center, which then assigns it to a new region. If all regions are assigned, the center assigns the robot to a region with the longest remaining spiral path. The robot then travels on the spiral path from the end, allowing two robots to search the same path from opposite directions. If there are no available paths to assign or share, the robot becomes idle.

Otherwise, the central server allocates one robot to each region. It then allocates one more robot to each region until there are no more robots left. Therefore, the maximum number of robots that can be allocated in a search arena is twice the number of regions. If the number of robots exceeds this limit, the search arena can be partitioned into more regions to accommodate more robots. If a robot completes its search in a region, it will return to the center. The center will check whether there is a path that can be shared in the region. If not, the robot will be idle until there is a path available for it to search.

We created two versions of MDBSA: MDBSA$_{Centralized}$ and MDBSA$_{Distributed}$. In MDBSA$_{Centralized}$, robots collect resources in their assigned regions and deliver them to the center. In MDSA$_{Distributed}$, robots deliver resources to their regional centers. We assumed that the collected resources would eventually be delivered to the center from the regional centers using larger logistic robots. Therefore, the performance of MDBSA$_{Distributed}$ serves as an upper bound for MDBSA$_{Centralized}$.

Robots are responsible for searching for resources within their designated regions. When a robot discovers a resource, it delivers it to either the central server in MDBSA$_{Centralized}$ or its regional center in MDBSA$_{Distributed}$. Once completed, the robot returns to the previous location where it found the resource and resumes its search along the spiral path. If the robot completes the search within the region, it returns to the center for a new assignment. The center assigns the robot to a new unassigned region if one is available. If all regions are assigned, the robot shares a spiral path with another robot in a region as previously described. If there are no unassigned regions, the robot becomes idle.

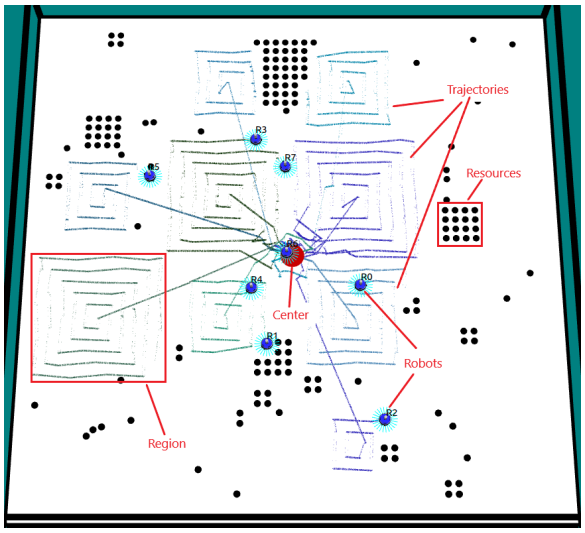

Fig. 4: The MDBSA squared spiral trajectories of 8 robots are in 16 (4×4) regions, overhead view.

Fig. 4 shows the squared spiral trajectories of 8 robots in the MDBSA$_{Centralized}$ as they search for resources in the partitioned arena of 16 regions. Each robot is assigned to search in a specific region, and their trajectories are shown in different colors. Robot "$R2$" completed its search in a region and its trajectory is shown in blue. It was then assigned to a new region at the bottom of the arena to continue its search for resources.

Fig. 5 demonstrates the trajectories of 8 robots in the MDBSA$_{Distributed}$, where two robots are assigned to each of the 4 regions. The trajectories show that one robot starts from its regional center, while the other starts from the end point of the spiral path. They will eventually meet at some point in their spiral path and complete the search in the region. Afterward, they will return to the center and be assigned to new regions. Each region can accommodate up to two robots, and no region can be assigned to more than two robots in this case. If a robot completes its search, it will become idle.

We compared the proposed MDBSA algorithm to the DDSA algorithm. We aimed to quantify the difference in performance between MDBSA$_{Centralized}$ and MDBSA$_{Distributed}$. Videos of

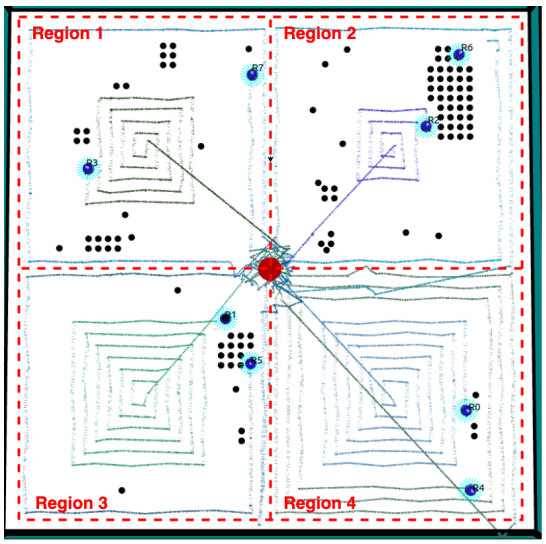

Fig. 5: The MDBSA squared spiral trajectories of 8 robots are in 4 regions, overhead view.

the algorithm running in the simulation are available on YouTube[1].

V. Experiment Setup

We performed three sets of experiments to assess the foraging performance and scalability of the proposed MDBSA algorithm for robot swarms, using the multi-robot simulator ARGoS [12]. In Experiment 1 (as shown in Table I), we used 8 robots and 160 resources, which were randomly distributed in the search arena as clusters of various sizes. To obtain a statistically robust evaluation of performance, we conducted 100 runs for each configuration. The simulation lasted for 12 minutes. We varied the number of regions while keeping other settings constant. The search space, spanning 8×8 meters, was partitioned into 4, 16, and 64 equal-sized regions to evaluate the impact of the number of regions on foraging performance.

TABLE I: Configuration in Experiment 1

Search space (m)	8×8
Number of robots	8
Number of resources	160
Number of runs	100
Resource distribution	Different sizes of clusters
Number of regions	4, 16, 64
Foraging time (minute)	12

In Experiment 2 (refer to Table II), we used a search space of 8×8 meters, with 16 equal-sized regions, and 200 resources randomly distributed in clusters of different sizes. We varied the number of robots (8, 14, 20, 26, and 32) to evaluate the scalability of the algorithm.

[1]https://youtu.be/CzJclBx1co4

TABLE II: Configuration in Experiment 2

Search space (m)	8×8
Number of robots	8, 14, 20, 26, 32
Number of resources	200
Resource distribution	Different sizes of clusters
Number of regions	16
Foraging time (minute)	16

Experiment 3, as presented in Table III, followed a similar setting as Experiment 2. We varied the number of resources to assess the algorithm's adaptability to different densities of resources in the search space.

TABLE III: Configuration in Experiment 3

Search space (m)	8×8
Number of robots	16
Number of resources	100, 150, 200, 250, 300
Resource distribution	Different sizes of clusters
Number of regions	16
Foraging time (minute)	12

VI. Results

Our evaluation focuses on two key metrics: the average number of collected resources per robot and the time spent avoiding collisions per robot in each simulation. All of the figures presented in this work were generated by averaging the results of 100 independent runs for each configuration. For experiments 1 and 2, we have conducted the Mann-Whitney U test for statistical analysis between each pairing of the 3 algorithms. The notch on each plot indicates the 95% confidence interval of the medians. If the notches of the two boxes do not overlap, this indicates a statistically significant difference between the medians. The '***' represents the significant difference ($p < 0.001$). The '**' represents a considerable difference, '*' represents a slight difference, and '-' represents a negligible difference.

Fig. 6 presents the results for Experiment 1. DDSA's results remained the same as the algorithm does not utilize regions. When using 4 and 16 regions, MDBSA$_{Centralized}$ achieved similar performance to DDSA regarding the number of collected resources. However, we observed a decrease of approximately 25% in the number of collected resources using 64 regions. On average, MDBSA$_{Distributed}$ collected around 101% more resources than DDSA and around 112% more resources than MDBSA$_{Centralized}$. MDBSA$_{Distributed}$ had the lowest standard deviation, achieving the highest number of collected resources when using 16 regions. The collision time in MDBSA$_{Centralized}$ increased as the number of regions increased, with both collision time and standard deviation being the highest among all algorithms. When using 4 and 64 regions, MDBSA$_{Distributed}$ has approximately 33% less time in collisions than DDSA, and around 72% less time when using 16 regions. Additionally, it

had the lowest standard deviation in collision time compared to the other two algorithms.

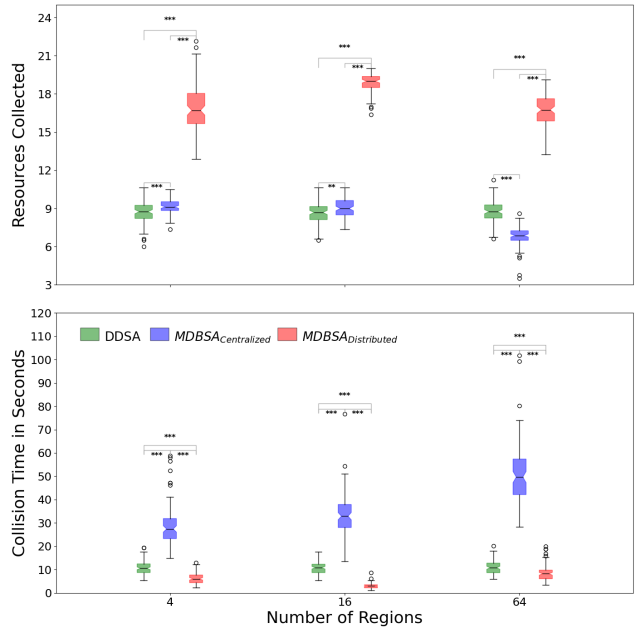

Fig. 6: Resources collected and collision times per robot in Exp. 1

Fig. 7 displays the average number of resources collected per robot per minute in Experiment 1. The results show that MDBSA$_{Distributed}$ outperforms both MDBSA$_{Centralized}$ and DDSA consistently. MDBSA$_{Centralized}$ and DDSA exhibit similar trends for 4 and 16 regions, with DDSA being slower to start during the first 2 minutes. However, when using 64 regions, both MDBSA$_{Centralized}$ and DDSA consistently have low performance. As the number of regions increases, a more pronounced arc is observed in the trend for MDBSA$_{Distributed}$, where robots are most efficient at collecting resources during the first 9 minutes of the simulation.

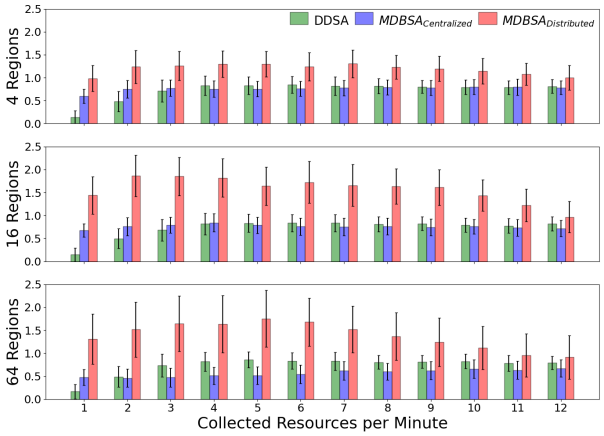

Fig. 7: Foraging rate per minute in Exp. 1

Fig. 8 presents the results for Experiment 2. We observe that as the number of robots increases, MDBSA$_{Distributed}$ becomes increasingly more efficient at collecting resources per robot compared to DDSA. Specifically, MDBSA$_{Distributed}$ outperforms DDSA by $\approx 100\%$ to $\approx 200\%$ in terms of the number of resources collected per robot as the number of robots increases. On the other hand, MDBSA$_{Centralized}$ collects $\approx 44\%$ fewer resources than DDSA as the number of robots increases from 14 to 32. Additionally, the standard deviation in the number of resources collected decreases as the number of robots increases in all algorithms, except for a few outliers in MDBSA$_{Distributed}$. Increasing the number of robots has a significant effect on the collision times for MDBSA$_{Centralized}$, with the collision time increasing sharply compared to DDSA. As the number of robots increases, MDBSA$_{Distributed}$ spends the least amount of time in collisions, achieving $\approx 21\%$ to $\approx 58\%$ less collision time than DDSA.

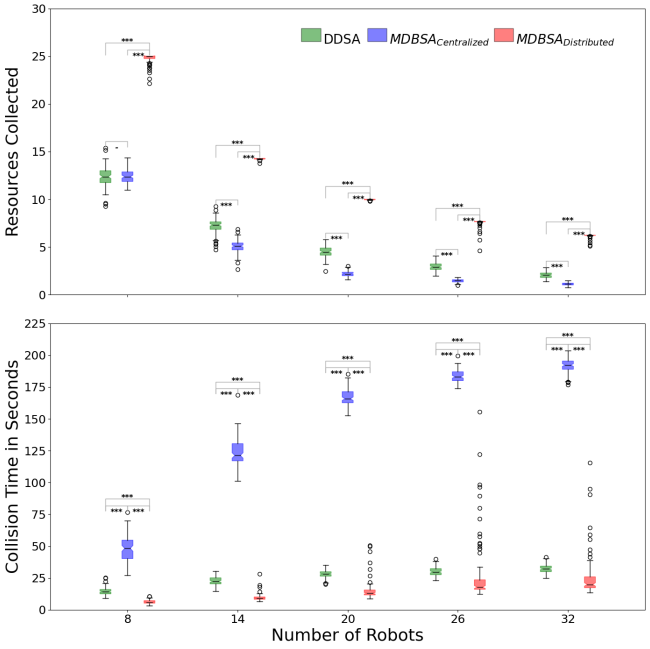

Fig. 8: Resources collected and collision time per robot in Exp. 2

Fig. 9 depicts the average number of resources collected per robot per minute in Experiment 2. Across all algorithms, there is a decreasing trend in the number of resources collected per robot as the number of robots increases. This trend is more pronounced in DDSA and MDBSA$_{Centralized}$ across all minutes. MDBSA$_{Distributed}$ shows a noticeable change in trend, where the decline in resource collection begins earlier as the number of robots increases. In all simulations except for those with 4 robots, MDBSA$_{Centralized}$ consistently underperforms against both other algorithms. In simulations with 16 or more robots, robots collect $\approx 24\%$ fewer resources than DDSA in MDBSA$_{Centralized}$. Despite the change in the number of robots, MDBSA$_{Distributed}$ collected the most resources on average

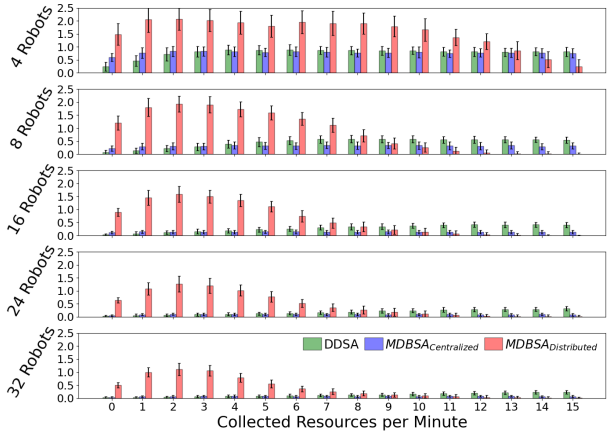

Fig. 9: Foraging rate per minute in Exp. 2

during the first 6 minutes.

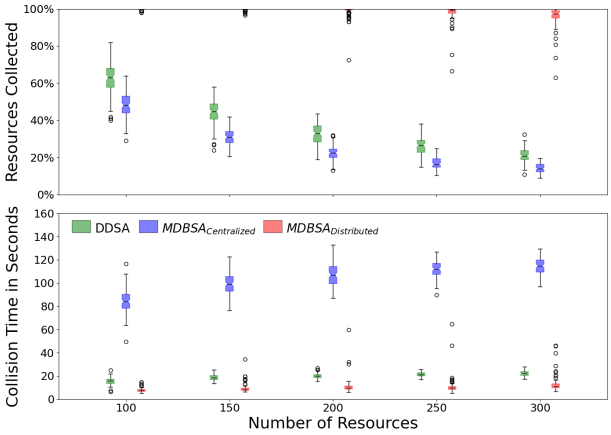

Fig. 10: Resources collected and collision time per robot in Exp. 3

Lastly, Fig. 10 displays the results for Experiment 3. MDBSA$_{Centralized}$ collects $\approx$ 30.4% fewer resources than DDSA. As the total number of resources increases, MDBSA$_{Distributed}$ demonstrates an almost 100% collection rate on average with a $\approx$ 60% to $\approx$ 350% improvement in the number of collected resources as the number of robots increases in comparison to DDSA. MDBSA$_{Centralized}$ displays an upward trend with decreasing momentum in the collision time as the number of resources increases. It contains the highest collision time per bot in comparison to the others.

VII. Discussion

In this study, we compared the performance of three foraging algorithms: DDSA, MDBSA$_{Centralized}$, and MDBSA$_{Distributed}$. Our results show that in all simulated scenarios, MDBSA$_{Distributed}$ outperformed the other two algorithms in terms of the number of collected resources and collision time. This is because the travel time for robots is significantly reduced, and collisions between robots delivering resources to a central location are avoided in this algorithm. When using 4 and 16 regions, MDBSA$_{Centralized}$ showed performance comparable to DDSA in terms of the number of collected resources, but its performance decreased in the scenario with 64 regions (see Fig.6 and Fig.7). This is because robots are distributed across smaller regions around the center, which leads to more collisions. However, in MDBSA$_{Distributed}$, the performance may experience a slight decrease since robots spend more time traveling to the center for new assignments to other regions.

As we increase the number of robots in all three algorithms, the foraging performance per robot decreases due to increased competition over resources (see Fig. 8). However, MDBSA$_{Distributed}$ is the least affected by this since the reduced travel time and collisions enable robots to collect resources more efficiently. DDSA still outperforms MDBSA$_{Centralized}$ because more robots can find resources quickly in the MDBSA$_{Centralized}$ and deliver them to the center, resulting in more collisions (see Fig. 8). However, the highly efficient performance of MDBSA$_{Distributed}$ implies that delivering to the center is a major issue. Additionally, we find that the standard deviations in both the resource collection and collision time decrease as the number of robots increases in all three algorithms. This suggests that increasing the number of robots stabilizes the algorithms' performance, regardless of their individual performance, as the performance of a smaller number of robots is more sensitive to the distribution of resources.

Experiment 3 also showed a similar trend in the standard deviations of resource collection for DDSA and MDBSA$_{Centralized}$ (see Fig. 10). Increasing the number of resources further stabilized the two algorithms because the distribution of resources became closer to random, with each region having a similar number of resources. However, this had the opposite effect on MDBSA$_{Distributed}$. Even though this algorithm outperformed the others, as the number of resources increased, the standard deviation increased, suggesting some instability of the algorithm as the environment became increasingly dynamic in terms of resource distribution.

Our experiments have demonstrated that MDBSA$_{Distributed}$ is highly efficient in collecting resources in a scalable way, outperforming DDSA and MDBSA$_{Centralized}$ in terms of the number of collected resources and collision time. Experiment 2 showed that as the number of robots increases, MDBSA$_{Distributed}$ becomes more efficient, with a higher foraging rate achieved early on in the simulation. However, as the number of resources increases in Experiment 3, the standard deviation of resource collection time increases, indicating some instability in the algorithm. Nevertheless, MDBSA$_{Distributed}$ remains highly efficient in collecting nearly 100% of resources within a 12-minute time frame. Our findings suggest that a distributed approach, such as MDBSA$_{Distributed}$, can significantly improve the efficiency and scalability of resource collection in foraging robot swarms.

The results of this study have significant implications for the development of foraging robot swarms. The MDBSA$_{Distributed}$

algorithm has the potential to substantially improve resource collection efficiency, especially in scenarios where resources are limited or time is critical. Additionally, implementing a distributed approach can reduce collision time, enhancing the system's reliability, scalability, and robustness. Our study shows that a distributed approach can significantly enhance the performance of deterministic foraging robot swarms in terms of efficiency, reliability, scalability, and robustness, making it highly suitable for time-critical applications. In conclusion, this study establishes a solid basis for future research into the advancement of multi-robot systems designed for resource collection tasks. It has the potential to contribute significantly to the development of robotic systems that can help address challenges in areas such as agriculture, disaster response, and exploration, among others.

This study has opened up several avenues for future research to build upon its findings. Firstly, the simulations were conducted in a controlled environment, which may not reflect the complexity and dynamics of real-world scenarios. Hence, it is necessary to evaluate the algorithm's robustness in more complex scenarios, such as those with sensor errors or obstacles, to better understand the algorithm's performance. Secondly, the current implementation of $\text{MDBSA}_{Distributed}$ separates the foraging and transportation tasks. Future research could focus on developing a more comprehensive algorithm that integrates both tasks to improve the efficiency and effectiveness of the foraging robot swarm system in practical applications. An integrated approach could be achieved by adding a specialized transportation vehicle to the system that periodically delivers multiple resources to the center. Overall, this study provides a solid foundation for future research to develop more efficient and effective foraging robot swarm systems for a variety of applications.

ACKNOWLEDGMENT

This work is supported by the GAANN program (P200A210144 - 22) from the U.S. Department of Education. The authors would also like to acknowledge the partial funding provided by the CREST Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS) through NSF Award No. 2112650.

REFERENCES

[1] A. F. T. Winfield, *Towards an Engineering Science of Robot Foraging*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 185–192.

[2] S.-k. Yun and D. Rus, "Adaptive coordinating construction of truss structures using distributed equal-mass partitioning," *Trans. Rob.*, vol. 30, no. 1, pp. 188–202, Feb. 2014.

[3] W. Fink, J. M. Dohm, M. A. Tarbell, T. M. Hare, and V. R. Baker, "Next-generation robotic planetary reconnaissance missions: A paradigm shift," *Planetary and Space Science*, vol. 53, no. 14–15, pp. 1419 – 1426, 2005.

[4] R. Groß and M. Dorigo, "Towards group transport by swarms of robots," *International Journal of Bio-Inspired Computation*, vol. 1, no. 1-2, pp. 1–13, 2009.

[5] C. W. Bac, E. J. Henten, J. Hemming, and Y. Edan, "Harvesting robots for high-value crops: State-of-the-art review and challenges ahead," *Journal of Field Robotics*, vol. 31, no. 6, pp. 888–911, 2014.

[6] V. Gazi and K. M. Passino, "Stability analysis of social foraging swarms," *Trans. Sys. Man Cyber. Part B*, vol. 34, no. 1, pp. 539–557, Feb. 2004.

[7] S. M. Ackerman, G. M. Fricke, J. P. Hecker, K. M. Hamed, S. R. Fowler, A. D. Griego, J. C. Jones, J. J. Nichol, K. W. Leucht, and M. E. Moses, "The swarmathon: An autonomous swarm robotics competition," in *2018 IEEE Intl. Conference on Robotics and Automation (ICRA)*, 2018.

[8] C. Ju and H. I. Son, "Multiple UAV systems for agricultural applications: Control, implementation, and evaluation," *Electronics*, vol. 7, no. 9, 2018.

[9] D. Albani, J. IJsselmuiden, R. Haken, and V. Trianni, "Monitoring and mapping with robot swarms for agricultural applications," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6.

[10] 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.

[11] G. M. Fricke, P. H. Joshua, D. G. Antonio, T. T. Linh, and E. M. Melanie, "A Distributed Deterministic Spiral Search Algorithm for Swarms," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4430–4436.

[12] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, and F. Ducatelle, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[13] M. Muller and R. Wehner, "The hidden spiral: systematic search and path integration in desert ants, cataglyphis fortis," *The Journal of Comparative Physiology*, vol. 175, pp. 525–530, 1994.

[14] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini uav," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 89–110, 2008.

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

[16] E. Langetepe, "On the optimality of spiral search," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '10. USA: Society for Industrial and Applied Mathematics, 2010, p. 1–12.

[17] M. A. A. El-Hadidy, "Optimal spiral search plan for a randomly located target in the plane," *International Journal of Operational Research*, vol. 22, no. 4, pp. 454–465, 2015.

[18] H. M. A. Gabal and M. A. A. El-Hadidy, "Optimal searching for a randomly located target in a bounded known region," *International Journal of Computing Science and Mathematics*, vol. 6, no. 4, pp. 392–403, 2015.

[19] A. Ryan and J. Hedrick, "A mode-switching path planner for uav-assisted search and rescue," in *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005, pp. 1471–1476.

[20] R. Baeza-Yates and R. Schott, "Parallel searching in the plane," *Computational Geometry*, vol. 5, no. 3, pp. 143–154, 1995.

[21] A. Hayes, A. Martinoli, and R. Goodman, "Swarm robotic odor localization," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2001, pp. 1073–1078 vol.2.

[22] N. Yang, Q. Lu, K. Xu, B. Ding, and Z. Gao, "Multi-actor-attention-critic reinforcement learning for central place foraging swarms," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–6.

[23] Q. Lu, A. D. Griego, G. M. Fricke, and E. M. Moses, "Comparing physical and simulated performance of a deterministic and a bio-inspired stochastic foraging strategy for robot swarms," in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2019.

[24] G. Wang, F. Wei, Y. Jiang, M. Zhao, K. Wang, and H. Qi, "A multi-auv maritime target search method for moving and invisible objects based on multi-agent deep reinforcement learning," *Sensors*, vol. 22, no. 21, 2022.

[25] J. Ericksen, G. M. Fricke, S. Nowicki, T. P. Fischer, J. C. Hayes, K. Rosenberger, S. R. Wolf, R. Fierro, and M. E. Moses, "Aerial survey robotics in extreme environments: Mapping volcanic co2 emissions with flocking uavs," *Frontiers in Control Engineering*, vol. 3, 2022.

[26] A. Aggarwal, D. Gupta, W. F. Vining, G. M. Fricke, and M. E. Moses, "Ignorance is not bliss: An analysis of central-place foraging algorithms," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6510–6517.

[27] R. Baezayates, J. Culberson, and G. Rawlins, "Searching in the plane," *Information and Computation*, vol. 106, no. 2, p. 234–252, oct 1993.