Accepted for publication in the IEEE International Conference on Robotics and Automation (ICRA 2021)

Multiple-Place Swarm Foraging with Dynamic Robot Chains

Dohee Lee[1] and Qi Lu[2] and Tsz-Chiu Au[1]

***Abstract*— The goal of foraging robot swarms is to search and deliver resources to a specific central collection zone quickly. In the previously proposed multiple-place foraging algorithm with dynamic depots, foraging performance decreases as search areas and swarm sizes increase: depots need to travel long distances to deliver resources to the center, and more robots produce more congestion on their journeys. We propose a novel extension to the multiple-place foraging in which multiple robot chains are deployed dynamically. Each robot chain connects a foraging location to the central collection zone. Instead of delivering resources by a single robot, resources are passed on robot chains from foraging locations to the center directly such that congestion near the central collection zone can be avoided. Dynamic robot chains can also relocate themselves to get closer to the resources while avoiding obstacles. We simulate our robot swarms in the robot simulator ARGoS. Our experiments show that robots using the MPFA with dynamic chains outperform the MPFA with dynamic depots and have less congestion.**

I. INTRODUCTION

Swarm robotics has been successfully applied to handle foraging tasks in which the objective of the swarm is to search for resources distributed in an arena and bring them back to a collection zone called *central depot* [1], [2]. Some common resources such as minerals, water, and fuels are typically deposited in clusters at unknown locations in a large area. Existing foraging swarm robot systems focus on developing an effective decentralized search-and-collection foraging algorithm for ant-like robot swarms to collect resources [3]–[5]. Lu et al. proposed a foraging system called *multiple-place foraging systems*, which utilizes helper robots called *dynamic depots*, which help to transfer resources to the central depot [6]–[8]. A dynamic depot is a mobile robot that acts as a depot with limited storage capacity for holding resources. Foraging robots can put the collected resources to the nearby dynamic depots, which will return to the central depot from time to time and dump the resources on behalf of the foraging robots. Dynamic depots can relocate themselves according to the demand of foraging robots over time.

The use of dynamic depots can substantially improve foraging performance. However, foraging systems based on dynamic depots can suffer from congestion near the central depot when dynamic depots return to the central depot to unload the collected resources, especially when there are many dynamic depots and foraging robots [8]. The congestion becomes bottlenecks that can quickly degrade the entire system. We therefore propose to replace dynamic depots with *dynamic robot chains*, which are essentially sequences of mobile robots with the ability of passing resources *at a distance*. Two consecutive robots in a robot chain can establish a *link* on which resources can be moved from one robot to another. One possible implementation of links is based on mobile conveyors [9], as shown in Fig. 1. A link can be implemented by other mechanisms as well (e.g., rope tows). Suppose some robot chains connect to the central depot, as shown in Fig. 2. A foraging robot can put resources on the last robot in a nearby robot chain. The resource will then be passed along the robot chain to the central depot. The robot chains are *dynamic* because the robots are mobile and hence robot chains can relocate themselves such that the last robot to get closer to uncollected resources.

Fig. 1: A robot chain consists of three mobile conveyors.

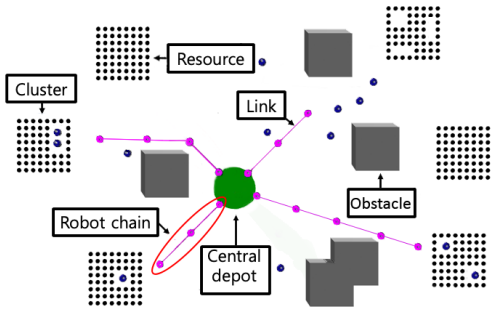


Fig. 2: A robot chain foraging system with four robot chains. The blue dots are foraging robots and the magenta dots are robot-chain robots.

The use of robot chains can avoid congestion at the central depot because our system ensures that one end of a robot chain is always connected to the central depot. Unlike dynamic depots, there is no need to compete with other robots when going to the central depot. Moreover, the robot chain can transfer resources continuously to the central depot without any interruption. While congestion of foraging robots can still occur near the last robots of robot chains, the congestion does not concentrate at the central depot but is distributed to multiple robot chains. Our experimental results in Sec. VI show that multiple-place swarm foraging systems based on robot chains are more efficient than dynamic depots.

[1]Department of Computer Science and Engineering, Ulsan National Institute of Science and Technology (UNIST), South Korea. {dohee,chiu}@unist.ac.kr

[2]Department of Computer Science, The University of Texas at San Antonio, USA. qi.lu@utsa.edu

This paper is organized as follows. After presenting the related work, we define our foraging task in Sec. III, describe the foraging robots' controller in Sec. IV, and outline the robot chains' relocation procedure in Sec. V. We present the experimental results in Sec. VI and conclude in Sec. VIII.

II. RELATED WORK

In central-place foraging, robots collect resources scattered in a large area and transported them to a central collection zone [3], [4]. Hecter and Moses [5] devised the stochastic central-place foraging algorithm (CPFA) for robot swarms. Flanagan et al. [10] significantly improved the foraging performance of CPFA by sharing the location information of resources. Fricke et al. [11] presented a distributed deterministic spiral search algorithm (DDSA) that guarantees a complete search of the entire arena. Although the DDSA outperforms the CPFA in simulation, the CPFA is slightly better than the DDSA in physical experiments [12].

Inspired by foraging behavior observed in the polydomous colonies of Argentine ants and wasps [13] with multiple nests and spider monkeys with multiple sleep site [14], Lu et al. [6], [7] proposed the multiple-place foraging algorithm (MPFA), which is analogous to global courier and transportation services in which many distributed warehouses collect and distribute resources efficiently. When compared with the CPFA,the MPFA produces higher foraging rates and lower travel distance. Lu et al. [7] improved the foraging performance of the MPFA by introducing the dynamic depots $\text{MPFA}_{dynamic}$, in which depots transport resources to the center directly [15]. However, foraging robots still need to travel long distances to transport resources. Pini et al. [16] presented a static partitioning strategy for foraging robot swarms. Ferrante et al. [17] uses Grammatical Evolution to divide a foraging task into searching and delivering tasks.

In this paper, we extend the work of MPFA with dynamic depots in [8] by replacing dynamic depots with mobile robot chains. Previously, robot chains have been used to localize other robots by acting as a set of stationary beacons [18] or as a navigation to relay images of an environment [19]. However, our robot chains are used to transfer resources by mobile conveyor lines [9] or delivery drones [20].

III. FORAGING TASKS WITH ROBOT CHAINS

We consider a foraging task in which a team of robots cooperates to collect resources in a large area called *an arena*. Fig. 2 shows an example of our foraging task. We assume the resources are deposited in several *resource clusters* whose locations are initially unknown. Our robots have to move around in the arena to discover the locations of the resource clusters. After locating the resource clusters, the robots work together to collect the resources and carry them back to the central depot, which is located at the center of the arena. The movement and the robots' visibility are hindered by some *obstacles* in the arena. Obstacles are polygon-shaped regions that no robot and no link cannot pass through. The locations and the shapes of obstacles are initially unknown, and the robots can discover them by sensors.

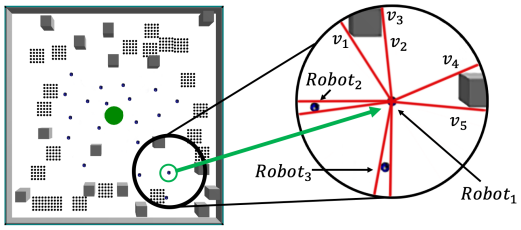


Fig. 3: The sensory information collected by the lidar on a robot. The red dotted line represents the range of the lidar.

Every mobile robot is equipped with a gripper, a lidar sensor, a resource detection device, a wireless communication device, a resource-holder with a finite storage capability, and a link-forming component. Each robot can be in one of the four states at any time: 1) the mobile state, 2) the resource-collecting state, 3) the resource-dumping state, and 4) the robot-chain state. In the mobile state, a robot can move freely with its lidar and its resource detection device enabled. When a robot arrives at a resource cluster, it can enter the resource-collecting state in which the robot can use its gripper to put the resources in the resource-holder. After collecting a resource, the robot enters the mobile state again and moves to the last robot in a robot chain or the central depot. Then the robot will enter the resource-dumping state and put all resources in either the resource-holder of the last robot of a robot chain or the central depot. When a robot is informed to form a robot chain with other robots, it enters the robot-chain state in which the robot cannot move, its sensor is disabled, and the link-forming component will start to establish links with the adjacent robots. The wireless communication devices are always enabled and robots can use the devices to share information about the resource clusters and the obstacles they encounter. Each robot can play one of two *roles* at any time: 1) foraging robots and 2) the robot-chain robots. As a foraging robot, a robot can switch between the mobile state, the resource-collecting state, the resource-dumping state. As a robot-chain robot, a robot can switch between the mobile state and the robot-chain state.

As discussed in Section I, there are many mechanical mechanisms for forming links between two robots. We assume the behavior of these mechanisms can be mathematically described by a *link model* $\langle d_{\max}, d_{\min}, C, v \rangle$, where 1) $d_{\max}$ and $d_{\min}$ are the maximum and minimum distances between the robots, respectively; 2) C is the capability of the link, which is the maximum number of resources moving on the link simultaneously; and 3) v is the speed of the resources moving on the link. Each robot-chain robot belongs to one robot chain, and all robot-chain robots in a robot chain have to enter the robot-chain states simultaneously to form a robot chain. A foraging robot can put resources in the resource-holder of the last robot anytime as long as the resource-holder is not full. Likewise, the i'th robot in a robot chain can send the resources in its resource-holder to the (i-1)'th robot in the robot chain anytime, as long as 1) it does not exceed the capability of the link and 2) the number of resources on the link does not exceed the empty space in the resource-

holder of the (i-1)'th robot. After a resource is put on a link, the resource will eventually enter the resource-holder of the (i-1)'th robot. The first robot always puts the resources it receives into the central depot immediately.

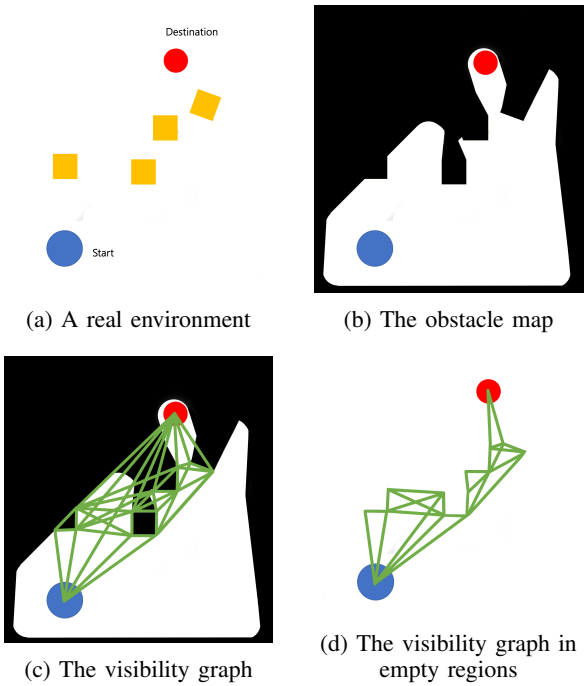
Our robots use lidar to detect obstacles and other robots as shown in Fig. 3. A robot can distinguish the edges of other robots from the edges of the obstacles based on the current location of the other robots broadcasted by the other robots. If an edge's location is not consistent with the edge of any other robot, the edge must belong to an obstacle.

IV. THE CONTROLLER FOR FORAGING ROBOTS

Initially, each robot is assigned a role and the robot-chain robots will start forming robot chains that reach different locations in the arena. In our experiments, there are four initial robot chains, and the last robots of the initial robot chains are close to the corners of the arena. The procedure of forming the initial robot chain is the same as the procedure of relocating robot chains that will be described in Sec. V. A robot chain will relocate itself from time to time based on the conditions in Sec. V. Between relocation, the foraging robots near the last robot of a robot chain will collect resources and put them in the resource-holder of the last robot. More precisely, the controller of the foraging robots is the same as the one in [8], except that our foraging robots have to avoid obstacles. A foraging robot always remembers the location of the last resource cluster it visited, and go there to collect resources until the cluster becomes empty. After collecting some resources, the foraging robot will put the resources in the resource-holder of the closest last robot of a robot chain. Note that foraging robots can also choose to dump the resources to the central depot directly if it is closer than the last robots of any robot chain. When the cluster becomes empty, the foraging robot will explore the arena to find other resource clusters using the exploration strategy in [8].

Since the locations of robots are shared among the robots, foraging robots have no trouble locating the last robots of robot chains. However, a foraging robot has to find a route to the last robot by avoiding obstacles. An *obstacle map* is a shared data structure that stores the information about the obstacles detected by all robots. An obstacle map partitions a 2D arena map into regions of three different kinds: 1) unknown regions, 2) empty regions, and 3) obstacle regions. These regions are shown as the black regions, the white regions, and the magenta regions in Fig. 4b, respectively. The entire obstacle map is initially one unknown region but is later updated according to the sensing information collected by the robots, as shown in Fig. 3. The sensing information can be used to determine the edges and the corners of obstacles and the empty regions that have no obstacle. The robots can integrate all information into the obstacle map by expanding the region of empty regions and mark the obstacle regions enclosed by the detected edges.

When a foraging robot needs to move to a new location, it will first compute the visibility graph in the obstacle map (Fig. 4c) and then remove all edges in the visibility graphs



(a) A real environment (b) The obstacle map

(c) The visibility graph (d) The visibility graph in empty regions

Fig. 4: The visibility graphs in an obstacle map.

that are not in the empty regions (Fig. 4d). The foraging robot will then move along the shortest path connecting the current location to the destination in the visibility graph. However, we also encourage the foraging robot to explore the unknown regions by occasionally moving into unknown regions. We employed the epsilon greedy exploration strategy to decide when a foraging robot should explore unknown regions. Whenever a foraging robot visits a node in the visibility graph that is incident to an edge that reaches an unknown region, there is a small probability that the robot will choose this path instead of continuing on the shortest path in the empty region. The foraging robot will update the visibility graph upon getting more information about the unknown region and then recompute the shortest path accordingly. When a foraging robot reaches a dead end in the exploration, it will return to the previous node in the last shortest path. If there is no shortest path in the empty region to reach the destination, the foraging robot will explore the unknown region randomly until there is a path to the destination.

V. RELOCATION OF ROBOT CHAINS

Dynamic depots can greatly reduce the amount of time the foraging robots take to transport the resources by relocation [8]. Likewise, a robot chain should relocate itself such that its last robot can be closer to resource clusters. The questions are when and how to relocate a robot chain.

Our system decides whether a robot chain should relocate only after a duration $t_{protect}$ since the last relocation, such that the relocation would not occur too frequently. After a duration $t_{protect}$, the robot chain will check whether the utilization of the robot chain remains high. The utilization of a robot chain is the rate at which foraging robots put resources to the resource-holder of the last robot. If the rate

is higher than a threshold $\alpha_{protect}$, there is no relocation so that the foraging robots can keep using the robot chain. Otherwise, our system will check whether there is a better location for the robot chain. First, it computes a *target location*. When relocation occurs, the relocation procedure, as described later in this section, aims to put the last robot as close to the target location as possible. The target location (x,y) can be computed by the following equations:

$$x = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} \text{ and } y = \frac{\sum_{i=1}^{N} w_i y_i}{\sum_{i=1}^{N} w_i}. \quad (1)$$

where (x_i, y_i) is the coordinate of the resource cluster i in a set $\mathscr{R}$ of resource clusters, w_i is the estimated number of remaining resources in the resource cluster i, and N is the total number of locations where robots have detected resources in $\mathscr{R}$. $\mathscr{R}$ is a set of non-empty resource clusters near the last robot of the robot chain. $\mathscr{R}$ includes all non-empty resource clusters for calculating the target point in the last relocation. In addition, our system will add more resource clusters to $\mathscr{R}$ (e.g, the next known resource cluster on the right side of the robot chain that is not considered when computing the last target point).

If our system decides not to relocate a robot chain, it will wait for a short duration $t'_{protect}$ and then decide again. Otherwise, the robot chain will stop receiving resources from foraging robots and wait until all existing resources on the chain arrive at the central depot. Then all links on the robot chain are disengaged, and all robot-chain robots will switch to the mobile state. After that, the robots will try to form a new robot chain such that the last robot is as close to the target location as possible. Due to unknown obstacles in the unknown regions in the arena, the ideal shape of the new robot chain cannot be pre-computed, and the robots have to search for the best positions while exploring the arena.

The key idea of the relocation procedure is to maintain the best configuration of a robot chain during the exploration and to make sure that there is enough time for robots to return to this best configuration before the allotted exploration time T_{explore} is expired. A *configuration* of a robot chain is a sequence of positions and orientations of the robots on the robot chain. Initially, the best configuration is the one before any robot moves. Based on the visibility graph derived from the current obstacle map, we extend the visibility graph by connecting the central depot and the target location to the nodes in the graph. In the subgraph in the empty regions, find the shortest path connecting the central depot to the target location. Then we calculate the robots' positions that can be put on or near the shortest path to form a robot chain in the empty region along the shortest path starting at the central depot. These positions are the next best configuration, and all robots will then move to these positions. After the robots arrive at these locations successfully and there is time left for exploration, our system will send the robots to explore the closest unknown regions. The robots will reveal more empty regions in the obstacle map, and the visibility graph will be updated accordingly. If the shortest path between the central depot and the target location changes, it implies that a better configuration is found, and the robots will move to the new locations in the new configuration. After that, the exploration continues. The robots have to keep track of the time t_{return} they need to return to the positions in the current best configuration. When the remaining exploration time is equal to t_{return} or the current best configuration can be proved to be the shortest, the robots will immediately stop exploration, return to the positions in the best configuration, and form the robot chain.

After relocation, the robot chain can recruit more robots to join the robot chain if the last robot is still far away from the target location. Likewise, the robot chain can drop some robots to reduce its length if it is longer than the shortest path between the central depot and the target location.

VI. EXPERIMENTAL CONFIGURATIONS

To evaluate our robot chain algorithm, we conducted two sets of experiments in the Autonomous Robots Go Swarming (ARGoS) [21]. We checked whether the foraging performance varies systematically with different configurations and statistically analyzed the results. In both experiments, we compared our proposed algorithm $\text{RC}_{dynamic}$ (i.e., dynamic robot chains) with two MPFA with dynamic depots algorithms, ($\text{MPFA}^{3}_{dynamic}$ and $\text{MPFA}^{16}_{dynamic}$). In the two MPFA algorithms, 4 dynamic depots are distributed and can move to new locations dynamically as described in [8]. The difference between the two MPFA algorithms is the capacities of depots: 3 in the $\text{MPFA}^{3}_{dynamic}$ and 16 in the $\text{MPFA}^{16}_{dynamic}$.

We implemented another robot chain algorithm RC_{static} that does not allow moving or relocating robot chains. In two experiment, we set a link model as follows;$d_{\max} = 0.8m, d_{\min} = 0.34m, C = 3, v = 0.32m/s$. In the first set of experiments, the number of clusters is 20, and the shapes of clusters are 5×5, 5×10, and 10×10 for 500, 1000, and 2000 resources, respectively. We scaled up the number of resources, the number of robots, and the arena size to evaluate the foraging performance of the algorithms. A certain percentage of robots were used to initialize the robot chains.The parameters of relocation are $t_{protect} = 360s$ and $\alpha_{protect} = 2$. The experimental setup is summarized in Table I.

TABLE I: The Configuration in Experiment 1

Arena Size($m \times m$)	10×10	20×20	40×40
Number of resource	500	1000	2000
Number of robots	20, 30, 40 50, 60	40, 50, 60 70, 80	60, 80, 100 120, 140
Foraging time (minute)	30		
% of robots for the initial robot chains	30%		

In the second experiment, we measured the foraging performance in the arenas 20×20 meter with a different number of obstacles: 4, 8, 16, and 32. An obstacle is a simulated box of size 0.5m $\times$0.5m $\times$0.5m with a random orientation. 1000 resources are in $5m \times 10$m clusters. 40, 60, and 80 robots are distributed in the center..

VII. EXPERIMENTAL RESULTS

The foraging performance is the number of resources collected and delivered to the central collection zone. The collision time is the time robots spent to avoid collision with other robots and the boundary of the arena. We checked whether the foraging performance varies systematically with different configurations and statistically analyze the results. Each data set in the figures is an average of 30 runs, and each error bar indicates 95% confidence intervals.

Fig. 5 demonstrates the foraging performance of all 4 algorithms in Experiment 1. The two robot chain algorithms outperform the two MPFA algorithms. All performance increase as the number of robots and the arena size increase, except $\text{MPFA}^{3}_{dynamic}$. Both robot chain algorithms increase faster than the two MPFA algorithms. The results indicate that the performance of $\text{RC}_{dynamic}$ is 237% higher than $\text{MPFA}^{3}_{dynamic}$ and 107% higher than $\text{MPFA}^{16}_{dynamic}$ when the number of robots is 140, and the arena size is $40m \times 40m$.

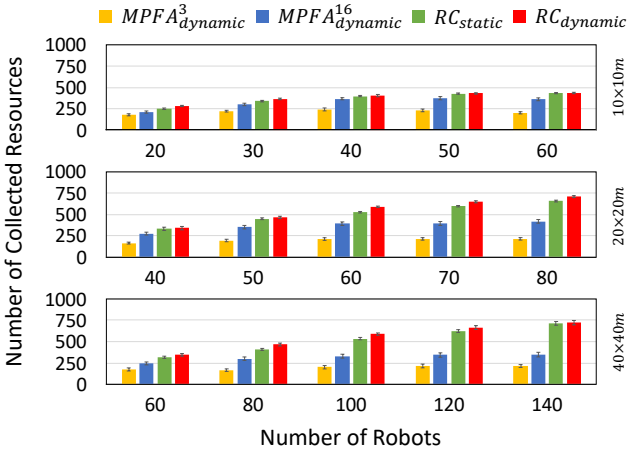


Fig. 5: Foraging performance using a different number of robots at different arenas in Experiment 1.

Fig. 6 compares the collision time of all 4 algorithms in Experiment 1. The collision time in the MPFA with dynamic depots with larger capacity 16 is 27% higher than the time in $\text{RC}_{dynamic}$. The depots with the larger capacity 16 have more collision time than the depots with the smaller capacity 3. The difference between the collision time in the two robot chain algorithms is not obvious.

Fig. 7 demonstrates the foraging performance of all 4 algorithms with a different number of randomly distributed obstacles in experiment 2. All performance decrease as the number of obstacles increases. $\text{MPFA}^{16}_{dynamic}$ outperforms the other three algorithms using 40 robots. It is 40%, 44%, 47%, and 46% higher than $\text{RC}_{dynamic}$, respectively. The performance of the other three algorithms does not have a significant difference. When the number of robots increases to 60, RC_{static} and $\text{RC}_{dynamic}$ slightly outperform $\text{MPFA}^{16}_{dynamic}$. Comparing their performance with 40 robots,

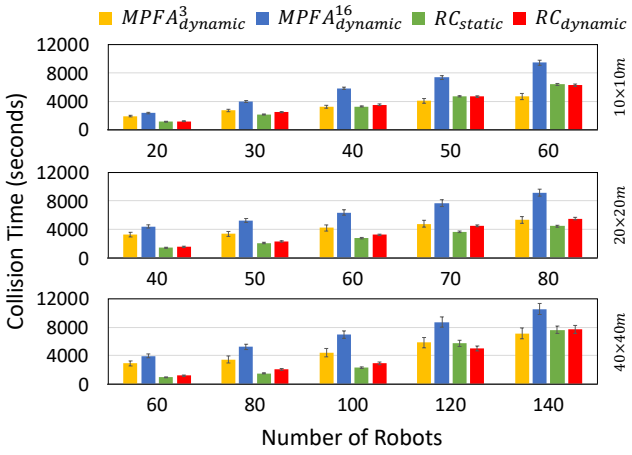


Fig. 6: The collision time of each swarm in Experiment 1.

their performance increases faster than the two MPFA algorithms. When the number of robots increases to 80, we can see a more clear improvement in RC_{static} and $\text{RC}_{dynamic}$. Besides, $\text{RC}_{dynamic}$ outperforms RC_{static} when the number of obstacles are 8, 16, and 32.

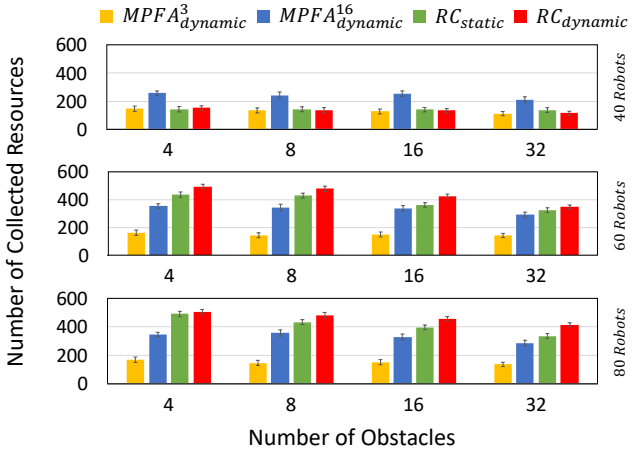


Fig. 7: Foraging performance using a different number of the robots and obstacles in Experiment 2.

VIII. DISCUSSION AND FUTURE WORK

In this paper, we showed that by giving robots in a robot swarm the ability to pass objects at a distance between them, they can collectively solve the foraging tasks more efficiently. The key is the formation of robot chains that can overcome the two major limitations of existing dynamic depot foraging systems: congestion near the central collection zone and the long travel distance for delivering. To this end, we introduced a novel foraging swarm robot system based on robot chains. We presented the robot chain algorithm for controlling the robots and described the relocation procedure of dynamic robot chains. Our experiments show that dynamic robot chains outperform dynamic depots in multiple-place foraging tasks given the same number of robots.

The reason for the superior performance of our robot chain algorithms is as follows. In $\text{MPFA}^{3}_{dynamic}$ and $\text{MPFA}^{16}_{dynamic}$,

when the capacity of resources of the depot robots is as low as 3, the depots have to visit the central depot frequently. Therefore, the foraging robots have to spend more time waiting for the depots to return. RC_{static} and $\text{RC}_{dynamic}$ do not have this problem since it can continuously receive the resources from the foraging robots after the robot chains are established; thus, the foraging robots have a high utility rate. In $\text{RC}_{dynamic}$, the robot chains only relocate a few times, and hence they keep operating most of the time. $\text{RC}_{dynamic}$ suffers from slightly more collisions when the robot chains relocate. However, the relocation updates robot chains to better locations for collecting more resources.

Most of the collisions in $\text{MPFA}^{3}_{dynamic}$ and $\text{MPFA}^{16}_{dynamic}$ are due to the congestion near the depot robot and the central depot in Fig. 6. Due to a large number of robots near the depot robot or the central depot, the robots have to take more time to unload the resources to the depot robot or the central depot. Additionally, the difference collision time between $\text{MPFA}^{3}_{dynamic}$ and $\text{MPFA}^{16}_{dynamic}$ is the capacity of the depot robot. The smaller capacity results in more trips of delivering resources, and therefore results in more collisions. When the foraging robots are waiting for the depot robot, there is no collision between the robots that want to unload resources to the depot robot. Therefore, the $\text{MPFA}^{3}_{dynamic}$ shows more small collision times than $\text{MPFA}^{16}_{dynamic}$. In RC_{static} and $\text{RC}_{dynamic}$, the collision at the central depot can be avoided almost completely; instead, the collisions in RC_{static} and $\text{RC}_{dynamic}$ occur mostly at the end of the robot chains. Since there are four robot chains, the collisions are evenly distributed to these robot chains, causing less congestion. In $\text{RC}_{dynamic}$, there are some collisions near the central depot when the robot chains are relocated. Robots collide with other robots when they move to new locations to form a new robot chain. When a depot travels to the central collection zone to deliver resources, foraging robots around it is idle and wait for its returns. The depot with a small capacity 3 travels more frequently than the one with a large capacity 16. Therefore, there is less collision and the robots have a longer waiting time.

When obstacles are distributed in searching arenas, all foraging performance is disrupted and decrease rapidly. The performance of RC_{static} is very close to $\text{RC}_{dynamic}$. It indicates that the relocation is not efficient enough in small swarm sizes and small arenas. The performance of $\text{RC}_{dynamic}$ has the potential to be much higher when the number of robots is larger.

This work shows that the use of dynamic robot chains can greatly improve the performance of foraging systems when compared with existing approaches. The advantages of this novel approach are: 1) it provides more efficient transportation than central place foraging algorithms and the multiple places foraging algorithm MPFA with limit depot capacities; 2) it has less collision time among robots, and 3) it can detect and avoid obstacles. In the future, we will design a more efficient strategy for avoiding obstacles in more complex environments.

ACKNOWLEDGMENT

This work has been taken place in the ART Lab at UNIST and UT San Antonio. ART research is supported by NRF (2016R1D1A1B0101359816 and 2016M3C4A795263722).

REFERENCES

[1] W. Liu, "Design and modelling of adaptive foraging in swarm robotic systems," Ph.D. dissertation, Faculty of Environment and Technology, University of the West of England, Bristol, 2008.

[2] W. Liu and A. F. Winfield, "Modeling and optimization of adaptive foraging in swarm robotic systems," *The International Journal of Robotics Research*, vol. 29, no. 14, pp. 1743–1760, 2010.

[3] E. Castello, T. Yamamoto, F. Dalla Libera, W. Liu, A. F. Winfield, Y. Nakamura, and H. Ishiguro, "Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach," *Swarm Intelligence*, vol. 10, no. 1, pp. 1–31, 2016.

[4] W. Liu, A. F. Winfield, J. Sa, J. Chen, and L. Dou, "Towards energy optimization: Emergent task allocation in a swarm of foraging robots," *Adaptive behavior*, vol. 15, no. 3, pp. 289–305, 2007.

[5] 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.

[6] Q. Lu, M. E. Moses, and J. P. Hecker, "A Scalable and Adaptable Multiple-Place Foraging Algorithm for Ant-Inspired Robot Swarms." Workshop on On-line decision-making in multi-robot coordination, 2016 Robotics Science and arXiv:1612.00480.

[7] Q. Lu, J. P. Hecker, and M. E. Moses, "The MPFA: A Multiple-Place Foraging Algorithm for Biologically-Inspired Robot Swarms," 2016.

[8] Q. Lu, J. P. Hecker, and M. Moses, "Multiple-place swarm foraging with dynamic depots," *Autonomous Robots*, vol. 42, no. 4, pp. 909–926, 2018.

[9] D. Lee and T.-C. Au, "Automatic configuration of mobile conveyor lines," in *ICRA*, 2016, pp. 3841–3846.

[10] T. P. Flanagan, K. Letendre, W. R. Burnside, G. M. Fricke, and M. E. Moses, "Quantifying the effect of colony size and food distribution on harvester ant foraging," *PloS one*, vol. 7, no. 7, p. e39427, 2012.

[11] 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 *IROS*, 2016, pp. 4430–4436.

[12] Q. Lu, A. D. Griego, G. M. Fricke, and M. E. Moses, "Comparing physical and simulated performance of a deterministic and a bio-inspired stochastic foraging strategy for robot swarms," in *ICRA*, 2019, pp. 9285–9291.

[13] T. P. Flanagan, N. M. Pinter-Wollman, M. E. Moses, and D. M. Gordon, "Fast and flexible: Argentine ants recruit from nearby trails," *PLOS ONE*, vol. 8, no. 8, pp. 1–7, August 2013.

[14] C. A. Chapman, L. J. Chapman, and R. McLaughlin, "Multiple central place foraging by spider monkeys: Travel consequences of using many sleeping sites," *Oecologia*, vol. 79, no. 4, pp. 506–511, 1989.

[15] Q. Lu, "An efficient multiple-place foraging algorithm for scalable robot swarms," 2019.

[16] G. Pini, A. Brutschy, A. Scheidler, M. Dorigo, and M. Birattari, "Task partitioning in a robot swarm: Object retrieval as a sequence of subtasks with direct object transfer," *Artificial life*, vol. 20, no. 3, pp. 291–317, 2014.

[17] E. Ferrante, A. E. Turgut, E. Duéñez-Guzmán, M. Dorigo, and T. Wenseleers, "Evolution of self-organized task specialization in robot swarms," *PLoS Comput Biol*, vol. 11, no. 8, pp. 1–21, 2015.

[18] S. Nouyan, A. Campo, and M. Dorigo, "Path formation in a robot swarm," *Swarm Intelligence*, vol. 2, no. 1, pp. 1–23, 2008.

[19] P. M. Maxim, W. M. Spears, and D. F. Spears, "Robotic chain formations," *International Federation of Automatic Control Proceedings Volumes*, vol. 42, no. 22, pp. 19–24, 2009.

[20] F. Wang, P. Liu, S. Zhao, B. M. Chen, S. K. Phang, S. Lai, T. H. Lee, and C. Cai, "Guidance, navigation and control of an unmanned helicopter for automatic cargo transportation," in *IEEE Xplore Proceedings of the 33rd chinese control conference*, 2014, pp. 1013–1020.

[21] 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.