Dynamic Robot Chain Networks for Swarm Foraging

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

***Abstract*— The objective of foraging robot swarms is to search for and collect resources in an unknown arena as quickly as possible. To avoid the congestion near the central collection zone, we previously proposed an extension to the multiple-place foraging in which robot chains are deployed dynamically so that foraging robots can deliver to the robot chains instead of the central collection zone. However, a robot chain can only reach one location at a time, and congestion can occur at the end of the robot chain. This paper presents an extension to dynamic robot chains called *dynamic robot chain networks*, which extends robot chains with branches, each of which reaches different resource clusters. We formulate the problem of finding the smallest dynamic robot chain networks as the Euclidean Steiner tree problem and explain how Steiner trees can be utilized to optimize the efficiency of foraging operations. We implemented our foraging robot swarms in a simulator called ARGoS. Our experiments showed that dynamic robot chain networks can avoid obstacles and collect more resources when compared with the original robot chain design.**

I. INTRODUCTION

One important application of swarm robotics is foraging—searching for resources such as minerals, water, and fuels that are distributed in an arena and bringing them back to a collection zone called *central depot* [1], [2]. Most existing foraging swarm robot systems use decentralized search-and-collection foraging strategies [3]–[8]. However, some of these foraging systems suffer from congestion near the central depot when many foraging robots return and unload the collected resources at the central depot [8]. To alleviate the congestion problem, Lu et al. proposed *multiple-place foraging systems*, which utilizes helper robots called *dynamic depots* to collect resources from foraging robots and transport the resources to the central depot [6]–[8]. Although dynamic depots can help reduce the number of robots near the central depot, congestion still occurs in heavy traffic. Previously, we proposed to replace dynamic depots with *dynamic robot chains*, which are sequences of robots with the ability to pass resources at a distance on them [9]. One possible implementation of dynamic robot chains is based on mobile conveyors [10]. A foraging robot can put resources on the last robot in a robot chain and let the robot chain pass the resources to the central depot. This approach can dramatically reduce congestion near the central depot [9].

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

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

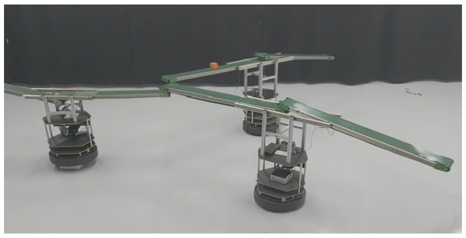


Fig. 1: The prototype of a robot chain network formed by mobile conveyor robots.

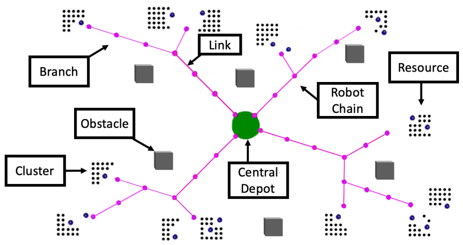


Fig. 2: A robot chain foraging system with four robot chain networks. The blue dots are foraging robots, whereas the magenta dots are robot-chain robots.

Dynamic robot chains, however, cannot eliminate congestion since congestion can still occur near the last robots of robot chains. When there are many resource clusters near the end of a robot chain, several foraging robots have to deposit the collected resources into the end of the robot chain. Thus, the ends of robot chains become the new bottleneck of the system. In this paper, we propose *dynamic robot chain networks*, which extend the idea of dynamic robot chains by having a network of robot chains that connect to multiple resource clusters. By allowing robot chains to have branches, the traffic near the end of a robot chain can be distributed to several branches and congestion can be avoided. More importantly, dynamic robot chain networks can achieve higher throughput by having multiple endpoints connecting to different resource clusters using fewer robots. The prototype of a dynamic robot chain network formed by mobile conveyor robots is shown in Fig. 1. We optimize the number of robots on a robot chain network by computing *Euclidean Steiner trees with obstacle avoidance*, which are the minimum spanning trees for connecting to multiple endpoints with the addition of Steiner points. We describe a high-level controller which decides when we should add new branches to an existing network and when we should relocate a subtree of a network to the locations on the Steiner tree. The experimental results in Sec. VIII show that multiple-place swarm foraging systems based on robot chain networks are more efficient than both dynamic depots and robot chains with no branch.

The rest of this paper is organized as follows. After presenting the related work, we define the foraging task in Sec. III. Sec. IV, V and VI describe our foraging system in detail. The experimental setups and results are presented in Sec. VII and VIII. Finally, we conclude in Sec. IX.

II. Related Work

In central-place foraging, robots collect resources scattered in a large area and transported them to a central collection zone [11], [12]. Hecker and Moses [13] devised the stochastic central-place foraging algorithm (CPFA) for robot swarms. Flanagan et al. [14] significantly improved the foraging performance of CPFA by sharing the location information of resources. Fricke et al. [15] 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 [16].

Inspired by foraging behavior observed in the polydomous colonies of Argentine ants and wasps [17] with multiple nests and spider monkeys with multiple sleep site [18], Lu et al. [19], [20] 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. Lu et al. [21] improved the foraging performance of the MPFA by introducing the dynamic depots MPFA$_{dynamic}$, in which depots transport resources to the center directly. Pini et al. [22] presented a static partitioning strategy for foraging robot swarms. Ferrante et al. [23] uses Grammatical Evolution to divide a foraging task into searching and delivering tasks. Lee et al. [9] extended the work of MPFA with dynamic depots in [21] 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 [24] or as navigation to relay images of an environment [25]. However, our robot chains are used to transfer resources by mobile conveyor lines [10] or delivery drones [26].

The Euclidean Steiner tree (EST) problem is an NP-hard problem, but there is a polynomial-time approximation scheme for finding a near-optimal solution in polynomial time [27]. [28] presented an exact but inefficient algorithm for the obstacle-avoiding EST problem. [29] proposed a nature-inspired approach to compute the typology of obstacle-avoiding Steiner trees. [30] proposed a heuristic strategy for the problem of constructing a EST when a 2D plane can be divided into polygonal regions. [31] proposed a heuristic for the EST problem using a large scale of simulated robot swarms. Our implementation of the EST solver is based on the algorithm in [32] with some modifications to handle obstacles.

III. Foraging with Robot Chain Networks

In a foraging task, a team of robots cooperates to collect resources in *an arena*. Fig. 2 shows an example of an arena

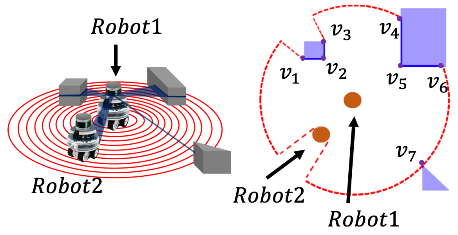


Fig. 3: The sensory information collected by the lidar on a robot. The red dotted line represents the range of the lidar.

in which resources are deposited in several *resource clusters*. Besides, there are *obstacles* that hinder the robot's visibility and movement. Initially, the robots do not know the locations of the resource clusters and the obstacles. The goal of the robot team is to find as many resources as possible and carry them back to the central depot.

A robot is an omnidirectional mobile robot with a gripper, a lidar sensor, a wireless communication device, a resource detection device, a resource-holder with limited storage space, and a link forming component (e.g., conveyors). A robot uses its lidar to detect the edges of the obstacles and other robots within the lidar's sensing range (see Fig. 3). The resource detection device can detect the exact locations of all resources within its sensing range. When the distance between two robots is less than $d_{\max}$, the two robots can use their link forming components to form a *link* between them such that one robot can send resources to the other robot via the link. The *capacity* of a link is the maximum number of resources that can move on the link simultaneously.

A *robot chain* is a sequence of robots in which every pair of adjacent robots are linked together. A *robot chain network* is a set of robot chains that join together in a tree-like structure. To form a robot chain network, robots must be capable of establishing links to three adjacent robots to form Y-junctions. Three mobile conveyors can form a one-way Y-junction by overlapping the endpoints of their belts such that objects from two incoming belts can drop on the outgoing belt. The problem of finding the minimum size robot chain networks can be formulated as an *obstacle-avoiding Euclidean Steiner tree problem*: given N points in a 2D plane with obstacles, find a tree structure in the space that connects all points by lines of minimum total length with the help of newly-created points called *Steiner points*. In our case, the N points are the locations of resource clusters, the lines are robot chains, and the Steiner points are Y-junctions. One nice theoretical result is that every vertex in a minimum Steiner tree must have a degree of two or three [33], [34]. Hence, it is sufficient to consider Y-junctions only when forming a robot chain network of minimum size.

At any time, a robot is in one of the four states: 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. When a robot arrives at a resource, it can enter the resource-collecting state and use its gripper to put resources in its resource-holder. After collecting the resource, the robot enters the mobile state again. When the robot arrives at an endpoint of a robot chain network or the central depot, it can enter the resource-dumping state to unload the resource and then enter the mobile state again. When a robot is asked to establish a link with other robots, it enters the robot-chain state and starts forming links with the other robots. When a branch in a robot chain network is disbanded, the links are retracted and the robots enter the mobile state. At any time, a robot plays one of the following *roles*: 1) *exploring robots*, 2) *foraging robots* and 3) *robot-chain robots*. An exploring robot can only enter the mobile state. A foraging robot can enter the mobile state, the resource-collecting state, and the resource-dumping state. A robot-chain robot can enter the mobile state and the robot-chain state. A robot can change its role only when it is in the mobile state.

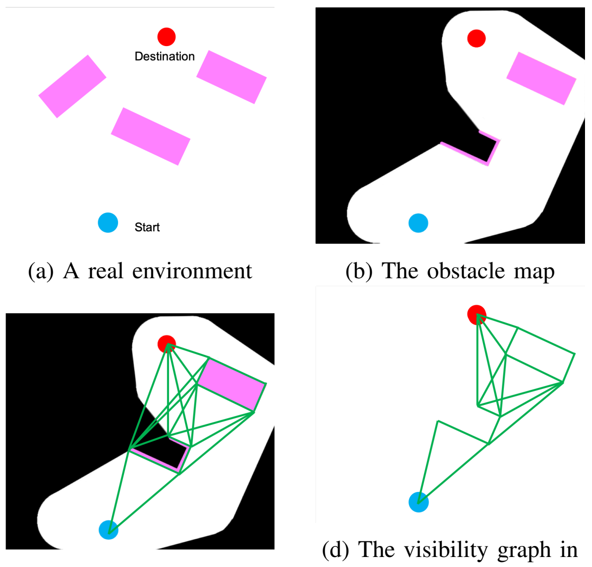
An *endpoint* is the last robot of a branch that connects to a resource cluster. Each foraging robot has a *preferred* endpoint, which is typically the nearest endpoint at the current position. A foraging robot can put its collected resources to the preferred endpoint, and then the resource will be transferred to the central depot via the network. If the resource-holder of an endpoint is full or another foraging robot is using the endpoint, the foraging robot has to wait until the endpoint becomes available. The resource-holder of an endpoint is full when *congestion* occurs on the network, causing the resources to pile up to the endpoint. Congestion is mainly caused by the limited capacity of the link on the *main branch* that connects to the central depot. But this congestion can be alleviated if resources can be transferred quickly through the main branch. In our experiments, there are four initial robot chain networks in the four quadrants of an arena at the beginning.

IV. Foraging Robots' Behavior

A foraging robot remembers the location of the last resource cluster it visited and collects resources from the cluster until there is no more resource in the cluster. When all resources in a cluster have been collected, the foraging robot will go to another cluster to collect resources or become an exploring robot and explore the arena to find other resource clusters and obstacles using the exploration strategy in [8].

Robots share their locations with the nearby robot, especially the adjacent robots on a robot chain. They also share the locations of the resources and the obstacles they know. The central depot acts as an information hub that redirects information via robot chain networks. Unless a robot is exploring a location that is far away from any robot chain networks, it can receive the shared information from the central depot instantly. Robots that are not close to any network will have to get closer to the network from time to time to exchange information with other robots.

When a robot discovers an obstacle, it will integrate the information of the obstacle collected by its lidars (Fig. 3) by updating a shared data structure called the *obstacle map* (Fig. 4), which partitions the arena map into three kinds of regions: 1) unknown regions, 2) empty regions and 3) obstacle regions. In Fig. 4b, they are the black regions,



(a) A real environment (b) The obstacle map
(c) The visibility graph (d) The visibility graph in the empty regions

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

the white regions, and the magenta regions, respectively. Foraging robots use the obstacle map to navigate in the arena. To visit a location, a foraging robot computes a *visibility graph* in the obstacle map (Fig. 4c) and removes all edges that are not in the empty regions (Fig. 4d). The foraging robot moves along the shortest path to the destination in the visibility graph. When moving toward the destination, the foraging robot will update the visibility graph upon getting more information about the unknown regions and then recompute the shortest path. If there is no path to reach the destination, the foraging robot will explore the unknown region randomly until there is a path to the destination.

V. Modification of Robot Chain Networks

A robot chain network supports five modification operations: (1) adding a branch to an existing network, (2) deleting a branch from an existing network, (3) relocating a subtree in an existing network, (4) establishing a new network, and (5) disbanding a network. A *high-level controller* in Sec VI decides when to apply these operations. Typically, when a new resource cluster is discovered, the high-level controller will consider either adding a branch connecting an existing network to the new resource cluster or relocating a subtree in an existing network to include the new resource cluster as a new endpoint in the network. When a resource cluster has no more resources, the branch to the resource cluster will be deleted. If all resource clusters at the endpoints of a network are empty, the entire network will be disbanded. If some resource clusters are not close to any existing networks, a new network for these clusters will be established.

The relocation of subtrees relies on the solution to the obstacle-avoiding EST problem. For each resource cluster in a subtree π including the new resource cluster, we pick a point within a distance R_{end} from the center of the cluster but outside the cluster as an endpoint. The endpoint should be between the center of the cluster and the network. Given the coordinates of a set of endpoints and the obstacle map, we first compute a nearly optimal Steiner tree that connects the central depot to the endpoints while avoiding the obstacles. We modified the EST solver in [32] to make it works with obstacle avoidance using some heuristics in [29] and [30]. Second, we compute the *target locations* of the robots on the edges of the Steiner tree. The target locations should satisfy all physical constraints, such as the maximum link distance and the Y-junction configurations. After computing the target locations, the high-level controller will ask the endpoints in π to stop receiving resources from foraging robots and wait until all existing resources on the chain leave π. Note that other parts of the network can continue to function while relocating π. After that, the robot-chain robots in π are disbanded and become foraging robots. Then the set of foraging robots that are close to the target locations moves to the target locations along the shortest paths in the visibility graphs.

To encourage robots to explore the regions around the new subtree, the robots are given a certain amount of time, namely t_{explore}, to freely explore the region near the subtree while moving to the target locations. The key idea of the exploration procedure is to maintain the current best configuration of the subtree during the exploration and make sure that there is enough time for robots to return to the last best configuration before the allotted exploration time t_{explore} is expired. When robots collect new information about the obstacles during exploration, the high-level controller will recompute the Steiner tree and the target locations and then ask the robots to move to the new target locations.

The operation of establishing a new network is the same as the relocation of a subtree, except no existing subtree will be disbanded and the subtree's root is the central depot. Adding a new branch is the same as the addition of new robot chains using the visibility graph in Fig. 4 as discussed in [9].

VI. THE HIGH-LEVEL CONTROLLER

The foraging robots and the dynamics of robot chain networks are managed by a high-level controller at the central depot. The controller divides all robots into three groups based on their roles as discussed in Sec III. An exploring robot will move to the nearest unknown regions in the obstacle maps as shown in Fig. 4, but visit the nearest endpoints of any network for every period t_{info} to report the collected information. The foraging robots will be evenly distributed to the endpoints of the networks and they repeatedly collect resources from nearby resource clusters and then dump the resources to the endpoints. The robot-chain robots act upon receiving resources from the upstream and pushing the resources downstream.

Robots can switch roles only when a modification of robot chain networks occurs. The high-level controller considers modifying any networks only at a fixed time interval t_{protect}. No new modification can be made before the end of a time interval to prevent modifications from happening too frequently. At the end of a time interval, the controller checks whether some branches or networks should be removed and whether there are resource clusters that have not been reached by any network. To reach these resource clusters, the high-level controller has three options: adding some new branches to an existing network, relocating a subtree in an existing network, or establishing a new network. The last option is selected only when the resource clusters are too far away from existing networks.

We opt for building large robot chain networks with many endpoints after setting aside a certain number of robots as exploring robots. Hence, the high-level controller prefers adding new branches instead of relocating a subtree. When adding a new branch, some foraging robots have to convert into robot-chain robots. As long as the average number of foraging robots for each endpoint in a robot network is not less than a given number N_{forage}, the addition of new branches should be allowed; otherwise, the high-level controller should relocate a subtree. Before relocating a subtree, the controller calculates the number of robots that are available to establish the new subtree. If the number of robots is large enough for filling out the minimum Steiner tree, the relocation proceeds; otherwise, the relocation is put on hold until some other branches are disbanded and more robots become available. Since the new Steiner tree can be smaller than the subtree before relocation, some robot-chain robots can become foraging robots after relocation.

VII. EXPERIMENTAL CONFIGURATIONS

To evaluate our robot chain network algorithm RC$_{netwrok}$, we conducted four sets of experiments in Autonomous Robots Go Swarming (ARGoS) [35]. In the first two experiments, we compared our proposed algorithm RC$_{network}$ with the other two robot chain algorithms in [9], (RC$_{no}$ and RC$_{one}$). In RC$_{no}$, robot chains can relocate close to multiple resource clusters, but the length of robot chains does not change. In RC$_{one}$, the robot chains can be extended to the center of multiple resource clusters. In the first experiment, the arena size is $20m \times 20m$. The number of clusters is 40, and the shape of clusters is 5×5. The number of robots is 40, 50, 60, and 70. Initially, 4 robot chains are distributed uniformly and 4 robots in each chain. The simulated foraging time is 30 minutes. The exploration time of robots in robot chains is 2 minutes and the frequency of relocation checking is 6 minutes. For testing the flexibility of the proposed algorithm, we conducted the second experiment where the arena, resources, and robot swarm size are larger. The experimental setup is summarized in Table I.

In the third experiment, we measured how the frequency of network modification and the frequency of robot chains' relocation t_{protect} affect the foraging performance in the 20×20 arena with obstacles. The experimental setup is summarized in Table II. In the last experiment, we measured how the exploration time t_{explore} affects the modification of the networks or robot chains in the $20m \times 20m$ arena with obstacles. We measure the total distance between the last robots of robot chains and the closest clusters of resources. Three groups of experiments were conducted based on the

TABLE I: The Configuration of Experiments 1 & 2

Arena Size ($m \times m$)	20×20	30×30
Number of resource	1000: $(40 \times 5 \times 5)$	2250: $(90 \times 5 \times 5)$
Number of robots	40, 50, 60, 70	60, 80, 100, 120
Number of robots in each initial robot chain	4	9
Foraging time (minute)	30	30
t_{explore} (minute)	2	2
t_{protect} (minute)	6	6

TABLE II: The Configuration of Experiments 3

Arena Size ($m \times m$)	20×20
Number of resource	1000
Number of robots	40, 60, 80
Foraging time (minute)	30
Number of obstacles	4, 8, 16, 32
Percentage of robots in the initial robot chains	30%
t_{explore} (minute)	2
t_{protect} (minute)	3, 6, 12, 24, 30

TABLE III: The Configuration of Experiment 4

Arena Size ($m \times m$)	20×20
Number of obstacles	2, 4, 6, 8
Number of robots in each robot chain	4, 5, 6
t_{explore} (s)	30, 60, 120, 240

number of robots in each robot chain. The experimental setup is summarized in Table III.

VIII. 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 spend to avoid collisions with other robots and the boundary of the arena. We checked whether the foraging performance varies systematically with different configurations and statistically analyzed 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 three algorithms in Experiment 1. The foraging performance increases as the number of robots increases. When the number of robots is 40, all performances are low and similar. When the number of robots is 60, the performance of RC$_{network}$ outperforms other algorithms. When the number of robots is 70, the performance of RC$_{network}$ is 13.6% higher than RC$_{one}$ and 19.5% higher than RC$_{no}$.

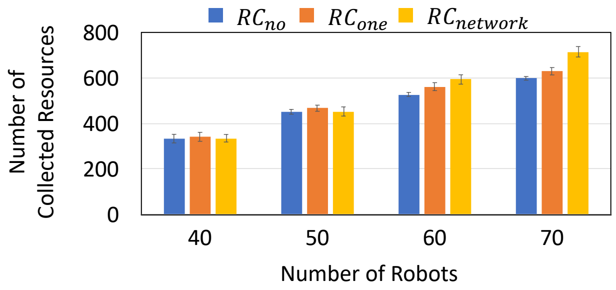


Fig. 5: Foraging performance in Experiment 1.

Fig. 6 compares the collision time of all three algorithms in Experiment 1. The collision time increases as the number of robots increases. The difference is higher when the number of robots is 70, the collision time in the RC$_{network}$ is 4.7% shorter than the time in RC$_{one}$ and 18.6% shorter than RC$_{no}$.

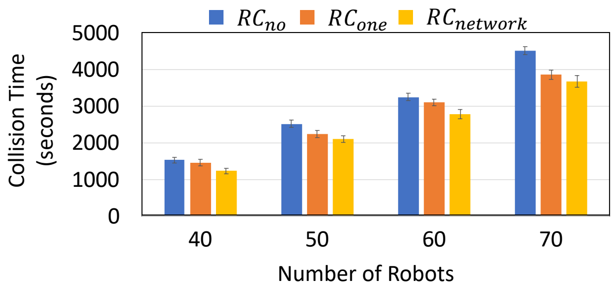


Fig. 6: The collision time of each swarm in Experiment 1.

Compared to the performance in Experiment 1, Fig. 7 demonstrates the same trend of the foraging performance in Experiment 2. When the number of robots is 120, the performance of RC$_{network}$ outperforms other algorithms. When the number of robots is 80, the performance of RC$_{network}$ is 14.9% higher than RC$_{one}$ and 21.0% higher than RC$_{no}$. Fig. 8 also demonstrates the same trend of the collision time in Experiment 2. When the number of robots is 120, the collision time in the RC$_{network}$ is 4.5% shorter than the time in RC$_{one}$ and 17.1% shorter than RC$_{no}$.

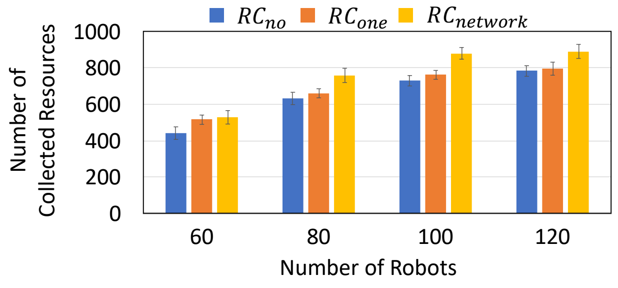


Fig. 7: Foraging performance in Experiment 2.

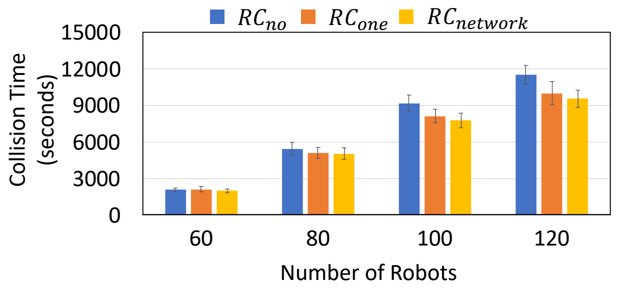


Fig. 8: The collision time of each swarm in Experiment 2.

Fig. 9 demonstrates the foraging performance in Experiment 3. The foraging performance of the robot chains or networks decreases as the number of obstacles increases. When the number of robots is 40, the trend of the performance is not obvious. When the number of robots is 60 and 80, the performance with $t_{\text{protect}} = 3$ is significantly lower than others. In all cases, the performance with $t_{\text{protect}} = 6$

is slightly better than the performance with $t_{\text{protect}} = 12$, but always better than others.

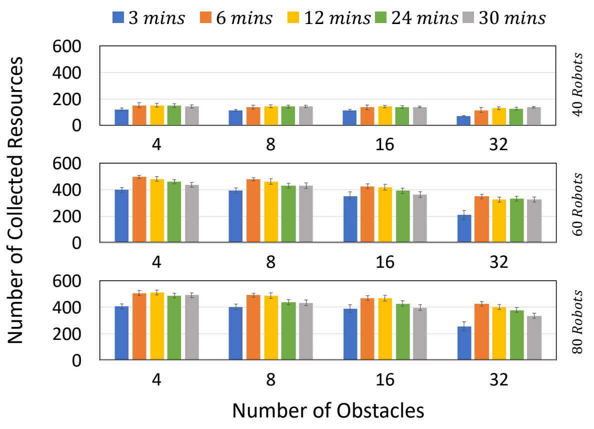


Fig. 9: Foraging performance using different t_{protect} in Experiment 3.

Fig. 10 demonstrates the performance of network modifications and robot chains' relocation in Experiment 4. The performance has the same trend in all three groups of experiments. The total distance d between the last robots of robot chains and the closest clusters increases as the number of obstacles increases. The shorter exploration time results in a longer distance d. The robot chain algorithm with the known obstacle map always has the best performance. When the number of robots in a robot chain is 6 and the number of obstacles is 8, the distance using the robot chain algorithm with obstacle maps is 35.8%, 52.6%, 181.2%, and 274.6% shorter than the distance using the robot chain algorithm with the exploration time, 240, 120, 60, and 30 seconds, respectively.

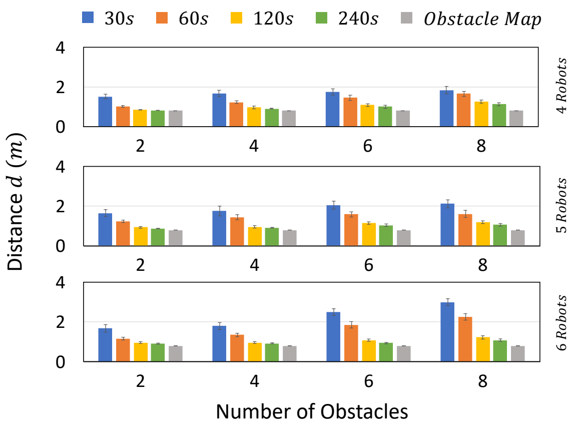


Fig. 10: Relocating performance using different t_{explore} in Experiment 4.

IX. Discussion and Future Work

In [9], we have already demonstrated that the dynamic robot chain algorithm outperforms the MPFA with dynamic depots. In this paper, we designed tree-like robot chain networks for foraging robot swarms, based on obstacle-avoiding Euclidean Steiner trees. The collected objects can be transported on the robot chain to the central collection zone. Compared to the existing dynamic depot foraging systems, our network formation can overcome two major limitations of the central place foraging algorithm (CPFA): congestion in transportation and the long travel distance between the location of one depot and the locations of multiple clusters. We presented the robot chain network algorithm $\text{RC}_{network}$ and described the modification procedure of subtrees. Our experiments (Fig. 5 and Fig. 7) show that $\text{RC}_{network}$ outperforms other two robot chain algorithms with no or one single branch, RC_{no} and RC_{one}, respectively.

The reasons for the superior performance of robot chain networks are as follows. When there are multiple discovered clusters, one robot chain is insufficient to serve the foraging in multiple clusters. When there are no extended branches, robots need to travel between the multiple clusters and the location of the last robot in the robot chain. Robots collide with each other as in the MPFA with dynamic depots. When there are multiple branches to the clusters, many robots are in the robot chains and branches, and a smaller number of foraging robots travel in the arena. Therefore, the collision times in $\text{RC}_{network}$ are lower (Fig. 6 and Fig. 8). A real-time adaptive strategy is a key component of $\text{RC}_{network}$ since it can create multiple branches that connect to multiple clusters. This reduces the travel time of foraging robots and hence increases the foraging performance.

The foraging performance in the third experiment (Fig. 9) indicates that the frequency of network modification or robot chain's relocation should be selected carefully. The robot chain, either with a high relocation frequency or without relocation, has a low performance. The relocation is efficient, but there is a cost of relocation. There are tradeoffs between the relocation and the cost. The results in the last experiment indicate the exploration time is also critical in the modification. The longer exploration time results in better robot chain networks, but the rate of improvement decreases.

Thus, by using dynamic robot chain networks that adapt to the distribution of resources, $\text{RC}_{network}$ is an efficient solution that mitigates the issue of bottlenecks and improves the foraging performance. This work shows that dynamic robot chain networks can significantly enhance the transportation performance in foraging swarm robotics systems. The advantages of this novel approach are: 1) it provides more efficient transportation than existing no branch or single branch robot chain algorithms; 2) it has less collision time among robots; and 3) it is adaptive to local environments. In the future, we will design more efficient robot chain networks to avoid obstacles in more complex environments.

Acknowledgments

This work has been taken place at UNIST and was supported by NRF (2016R1D1A1B0101359816 and 2022R1A2C10121681) and UNIST (1.210036.01 and 1.220048.01).

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. T. Winfield, "Modeling and optimization of adaptive foraging in swarm robotic systems," *International Journal of Robotics Research*, vol. 29, no. 14, pp. 1743–1760, Dec. 2010.

[3] Wenguo Liu, A. F. Winfield, Jin Sa, Jie Chen, and Lihua Dou, "Towards energy optimization: Emergent task allocation in a swarm of foraging robots," *Adaptive Behavior*, 2007.

[4] E. Castello, T. Yamamoto, F. D. 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*, 2016.

[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 Systems Conference, arXiv:1612.00480*, 2016.

[7] Q. Lu, J. P. Hecker, and M. E. Moses, "The MPFA: A Multiple-Place Foraging Algorithm for Biologically-Inspired Robot Swarms," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 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, Q. Lu, and T.-C. Au, "Multiple-place swarm foraging with dynamic robot chains," in *ICRA*, 2021, to appear.

[10] D. Lee and T.-C. Au, "Automatic configuration of mobile conveyor lines," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3841–3846.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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 *Intl. Conf. on Robotics and Automation (ICRA)*, May 2019, pp. 9285–9291.

[17] 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.

[18] 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.

[19] Q. Lu, M. E. Moses, and J. P. Hecker, "A scalable and adaptable multiple-place foraging algorithm for ant-inspired robot swarms." Robotics Science and Systems (RSS) workshop on On-line decision-making in multi-robot coordination, 2016.

[20] Q. Lu, J. P. Hecker, and M. E. Moses, "The MPFA: A multiple-place foraging algorithm for biologically-inspired robot swarms," *IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[21] Q. Lu, J. Hecker, and M. Moses, "Multiple-place swarm foraging with dynamic depots," *Autonomous Robots*, vol. 42(4), pp. 909–926, 2018.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] P. Crescenzi, V. Kann, M. Halldórsson, M. Karpinski, and G. Woeginger, "Minimum geometric steiner tree," *A Compendium of NP Optimization Problems*, 2000.

[28] M. Zachariasen and P. Winter, "Obstacle-avoiding euclidean steiner trees in the plane: an exact algorithm," in *Workshop on Algorithm Engineering and Experimentation*, 1999, pp. 286–299.

[29] V. Parque and T. Miyashita, "Obstacle-avoiding euclidean steiner trees by n-star bundles," in *IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2018, pp. 315–319.

[30] L. Garrote, L. Martins, U. J. Nunes, and M. Zachariasen, "Weighted euclidean steiner trees for disaster-aware network design," in *International Conference on the Design of Reliable Communication Networks (DRCN)*, 2019, pp. 138–145.

[31] H. Hamann and H. Wörn, "Aggregating robots compute: An adaptive heuristic for the euclidean steiner tree problem," in *From Animals to Animats 10*, M. Asada, J. C. T. Hallam, J.-A. Meyer, and J. Tani, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 447–456.

[32] W. D. Smith, "How to find steiner minimal trees in euclidean d-space," *Algorithmica*, vol. 7, pp. 137–177, 1992.

[33] E. N. Gilbert and H. O. Pollak, "Steiner minimal trees," *SIAM Journal on Applied Mathematics*, vol. 16, no. 1, pp. 1–29, 1968.

[34] F. K. Hwang, D. S. Richards, and P. Winter, "The steiner tree problem," *Annals of Discrete Mathematics*, vol. 53, 1992.

[35] 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.