Queue-Based Traffic Management for Collision Reduction in Scalable Swarm Foraging

Tameem Uz Zaman[1], Arturo Gonzalez[1], Tsz-Chiu Au[2], and Qi Lu[1]

***Abstract*— Swarm robotics provides robust and scalable solutions for tasks such as foraging; however, congestion near central collection zones remains a significant challenge, particularly as swarm size increases. Conventional approaches — such as static path planning or local repulsion-based methods — often fail to prevent inter-robot collisions and bottlenecks in these zones. Drawing inspiration from real-world queuing systems, we propose regulating traffic at central collection points by organizing robots into queues, allowing them to take turns when unloading resources. This paper presents a comparative analysis of two queuing strategies — single-queue and multi-queue systems — designed to alleviate congestion at central collection zones. Both systems ensure high throughput and eliminate gridlock. In addition, we evaluate two algorithms that guide robots to the endpoints of these queues. Experimental results using the ARGoS simulation environment demonstrate that both queuing strategies significantly reduce congestion around collection zones and lower the average delay in resource delivery. These findings suggest that queuing mechanisms can substantially improve the efficiency of swarm foraging. Furthermore, multi-queue systems outperform single-queue systems in terms of delay, as they allow robots to reach queue endpoints more quickly.**

I. INTRODUCTION

Swarm robotics investigates the coordination of numerous simple, autonomous robots to accomplish complex tasks in an efficient and scalable manner. Contemporary research in this field encompasses a range of collective behaviors, including self-organization [1], [2], task allocation [3], aggregation [4], [5], object sorting [6], [7], and foraging [8]–[11]. Among these, *foraging* is one of the most widely studied problems, wherein robots autonomously navigate and explore an environment to locate, retrieve, and deliver objects—such as targets, resources, or data—to a predefined collection zone. Robotic foraging is strongly inspired by the behavior of social insects (e.g., ants and bees), which exhibit effective decentralized strategies for gathering food across large, unfamiliar terrains to sustain their colonies.

Traditional foraging algorithms—such as the Centrally Placed Foraging Algorithm (CPFA)—typically assume that increasing the number of robots in a swarm enhances overall efficiency [9]. However, both empirical studies and simulation results challenge this assumption, revealing that robots often face significant congestion near the central collection zone (i.e., the central depot). This congestion forces robots to take inefficient detours or wait for access to the drop-off area. This behavior is evident in ARGoS simulations [12], where 64 robots forage within an 8×8 arena (see Fig. 1). As the number of robots delivering resources to the central depot increases, the time spent on collision avoidance grows substantially. This effect presents challenges in achieving optimal system efficiency in large-scale swarms.

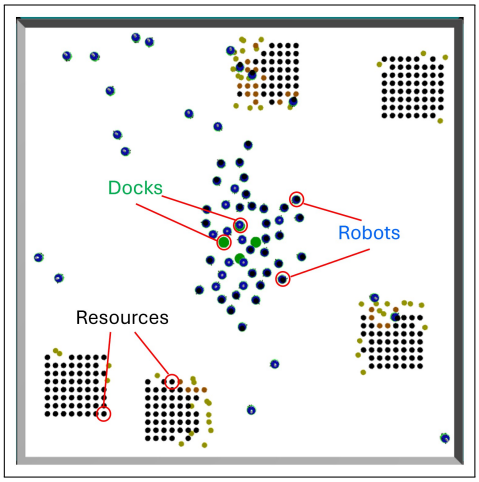

Fig. 1: An illustration of congestion around the central collection zone in the ARGoS simulation. 64 robots are deployed in the 8×8 m arena. 320 resources are distributed in five 8×8 clusters.

Alternative strategies have been proposed to mitigate congestion at the central depot. For instance, Lu et al. introduced multiple-place foraging systems, which employ helper robots called dynamic depots to collect resources from foraging robots and deliver them to the central depot [13]–[15]. While dynamic depots help reduce robot density near the central depot, congestion can still occur under heavy traffic conditions. To address this, dynamic robot chains—sequences of robots capable of passing resources over a distance—have been proposed as a replacement for dynamic depots [13], [16]. In this approach, a foraging robot places collected resources onto the last robot of a mobile conveyor-like robot chain [17], which then relays the resources to the central depot. This mechanism has been shown to alleviate congestion near the central depot significantly [13], [16].

In this paper, we address the congestion problem by drawing inspiration from queueing systems and simple coordination mechanisms used in modern transportation networks and supermarkets. At intersections, vehicles form queues and resolve conflicts through stop signs and traffic signals. In supermarkets, customers queue to take turns at checkout counters. Motivated by these systems, we propose several

[1]Department of Computer Science, The University of Texas Rio Grande Valley, USA. {tameemuz.zaman01,arturo.gonzalez11, qi.lu}@utrgv.edu

[2]Department of Computer Science, Texas State University, USA. chiu.au@txstate.edu

queue-based strategies for managing traffic near the central depot. Specifically, we explore configurations where the central depot is surrounded by either a single queue or multiple queues, ensuring that gridlock is avoided. Conventional wisdom from retail environments suggests that single-queue systems offer greater fairness to customers by reducing uncertainty in wait times. However, in robotic systems, finding the endpoint of a queue can be challenging and may introduce delays. In contrast, multi-queue systems provide multiple endpoints, enabling robots to join queues more quickly. To support this approach, we introduce endpoint-finding algorithms that help robots locate the endpoints of queues. In summary, the key contributions of this paper are:

- We propose two queue wrapping strategies—the single-queue wrapping strategy and the multi-queue wrapping strategy—for managing traffic at the central depot in foraging swarm systems.
- We develop endpoint-finding algorithms that enable robots to efficiently locate and join the appropriate queues, minimizing delay and improving throughput.
- We implement and evaluate the proposed queueing strategies and endpoint-finding algorithms through extensive simulations in the ARGoS environment.

This paper is organized as follows. After presenting the related work in Sec. II, we define the multirobot foraging problem in Sec. III. Then, we describe the single-queue and multi-queue wrapping strategies in Sec. IV, and present the endpoint finding algorithms in Sec V. After that, we define the delivery query strategies in Sec VI. Finally, we present our experimental evaluation in Sec. VII and Sec. VIII, discuss the results in Sec. IX, and conclude in Sec. X.

II. RELATED WORK

Congestion in swarm robotics—particularly in central-place foraging tasks—has received growing attention in recent years. Many algorithms have been proposed to enhance task efficiency by minimizing robot collisions and alleviating traffic bottlenecks. Spiral-based foraging algorithms have demonstrated promise for efficient area coverage and resource collection. Aggarwal et al. [18] compared deterministic spiral trajectories with stochastic, ant-inspired strategies, showing that spiral paths generally yield higher resource collection rates in simulation. However, their effectiveness tends to diminish under real-world conditions, where factors such as physical obstacles and sensor noise introduce additional challenges. Building on this approach, Bharaswadkar et al. [19] proposed the Clustered-CPFA, which incorporates waypoint planning and deliberate delays to mitigate robot interference in high-density zones. While this method effectively reduces congestion around clustered resources, it remains largely reactive and depends heavily on prior knowledge of potential congestion areas.

Other studies have tackled congestion through predictive and distributed control strategies. Marcolino et al. investigated various techniques—including random delays, region separation, and hybrid approaches—to manage competition among robots near shared resources [20], [21]. Although these methods help smooth robot arrival patterns, they often reduce overall efficiency by introducing intentional delays or limiting access to certain areas. More recently, Passos et al. proposed algorithms such as Single Queue Former (SQF) and Touch and Run Vector Fields (TRVF), which aim to organize robots into queues or guide them via vector fields to improve throughput in high-demand regions [22]. However, these algorithms typically rely on potential fields, making them susceptible to problems such as local minima, and their performance can vary significantly depending on robot density and environmental scale. Hierarchical control strategies have also been explored for congestion mitigation. For example, Vinicius and Chaimowicz introduced an abstraction-based method using elliptical formations to coordinate robot groups and prevent navigation deadlocks in multi-group scenarios [23]. However, these strategies may lack the flexibility to adapt to changing environments.

Learning-based approaches, such as decentralized deep reinforcement learning (DRL), provide adaptive capabilities for collision avoidance and dynamic path planning. Long et al. [24] demonstrated that DRL policies operating at the sensor level can effectively manage unforeseen obstacles and varying robot densities, outperforming traditional rule-based strategies in complex multi-robot environments. However, DRL techniques often require substantial training data and computational resources, and their decision-making processes are typically opaque, posing concerns for deployment in safety-critical applications. Soma et al. [25] found that increases in swarm size can degrade convergence speed and task efficiency in the absence of adequate congestion mitigation strategies. These findings underscore the need for scalable and proactive control mechanisms to ensure the effective operation of large-scale swarms. Together, these studies illustrate the strengths and limitations of both conventional and learning-based methods. Building on these insights, our research integrates biologically inspired spiral foraging trajectories with adaptive congestion detection mechanisms to enhance swarm foraging performance.

III. THE MULTIROBOT FORAGING PROBLEM

In a foraging task, a team of robots collaborates to collect resources distributed throughout an *arena*. Fig. 1 illustrates an arena where multiple resource clusters are dispersed. These resources are grouped into several clusters, and their locations are initially unknown to the robots. The robots must explore the arena to discover the locations of these clusters. Once a cluster is located, a robot retrieves a single unit of resource and transports it to a centrally located *central depot*. The environment also contains obstacles that obstruct the robots' movement and limit their sensing capabilities. Initially, the robots have no prior knowledge of the locations of either the resource clusters or the obstacles. The objective of the robot team is to discover and retrieve as many resources as possible and deliver them to the central depot.

The formal definition of multiple robot foraging is:

- $\mathcal{E} \subseteq \mathbb{R}^n$ be the **workspace** (e.g., 2D or 3D environment) for the arena.

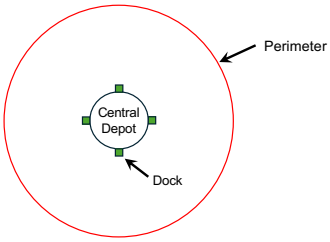

Fig. 2: A central depot (the black circle) and its perimeter (the red circle). The green squares are the docks.

- $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$ be the set of **robots**, where N is the total number of robots. Each robot r_i has:
  - A **state** $x_i(t) \in \mathcal{X}$, where $\mathcal{X}$ is the robot's state space (e.g., position, velocity).
  - A **policy** $\pi_i : \mathcal{X} \times \mathcal{E} \to \mathcal{U}$, mapping observations to control inputs $u_i(t) \in \mathcal{U}$.
  - A **sensing radius** $\rho > 0$.
  - A **communication range** $\gamma > 0$ for local communication with nearby robots.
- $\mathcal{T} = \{t_1, t_2, \ldots, t_M\}$, where $t_i \in \mathcal{E}$ for $1 \leq i \leq M$, be a finite set of **target objects** (or resources) to be collected.
- $\mathcal{B} \subseteq \mathcal{E}$ denote the **central depot** (i.e., the **home base**).

Each robot is an omnidirectional mobile platform equipped with a gripper, a LiDAR sensor, a wireless communication module, a resource detection device, and a resource holder with limited storage capacity. The LiDAR sensor enables the robot to detect obstacles and identify other robots within its sensing range. The resource detection device allows the robot to pinpoint the locations of all resources within its detection radius. Communication between robots is enabled when the distance between any two robots is less than γ, allowing them to exchange information and coordinate their actions.

Most existing studies on multirobot foraging assume a central depot where robots can unload resources at any location within the depot area. In contrast, this paper introduces a more structured model in which robots are required to unload resources at designated *docks* along the boundary of the central depot. The central depot is defined as a two-dimensional closed shape with K docks positioned on its perimeter, where $K \geq 1$, as illustrated in Fig. 2. To complete the unloading process, a robot must remain at a dock for a fixed duration of T_{unload}.

The *perimeter* of a central depot refers to the boundary of the designated area surrounding the depot where queues may be formed. Only robots that intend to unload resources at the central depot are permitted to enter this perimeter. Once a robot completes the unloading process, it must exit the perimeter. All robots possess knowledge of the queue structures within the perimeter. The *time delay* associated with unloading is defined as the time interval between a robot's entry into the perimeter and its subsequent departure.

IV. QUEUE WRAPPING STRATEGIES

If a queue is formed as a straight line passing through a dock at the central depot, it cannot accommodate many robots due to spatial limitations. To maximize the use of available space within the perimeter, the queue must wrap around the central depot. In this section, we describe two queue wrapping strategies—one designed for single-queue systems and the other for multi-queue systems—that enable incoming robots to efficiently queue within the perimeter while allowing outgoing robots to exit the perimeter quickly.

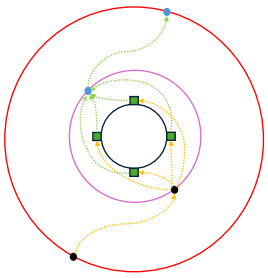

Fig. 3: Single-Queue Wrapping Strategies.

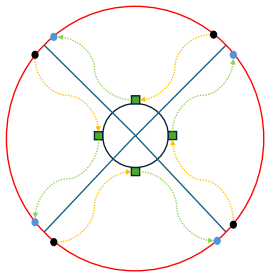

Fig. 4: Multi-Queue Wrapping Strategies.

A. Single-Queue Wrapping Strategies

In single-queue wrapping strategies, the perimeter of the central depot is divided into two regions: the *inner region* and the *outer region*. As illustrated in Fig.3, the area enclosed by the purple circle constitutes the inner region, while the area between the red and purple circles defines the outer region. The outer region has one *entry point* and one *exit point*, indicated by the black dot and the blue dot on the red circle in Fig.3, respectively. Similarly, the inner region also includes a single *entry point* and *exit point*, represented by the black dot and blue dot on the purple circle.

A *line* refers to a predefined path along which robots can move. Within the inner region, an *inner queue* is defined as a line extending from the entry point to the exit point, passing through a dock. Each dock is associated with a unique inner queue. To prevent congestion in the inner region, each inner queue accommodates at most two robots at any given time. A robot may be dispatched to an inner queue if and only if (1) the dock is unoccupied, and (2) there is no robot positioned between the dock and the entry point. Similarly, after completing the unloading process at the dock, a robot may proceed toward the exit point if and only if (1) the exit point is unoccupied, and (2) there is no robot between the dock and the exit point.

The outer region contains two lines: the *entry line* and the *exit line*. The entry line connects the entry point of the outer region to the entry point of the inner region, while the exit line connects the exit point of the inner region to the exit point of the outer region. Robots queue along the entry line as they wait to enter the inner region for resource unloading. Ideally, the exit line should be as short as possible, as robots on this line exit the perimeter immediately after unloading. In contrast, the entry line should be significantly longer to accommodate the higher number of robots waiting, due to the time required to complete the unloading process at the docks in the inner region.

B. Multi-Queue Wrapping Strategies

In multi-queue wrapping strategies, the central depot's surrounding area is divided into distinct *sectors*, with each sector associated with a single dock. For example, in Fig. 4, there are four sectors corresponding to four docks. These sectors are mutually exclusive, with no overlap between them. Each sector has one entry point and one exit point. Unlike the shared entry and exit lines in single-queue strategies, each sector has its own dedicated *entry line*, which directly connects the sector's entry point to its dock, and a dedicated *exit line*, which directly connects the dock to the sector's exit point. The entry and exit lines are non-intersecting. Typically, the exit line is designed to be as short as possible, allowing robots to exit the perimeter promptly after unloading. In contrast, the entry line is designed to utilize most of the space within the sector to accommodate a larger number of robots waiting to unload their resources at the dock.

C. Comparisons between the Queue Wrapping Strategies

In multi-queue wrapping strategies, robots can quickly locate the endpoint of one of the queues due to the presence of multiple queues. However, when a large number of robots approach from the same direction, queue lengths can become highly imbalanced. In extreme cases, a single queue may contain all incoming robots while the others remain empty, potentially leading to increased average delay. In contrast, robots in single-queue wrapping strategies need more time to locate the queue's endpoint, but since all robots ultimately join the same queue, the average delay can be lower—particularly when the time spent locating the queue is negligible compared to the time spent waiting within it.

D. Intersections and Gridlock Prevention

In both queue wrapping strategies, lines may intersect or even overlap within the queueing system. When two robots reach an intersection simultaneously, a right-of-way policy must be enforced to prevent collisions. More critically, the system must ensure that no robot becomes indefinitely blocked at an intersection due to gridlock. When robots from different inner queues arrive at an intersection, conflicts are resolved based on a first-come-first-served (FCFS) policy—the robot that arrives first is granted the right of way. If two or more robots reach the intersection at the same time, right-of-way is assigned sequentially in a counter-clockwise order. To prevent deadlock, the design must guarantee that no robot is stuck at an intersection, permanently blocking other robots from entering the intersection. Since the traffic flows only from the entry points to the exit points on the perimeter and all robots will eventually leave the perimeter from the exit points, a robot at an intersection can eventually leave the intersection.

V. ENDPOINT FINDING ALGORITHMS

When a robot enters the perimeter, it begins searching for the tail of a queue and joins the queue by positioning itself behind the last robot. We refer to this last robot as the *endpoint* of the queue. In this section, we present two algorithms designed to enable robots to efficiently locate the endpoint of a queue.

A. The Entry Point Joining Algorithm

In both queue wrapping strategies, entry lines originate from designated entry points on the perimeter. A simple method for locating an endpoint is to first search for an entry point on the perimeter and then follow the corresponding entry line to the endpoint. This approach guarantees that the endpoint will eventually be found. Specifically, the perimeter is divided into several arc segments, each associated with a *preferred direction* that guides robots toward the nearest entry point. The preferred direction is defined as a tangent to the perimeter and can be either to the left or right, relative to the incoming robot's orientation. When a robot reaches the perimeter, it proceeds along the perimeter in the preferred direction until it locates an entry point.

B. The Tail Joining Algorithm

While the entry point joining algorithm is simple to implement, it can be overly conservative, as it requires robots to follow potentially long and winding entry lines. To improve efficiency, we propose the *tail joining* algorithm, which enables robots to skip empty segments of the entry line by directly searching for the queue endpoint. In single-queue systems, when an incoming robot, denoted as v_1, enters the perimeter, it does not follow the perimeter to locate an entry point. Instead, it moves in a straight line toward the inner region. If v_1 encounters another robot v_2, it initiates communication to determine whether v_2 is already in the queue. If v_2 is not in the queue, then both robots are still searching for the endpoint and will resolve the potential collision by randomly yielding to one another. If v_2 is on the queue, v_1 will proceed in the opposite direction of the line on which v_2 is located. If v_2 is not the last robot in the queue, v_1 will eventually encounter another robot v_3 positioned behind v_2, and the search process will continue accordingly. However, if v_2 is the last robot in the queue, v_1 will simply join the queue behind it. If v_1 reaches the boundary of the inner region without encountering any robot, it will switch to a fallback strategy by moving to the nearest line and following it to locate the endpoint. This algorithm is also applicable to the multi-queue systems if we define the inner region appropriately.

VI. DELIVERY QUEUE STRATEGIES

Each combination of a queue wrapping strategy and an endpoint-finding algorithm defines a distinct *delivery queue strategy*. Given the two queue wrapping strategies and two endpoint-finding algorithms introduced earlier, we identify four possible delivery queue strategies, as illustrated in Fig. 5. In single-queue systems, if robots join the queue using the entry-point joining method, the strategy is denoted as Q_{SE}. If they use tail joining instead, the strategy is denoted as Q_{ST}. Similarly, in multi-queue systems, we define two strategies based on the method of joining queues: Q_{ME} for entry-point joining and Q_{MT} for tail joining.

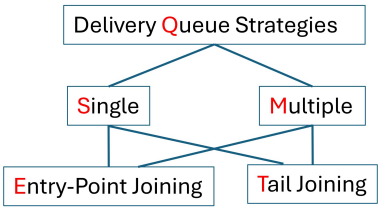

Fig. 5: An overview of proposed delivery queue strategies.

VII. EXPERIMENTAL SETUP

We designed two types of queue layouts: a single spiral queue and multiple sector-based serpentine queues. Each queue is planned within a circular red region with a diameter of 4 meters, considering the diameter of the footbots to be 0.17 meters. The layouts are optimized to utilize the available space within the red region as efficiently as possible.

When the queue changes direction, such as in serpentine or spiral turns—the spacing between adjacent lanes is set to 0.20 meters, ensuring collision-free movement as robots traverse in opposite or parallel directions. An illustration of both the single and multi-queue configurations used in the simulation is shown in Figs. 6 and 7.

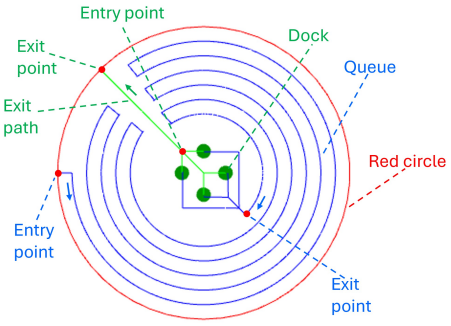

Fig. 6: The single queue in ARGoS simulation

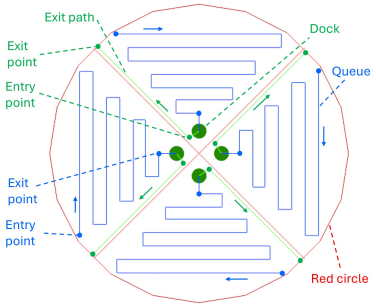

Fig. 7: The multiple queues

The configuration of the ARGoS simulation environment is summarized in Table I. To evaluate the effectiveness of the proposed congestion mitigation strategies, we conducted 50 simulation runs for each configuration in a 12×12 meter arena, using robot team sizes of 48, 64, 80, 96, and 112. Each simulation lasted 18 minutes. We compare our proposed queuing strategies against a baseline approach, CPFA, without any advanced traffic management mechanisms.

We evaluate the performance of the proposed strategies using the following three metrics: 1) **Resource Collection**: The total number of resources collected by the swarm during the simulation. 2) **Collision Time**: The cumulative time (in minutes) that all robots spend in collision states. 3) **Resource Drop-Off Time**: The time interval between a robot entering the red circular region and successfully dropping off a resource at a designated dock.

TABLE I: Experimental Parameters

Arena Size	12 × 12 meter
Radius of Red Circle	2 meter
# of Robots	48, 64, 80, 96, 112
# of Resources	320
Resource Distribution	5 clusters
Cluster Size	8 × 8
Simulation Time	18 minutes
# of Runs	50

VIII. RESULTS

The results are presented in the following figures. Each experiment was replicated over 50 trials, and we report the median resource collection time for the swarm in each configuration. We used **statistical significance testing**to determine whether their performance differences are statistically meaningful. We employed an independent two-sample t-test (using `ttest_ind` from the SciPy library in Python). We compute a p-value, and the level of significance is then visualized on the box plots using star notation, ***, $p < 0.001$, as highly significant.

A. Foraging performance

The foraging performance of CPFA shows a slight decline as the number of robots increases. Given a total of 320 available resources, the average number of collected resources decreases from 49 with 48 robots to 37 with 112 robots. In contrast, the single-queue strategies consistently outperform CPFA across all configurations. Notably, the multi-queue strategies with tail joining, denoted as Q_{ME} and Q_{MT}, demonstrate significant improvements in performance. Their effectiveness increases with larger swarm sizes, indicating enhanced scalability. Among all strategies, Q_{MT} achieves the best performance, collecting 202 resources with 48 robots and 264 resources with 112 robots.

Table II summarizes the average percentage of resources collected in Fig. 8. We can see that 82.39% of resources are collected by 112 robots in Q_{MT}.

TABLE II: Resource Collection (%)

Algorithms	# of Robots				
	48	64	80	96	112
CPFA	15.17	13.33	12.17	11.47	11.39
Q_{SE}	17.39	18.09	17.36	16.63	18.06
Q_{ST}	37.98	38.88	42.45	37.44	34.60
Q_{ME}	46.61	54.29	59.19	61.42	64.22
Q_{MT}	62.94	71.93	76.39	**82.31**	**82.39**

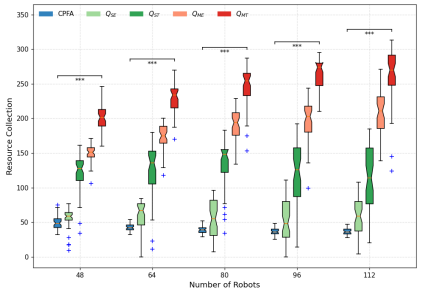

Fig. 8: Resource Collection with different numbers of robots.

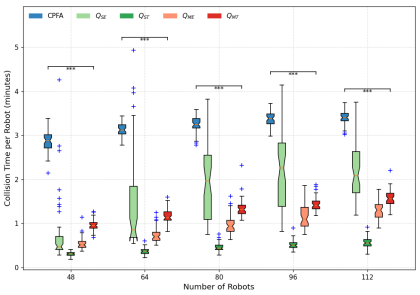

Fig. 9: Collision time per robot

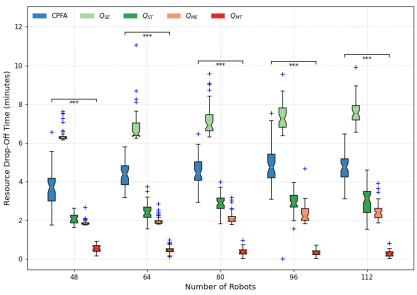

Fig. 10: Resource Drop-Off Time

B. Collision time

As expected, CPFA exhibits the highest collision time (see Fig. 9). In contrast, all four of our proposed algorithms demonstrate significantly lower collision times. Among them, Q_{SE} shows a notably faster increase in collision time as the number of robots grows. This is primarily due to frequent collisions near the entry point of the single queue, where congestion is more likely to occur. The other proposed algorithms maintain relatively stable collision times, with only slight increases observed as the swarm size increases.

C. Resource Drop-Off Time

Among all strategies, Q_{SE} exhibits the highest resource drop-off time, primarily because robots must traverse the entire queue from the entry point to the exit to complete the drop-off. Notably, its drop-off time is even higher than that of CPFA. In contrast, Q_{MT} consistently achieves the lowest drop-off time across all experiments (see Fig. 10). While the drop-off time for most algorithms increases slightly with larger swarm sizes, Q_{MT} remains remarkably stable.

Table II summarizes the average number of resources collected under each configuration. As shown, Q_{MT} achieves the best overall performance, collecting more than 80% of the available resources within the 18-minute simulation window.

IX. DISCUSSION

The analysis of the four queue-based drop-off strategies reveals several design insights for swarm traffic management. First, we see that adding queue management to central drop-off systems significantly improves throughput and reduces congestion levels. The multi-queue model excels in parallelizing access to drop-off points, thereby alleviating bottlenecks at individual docks. In contrast, although the single-queue approaches are much simpler, they are considerably less effective in handling larger swarms.

Additionally, the tail-joining algorithm minimizes idle time spent searching for entry points and better utilizes available docks. This is particularly beneficial in dynamic swarm environments like ours, where robot arrival patterns are often unpredictable. Notably, the superior performance achieved by Q_{MT} highlights its scalability—maintaining efficiency without significant performance degradation, even at high swarm sizes. However, while Q_{MT} performs best overall, the optimal strategy may depend on specific environmental constraints. For instance, environments with high symmetry and uniform traffic patterns (e.g., resource clusters located on one side) might benefit from single-queue systems due to their predictability and fairness.

X. CONCLUSION AND FUTURE WORK

Our results demonstrate that queue-based traffic management significantly reduces congestion and improves the scalability of swarm foraging systems. The robot system does not perform optimally at low swarm sizes (e.g., 48 robots) in the CPFA algorithm. With the multi-queue and tail-joining strategy (Q_{MT}), performance increases substantially with larger swarms (up to 112 robots). Collision time increases only slightly, and the average resource drop-off time remains nearly constant, even as the number of robots grows.

The key takeaway from this study is that pre-planned multi-queue structures combined with tail-joining algorithms offer an effective and scalable solution for traffic coordination in robotic swarms. This underscores the importance of careful queue structure design in high-density swarm scenarios. While our findings validate the effectiveness of queue-based traffic management, several promising directions remain for future work. One such direction is the use of evolutionary algorithms to dynamically evolve queue structures. Rather than relying on predefined single or multi-queue configurations, the system could adapt in real-time based on environmental cues such as robot density and task performance metrics. In the future, we can explore adaptive queue configurations, including changes in geometry, length, and entry rules based on local congestion levels. This would make the system more resilient to fluctuations in swarm size and uneven resource distributions.

ACKNOWLEDGMENTS

The authors would like to acknowledge funding provided by the GAANN program (P200A210144 - 22) from the U.S. Department of Education, the National Science Foundation (NSF) Minority Serving Institute (MSI) program (NSF Award No. 2318682), and the Expand AI program (NSF Award No. 2434916). Tsz-Chiu Au was supported by NRF RS-2022-NR069751 (2022R1A2C101216813).

REFERENCES

[1] B. Khaldi, F. Harrou, F. Cherif, and Y. Sun, "Self-organization in aggregating robot swarms: A dw-knn topological approach," *Biosystems*, vol. 165, pp. 106–121, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0303264717302897

[2] M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß, "Self-organized aggregation without computation," vol. 33, no. 8, 2014. [Online]. Available: https://doi.org/10.1177/0278364914525244

[3] S. Nayak, S. Yeotikar, E. Carrillo, E. Rudnick-Cohen, M. K. M. Jaffar, R. Patel, S. Azarm, J. W. Herrmann, H. Xu, and M. Otte, "Experimental comparison of decentralized task allocation algorithms under imperfect communication," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 572–579, 2020.

[4] F. Arvin, A. E. Turgut, F. Bazyari, K. B. Arikan, N. Bellotto, and S. Yue, "Cue-based aggregation with a mobile robot swarm: a novel fuzzy-based method," *Adaptive Behavior*, vol. 22, no. 3, pp. 189–206, 2014.

[5] F. Arvin, A. E. Turgut, T. Krajník, and S. Yue, "Investigation of cue-based aggregation in static and dynamic environments with a mobile robot swarm," *Adaptive Behavior*, vol. 24, no. 2, pp. 102–118, 2016.

[6] A. Vardy, "Accelerated patch sorting by a robotic swarm," in *2012 Ninth Conference on Computer and Robot Vision*, 2012, pp. 314–321.

[7] A. Vardy, G. Vorobyev, and W. Banzhaf, "Cache consensus: Rapid object sorting by a robotic swarm," *Swarm Intelligence*, vol. 8, no. 1, pp. 61–87, March 2014.

[8] B. Jin, Y. Liang, Z. Han, and K. Ohkura, "Generating collective foraging behavior for robotic swarm using deep reinforcement learning," *Artif. Life Robot.*, vol. 25, no. 4, p. 588–595, November 2020. [Online]. Available: https://doi.org/10.1007/s10015-020-00642-2

[9] J. P. Hecker and M. E. Moses, "Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms," *Swarm Intelligence*, vol. 9, pp. 43–70, 2015.

[10] A. Font Llenas, M. S. Talamali, X. Xu, J. A. R. Marshall, and A. Reina, "Quality-sensitive foraging by a robot swarm through virtual pheromone trails," in *Swarm Intelligence*, M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. Reina, and V. Trianni, Eds. Cham: Springer International Publishing, 2018, pp. 135–149.

[11] L. Qi, J. P. Hecker, and M. E. Moses, "Multiple-place swarm foraging with dynamic depots," *Autonomous Robots*, vol. 42, pp. 909–926, 2018.

[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] D. Lee, Q. Lu, and T.-C. Au, "Multiple-place swarm foraging with dynamic robot chains," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[14] 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.

[15] 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.

[16] D. Lee, Q. Lu, and T.-C. Au, "Dynamic robot chain networks for swarm foraging," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4965–4971.

[17] 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.

[18] 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.

[19] A. Bharaswadkar, V. Vakilian, B. Thoms, and J. T. Isaacs, "Congestion strategies for clustered central place foraging," in *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2021, pp. 185–190.

[20] L. S. Marcolino and L. Chaimowicz, "Traffic control for a swarm of robots: Avoiding target congestion," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 1955–1961.

[21] L. Soriano Marcolino, Y. Tavares Dos Passos, A. A. Fonseca De Souza, A. Santos Rodrigues, and L. Chaimowicz, "Avoiding target congestion on the navigation of robotic swarms," vol. 41, no. 6, 2017.

[22] Y. T. dos Passos, X. Duquesne, and L. S. Marcolino, "Congestion control algorithms for robotic swarms with a common target based on the throughput of the target area," *Robotics and Autonomous Systems*, vol. 159, p. 104284, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889022001737

[23] V. Graciano Santos and L. Chaimowicz, "Hierarchical congestion control for robotic swarms," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4372–4377.

[24] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2018, p. 6252–6259. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8461113

[25] K. Soma, V. S. Varadharajan, H. Hamann, and G. Beltrame, "Congestion and scalability in robot swarms: A study on collective decision making," in *International Symposium on Multi-Robot and Multi-Agent Systems (MRS 2023)*. IEEE, 2023, pp. 199–206. [Online]. Available: https://publications.polymtl.ca/57764/