Estimating Robot Swarm Properties by Direct Calculations instead of Simulation

Jacob Cadavez[1] and Daniel Boley[1]

University of Minnesota, Minneapolis MN 55455, USA boley@umn.edu

Abstract. We propose a novel way to estimate some properties of the motion of a swarm of robots which does not require carrying out extensive simulations. We leverage the large body of literature on graphs to compute estimates for properties like hitting time, cover time, centrality measures of different regions of an arena, etc. We do not aim to produce accurate physical models of robot behavior, but rather aim to produce estimates of major properties of the robot behavior in the presence of a swarm of robot obstacles using quick closed-form formulas. This would allow the swarm designer to explore a variety of scenarios quickly before advancing to the stage of expensive simulations for more accurate measurements. In this paper, we model the correlated random walk (CRW) of a robot on a discrete arena in discrete time, and then extend this simple model to include multiple robot obstacles. We show how several critical properties for the discrete model can be estimated rapidly using linear algebra tools from spectral graph theory.

Keywords: Robot swarms · graph theory · Markov chains · graph Laplacian

1 Introduction

In swarm robotics, a large group of simple robots cooperates implicitly to collectively perform tasks. The robots in general have limited individual capabilities in terms of sensing, processing power, and inter-agent communications, but the number of robots makes up for the individual limitations.

Robot swarms can be used for many real-world tasks, such as cleaning an area [27], demining, rescuing people, or mowing a lawn, which require complete coverage of the space and where ideally each location should be visited only once to avoid unnecessary repetitions, or tasks that require maintaining surveillance over an area to detect intruders, fires, traffic jams or enemies [14,9], or to map an unknown area [13].

A task of special interest in the swarm community, due to its real-world applications, is "foraging", where robots search for target objects and bring them back to a centralized location called the nest. Multiple robots could be released to search for targets within a predefined area. We denote this process as "searching" because robots don't know where the target locations are. In this paper, we model foraging by a single robot in the presence of multiple roving robot obstacles. The foraging behavior by a swarm can in principle be extrapolated from the estimated behavior of a single robot in the presence of other roving robots.

In these situations, it is often necessary for engineers to build robot swarms that will accomplish the given task within a set time constraint. An important metric that

is used to decide the configuration for a swarm mission (such as number of robots, movement behavior, etc.) is the expected search time or 'hitting time' of a simulation. This measurement can be very resource and time intensive to obtain. The most accurate method to collect data is to run experiments involving real robots, taking a lot of time and resources to perform. Instead, researchers typically run a large number of computer simulations and perform data analysis on the results to obtain empirical models. However, these trials can become very taxing when scaling to larger arena sizes and may not accurately represent how these swarms may behave in practice, even with ARGoS, the most accepted software to run such simulations [26].

We propose an alternative approach by using mathematical models to approximate robot swarms' average hitting times over legacy methods. We specifically do ***not*** aim for physical accuracy, but rather aim to quickly produce approximations using closed-form formulas that are close enough to give a good qualitative measure of the robot behaviors under a variety of scenarios of interest to the designer.

The mathematical computations we propose in this paper are based off a variant of the random walk (RW) model - a movement policy that dictates how each robot moves from step to step. Although very simplistic, the RW model is frequently used for exploration due to its scalability and modest processing requirements [24,19,6]. The RW model is a discrete-time stochastic process where successive random steps are used. When the number of possible states is finite or countable and the step taken at each discrete time interval depends only on the current state (or at most a fixed sized history), the process can be described by a discrete-time Markov chain [15]. An important property of random walks is the "first passage time," or Hitting time (HT), which is the average time to first visit a node while doing a random walk from another node in a given network [17,20].

An extension of the RW model is the Correlated Random Walk (CRW) model where the probability of each robot's next step depends on its current orientation and location, which is the aforementioned variant explored in this paper [22,11,10,25,24]. By modeling CRW model's movement behavior as a graph, one can observe that each state transition is directed. For example, transitioning from a location A to a neighboring location B and then back to A, one ends up at A with different orientation, represented by a different state in the Markov chain. Thus the links in the Markov chain are only one-way. Although not the most optimized search method, CRWs are commonly used to model insect behaviors, which is the motivation behind a large fraction of the research in swarms. As real robots are often non-holonomic or require extra steps to rotate in place, CRWs are a better model than regular RW as they take direction into consideration.

As there is previously developed work that showcases how computational models can estimate probability distributions of a solo robot, this paper will focus on multi-robot systems of obstacles. In this work, we develop computation models that can rapidly compute estimates of probability distributions of a specific robot among a swarm of other robots assuming each robot follows the same movement policy and acts independently from other robots.

2 Related Work

An enormous body of work exists on the analysis of random walks on graphs such as estimating hitting times, centrality measures, cover times, and other aggregate properties. Space does not permit an extensive summary of existing literature on Markov chains, graphs, random walks, etc. Many variations on robot swarm models have appeared. In the present paper we study robots that move independently within an arena to accomplish a task. This is in contrast to analysis where each robot coordinates its behavior based on its neighboring robots (e.g., [27]).

There is a long history of study of Markov chain properties that can be obtained directly from the probability transition matrix and matrices derived therefrom, too much to fit in this paper. Only been recently a study of the computational issues for directed graphs or non-reversible Markov chains has been addressed (e.g., [30,5]).

Many swarm algorithms use random or correlated random walk, because of their simplicity. Some instead synchronize the robot motions through the environment, like [27], where robots clean a non-convex region using the dirt on the floor as the main means of inter-robot communication and follow a strict protocol to guarantee full coverage.

A recent survey of foraging algorithms [18] analyzes the state of the art in foraging, with special attention to the foundations of swarm research as well as to applications of robot swarms. A specific version of foraging called "Central Space Foraging," is characterized by having the collection area (goal node) in the center of the space, which is circular. The problem has been studied systematically [1] to analyze the performance of a few different algorithms. Specifically, the robots move randomly, or follow a deterministic path such as a multi-robot Archimedes spiral, or move radially from the center. The comparisons in the paper were done experimentally, but the paper provided theoretical upper bounds for the time needed to complete the task.

A theoretical study of emergent behaviors in systems of many simple agents with limited memory and limited communication has yielded some theoretical guarantees for a model problem. A typical result [4,23] involves a uniform triangular/hexagonal grid with periodic boundary conditions. Theoretical guarantees are given that congregation or dispersion will naturally arise depending on the setting of a presence of "food."

3 Proposed Approach for Direct Computations

In the swarm community, the environment used for searching tasks is typically 2D, most often with continuous space. In our approach, instead, we use a discrete space and a correlated random walk (CRW) to model the motion of the robots in discrete time. We generalize the grid representation with a graph, and use methods developed for graphs to estimate the motion behavior of the robots in the swarm. Our objective is not to design controllers for the robots, but to get quick estimates of swarm properties by direct calculation. This allows one to explore quickly a variety of scenarios, reserving the expensive simulations for scenarios of specific interest.

We show next how to obtain the approximate distribution of the basic property, the hitting time (HT), by a direct calculation, thereby avoiding the expense of running simu-

lations. Analogous properties can be obtained using the non-symmetric graph Laplacian and other matrices derived therefrom [7,16].

In later sections, an outline of how the simulation works and how the Markov-chain approach can yield values for important moments is given.

3.1 Computation of HT Mean and Variance

Let $\mathbf{P}$ denote the probability transition matrix for the Markov chain, such that p_{ij} (the entry in row i column j) is the probability that the robot in state i will move to state j at the next time step. We define two vectors: the mean $\mathbf{h}_\mu$ and variance $\mathbf{h}_{\sigma^2}$, whose i-th entry is the mean and variance (respectively) of the Hitting Time from the i-th state to the nest: the number of time steps needed to reach the nest from state i.

Following [8,28], we partition $\mathbf{P}$ to get $\mathbf{Q}$, the probability transition matrix corresponding to the non-absorbing states, via:

$$\mathbf{P} = \begin{bmatrix} \mathbf{Q} & \mathbf{r} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{1}$$

Here $\mathbf{r}$ is a single vector whose i-th entry is the probability of transitioning from state i to the absorbing state in the single time step. These formulas can be easily generalized to the case of multiple absorbing states where $\mathbf{r}$ is a matrix. The matrix $\mathbf{P}$ is row stochastic since the matrix is non-negative and all the row sums equal 1. So $\mathbf{Pe} = \mathbf{e}$, where $\mathbf{e}$ is a vector of all ones of appropriate dimension. With some algebra, This implies [28] that $\mathbf{r} = (\mathbf{I} - \mathbf{Q})\mathbf{e}$. We define the fundamental matrix $\mathbf{N}$ of $\mathbf{P}$ as [8]:

$$\mathbf{N} \stackrel{\text{def}}{=} (\mathbf{I} - \mathbf{Q})^{-1}. \tag{2}$$

Then using power series expansions in $\mathbf{Q}$ for $\mathbf{N}$, $\mathbf{N}^2$ and $2\mathbf{N}^3 - \mathbf{N}^2$ we obtain the following expressions for the HT mean and variance [28]:

$$\begin{aligned} \mathbf{h}_\mu &= \mathbf{N}^2\mathbf{r} = \mathbf{Ne} \\ \mathbf{h}_{\sigma^2} &= 2\mathbf{N}\mathbf{h}_\mu - \mathbf{h}_\mu - \mathbf{h}_\mu^2, \end{aligned} \tag{3}$$

This gives a closed-form expression for the mean and variance of the hitting time. With these two parameters in hand, one can fit a 2-parameter distribution to the hitting time and use that to estimate quantities such as the probability that the nest will be reached within a given time limit. This is done later in Section 3.3. However we first discuss the process to compute the quantities in Eqs. (3).

3.2 Fast Solution of Linear Systems

Multiplying a vector $\mathbf{v}$ by the matrix $\mathbf{N}$ is equivalent to solving the system of linear equations of the form $(\mathbf{I} - \mathbf{Q})\mathbf{u} = \mathbf{v}$ for the vector $\mathbf{u}$. In the next subsection we show how the system of equations involving $\mathbf{I} - \mathbf{Q}$ can be solved efficiently, taking full advantage of the high degree of sparsity in the matrix $\mathbf{I} - \mathbf{Q}$. This system can be efficiently solved for by iterative methods such as Restarted GMRES [12,21], even if

$\mathbf{I} - \mathbf{Q}$ is extremely large (e.g., $100,000 \times 100,000$ or larger), since it is very sparse (at most 6 entries in every row in our current setup).

Note that $\mathbf{Q}$ can be embedded within a larger irreducible probability transition matrix for a random walk with no transient states:

$$\widetilde{\mathbf{P}} \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{Q} & \mathbf{r} \\ \mathbf{s}^T & t \end{bmatrix}, \tag{4}$$

such that $\widetilde{\mathbf{P}}$ is row stochastic and $\mathbf{s}$ is not entirely zero. This modified matrix satisfies the preconditions to guarantee convergence of an iterative method like GMRES [2].

To illustrate the computation costs, we show in Table 1 some representative times on a linux office desktop using the scipy.sparse matrix package in python. For example, a 150×150 arena generates a probability transition matrix $\mathbf{P}$ of dimension $136,807 \times 136,807$ but has only 807,864 non-zero entries (at most 6 non-zero entries per row), or about $.004\% = 4 \times 10^{-5}$ of all the entries. Solving a system of equations with this matrix using restarted GMRES with an off-the-shelf preconditioner takes under 6 seconds. The time to setup the system can vary, partially because that part of the computation was not heavily optimized.

Table 1. Computation costs for different sized arenas.

arena size	matrix $\mathbf{P}$ dimension	number non-zeros	setup time (sec)	GMRES time (sec)
30×30	5767	31223	1.5	0.05
150×150	136807	807863	38.4	5.4
300×300	543607	3236663	44.9	35.6

3.3 Distribution from Moments

Starting with the estimates for the first two moments (mean and variance) one can fit an appropriate distribution for the hitting time. This can be used to extrapolate the motion of a set of robots under different circumstances. Since the hitting time is non-negative and has no intrinsic upper limit, we fit a Gamma Distribution whose cumulative distribution function (CDF) [the probability the target will be reached in time less than or equal to T] is:

$$\text{CDF}(T) \stackrel{\text{def}}{=} \int_0^T \frac{\beta^\alpha}{\Gamma(\alpha)} \tau^{\alpha-1} e^{-\beta\tau} d\tau \tag{5}$$

with

$$\alpha = \frac{(\text{mean}(T))^2}{\text{var}(T)},\ \beta = \frac{\text{mean}(T)}{\text{var}(T)}, \tag{6}$$

where T is the random variable ($\mathbf{h} - \mathbf{h}_{\min}$ in this case), and α, β are the shape and rate parameters written in terms of the mean and variance of T. Here $\Gamma(\alpha)$ is the Gamma function. The hitting time cannot be less than the shortest path length $\mathbf{h}_{\min}$, hence we shift the Gamma distribution by this amount.

4 Verification of Properties via Simulations

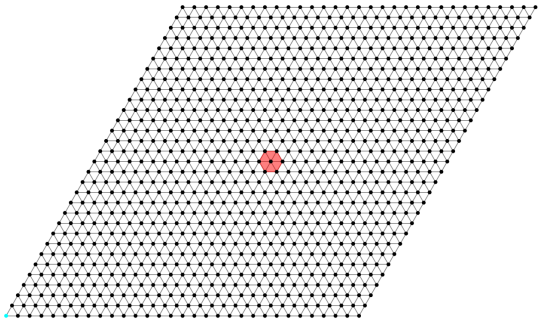

Fig. 1. A 31x31 arena for a single robot starting at the origin (lower left). The red nodes are in the robot sensing range and will absorb the robot.

We illustrate a 31×31 arena in Fig. 1, though we experiment with arenas of various sizes. The arena is discretized into equilateral triangles as it allows a robot to traverse to six neighbors with equal length distances opposed to using squares only having four connectors. Each state of the correlated random walk represents a combination of the location in the arena and the orientation of the robot, where the orientation is one of the six possible incoming directions.

4.1 Average Times to Reach Nests

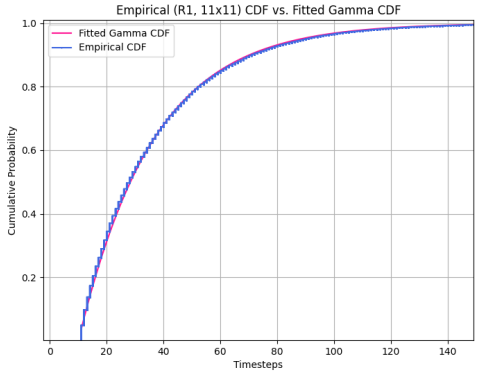

Fig. 2. Sample CDF distribution for a single-robot swarm simulation over 3000 trials. The robot started at the origin with no orientation, with no laziness, in an 11x11 arena. Underneath the plotted points, in pink, is a fitted gamma distribution off of the data.

The CRW follows the model methodology laid out by previous papers on the same topic [29,3]. There are six different states a robot can achieve when traversing an arena: I (has all six connected neighbors), CW (collision against an arena wall), PW (parallel to

the wall), ASC (against sharp corner), AOC (against obtuse corner). Our team believes that going forward is the best methodology of traversing an unknown grid, so therefore we configured each robot to have a higher likelihood of moving forward than turning. At each timestep, the robot will randomly pick a state from the transition matrix to dictate its next move.

Additionally, absorbing nodes have been set around the target node, meaning that when a robot moves into these nodes, the probability of moving into the target node is 1. Our team developed a simulation that follows the above rules and will provided in a GitHub if this paper is approved. By developing our own simulator, the transition matrices done in the computation are easier and reduces the potential human-error when analyzing. Data was collected off of the simulator and was compared to the Markov-chain computation approach; this methodology has been used in previous papers and will be the baseline of analysis for other configurations in this paper.

5 Multiple Robots and Obstacles

5.1 Problem Definition

In real-life situations, the area that a robot swarm traverses can be filled with objects and terrain that may impede movement. These are obstacles that can be identified by two categories: static and dynamic obstacles.

Static obstacles are entities that do not move throughout a simulation; instead, they block off different paths and states, making robots have to find alternative routes in order to reach their goal. If these static obstacle locations are fixed ahead of time, the transition matrix is set to have a 0% probability of moving to a state occupied by an obstacle.

Dynamic obstacles are defined as entities that move alongside the swarm, having a chance to interfere with a given robot's path, impeding its movement. As each robot in a given swarm moves independently of each other in a CRW policy, all of the other robots can be considered dynamic obstacles.

Given that robots and obstacles cannot inhabit the same position in the arena, a protocol is established for what should happen in these situations, and in this project two were explored: a wait-next and find-next protocol. In a wait-next protocol, if a robot tries to move to a position occupied by a dynamic obstacle, it will stay in its current position. In a find-next protocol, the robot will randomly select another state to transition to (or stay put if unable to find a feasible move after a given number of tries). When dynamic obstacles are introduced, the Markov chain model approach used above cannot be applied directly in this situation; the requirement for a fixed predefined transition matrix is not met. One workaround for this issue is find a model with a property that mimics the behavior of dynamic obstacles; we have found a "laziness" property to be a good candidate.

5.2 Adapting Model with Laziness Property

We have defined laziness to be a property in which a robot has a set probability of not changing state, which follows the behavior of robots with a wait-next collision protocol. Mathematically, the transition matrix is modified to incorporate the possibility

Table 2. Combined Analysis of KS Test Results and Hitting Times for Different Robot Counts in an 11x11 Arena. Each robot follows a wait-next protocol. The tracked robot is placed at the origin.

# Robots	Population Size	Sample Size	KS Statistic	P Value	Avg. HT	HT Std Dev.	Avg. Wait %
5	3000	100	0.088	0.395	39.102	29.157	4.48%
6	3000	100	0.097	0.289	38.024	26.689	5.65%
7	3000	100	0.095	0.311	39.280	28.851	6.72%
8	3000	100	0.105	0.208	39.273	28.969	7.90%
9	3000	100	0.083	0.479	39.531	28.948	9.04%
10	3000	100	0.095	0.307	41.830	30.830	10.11%

Table 3. Hitting Times Analysis for a Solo Robot with Varying Laziness in an 11x11 Arena

Laziness Probability	Avg HT	HT Std Dev
0.077	39.123	29.311
0.078	39.162	29.343
0.079	39.202	29.376
0.080	39.241	29.408

where a robot remains in its current state with probability p_l and transitions normally with probability $1 - p_l$. This property allows an isolated robot in the arena to delay state transitions, mirroring the effects of dynamic obstacles in multi-robot environments. Additionally, another important quality is that the transition matrix is static, meaning that methods to compute moments for mean and variance can be performed.

If a single robot model is able to mimic the behavior of a multi-robot swarm, then it is possible to use the above approach to rapidly approximate hitting times for a robot in terrains with dynamic obstacles. Here we chose to collect the hitting times of multi-robot simulations for 3,000 samples for each configuration and to calculate the moments on a single-robot swarm. Given that there is a proven way to calculate the mean and variance without having to run simulations, we found it more important to avoid margins of error when performing these comparisons. Tables **??**, 3 showcase the moments for each configuration as well as average wait % for multi-robot swarms, used to map to the appropriate laziness probability.

To compare the similarity in distributions between the multi-robot configuration and a fitted gamma distribution on the population, we performed a Kolmogorov-Smirnov (KS) test. The KS-test provides insight on the goodness-of-fit of a distribution compared to another. The yielded KS-statistic provides insights on the magnitude of differences between a given distribution and another distribution or a set of samples; the larger the KS-statistic, the greater the difference is. Utilizing the KS-statistic, we ran a statistical analysis procedure to verify whether a set of samples from the data collected on a given configuration will match the fitted gamma distribution provided from Python's MLE algorithms. The null hypothesis is that the gamma is a good fit for the samples taken from the population and do not significantly differ. This procedure was then ran on each robot configuration with a sample size taken of 300 and results are documented in Table 2.

The results in Table 2 show p-values greater than 0.05, so the null hypothesis cannot be rejected. One concludes that the fitted gamma distribution does not significantly differ from the samples, and is indicative of a decent fit.

After identifying that the fitted gamma distribution is appropriate for a multi-robot swarm, one can now conduct levels of fit between this distribution to the single-lazy robot simulation. In Table 3, this is the data collected for the single-robot simulation with laziness that matches the wait percentage for the multi-robots above.

5.3 Scaling to Larger Arenas

The trials and analysis were performed on smaller arenas to validate whether or not the proposed approach would work. After verifying a potential correlation, our next task was to scale this project to analyze whether it would hold up with larger arena sizes.

The same methodology above was conducted on the larger arenas, and Tables 4 6 showcase the results and data collected. For the fitted gamma distribution on each distribution, the KS-test yielded low KS-statistic. This indicates that the gamma distribution provided an appropriate fit on the number on the samples taken with no significant difference. Although there is a small range of variance when comparing the moments between the multi-robot simulations and the lazy single robot, this difference can be attributed to low sampling size and the stochastic nature of the system, leading to fluctuations due to randomness.

Table 4. Hitting time metrics for the robot placed at (0, 0) with an orientation of 0-degrees in a 31x31 Arena. This was tracked over multiple setups with different robot counts. Robots are randomly distributed except for the tracked robot, which is placed at the origin. Each robot follows a wait-next collision protocol.

# Robots	Population Size	Sample Size	KS Statistic	P Value	Avg. HT	HT Std Dev.	Avg. Wait %
15	3230	300	0.077	0.053	393.83	331.39	1.85%
16	3000	300	0.060	0.227	411.43	353.95	1.98%
17	3000	300	0.061	0.212	419.15	367.35	2.09%
18	3000	300	0.051	0.411	408.93	361.73	2.27%
19	3000	300	0.056	0.296	412.20	342.05	2.36%

Table 5. This is a table showcasing the results for the fastest robot in a swarm of varying sizes, 31x31 arena, filled all in one corner.

# Robots	Avg Timesteps	Std. Dev.
15	72.181	29.584
16	70.392	28.538
17	69.398	27.753
18	67.302	26.050

Table 6. Hitting Times Analysis for a Solo Robot with Varying Laziness in a 31x31 Arena.

Laziness Probability	Average HT	HT Std Dev
0.018	409.98	352.02
0.019	410.40	352.38
0.020	410.81	352.74
0.021	411.23	353.10

5.4 Approximating Expected First Arrival Time in a Swarm of Robots

Let $p(t)$ be a gamma probability density function (PDF) with mean μ and variance σ^2. Define the cumulative distribution function (CDF) as:

$$c(t) = \int_0^t p(\tau)\, d\tau$$

This gives the probability that a single robot reaches the nest by time t. Thus,

$$1 - c(t) = \Pr(\text{robot does not arrive by } t)$$

If we approximate a swarm of N independent robots avoiding collisions with N random walkers with an appropriate laziness value, then the probability no walker reaches the nest by time t would be $(1 - c(t))^N$. So the probability at least one walker reaches the nest is $G_N(t) = 1-(1-c(t))^N$. Differentiating gives the PDF for the first arrival time:

$$g_N(t) = G'_N(t) = N(1 - c(t))^{N-1} p(t)$$

The expected first arrival time and corresponding standard deviation are

$$\mu_N = \int_0^\infty t\, g_N(t)\, dt \quad \text{and} \quad \sigma_N = \sqrt{\int_0^\infty (t - \mu_N)^2\, g_N(t)\, dt}$$

Theoretically, this algorithm can be used to approximate for multiple robots using an approximated mean and standard deviation of a computed "lazy" solo robot. For example, a single walker in a corner of a 31×31 arena with hitting time moments of $\mu = 410$, $\sigma = 352$ indicates that the first of $N = 15$ walkers will reach the nest in $\mu_N \approx 64$ timesteps with a standard deviation of 29. This approximation yields a qualitative reflection of the performance of multiple robots while bypassing the need to simulate the system.

6 Conclusion

This paper has established a foundation for which Markov chains and graph algorithms can be utilized to obtain quick estimates of different properties of robot swarms, such as hitting times, without having to run simulations. Throughout this project, we have explored the possibility of dynamic and static obstacles being introduced in the simulation, and how the models can be adjusted to fit them. The static laziness parameter is imperfect in matching the nuance of dynamic robots as obstacles; however, it yields results that are consistent and similar to multi-robot swarms. After obtaining the distribution, one can then perform necessary analysis to see which configuration of robot swarm is needed in their project.

As Markov-chains haven't been extensively used in robot swarming, we propose that this could be a very promising direction to explore when it comes to optimizing robot swarm designs. Many local variations of robot movement policies are based off of a random walk approach, meaning that the approach outlined above can generate rapid approximations for more complex systems. Future work will outline different areas and directions to continue this research.

6.1 Future Work

The proposed methods have some limitations due to its simplification of how real robots operate. Using a static laziness parameter to approximate dynamic collisions is not perfect, but does yield results that are qualitatively consistent. We have modeled the time for one particulate robot to reach the goal. This work should be thought of as preliminary to a model for tracking when one of many robots reach the goal, a future direction.

Additionally, as the results for a laziness robot mapping may not be appropriate, further analysis and research will have to be done on potential conditions on why this difference occurs, especially with larger arena sizes. Future work will include evaluating the effects when the arena is more crowded and alternatice statistical tests such as Kullback–Leibler divergence and Earth Mover's Distance to compare distributions.

In this paper, we have shown that graph theory can offer a rich set of tools with which to analyze a multi-robot system and predict its global motion behavior without having to run expensive simulations. This suggests that further development of graph-based models for swarm robotics would be a very promising direction to pursue.

References

1. Aggarwal, A., Gupta, D., Vining, W., Fricke, G., Moses, M.: Ignorance is not bliss: an analysis of central-place foraging algorithms. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 6510–6517. IEEE, New York City, NY (2019), doi:10.1109/IROS40897.2019.8967665
2. Boley, D.: On fast computation of directed graph laplacian pseudo-inverse. Linear Algebra and its Applications **623**, 128–148 (2021)
3. Boley, D., Gini, M., Zhang, Y.: How do robot swarms behave? what graphs can tell us. In: Autonomous Robots and Multirobot Systems (ARMS) 2023 Workshop at AAMAS 2023 (May 2023), https://u.cs.biu.ac.il/~agmon/arms2023/program.html
4. Cannon, S., Daymude, J., Gokmen, C., Randall, D., Richa, A.: A local stochastic algorithm for separation in heterogeneous self-organizing particle systems. https://arxiv.org/abs/1805.04599 (2019)
5. Cohen, M.B., Kelner, J., Peebles, J., Peng, R., Sidford, A., Vladu, A.: Faster algorithms for aomputing the stationary distribution, simulating random walks, and more. In: IEEE 57th Annual Symp. on Found. Comput. Sci. (FOCS). pp. 583–592. IEEE, New York City, NY (Oct 2016)
6. Fujisawa, R., Dobata, S.: Lévy walk enhances efficiency of group foraging in pheromone-communicating swarm robots. In: IEEE/SICE Int'l Symposium on System Integration, SII 2013. pp. 808–813. IEEE, New York City, NY (Dec 2013)
7. Golnari, G., Zhang, Z.L., Boley, D.: Markov fundamental tensor and its applications to network analysis. Linear Algebra and its Applications **564**, 126–158 (2019)
8. Grinstead, C.M., Snell, J.L.: Introduction to Probability. Amer Math Soc., Boston, MA (1997)
9. Hamann, H.: Modeling swarm systems and formal design methods. In: Swarm Robotics: A Formal Approach, pp. 95–127. Springer, Berlin/Heidelberg, Germany (2018)
10. Harwell, J., Gini, M.: Broadening applicability of swarm-robotic foraging through constraint relaxation. In: 2018 IEEE Int'l Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR). pp. 116–122. IEEE, New York City, NY (2018)

11. Jeong, M., Harwell, J., Gini, M.: Analysis of exploration in swarm robotic systems. In: IAS-16. pp. 445–457. Springer, Berlin/Heidelberg, Germany (2021)
12. Joubert, W.: On the convergence behavior of the restarted GMRES algorithm for solving nonsymmetric linear systems. Num. Lin. Alg. Appl. **1**(5), 427–447 (1994)
13. Kegeleirs, M., Ramos, D.G., Birattari, M.: Random walk exploration for swarm mapping. In: Proc. Towards Autonomous Robotic Systems: 20th Annual Conference, TAROS 2019, Part II. pp. 211–222. Springer-Verlag, Berlin, Heidelberg (2019)
14. Lancaster, J.P., Gustafson, D.A.: Predicting the behavior of robotic swarms in search and tag tasks. Procedia Computer Science **20**, 77–82 (2013). https://doi.org/https://doi.org/10.1016/j.procs.2013.09.242, https://www.sciencedirect.com/science/article/pii/S1877050913010429, complex Adaptive Systems
15. Levin, D.A., Peres, Y., Wilmer, E.L.: Markov Chains and Mixing Times, vol. 107. American Mathematical Soc., Providence, RI (2017)
16. Li, Y., Zhang, Z.L.: Digraph Laplacian and the degree of asymmetry. Internet Mathematics **8**(4), 381–401 (2012)
17. Lovász, L.: Random walks on graphs. Combinatorics **2**(1-46), 4 (1993)
18. Lu, Q., Fricke, G.M., Ericksen, J.C., Moses, M.E.: Swarm foraging review: Closing the gap between proof and practice. Current Robotics Reports **1**, 1–11 (2020)
19. Martinez, F., Jacinto, E., Acero, D.: Brownian motion as exploration strategy for autonomous swarm robots. In: 2012 IEEE Int'l Conf. on Robotics and Biomimetics (ROBIO). pp. 2375–2380. IEEE, New York City, NY (2012)
20. Masuda, N., Porter, M.A., Lambiotte, R.: Random walks and diffusion on networks. Physics reports **716**, 1–58 (2017)
21. Morgan, R.B.: GMRES with deflated restarting. SIAM J Sci Comput **24**(1), 20–37 (2002)
22. Nain, R., Sen, K.: Transition probability matrices for correlated random walks. Journal of Applied Probability **17**(1), 253–258 (1980)
23. Oh, S., Randall, D., Richa, A.W.: Foraging in particle systems via self-induced phase changes. https://arxiv.org/abs/2208.10720 (2022)
24. Pang, B., Song, Y., Zhang, C., Wang, H., Yang, R.: A swarm robotic exploration strategy based on an improved random walk method. Journal of Robotics **2019**(Article ID 6914212), 9 (2019)
25. Patlak, C.S.: Random Walk with Persistence and External Bias: A Mathematical Contribution to the Study of Orientation of Organisms. University of Chicago, Committee on Mathematical Biology (1953)
26. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., et al.: ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In: IEEE/RSJ Intel Conf Intelligent Robots and Systems. pp. 5027–5034. IEEE, New York City (2011)
27. Wagner, I., Altshuler, Y., Yanovski, V., Bruckstein, A.: Cooperative cleaners: A study in ant robotics. The International Journal of Robotics Research (IJRR) **27**(1), 127–151 (2008)
28. Zhang, Y., Boley, D., Harwell, J., Gini, M.: A correlated random walk model to rapidly approximate hitting time distributions in multi-robot systems. In: Intelligent Autonomous Systems 17, Proc. 17th Int'l Conf IAS-17. pp. 724–736. Springer, Berlin/Heidelberg, Germany (June 2022)
29. Zhang, Y., Boley, D., Harwell, J., Gini, M.: A correlated random walk model to rapidly approximate hitting time distributions in multi-robot systems. In: Intelligent Autonomous Systems 17, Proc. 17th Int'l Conf (IAS-17). pp. 724–736. Springer (2022)
30. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: Proc. 22nd Int'l Conf. Machine Learning. pp. 1041–1048. ACM, New York City, NY (2005)