How do robot swarms behave? What graphs can tell us

Daniel Boley  
University of Minnesota  
boley@umn.edu

Maria Gini  
University of Minnesota  
gini@umn.edu

Yi Zhang  
University of Minnesota  
yi.zhang@vanderbilt.edu

ABSTRACT
This paper proposes a novel way of modeling swarms of robots. The main idea we propose is to apply to swarm robotics theoretical methods and algorithms developed for random walks on graphs. Most mathematical models used for swarm robotics are continuous models based on ODEs as opposed to discrete models. Many properties of random walks on graphs can be derived from the so-called graph Laplacian matrix and its pseudo-inverse. Examples include the first passage time to reach a goal, average round-trip times, probability of passing a landmark, and the like. Using discrete graph structures will also allow us to model swarms of robots having a mix of different behaviors. Using graph models is a major innovation in swarm robotics that we believe has the potential to provide new theoretical tools for the study of swarms of robots. In this paper we model the movement of multiple robot agents on an arena discrete in time and space. We then 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

ACM Reference Format:

1 INTRODUCTION

In swarm robotics, a large group of simple robots coordinates 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 [67], 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, fire, etc. [37, 45], or to map an unknown area [43].

A task of special interest in the swarm community, due to its real-world applications, is “foraging”, where robots search for target objects, such as food, and bring them back to a centralized location called the nest. Multiple robots can be released to search for targets within a predefined area. We denote this process as “searching” because robots don’t know the target locations.

Robot searching has many real-world applications. For instance, collecting trash in an area and looking for survivors are examples of real-world problems that can be modeled as foraging tasks. In this paper, we define the searching time as the time required by at least one of the robots to first reach a target. For many tasks, especially time-sensitive ones, an estimate of searching time is often necessary. Furthermore, having an estimated search time can help design multi-robot systems that are more efficient at searching, for instance, to decide how many robots to use, where to deploy them, and sensors they would need.

Traditionally, researchers run a large number of simulations and average the results to obtain empirical models of robot behaviors. Simulation (e.g. using ARGoS [61]) can be time-consuming, especially when the region to be searched is large. Experimental investigations involving real robots (see [22] for an example) are costly and can even be impractical. We are instead interested in developing mathematical models of swarm behaviors that could be used to estimate how the robots will behave before carrying out any experiments or simulations.

To model swarm behaviors we need to specify how robots move. Random walk (RW) is frequently used for exploration, because of its limited processing requirements and scalability [32, 52, 57]. Random walk is a discrete-time stochastic process where successive random steps are used. When the random steps are time-independent, the process can be described by a discrete-time Markov chain [47].

When the probability of the next step depends on the direction in which the robot has moved previously [38, 57, 60] the motion is called Correlated Random Walk (CRW) [42]. In CRW the state transition probabilities depend on both the location and orientation of the walker [55]. CRW is not the most efficient way of walking but it is commonly used to model insect behaviors, which is the motivation behind a large fraction of the research in swarms. CRWs are more feasible than random walk with real robots (e.g., most wheeled robots are non-holonomic and cannot rotate in place, as required for a standard random walk).

In this work, we develop computation models that can rapidly compute aggregate properties and probability distributions of search time, coverage time, time to collision, etc., as the number of robots, targets, and obstacles vary.

As an example, an important property of random walks that has been studied extensively is the “first passage time,” or Hitting time (HT), which is the average time to first visit a node while doing a random walk (RW) from another node in a given network [49, 53].

Not all properties can be modeled using a random walk model. Some properties may be modeled only approximately because they would involve varying the transition probabilities over time as the world state varies.
Our aim is to develop and measure empirically models that are able to capture swarm behaviors well enough to make global predictions about their performance.

To the best of our knowledge, the spectral analysis of directed graphs has not been applied directly to multi-robot systems or swarm robots. Thus, we aim to leverage some of the methods to rapidly compute aggregate properties of digraphs and to develop a computation model that can rapidly and accurately approximate properties such as the HT in the multi-robot searching process.

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. The relation between the undirected graph Laplacian and connectivity of undirected graphs was treated in, e.g., [16, 24, 29, 66]. The computation of graph connectivity properties based on the graph Laplacian was used in specific applications in [3] (expander graphs), [31] (recommender systems), [65] (computer vision). All of these were for undirected graphs.

Fewer papers exist discussing directed graphs (digraphs) in limited contexts: [11] has a short treatment within a text mostly devoted to the symmetric case; [14] treats the Matrix tree theorem and extends some results to the directed graph case; [17, 71] extend eigenvalue bounds for the Cheeger constant, previously developed for undirected graphs, to digraphs by using closely related undirected graphs. Computing the eigenvalues of the appropriate graph Laplacian matrix is much easier than computing the Cheeger constant itself, so these eigenvalues yield a quick estimate on the overall connectivity of the digraph.

Much fundamental material on random walks, Markov chains, recurrence times, and related topics can be found in these books [2, 44, 47] and survey on random walks [49] and the survey [53] which discusses a wide range of properties of random walks including cover and hitting times, stationary probabilities in both discrete and continuous time.

Another thread of research has given some theoretical bounds on how many robots are needed to explore a grid using a systematic finite-state algorithm with a collection of robots with constant memory and limited sensors [5], or can observe all other robots’ positions [21, 64], in the context of a cleaning or exploration task.

Some recent papers have addressed the analysis of different exploration strategies for one or more agents over a graph [45, 57, 58]. The papers [28, 41, 48, 70] use spectral properties of the underlying Markov chain to derive bounds on the hitting and cover times. The papers [10, 51] give hitting and cover times for specific graphs (e.g., cycles) or graphs that maximize the cover time, while [6] does the same for a tree. Hitting times for multiple simultaneous random walks are treated in [59, 63]. The paper [34] generalized the fundamental matrix to the fundamental tensor encapsulating betweenness measures.

Many results for HT exist. In particular, the average time to reach a given node can be computed based on the fundamental matrix associated with the probability transition matrix of the network [36], by treating the target node as an absorbing state. Boley et al. [9] showed how to obtain average HTs from any node to any node in a directed graph at once using the asymmetric graph Laplacian, and later showed one way to obtain this rapidly using sparse matrix methods [8]. In the HT literature, there are also papers about HT higher moments [19, 44], upper bounds [10], and distributions [46]. In addition, there are studies on the speed-up of a random walker search when there are multiple searchers [4, 26, 59] and random walker collisions [7, 33].

Obtaining a-priori estimates of hitting times, cover times, and the like is very useful if one can avoid the expense of long simulations since they allow one to adjust the system to optimize performance. Several authors [4, 10, 13, 26, 27] found the bounds on the cover time vary greatly depending on the starting node, the number of agents, and even the layout of the specific graph. For a two-dimensional n×n grid (the layout closest to the graphs of interest in this paper), it was found that a small number of independent agents can reduce the cover time proportionally, but beyond O(log n) agents it can quickly saturate [4, 27]. The papers [59, 63] also address how the cover time varies with the number of agents. These theoretical results suggest that some coordination between the agents, or carefully staging the agents initially in different parts of the domain, would make a big difference. These papers are focused on ordinary walks over an undirected graph.

A few papers derive overall estimates of the probabilities of collisions theoretically [7, 33, 62] but these do not involve simulations of robot agents. Collisions are hard to model using a memoryless process with transition probabilities fixed in advance, so some papers have incorporated an average collision probability fixed upfront (e.g., [39] & references therein).

Another body of papers derives facts about the probability distribution of hitting times, cover times, or mixing times beyond the mean values, some in terms of higher moments or fitting a standard probability distribution [15, 23, 25, 46, 72]. The papers [40, 42] obtained formulas for the hitting time probability distribution using the Laplace and Z transforms, respectfully. The higher moments are used for network seed-set selection in [30].

The computation of the moments involves the solution of large linear systems involving various forms of the graph Laplacian or probability transition matrix, as in [19, 20] based on the GTH algorithm [35]. Later works were based on a mix of direct and iterative methods [8, 18]. Unlike the other methods, the method sketched in [8] is a pure iterative method, and was used successfully in [68] to empirically fit a probability distribution to the hitting times.

Most of the work described so far is theoretical, with no experimental work with real robots or physics–based simulators. In the swarm community, instead, most of the work is experimental, done using simulators such as ARGoS [61].

Many swarm algorithms use random or correlated random walk, because of their simplicity, but not all. Some instead synchronize the robot motions through the environment, like in [67], 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 [50] 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 are done experimentally, but the paper provides theoretical upper bounds for the time needed to complete the task.

A theoretical study of emergent behavior in multi-agent systems of simple agents with limited memory and limited communication has yielded some theoretical guarantees for a model problem. A typical result [12, 56] involves a uniform triangular/hexagonal grid with periodic boundary conditions. They provide theoretical guarantees that a congregation or dispersion behavior will naturally arise in their system of simple robots depending on the setting of a parameter or the presence of “food.”

3 PROPOSED APPROACH

In the swarm community, the environment used for searching tasks is typically 2D, most often with continuous space. In our approach, we use a discrete space modeled as a graph, and use 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 behaviors of the robots in the swarm.

We show how we can obtain the approximate distribution of the basic property, the hitting time, by a direct calculation, thereby avoiding the expense of running simulations. Analogous properties can be obtained using the non-symmetric graph Laplacian and other matrices derived therefrom [8, 48]. The example is simple but sufficient to provide a better understanding of the idea. To simplify the notation, we number the states in the random walk so that the absorbing state is numbered last.

3.1 Computation of HT Mean and Variance

Using the transition matrix $P$ associated with this network, the HT mean $h_i$ and variance $h_{i2}$ from states outside the sensing range to the absorbing state can be determined. Here the $i$-th component of the vector $h_i$ is the average number of steps when starting from node $i$ before reaching the absorbing state (numbered $n$), and the $i$-th component of $h_{i2}$ is the corresponding variance. Following [36], we partition $P$ to get $Q$, the probability transition matrix corresponding to the non-absorbing states, via:

$$P = \begin{pmatrix} Q & r \\ 0 & 1 \end{pmatrix}.$$  (1)

Here $r$ is a single vector whose $i$ entry is the probability of transitioning from state $i$ to the absorbing state. These formulas can be easily generalized to the case of multiple absorbing states where $r$ is a matrix. This matrix is row stochastic: $Pe = e$, where $e$ is a vector of all ones of appropriate dimension. This implies that

$$Qe + r = e \implies r = (I - Q)e.$$  (2)

The fundamental matrix $N$ of $P$ is [36]:

$$N = (I - Q)^{-1}.$$  (3)

The probability of reaching the absorbing state on exactly the $k$-th time step, starting from node $i$, is $[Q^{k-1}r]_i$.

In the following, we use the following identities:

(a) $N^2 = 1 + 2Q + 3Q^2 + 4Q^3 + 5Q^4 + \cdots$

(b) $N^3 = 1 + 3Q + 6Q^2 + 10Q^3 + 15Q^4 + \cdots$

(c) $2N^3 - N^2 = 1 + 4Q + 9Q^2 + 16Q^3 + 25Q^4 + \cdots$  (4)

The vector of means, $h_{i2}$, can be determined by:

$$h_{i2} = 1 \cdot r + 2 \cdot Qr + 3 \cdot Q^2r + \cdots = N^2r = Ne$$  (5)

where $e$ is a vector of all 1s of appropriate dimension. One computes this by solving the system of equations:

$$(I - Q)h_{i2} = e$$  (6)

This system can be efficiently solved for by iterative methods such as Restarted GMRES [8, 54], even if $I - Q$ is extremely large (e.g., 100, 000 x 100, 000 or larger) as long as it is sparse (see below).

The vector of variances, $h_{i2}$, can be calculated by first computing the uncentered second moment [44]:

$$\bar{h}_{i2} = (2N - I)h_{i2} = 2Nh_{i2} - h_{i2}$$  (7)

where $h_{i2}^2$ means elementwise squaring. This formula can be derived by applying the identity (4c) to the following definition of the uncentered second moment:

$$\bar{h}_{i2} = \sum_{k=1}^{\infty} k^2 Q^{k-1}r$$  (8)

Subtracting the squares of the means yields the centered second moment, namely the variance:

$$h_{i2} = 2Nh_{i2} - h_{i2}^2.$$  (9)

where $h_{i2}^2$ means elementwise squaring. The quantity $Nh_{i2}$ can be calculated by applying the same fast iterative method used to solve (6) to the system $(I - Q)x = h_{i2}$. Then the HT standard deviation $h_{i2}$ can be obtained.

3.2 Hitting Times for Different Targets

In the above analysis, we fixed the nest as an arbitrarily chosen target node, numbered $n$ for convenience. In this section, we show how the computations carried out for one target can be used to obtain similar quantities for another target node. As a simple matter, the set of equations corresponding to (6) can be obtained by dropping row and column $k$ instead of $n$, or equivalently by permuting the numbering of the nodes so that the new target is numbered $n$. If space allows, one can precompute the entire inverse $N$ for one target, and then rapidly compute the inverse for a different target by making low rank corrections, based on the following lemmas [8].

**Lemma 1** [8].

(a) Let $L = \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix}$ be an $n \times n$ irreducible matrix such that $\text{nullity}(L) = 1$. Let $M = L^T$ be the pseudo-inverse of $L$ partitioned similarly and assume $v^T L = 0$, $L u = 0$, where $u, v$ are partitioned as $u^T = (u_1, u_n)$ and $v^T = (v_1, v_n)$. Assume $u_n > 0$ and $v_n > 0$. Then the inverse of the $(n - 1) \times (n - 1)$ matrix $L_{11}$ exists and is given by

$$L_{11}^{-1} = \begin{pmatrix} I_{n-1} + \frac{u_1 u_n^T}{u_n} & M_{11} \left( I_{n-1} + \frac{v_1 v_n^T}{v_n} \right) \\ M_{11}^T m_{n1} & m_{n2} \end{pmatrix} \left( \begin{pmatrix} I_{n-1} & -v_1^T/v_n \end{pmatrix} \right).$$


(b) Conversely, we can write \( L \) and \( M = L^{-1} \) in terms of \( L_{11}, L_{11}^{-1}, u, v \) as follows
\[
L = \begin{bmatrix}
L_{11} & -u_{1}L_{11}u_{1} \\
\frac{1}{\alpha v_{1}}v_{1}L_{11}u_{1} & 1 - \frac{1}{\alpha v_{1}}v_{1}
\end{bmatrix}
= \begin{bmatrix}
L_{11}^{-1} & L_{11}^{-1}v_{1} \\
\frac{1}{\alpha v_{1}}v_{1} & 1 - \frac{1}{\alpha v_{1}}v_{1}
\end{bmatrix}
\]
\[
M = \begin{bmatrix}
M_{11} & M_{12} \\
M_{21} & m_{nn}
\end{bmatrix}
= \begin{bmatrix}
L_{11}^{-1} - u_{1}v_{1} & L_{11}^{-1}v_{1} \\
\frac{1}{\alpha v_{1}}v_{1} & 1 - \frac{1}{\alpha v_{1}}v_{1}
\end{bmatrix}
\]

Since the rows and columns can be permuted arbitrarily, these formulas allow us to compute the inverse of any \((n - 1) \times (n - 1)\) principal submatrix rapidly from the inverse of just one such principal submatrix \( I - Q \) of (1).

### 3.3 Fast Solution of Linear Systems

The system of linear equations (6) derived from a random walk enjoys special properties that make it possible to solve them using methods that are faster than using a standard elimination algorithm. Here we give a sketch on how this works. For details, see [8] and references therein. First we note that \( Q \) can be embedded within a larger irreducible probability transition matrix for a random walk with no transient states:

\[
\tilde{P} = \begin{bmatrix}
Q & r \\
\ast & t
\end{bmatrix}
\]

with strictly positive stationary probabilities \( \pi_i > 0, i = 1, \ldots, n \) such that \( \pi^T \tilde{P} = \pi^T \). Here \( \pi^T = (\pi_1, \ldots, \pi_n) \). The underlying graph is strongly connected, in the sense that there is a path from any node to any other node and back again. Let \( \Pi = \text{Diag}(\pi) \) denote the diagonal matrix with the stationary probabilities on the diagonal. Then the diagonally scaled matrix \( I - \tilde{P} = \Pi^{1/2}(I - \tilde{P})\Pi^{-1/2} \) enjoys the property that its left and right nullspaces are the same, spanned by the single vector \( \sqrt{\pi} = (\sqrt{\pi_1}, \ldots, \sqrt{\pi_n})^T \), and that \( I - \tilde{P} + \sqrt{\pi} \sqrt{\pi}^T \) is positive definite (though not symmetric). Hence an iterative method based on multiplying vectors by this matrix will converge [8]. Examples of such iterative methods include simple Richardson iteration, or Krylov space methods like GMRES. Because these iterative methods depend only on forming matrix vector products involving the matrix, the cost is proportional to the number of nonzero entries in the matrix, i.e., the number of edges in the underlying graph. The matrix \( I - \tilde{P} \) is singular, of course, but its pseudo-inverse can be written using an ordinary inverse:

\[
(I - \tilde{P})^{-1} = (I - \tilde{P} + \sqrt{\pi} \sqrt{\pi}^T)^{-1} - \sqrt{\pi} \sqrt{\pi}^T.
\]

Hence we can compute the matrix-vector product \((I - \tilde{P})^{-1}v\) for any vector \(v\) by using a fast iterative method to solve the linear system \((I - \tilde{P} + \sqrt{\pi} \sqrt{\pi}^T)x = v\). Putting this all together means that solving the system (6) can be accomplished by embedding it within a strongly connected graph, scaling it by \(\Pi^{1/2}\), and using Lemma 1 to reduce the problem to solving a system involving the pseudo-inverse (11), solved using a fast iterative method.

### 3.4 Peripheral Nodes

In certain foraging situations it may be important to ensure one has explored all parts of a graph that might be assembled from sensor data. Instead of using a visualization of the graph to manually identify the hard-to-reach areas of the graph, an automated method can better ensure consistency in identifying such nooks and crannies. The spectral analysis of graphs, or specifically the pseudo inverse provides one possible way to do this. The Kirchhoff index is a measure of the connectivity of the graph, and is defined as the sum average round-trip commute time in a random walk between any pair of nodes. It has been shown that this quantity is equal to the trace of the pseudo-inverse of the scaled combinatorial Laplacian \(L = \Pi(I - P)\), even for strongly connected directed graphs [8]:

\[
K = \sum_{ij} H(i, j) = n \cdot \sum_{j} \tilde{m}_{jj} = n \cdot \text{Tr} (\tilde{M}),
\]

where \(\tilde{m}_{ij}\) is the \(i, j\)-th element of the matrix \(\tilde{M} = \tilde{L}^T\), and \(\text{Tr}(A) = \sum_j A_{jj}\) stands for the trace of matrix \(A\), namely the sum of the diagonal elements. We can measure how peripheral a node \(q\) is by measuring the average hitting time between any pair of nodes but restricting oneselvess to paths that go through the node \(q\). If we denote this “average node peripherality” as \(K_q\), it can be shown [9] that this quantity equals

\[
K_q = \sum_{ij} H_q(i, j) = n \cdot \text{Tr} (\tilde{M}) + n^2 m_{qq} = n \cdot (K + n \cdot m_{qq}).
\]

Hence the diagonal entries of the pseudo-inverse \(M\) can immediately identify which nodes are more central (and perhaps be a bottleneck) and which nodes are more peripheral. This measure of “peripherality” differs from the stationary probability \(\pi_q\) mainly for graphs that have loosely connected components or long stringy paths.

### 3.5 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 behavior of a set of robot agents under different circumstances. Since the hitting time is non-negative and has no intrinsic upper limit, we can 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:

\[
\text{CDF}(T) = \int_0^T \frac{\beta^\alpha}{\Gamma(\alpha)} \tau^{\alpha - 1} e^{-\beta \tau} d\tau
\]

with

\[
\alpha = \frac{(\text{mean}(T))^2}{\text{var}(T)}, \quad \beta = \frac{\text{mean}(T)}{\text{var}(T)}.
\]

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

### 4 COMPUTATIONAL EXAMPLE

We use the methods described in Section 3 to directly compute the probability distribution of hitting times and related properties for a specific example. We validate the results by comparing the estimates obtained by direct calculation with those from simulation. Analogous properties can be obtained using the non-symmetric graph Laplacian and other matrices derived therefrom, as sketched in Section 3.
We use a $30 \times 30$ arena illustrated in Fig. 1 from [69]. In this example, each node is connected to six neighboring nodes. Each state of the 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. The correlated random walk (CRW) is modelled by having the probability to continue straight to be much higher than the probability to turn in a different direction. For example, if the robot arrives at a node from the west, then it will have a 49% probability to continue to the node to its east, but only a 24% probability to turn $60^\circ$ to the right or to the left, and only a 3% chance to take any of the other directions including reverse. The goal node is the nest in the center of the arena. To simplify the notation, we coalesce the nest into a single absorbing node, and we number the states in the random walk so that the absorbing state is numbered last. By encoding the orientation as part of the state, the graph becomes directed, hence we must use formulas such as those shown in Section 3 which are valid for directed graphs.

The distribution of hitting times observed in the $30 \times 30$ arena, shown in Fig. 2, shows a good match with the theoretical Gamma distribution calculated directly from the moments, and with the behavior observed in an ARGoS simulation of robots in physical space (figs. 3, 4).

4.1 Multiple Robots

Using the computed Gamma Distribution, we can then compute an estimated distribution for the probability that at least one robot in a swarm of $n$ robots will reach the target in at most $T$ steps:

$$C_n(T) = 1 - \left(1 - CDF(T)\right)^n. \quad (16)$$

For small $n$ on a reasonably small arena, it is possible to compare this calculated distribution with empirical observations obtained from simulation. We carried out some preliminary computations using an arena consisting of a 2D grid with a collection area (goal node) in the center. This setup is a discrete model for a foraging task in which robots are supposed to collect items encountered while wandering through the arena and carry them to the goal node. Figure 5 shows there is a good match between the theoretical distribution and the empirical distribution observed from 300 runs over a $30 \times 30$ arena with 10 robots.

Using (16), we can answer questions for much larger arenas for which the expense of simulation could be prohibitive. As an example, using the methods of [8, 68], we can quickly answer the question: how many robots would be needed to have an 80% chance to reach the goal in the center of a $150 \times 150$ arena within 500 time steps. By computing the estimated CDF on the $150 \times 150$ arena for various $n$, one can see from the resulting Figure 6 that 200 robots would be needed.
In this paper, we have attempted to show that graph theory can offer a rich set of tools with which to analyze a multi-robot system and predict its global behavior without expensive simulations. This suggests that further development of graph-based models for swarm robotics would be a very promising direction to pursue.

REFERENCES


