How do robot swarms behave? What graphs can tell us

Daniel Boley
University of Minnesota
Minneapolis, United States
boley@umn.edu

Maria Gini
University of Minnesota
Minneapolis, United States
gini@umn.edu

Yi Zhang
University of Minnesota
Minneapolis, United States
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:
Daniel Boley, Maria Gini, and Yi Zhang. 2023. How do robot swarms behave? What graphs can tell us. In *ARMS workshop at 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29, 2023*, 7 pages.

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 were 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 graph, 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 $n \times n$ 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 $\mathbf{P}$ associated with this network, the HT mean $\mathbf{h}_\mu$ and variance $\mathbf{h}_{\sigma^2}$ from states outside the sensing range to the absorbing state can be determined. Here the i-th component of the vector $\mathbf{h}_\mu$ is the average number of steps when starting from node i before reaching the absorbing state (numbered n), and the i-th component of $\mathbf{h}_{\sigma^2}$ is the corresponding variance. Following [36], we partition $\mathbf{P}$ to get $\mathbf{Q}$, the probability transition matrix corresponding to the non-absorbing states, via:

$$\mathbf{P} = \left[\begin{array}{cc} \mathbf{Q} & \mathbf{r} \\ \mathbf{0} & 1 \end{array} \right]. \tag{1}$$

Here $\mathbf{r}$ 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 $\mathbf{r}$ is a matrix. This matrix is row stochastic: $\mathbf{Pe} = \mathbf{e}$, where $\mathbf{e}$ is a vector of all ones of appropriate dimension. This implies that

$$\mathbf{Qe} + \mathbf{r} = \mathbf{e} \qquad \Longrightarrow \qquad \mathbf{r} = (\mathbf{I} - \mathbf{Q})\mathbf{e} \tag{2}$$

The fundamental matrix $\mathbf{N}$ of $\mathbf{P}$ is [36]:

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

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

In the following, we use the following identities:

$$\begin{array}{lrcl} \text{(a)} & \mathbf{N}^2 & = & \mathbf{I} + 2\mathbf{Q} + 3\mathbf{Q}^2 + 4\mathbf{Q}^3 + 5\mathbf{Q}^4 + \cdots \\ \text{(b)} & \mathbf{N}^3 & = & \mathbf{I} + 3\mathbf{Q} + 6\mathbf{Q}^2 + 10\mathbf{Q}^3 + 15\mathbf{Q}^4 + \cdots \\ \text{(c)} & 2\mathbf{N}^3 - \mathbf{N}^2 & = & \mathbf{I} + 4\mathbf{Q} + 9\mathbf{Q}^2 + 16\mathbf{Q}^3 + 25\mathbf{Q}^4 + \cdots \end{array} \tag{4}$$

The vector of means, $\mathbf{h}_\mu$, can be determined by:

$$\mathbf{h}_\mu = 1 \cdot \mathbf{r} + 2 \cdot \mathbf{Qr} + 3 \cdot \mathbf{Q}^2\mathbf{r} + \cdots = \mathbf{N}^2\mathbf{r} = \mathbf{Ne} \tag{5}$$

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

$$(\mathbf{I} - \mathbf{Q})\mathbf{h}_\mu = \mathbf{e} \tag{6}$$

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

The vector of variances, $\mathbf{h}_{\sigma^2}$, can be calculated by by first computing the uncentered second moment [44]:

$$\widehat{\mathbf{h}}_{\sigma^2} = (2\mathbf{N} - \mathbf{I})\mathbf{h}_\mu = 2\mathbf{N}\mathbf{h}_\mu - \mathbf{h}_\mu \tag{7}$$

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

$$\widehat{\mathbf{h}}_{\sigma^2} = \sum_{k=1}^{\infty} k^2 \mathbf{Q}^{k-1}\mathbf{r} \tag{8}$$

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

$$\mathbf{h}_{\sigma^2} = 2\mathbf{N}\mathbf{h}_\mu - \mathbf{h}_\mu - \mathbf{h}_\mu^2, \tag{9}$$

where $\mathbf{h}_\mu^2$ means elementwise squaring. The quantity $\mathbf{N}\mathbf{h}_\mu$ can be calculated by applying the same fast iterative method used to solve (6) to the system $(\mathbf{I} - \mathbf{Q})\mathbf{x} = \mathbf{h}_\mu$. Then the HT standard deviation $\mathbf{h}_\sigma$ 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 $\mathbf{N}$ 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 $\mathbf{L} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{l}_{12} \\ \mathbf{l}_{21}^T & l_{nn} \end{pmatrix}$ be an $n \times n$ irreducible matrix such that $\mathsf{nullity}(\mathbf{L}) = 1$. Let $\mathbf{M} = \mathbf{L}^\dagger$ be the pseudo-inverse of $\mathbf{L}$ partitioned similarly and assume $\mathbf{v}^T\mathbf{L} = 0$, $\mathbf{Lu} = 0$, where $\mathbf{u}, \mathbf{v}$ are partitioned as $\mathbf{u}^T = (\mathbf{u}_1^T, u_n)$ and $\mathbf{v}^T = (\mathbf{v}_1^T, v_n)$. Assume $u_n > 0$ and $v_n > 0$. Then the inverse of the $(n-1) \times (n-1)$ matrix $\mathbf{L}_{11}$ exists and is given by

$$\begin{aligned} \mathbf{L}_{11}^{-1} &= \left(\mathbf{I}_{n-1} + \frac{\mathbf{u}_1\mathbf{u}_1^T}{u_n^2}\right)\mathbf{M}_{11}\left(\mathbf{I}_{n-1} + \frac{\mathbf{v}_1\mathbf{v}_1^T}{v_n^2}\right) \\ &= (\mathbf{I}_{n-1}\,,\ -\mathbf{u}_1/u_n)\begin{pmatrix} \mathbf{M}_{11} & \mathbf{m}_{12} \\ \mathbf{m}_{21}^T & m_{nn} \end{pmatrix}\begin{pmatrix} \mathbf{I}_{n-1} \\ -\mathbf{v}_1^T/v_n \end{pmatrix}. \end{aligned}$$

(b) Conversely, we can write $\mathbf{L}$ and $\mathbf{M} = \mathbf{L}^\dagger$ in terms of $\mathbf{L}_{11}$, $\mathbf{L}_{11}^{-1}$, $\mathbf{u}$, $\mathbf{v}$ as follows

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{11} & -\frac{1}{u_n}\mathbf{L}_{11}\mathbf{u}_1 \\ -\frac{1}{v_n}\mathbf{v}_1^T\mathbf{L}_{11} & \frac{\mathbf{v}_1^T\mathbf{L}_{11}\mathbf{u}_1}{u_n v_n} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n-1} \\ -\frac{1}{v_n}\mathbf{v}_1^T \end{bmatrix} \mathbf{L}_{11} \begin{bmatrix} \mathbf{I}_{n-1}, & -\frac{1}{u_n}\mathbf{u}_1 \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{m}_{12} \\ \mathbf{m}_{21}^T & m_{nn} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n-1} - \frac{\mathbf{u}_1\mathbf{u}_1^T}{\mathbf{u}^T\mathbf{u}} \\ -\frac{u_n}{\mathbf{u}^T\mathbf{u}} \cdot \mathbf{u}_1^T \end{bmatrix} \mathbf{L}_{11}^{-1} \begin{bmatrix} \mathbf{I}_{n-1} - \frac{\mathbf{v}_1\mathbf{v}_1^T}{\mathbf{v}^T\mathbf{v}}, -\frac{v_n}{\mathbf{v}^T\mathbf{v}} \cdot \mathbf{v}_1 \end{bmatrix}$$

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

$$\widetilde{\mathbf{P}} = \begin{bmatrix} \mathbf{Q} & \mathbf{r} \\ \mathbf{s}^T & t \end{bmatrix} \tag{10}$$

with strictly positive stationary probabilities $\pi_i > 0$, $i = 1, \ldots, n$ such that $\boldsymbol{\pi}^T\widetilde{\mathbf{P}} = \boldsymbol{\pi}^T$. Here $\boldsymbol{\pi}^T = (\pi_1, \ldots, \pi_n)$. 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 $\mathbf{\Pi}$ =Diag($\boldsymbol{\pi}$) denote the diagonal matrix with the stationary probabilities on the diagonal. Then the diagonally scaled matrix $\mathbf{I} - \widehat{\mathbf{P}} = \mathbf{\Pi}^{1/2}(\mathbf{I} - \widetilde{\mathbf{P}})\mathbf{\Pi}^{-1/2}$ enjoys the property that its left and right nullspaces are the same, spanned by the single vector $\sqrt{\boldsymbol{\pi}} = (\sqrt{\pi_1}, \ldots, \sqrt{\pi_n})^T$, and that $\mathbf{I} - \widehat{\mathbf{P}} + \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T$ 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 nunber of edges in the underlying graph. The matrix $\mathbf{I} - \widehat{\mathbf{P}}$ is singular, of course, but its pseudo-inverse can be written using an ordinary inverse:

$$(\mathbf{I} - \widehat{\mathbf{P}})^\dagger = (\mathbf{I} - \widehat{\mathbf{P}} + \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T)^{-1} - \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T. \tag{11}$$

Hence we can compute the matrix-vector product $(\mathbf{I} - \widehat{\mathbf{P}})^\dagger\mathbf{v}$ for any vector $\mathbf{v}$ by using a fast iterative method to solve the linear system $(\mathbf{I} - \widehat{\mathbf{P}} + \sqrt{\boldsymbol{\pi}}\sqrt{\boldsymbol{\pi}}^T)\mathbf{x} = \mathbf{v}$. Putting this all together means that solving the system (6) can be accomplished by embedding it within a strongly connected graph, scaling it by $\mathbf{\Pi}^{1/2}$, 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 Kirchoff 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 $\widehat{\mathbf{L}} = \mathbf{\Pi}(\mathbf{I} - \mathbf{P})$, even for strongly connected directed graphs [8]:

$$\mathbf{K} = \sum_{ij} \mathbf{H}(i,j) = n \cdot \sum_j \hat{m}_{jj} = n \cdot \mathrm{Tr}(\widehat{\mathbf{M}}), \tag{12}$$

where $\hat{m}_{ij}$ is the i,j-th element of the matrix $\widehat{\mathbf{M}} = \widehat{\mathbf{L}}^\dagger$, and $\mathrm{Tr}(\mathbf{A}) = \sum_j \mathbf{A}_{jj}$ stands for the trace of matrix $\mathbf{A}$, 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 oneselves to paths that go through the node q. If we denote this "average node peripherality" as $\mathbf{K}_q$, it can be shown [9] that this quantity equals

$$\mathbf{K}_q = \sum_{ij} \mathbf{H}_q(i,j) = n \cdot \mathrm{Tr}(\mathbf{M}) + n^2 m_{qq} = n \cdot (\mathbf{K} + n \cdot m_{qq}). \tag{13}$$

Hence the diagonal entries of the pseudo-inverse $\mathbf{M}$ 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 π_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:

$$\mathrm{CDF}(T) = \int_0^T \frac{\beta^\alpha}{\Gamma(\alpha)} \tau^{\alpha-1} e^{-\beta\tau} d\tau \tag{14}$$

with

$$\alpha = \frac{(\mathrm{mean}(T))^2}{\mathrm{var}(T)}, \ \beta = \frac{\mathrm{mean}(T)}{\mathrm{var}(T)}, \tag{15}$$

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. The hitting time cannot be less than the shortest path length $\mathbf{h}_{\min}$, 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.

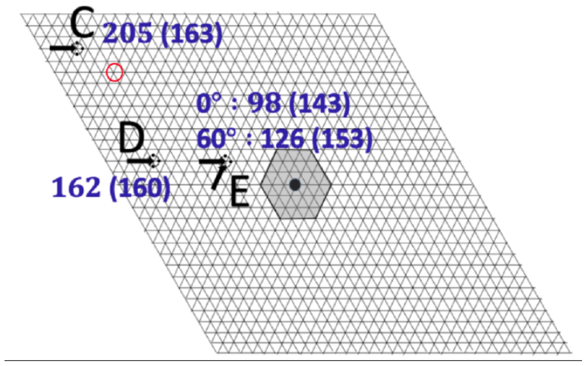

Figure 1: 30×30 arena showing 3 starting positions. Numbers are mean(std-dev) of hitting times to reach target nest (gray area in the middle) from each position.

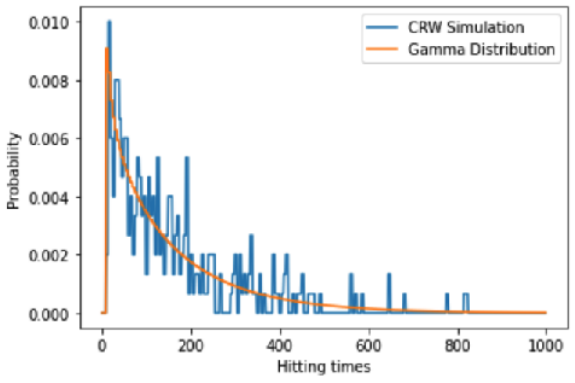

Figure 2: Hitting times for random walks starting from position D in Fig. 1.

We use a 30×30 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° 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 30×30 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).

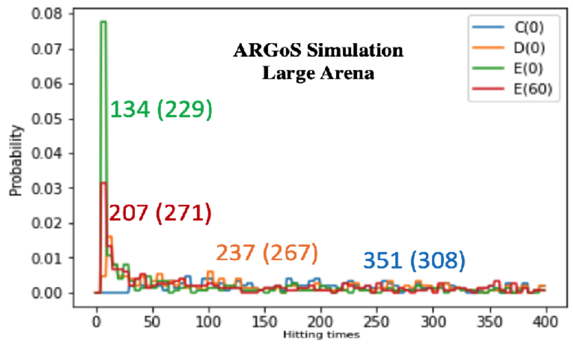

Figure 3: Hitting times from ARGoS simulations on 30×30 arena, starting in the upper left corner of the arena.

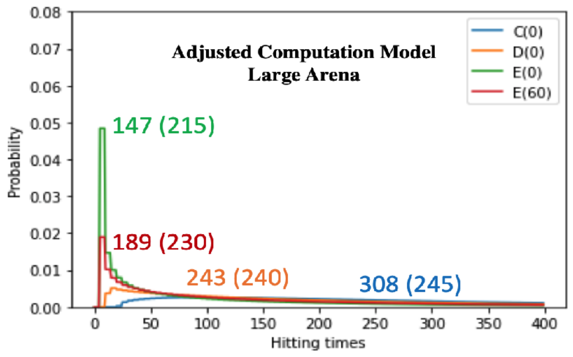

Figure 4: Hitting times from Correlated Random Walk simulations of the example of Fig. 3.

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:

$$C_n(T) = 1 - (1 - \mathrm{CDF}(T))^n. \tag{16}$$

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 30×30 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 150×150 arena within 500 time steps. By computing the estimated CDF on the 150×150 arena for various n, one can see from the resulting Figure 6 that 200 robots would be needed.

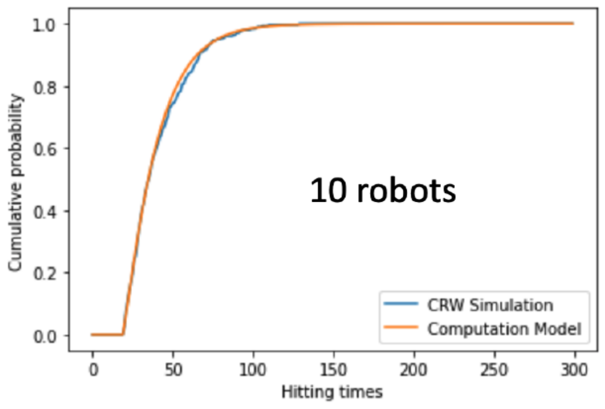

Figure 5: Times for the first of 10 robots to reach the goal node in the middle of a 30×30 arena: simulated vs calculated.

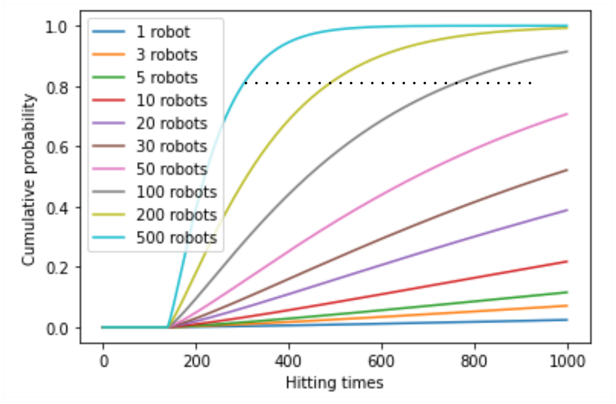

Figure 6: Calculated CDFs of hitting times for the first of many robots to reach the goal in the center of a 150×150 **arena.**

5 CONCLUSIONS AND FUTURE WORK

We have proposed using Markov chains and graph algorithms to study properties of swarms of robots engaged in foraging tasks. We have shown how to compute the mean and standard deviation of HT and the estimated distribution for the probability that at least one robot in the swarm will reach a target within a given upper-bound on the number of steps. This work is preliminary, but we hope it will encourage the community to use graph-based algorithms when studying swarms.

More theoretical work is needed in particular when robots are likely to collide, for instance, because their density is high as when they all start in a constrained area. Currently, most methods ignore collisions, assuming the density of the robots is low, so the probability of collision is low.

Work is also needed to model the probability of failures that could be caused by hardware breakdown, battery depletion, etc. This might entail extending our models, for instance using Markov decision processes to handle time-varying environments or even game theoretic methods to deal with adversarial agents.

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

[1] Aggarwal A, Gupta D, Vining WF, Fricke GM, and Moses ME. 2019. Ignorance is not bliss: an analysis of Central-Place Foraging algorithms. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, New York City, NY, 6510–6517. doi:10.1109/IROS40897.2019.8967665

[2] David Aldous and James Allen Fill. 2014. Reversible Markov Chains and Random Walks on Graphs. `http://www.stat.berkeley.edu/users/aldous/RWG/book.html`.

[3] Noga Alon. 1986. Eigenvalues and expanders. *Combinatorica* 6, 2 (1986), 83–96.

[4] Noga Alon, Chen Avin, Michal Kouckỳ, Gady Kozma, Zvi Lotker, and Mark R Tuttle. 2011. Many random walks are faster than one. *Combinatorics, Probability and Computing* 20, 4 (2011), 481–502.

[5] Yaniv Altshuler and Alfred Bruckstein. 2011. Static and Expanding Grid Coverage with Ant Robots: Complexity Results. *Theoretical Computer Science (TCS)* 412, 35 (2011), 4661–4674.

[6] RB Bapat. 2011. On the first passage time of a simple random walk on a tree. *Statistics & probability letters* 81, 10 (2011), 1552–1558.

[7] Martin T Barlow, Yuval Peres, and Perla Sousi. 2012. Collisions of random walks. *Annales de l'IHP Probabilités et statistiques* 48, 4 (2012), 922–946.

[8] Daniel Boley. 2021. On fast computation of directed graph Laplacian pseudo-inverse. *Linear Algebra Appl.* 623 (2021), 128–148.

[9] Daniel Boley, Gyan Ranjan, and Zhi-Li Zhang. 2011. Commute times for a directed graph using an asymmetric Laplacian. *Linear Algebra Appl.* 435, 2 (2011), 224–242.

[10] Graham Brightwell and Peter Winkler. 1990. Maximum hitting time for random walks on graphs. *Random Structures & Algorithms* 1, 3 (1990), 263–276.

[11] Richard A. Brualdi and Herbert J. Ryser. 1991. *Combinatorial Matrix Theory.* Cambridge Univ. Press, Cambridge, UK.

[12] S. Cannon, J. Daymude, C. Gokmen, D. Randall, and A. Richa. 2019. A local stochastic algorithm for separation in heterogeneous self-organizing particle systems. https://arxiv.org/abs/1805.04599.

[13] Pietro Caputo and Matteo Quattropani. 2020. Stationary distribution and cover time of sparse directed configuration models. *Probability Theory and Related Fields* 178, 3 (2020), 1011–1066.

[14] Pavel Chebotarev and Rafig Agaev. 2002. Forest matrices around the Laplacian matrix. *Lin. Alg. and its Appl.* 356, 1-3 (2002), 253–274.

[15] Haiyan Chen. 2007. The generating functions of hitting times for random walk on trees. *Statistics & probability letters* 77, 15 (2007), 1574–1579.

[16] F.R. Chung. 1997. *Spectral Graph Theory.* Am. Math. Soc., Providence, RI.

[17] Fan Chung. 2005. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics* 9, 1 (2005), 1–19.

[18] Michael B. Cohen, Jon Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. 2016. Faster algorithms for aomputing the stationary distribution, simulating random walks, and more. In *IEEE 57th Annual Symp. on Found. Comput. Sci. (FOCS)*. IEEE, New York City, NY, 583–592.

[19] Tugrul Dayar and Nail Akar. 2005. Computing moments of first passage times to a subset of states in Markov chains. *SIAM J. Matrix Anal. Appl.* 27, 2 (2005), 396–412.

[20] Tuğrul Dayar and William J. Stewart. 1996. On the effects of using the Grassmann–Taksar–Heyman method in iterative aggregation–disaggregation. *SIAM Journal on Scientific Computing* 17, 1 (1996), 287–303.

[21] Stéphane Devismes, Anissa Lamani, Franck Petit, Pascal Raymond, and Sébastien Tixeuil. 2012. Optimal grid exploration by asynchronous oblivious robots. In *Stabilization, Safety, and Security of Distributed Systems*, Andréa W. Richa and Christian Scheideler (Eds.). Springer, Berlin, Heidelberg, 64–76.

[22] Cristina Dimidov, Giuseppe Oriolo, and Vito Trianni. 2016. Random walks in swarm robotics: an experiment with kilobots. In *International Conference on Swarm Intelligence*. Springer, Berlin/Heidelberg, Germany, 185–196.

[23] Mucong Ding and Kwok Yip Szeto. 2018. First-passage time distribution for random walks on complex networks using inverse Laplace transform and mean-field approximation. arXiv preprint arXiv:1812.05598.

[24] W. E. Donath and A. J. Hoffman. 1973. Lower bounds for the partitioning of graphs. *IBM J. of Res. and Dev.* 17 (1973), 410–425.

[25] Qinglai Dong and Lirong Cui. 2019. First hitting time distributions for Brownian motion and regions with piecewise linear boundaries. *Methodology and Computing in Applied Probability* 21 (2019), 1–23.

[26] Klim Efremenko and Omer Reingold. 2009. How well do random walks parallelize? In *Approximation, Randomization, and Combinatorial Optimization.*

Algorithms and Techniques. APPROX RANDOM 2009 2009, Lecture Notes in Computer Science, I. Dinur, K. Jansen, J. Naor, and J. Rolim (Eds.). Vol. 5687. Springer, Berlin, Heidelberg.

[27] Robert Elsässer and Thomas Sauerwald. 2011. Tight bounds for the cover time of multiple random walks. *Theoretical Computer Science* 412, 24 (2011), 2623–2641.

[28] Uriel Feige. 1995. A tight upper bound on the cover time for random walks on graphs. *Random structures and algorithms* 6, 1 (1995), 51–54.

[29] Miroslav Fiedler. 1975. A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czechoslovak Math. J.* 25, 100 (1975), 619–633.

[30] Alexander H Foss, Richard B Lehoucq, W Zachary Stuart, J Derek Tucker, and Jonathan W Berry. 2020. A deterministic hitting-time moment approach to seed-set expansion over a graph. arXiv preprint arXiv:2011.09544.

[31] Francoise Fouss, Alain Pirotte, Jean michele Renders, and Marco Saerens. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. on Knowledge and Data Engineering* 19 (2007), 355–369.

[32] Ryusuke Fujisawa and Shigeto Dobata. 2013. Lévy walk enhances efficiency of group foraging in pheromone-communicating swarm robots. In *IEEE/SICE Int'l Symposium on System Integration, SII 2013*. IEEE, New York City, NY, 808–813.

[33] Alexandre Gaudillière. 2009. Collision probability for random trajectories in two dimensions. *Stochastic processes and their applications* 119, 3 (2009), 775–810.

[34] Golshan Golnari, Zhi-Li Zhang, and Daniel Boley. 2019. Markov fundamental tensor and its applications to network analysis. *Linear Algebra Appl.* 564 (2019), 126–158.

[35] Winfried K. Grassmann, Michael I. Taksar, and Daniel P. Heyman. 1985. Regenerative analysis and steady atate distributions for Markov Chains. *Operations Research* 33, 5 (1985), 1107–1116.

[36] Charles Miller Grinstead and James Laurie Snell. 1997. *Introduction to Probability*. Amer Math Soc., Boston, MA.

[37] Heiko Hamann. 2018. Modeling swarm systems and formal design methods. In *Swarm Robotics: A Formal Approach*. Springer, Berlin/Heidelberg, Germany, 95–127.

[38] John Harwell and Maria Gini. 2018. Broadening applicability of swarm-robotic foraging through constraint relaxation. In *2018 IEEE Int'l Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, New York City, NY, 116–122.

[39] John Harwell, Angel Sylvester, and Maria Gini. 2021. Characterizing the limits of linear modeling of non-linear swarm behaviors. https://arxiv.org/abs/2110.12307.

[40] Jeffrey J Hunter. 2013. The distribution of mixing times in Markov chains. *Asia-Pacific Journal of Operational Research* 30, 01 (2013), 1250045.

[41] Satoshi Ikeda, Izumi Kubo, and Masafumi Yamashita. 2009. The hitting and cover times of random walks on finite graphs using local degree information. *Theoretical Computer Science* 410, 1 (2009), 94–100.

[42] Minyoung Jeong, John Harwell, and Maria Gini. 2021. Analysis of exploration in swarm robotic systems. In *IAS-16*. Springer, Berlin/Heidelberg, Germany, 445–457.

[43] Miquel Kegeleirs, David Garzón Ramos, and Mauro Birattari. 2019. Random walk exploration for swarm mapping. In *Proc. Towards Autonomous Robotic Systems: 20th Annual Conference, TAROS 2019, Part II*. Springer-Verlag, Berlin, Heidelberg, 211–222.

[44] John G Kemeny and J Laurie Snell. 1983. *Finite Markov chains (with a new appendix "Generalization of a fundamental matrix")*. Springer, Berlin/Heidelberg, Germany.

[45] Joseph P. Lancaster and David A. Gustafson. 2013. Predicting the behavior of robotic swarms in search and tag tasks. *Procedia Computer Science* 20 (2013), 77–82. https://doi.org/10.1016/j.procs.2013.09.242 Complex Adaptive Systems.

[46] Hon Wai Lau and Kwok Yip Szeto. 2010. Asymptotic analysis of first passage time in complex networks. *EPL (Europhysics Letters)* 90, 4 (2010), 40005.

[47] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. 2017. *Markov Chains and Mixing Times*. Vol. 107. American Mathematical Soc., Providence, RI.

[48] Yanhua Li and Zhi-Li Zhang. 2012. Digraph Laplacian and the degree of asymmetry. *Internet Mathematics* 8, 4 (2012), 381–401.

[49] László Lovász. 1993. Random walks on graphs. *Combinatorics* 2, 1-46 (1993), 4.

[50] Qi Lu, G Matthew Fricke, John C Ericksen, and Melanie E Moses. 2020. Swarm foraging review: Closing the gap between proof and practice. *Current Robotics Reports* 1 (2020), 1–11.

[51] Saran Ishika Maiti and Jyotirmoy Sarkar. 2019. Symmetric walks on paths and cycles. *Mathematics Magazine* 92, 4 (2019), 252–268.

[52] Fredy Martinez, Edwar Jacinto, and Diego Acero. 2012. Brownian motion as exploration strategy for autonomous swarm robots. In *2012 IEEE Int'l Conf. on Robotics and Biomimetics (ROBIO)*. IEEE, New York City, NY, 2375–2380.

[53] Naoki Masuda, Mason A Porter, and Renaud Lambiotte. 2017. Random walks and diffusion on networks. *Physics reports* 716 (2017), 1–58.

[54] Ronald B Morgan. 2002. GMRES with deflated restarting. *SIAM J Sci Comput* 24, 1 (2002), 20–37.

[55] RB Nain and Kanwar Sen. 1980. Transition probability matrices for correlated random walks. *Journal of Applied Probability* 17, 1 (1980), 253–258.

[56] Shunhao Oh, Dana Randall, and Andréa W. Richa. 2022. Foraging in Particle Systems via Self-Induced Phase Changes. https://arxiv.org/abs/2208.10720. , 51:1-51:3 pages.

[57] Bao Pang, Yong Song, Chengjin Zhang, Hongling Wang, and Runtao Yang. 2019. A swarm robotic exploration strategy based on an improved random walk method. *Journal of Robotics* 2019, Article ID 6914212 (2019), 9.

[58] Bao Pang, Yong Song, Chengjin Zhang, and Runtao Yang. 2021. Effect of random walk methods on searching efficiency in swarm robots for area exploration. *Applied Intelligence* 51, 7 (2021), 5189–5199.

[59] Rushabh Patel, Andrea Carron, and Francesco Bullo. 2016. The hitting time of multiple random walks. *SIAM J. Matrix Anal. Appl.* 37, 3 (2016), 933–954.

[60] Clifford S. Patlak. 1953. *Random Walk with Persistence and External Bias: A Mathematical Contribution to the Study of Orientation of Organisms*. University of Chicago, Committee on Mathematical Biology.

[61] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, et al. 2011. ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In *IEEE/RSJ Intel Conf Intelligent Robots and Systems*. IEEE, New York City, 5027–5034.

[62] Jacob Richey. 2018. Collisions of random walks and related diffusions. https://arxiv.org/abs/1003.3255.

[63] Nicolás Rivera, Thomas Sauerwald, and John Sylvester. 2020. Multiple random walks on graphs: mixing few to cover many. arXiv preprint arXiv:2011.07893.

[64] Madhumita Sardar, Deepanwita Das, and Srabani Mukhopadhyaya. 2022. Grid exploration by a swarm of autonomous robots with minimum repetitions. *Theoretical Computer Science* 933 (2022), 67–87.

[65] Jambo Shi and Jitendra Malik. 2000. Normalised cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905.

[66] Ulrike von Luxburg. 2007. A Tutorial on spectral clustering. *Statistics and Computing* 17, 4 (December 2007), 395–416. Max Planck Institute for Biological Cybernetics. Technical Report No. TR-149.

[67] Israel Wagner, Yaniv Altshuler, Vladimir Yanovski, and Alfred Bruckstein. 2008. Cooperative cleaners: A study in ant robotics. *The International Journal of Robotics Research (IJRR)* 27, 1 (2008), 127–151.

[68] Yi Zhang, Daniel Boley, John Harwell, and Maria Gini. 2022. 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*. Springer, Berlin/Heidelberg, Germany, 724–736.

[69] Yi Zhang, Daniel Boley, John Harwell, and Maria Gini. 2022. 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)*. Springer, 724–736.

[70] Zhongzhi Zhang, Alafate Julaiti, Baoyu Hou, Hongjuan Zhang, and Guanrong Chen. 2011. Mean first-passage time for random walks on undirected networks. *The European Physical Journal B* 84, 4 (2011), 691–697.

[71] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2005. Learning from labeled and unlabeled data on a directed graph. In *Proc. 22nd Int'l Conf. Machine Learning*. ACM, New York City, NY, 1041–1048.

[72] Nikola Zlatanov and Ljupco Kocarev. 2009. Random walks on networks: Cumulative distribution of cover time. *Physical Review E* 80, 4 (2009), 041102.