Re-establishing communication in teams of mobile robots

Isaac Vandermeulen[1], Roderich Groß[1], and Andreas Kolling[2]

***Abstract*— As communication is important for cooperation, teams of mobile robots need a way to re-establish a wireless connection if they get separated. We develop a method for mobile robots to maintain a belief of each other's positions using locally available information. They can use their belief to plan paths with high probabilities of reconnection. This approach also works for subteams cooperatively searching for a robot or group of robots that they would like to reconnect with. The problem is formulated as a constrained optimization problem which is solved using a branch-and-bound approach. We present simulation results showing the effectiveness of this strategy at reconnecting teams of up to five robots and compare the results to two other strategies.**

I. INTRODUCTION

Communication is essential for the successful completion of most tasks performed by teams of mobile robots. In real environments, robots often communicate over inexpensive ad-hoc networks which have limited connectivity that is affected by distance and line of sight [1]. The robots may lose connectivity as they move throughout their environment. One solution to this problem is to restrict robots' motion to maintain connectivity, making them less effective at other tasks. Another solution is to have the team separate temporarily and meet occasionally to share information. Regular or preplanned meetings give robots some flexibility to separate, but are inconvenient when tasks take unpredictable lengths of time. If instead, the robots do not have a prearranged meeting, they have to find each other without sharing any common information. This problem can be described in one of three ways depending on the target robot's behavior. Its behavior can be a) cooperative, b) adversarial, or c) neutral. These problems are commonly known as rendezvous, pursuit-evasion, and search. In practice, a searcher often does not know whether its target is cooperative, adversarial, or neutral and should use a strategy which can be effective regardless of its target's objectives.

In this paper, we design a flexible communication strategy that can be used when completing a cooperative task. This strategy allows for varying degrees of communication so that robots can benefit from cooperation without wasting excessive energy to communicate. We do not require robots to communicate constantly or at fixed intervals. Without the objective of constant communication, the team of robots will in general be disconnected. If a robot wants to communicate with a disconnected robot, it searches for that target robot using its belief of the target's position. This belief is estimated using a probabilistic model of the target's motion. This problem is unique because it explicitly considers the communication structure of the environment. Reconnection is successful if two robots are within a communication range, which depends on the environment's and robot's, properties. The robots do not need to be in the exact same location to successfully reconnect.

[1] Isaac Vandermeulen and Roderich Groß are with the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK {iavandermeulen1,r.gross}@sheffield.ac.uk

[2] Andreas Kolling is with iRobot, Pasadena, California, USA akolling@irobot.com

A. Related work

When there is limited communication, a common approach is to enforce that connectivity be maintained at all times. Connectivity can be enforced by following a gradient ascent of the Fiedler eigenvalue [2], [3], by using a potential field [4], or by cooperatively planning paths [5], [6]. Alternatively, robots can meet periodically at a prearranged meeting time and place [7] or at prearranged locations but not times which results in some waiting [8].

Rendezvous can be symmetric or asymmetric depending on if all the robots use the same strategy. In asymmetric rendezvous, the optimal strategy is for a robot with a known ID to remain still while the other robot visits every vertex [9]. The symmetric version of this strategy is for each robot to randomly choose to visit every vertex or wait for a fixed length of time. Another symmetric approach is for robots to move randomly between several unique vertices that they have identified [10], [11], [12].

Search can involve a stationary or mobile target. For a stationary target, search is equivalent to the traveling salesman problem if the target is located on vertices [13] or the Chinese postman problem if it is located on edges [14]. A moving target can be modeled using a Markov model with the searcher attempting to maximize the probability of detection over a given time horizon [15]. For multiple cooperative searchers, the optimization problem is exponential in the number of searchers but this complexity can be reduced through implicit coordination [16]. Searchers who only communicate occasionally can fuse their beliefs of a target's location to obtain a better combined belief when they meet [17].

As a target's behavior can have a significant effect on the searcher's strategy, approaches generally assume that the target's behavior is known. If its behavior is unknown, a hybrid approach such as the rendezvous-evasion can be used [18]. To the best of our knowledge no approaches exist for the combination of cooperative rendezvous and target tracking.

B. Notation

We will rely heavily on linear algebraic objects including scalars, vectors, covectors, matrices, and tensors. Note that all of objects can be described in a uniform way as tensors. Any tensor space can be described as the tensor product of vector spaces and their duals, covector spaces. $V_1 \otimes V_2$ denotes the tensor product of (co)vector spaces V_1 and V_2. V^* represents the dual vector space of V. We will write tensors using the Einstein summation notation which implies summation over pairs of superscripts and subscripts.

In this paper, $(0,0)$-Tensors (i.e. scalars) are denoted by lower case (non-bold) characters. $(1,0)$-Tensors (i.e. vectors) are denoted by lower case bold characters with subscripts for basis vectors. $(0,1)$-Tensors (i.e. covectors) are denoted by lower case bold characters with superscripts for basis covectors. $(1,1)$-Tensors (i.e. matrices) are denoted by upper case bold characters. Scalars, vectors, and matrices are all written in an italic font. Higher order tensors are also denoted by upper case bold characters, but they use an upright font. Sets related to graph theory are denoted by script characters.

II. MULTI-ROBOT COORDINATION WITH INTERMITTENT COMMUNICATION

A. Environment model

Consider a team of p robots moving in a known undirected graph $\mathcal{G}_\mathrm{e} = (\mathcal{V}, \mathcal{E}_\mathrm{e})$ with $|\mathcal{V}| = n_\mathrm{e}$. This graph can be constructed using an exact or approximate cellular decomposition [19]. Let $\boldsymbol{A}_\mathrm{e} \in \mathrm{Hom}(\mathbb{R}^{n_\mathrm{e}})$ be the adjacency matrix of $\mathcal{G}_\mathrm{e}$ where $\mathrm{Hom}(V) = V \otimes V^*$ is the set of homomorphisms (i.e. linear maps) on V. At time $t \in \mathbb{Z}_{\geq 0}$, the robots' positions can be represented by an indicator matrix,

$$\boldsymbol{Q}[t] = q_j^v[t]\boldsymbol{e}_v^j \in \mathbb{R}^{n_\mathrm{e}} \otimes (\mathbb{R}^p)^*$$

where $\mathbb{R}^{n_\mathrm{e}} \otimes (\mathbb{R}^p)^*$ is the set of linear maps from $\mathbb{R}^p$ to $\mathbb{R}^{n_\mathrm{e}}$ and $\boldsymbol{e}_v^j$ is a basis element which maps $\boldsymbol{e}_j \in \mathbb{R}^p$ to $\boldsymbol{e}_v \in \mathbb{R}^{n_\mathrm{e}}$ and all other bases of $\mathbb{R}^p$ to $\boldsymbol{0} \in \mathbb{R}^{n_\mathrm{e}}$. The components of $\boldsymbol{Q}[t]$ are defined as

$$q_j^v[t] = \begin{cases} 1 & \text{if robot } j \text{ is at vertex } v \text{ at time } t \\ 0 & \text{otherwise.} \end{cases}$$

As each robot is only located at one vertex, 1 appears exactly once in each column. Therefore $\boldsymbol{Q}^\top[t]\boldsymbol{Q}[t] = \boldsymbol{I} \in \mathrm{Hom}(\mathbb{R}^p)$ and so $\boldsymbol{Q}^\top[t]$ is a left inverse for $\boldsymbol{Q}[t]$.

B. Communication model

Two identical robots may or may not be able to communicate depending on where they are located in the environment. Their ability to communicate depends on their distance from each other and any obstacles between them [20]. We use a second graph $\mathcal{G}_\mathrm{c} = (\mathcal{V}, \mathcal{E}_\mathrm{c})$ to describe when robots can communicate. This graph is based on the cellular decomposition of the environment and shares the same vertex set, $\mathcal{V}$, with $\mathcal{G}_\mathrm{e}$. Its edge set, $\mathcal{E}_\mathrm{c}$, contains an edge (v_1, v_2) if and only if a robot located at v_1 can communicate with a robot located at v_2. This topological definition of communication can be used to represent many different types of communication such as line-of-site, distance limited, and full communication.

At any time t, the ad-hoc network formed by the robots, $\mathcal{G}_\mathrm{r}$, depends on the communication properties of the environment and on the robot positions. Let $\boldsymbol{A}_\mathrm{c} \in \mathrm{Hom}(\mathbb{R}^{n_\mathrm{e}})$ and $\boldsymbol{A}_\mathrm{r} \in \mathrm{Hom}(\mathbb{R}^p)$ be the adjacency matrices of $\mathcal{G}_\mathrm{c}$ and $\mathcal{G}_\mathrm{r}$. These adjacency matrices are related by

$$\boldsymbol{A}_\mathrm{r}[t] = \boldsymbol{Q}^\top[t]\boldsymbol{A}_\mathrm{c}\boldsymbol{Q}[t].$$

We can easily check if the robots are connected by looking at λ_2, the second smallest eigenvalue of the Laplacian $\boldsymbol{L}_\mathrm{r}[t]$ which can be computed from $\boldsymbol{A}_\mathrm{r}[t]$ [21].

C. Reconnection objective

When robots are disconnected, they will eventually need to reconnect. Each robot has one or more target robots. Its individual objective is to find a path which maximizes the probability of reconnecting with at least one of its targets. If multiple robots are connected, they can plan their paths together to maximize the probability of finding one of their targets while remaining connected to each other.

The team objective determines which targets each robot has at a given time. This objective depends on what other tasks the robots are completing. If the only task is to connect all robots, each robot might have all disconnected robots as targets. This approach may result in livelock which can be avoided by using an asymmetric strategy where different robots have different targets. The target graph is a directed graph with one vertex per robot and an edge from robot i to robot j if robot j is one of robot i's targets. Livelock can be avoided by using a target graph which is weakly connected, has no directed cycles, and has exactly one sink vertex. Once a robot has no more targets, it stops moving. This approach guarantees eventual connectedness of the team of robots.

Other applications may not require all robots to be searching simultaneously. For example, in coverage, each robot is assigned a list of tasks that it must complete independently. As realistic tasks take variable times, some robots will finish their tasks sooner than others. When a robot still has tasks to complete, its objective is its own tasks but it may communicate with other robots which happen to be nearby. After it has finished its own tasks, a robot can be assigned all disconnected robots as targets. Similarly, in monitoring tasks, robots do not have targets while they are gathering data; once a robot has gathered enough data, it can choose any robots it wants share those data with as targets. As this paper is not concerned with a specific application, we consider the general problem where each individual robot can have any set of targets.

III. ESTIMATION OF OTHER ROBOTS' POSITIONS

Robots search for disconnected robots by following the path which maximizes the probability of finding at least one of their targets. This probability is computed from the belief of the other robots' position which is based on a model of a target's motion. This approach is general enough to account for the robots moving and performing tasks at variable speeds and stopping for indeterminate amounts of time.

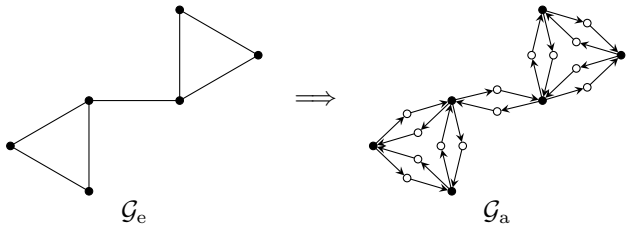

Fig. 1. Comparison of a graph, $\mathcal{G}_\mathrm{e}$, (left) and its augmented graph, $\mathcal{G}_\mathrm{a}$ (right)

A. Augmented graph

The probabilistic motion model we are using is based on a cellular decomposition of the continuous real-world environment. Each vertex of $\mathcal{G}_\mathrm{e}$ represents a discrete region of this environment. The edges describe which regions a robot can move between. This graph does not, however, provide a way to indicate that a robot is currently traveling between two regions.

To explicitly allow for states representing travel, we create a new graph with extra vertices. This augmented graph, $\mathcal{G}_\mathrm{a} = (\mathcal{V}_\mathrm{a}, \mathcal{E}_\mathrm{a})$, is directed and is obtained from $\mathcal{G}_\mathrm{e}$ by replacing each edge with two vertices and four directed edges (Figure 1). Each new vertex represents one direction of motion along the original edge. The augmented graph is a larger graph. If the environment graph, $\mathcal{G}_\mathrm{e}$ has n_e vertices and N_e edges, the augmented graph, $\mathcal{G}_\mathrm{a}$ has $n_\mathrm{a} = n_\mathrm{e} + 2N_\mathrm{e}$ vertices and $N_\mathrm{a} = 4N_\mathrm{e}$ edges. For an arbitrary environment graph, there may be as many as $\frac{n_\mathrm{e}(n_\mathrm{e}-1)}{2}$ edges; however cellular decompositions generally result in planar graphs. Euler's characteristic relates the number of faces, edges, and vertices of a convex polyhedron and can be used to establish that $N_\mathrm{e} \leq 3n_\mathrm{e} - 6$ for planar graphs with $n \geq 3$ [22]. Alternatively, if a decomposition results in a graph whose vertices have maximum degree D which does not depend on n_e, then $N_\mathrm{e} \leq \frac{D}{2} n_\mathrm{e}$. In both cases, N_e is $\mathcal{O}(n_\mathrm{e})$ so n_a is also $\mathcal{O}(n_\mathrm{e})$ and the complexity of the algorithm does not increase when using $\mathcal{G}_\mathrm{a}$ instead of $\mathcal{G}_\mathrm{e}$.

B. Semi-Markov motion model

A semi-Markov process is characterized by a sequence of states $q_1, q_2, \cdots \in \mathcal{V}_\mathrm{a}$ and transition times $s_1 \leq s_2 \leq \cdots \in \mathbb{T}$ where $\mathcal{V}_\mathrm{a}$ is a finite set of states and $\mathbb{T}$ is a set of possible transition times (usually $\mathbb{Z}_{\geq 0}$ or $\mathbb{R}_{\geq 0}$). During the interval $[s_i, s_{i+1})$, the system is in state q_i. At time s_{i+1}, the system changes to state q_{i+1}. Semi-Markov processes are related to Markov processes as the sequence of states, $q(s_1), q(s_2), \ldots$ is Markov. The advantage of using a semi-Markov model instead of a Markov model as was done in [16], is that it allows us to explicitly model variable task and transit times. The time between transitions follows a distribution that only depends on the current state. The state and time transition distributions can be described by

$$\begin{aligned} \mu_{v_1}(v_2) &= \mathbb{P}(q_{i+1} = v_2 \mid q_i = v_1) \\ \gamma_v(\Delta t) &= \mathbb{P}(s_{i+1} = t + \Delta t \mid s_i = t \wedge q_i = v) \end{aligned} \tag{1}$$

where $\mu_{v_1}(v_2)$ is the probability that the robot's next state is v_2 given that its current state is v_1 and $\gamma_v(\Delta t)$ is the probability that the next transition will happen once Δt time has passed.

The probability that a transition occurs between $s_i + a$ and $s_i + b$ is $\int_a^b \gamma_v(\mathrm{d}t)$. At $s_i + \Delta t_1$, the probability that the next transition will occur before $s_i + \Delta t_2$ is

$$\delta_{v,\Delta t_1}(\Delta t_2) = \frac{\int_{\Delta t_1}^{\Delta t_2} \gamma_v(\mathrm{d}t)}{\int_{\Delta t_1}^{\infty} \gamma_v(\mathrm{d}t)}. \tag{2}$$

Using μ and δ, we can update probability distributions using the semi-Markov model.

We consider a discrete-time semi-Markov model with a maximum holding time, $\Delta t = T$. The state of any robot can be described by a pair, $(v, \Delta t) \in \mathcal{V}_\mathrm{a} \times \{1, \ldots, T\}$, which completely determines its transition probabilities. Its position is v and the time since its last transition is Δt.

Suppose robot i is estimating the states of all other robots. The belief of a single robot's state is a probability distribution over $\mathcal{V}_\mathrm{a} \times \{1, \ldots, T\}$, which can be stored as a $(2,0)$-tensor in $\mathbb{R}^{n_\mathrm{a}} \otimes \mathbb{R}^T$. We aggregate all these distributions into a $(2,1)$-tensor, $\widehat{\mathbf{Q}} \in \mathbb{R}^{n_\mathrm{a}} \otimes \mathbb{R}^T \otimes (\mathbb{R}^p)^*$. Robot i's belief of robot j's position is a slice, $\widehat{\mathbf{Q}}\boldsymbol{e}_j$, of this tensor where $\boldsymbol{e}_j \in \mathbb{R}^p$ is the j^{th} basis vector of $\mathbb{R}^p$. This tensor can be expressed in coordinates as

$$\widehat{\mathbf{Q}} = q_j^{v,\Delta t} \boldsymbol{e}_v \otimes \boldsymbol{e}_{\Delta t} \otimes \boldsymbol{e}^j$$

where $\boldsymbol{e}_v$, $\boldsymbol{e}_{\Delta t}$, and $\boldsymbol{e}^j$ are basis (co)vectors for $\mathbb{R}^{n_\mathrm{a}}$, $\mathbb{R}^T$, and $(\mathbb{R}^p)^*$. The coefficient is the probability that robot j has been at vertex v for time Δt and is defined as

$$q_j^{v,\Delta t} = \mathbb{P}(q(j,t) = v \wedge s(j,t) = t - \Delta t \mid \mathcal{I}_i[t])$$

where $q(j,t)$ is the position of robot j at time t, $s(j,t)$ is its most recent transition time, and $\mathcal{I}_i[t]$ is robot i's information at time t.

As time progresses, robot i should update $\widehat{\mathbf{Q}}$. To update $\widehat{\mathbf{Q}}$, we need an endomorphism on the space of $(2,1)$-tensors. This endomorphism is a $(2,2)$-tensor $\mathbf{F} \in \mathrm{Hom}(\mathbb{R}^{n_\mathrm{a}} \otimes \mathbb{R}^T)$ which encodes all the transitions of the semi-Markov model. When $\mathbf{F}$ is multiplied by $\widehat{\mathbf{Q}}$, the resulting product is a $(2,1)$-tensor. This idea is analogous to how an $n_\mathrm{a} \times n_\mathrm{a}$ matrix (a $(1,1)$-tensor) is a linear map from $\mathbb{R}^{n_\mathrm{a}}$ to $\mathbb{R}^{n_\mathrm{a}}$ but can also be used as a map from $\mathbb{R}^{n_\mathrm{a} \times p}$ to $\mathbb{R}^{n_\mathrm{a} \times p}$ (which are also $(1,1)$-tensor spaces). It can be expressed as

$$\mathbf{F} = f_{v_1,\Delta t_1}^{v_2,\Delta t_2} \boldsymbol{e}_{v_2} \otimes \boldsymbol{e}_{\Delta t_2} \otimes \boldsymbol{e}^{v_1} \otimes \boldsymbol{e}^{\Delta t_1}$$

where

$$\begin{aligned} f_{v_1,\Delta t_1}^{v_2,\Delta t_2} = \mathbb{P}\big(&q(\,\cdot\,, t+1) = v_2 \wedge s(\,\cdot\,, t+1) = t + 1 - \Delta t_2 \\ &\big| q(\,\cdot\,, t) = v_1 \wedge s(\,\cdot\,, t) = t - \Delta t_1\big). \end{aligned}$$

$\mathbf{F}$ can be constructed from the $(1,1)$-tensors $\boldsymbol{M} = m_{v_1}^{v_2} \boldsymbol{e}^{v_1} \otimes \boldsymbol{e}_{v_2} \in \mathrm{Hom}(\mathbb{R}^{n_\mathrm{a}})$ which encodes the Markovian state transition probabilities and $\boldsymbol{D}(v) = d_{\Delta t_1}^{\Delta t_2}(v) \boldsymbol{e}^{\Delta t_1} \otimes \boldsymbol{e}_{\Delta t_2} \in \mathrm{Hom}(\mathbb{R}^T)$ which encodes the distribution for the holding time at vertex v. The coefficients for these tensors are

$$m_{v_1}^{v_2} = \mathbb{P}(q(\,\cdot\,, t+1) = v_2 \mid q(\,\cdot\,, t) = v_1)$$

$$d^{\Delta t_2}_{\Delta t_1}(v) = \mathbb{P}\big(s(\,\cdot\,, t+1) = t + 1 - \Delta t_2 \\ |s(\,\cdot\,, t) = t - \Delta t_1 \wedge q(\,\cdot\,, t) = v\big)$$

which are related to the μ's and δ's defined in (1) and (2). Trivially, $m^{v_2}_{v_1} = \mu_{v_1}(v_2)$. There are two cases to consider to understand how d relates to δ:

1) If a transition occurs at t, then $\Delta t_2 = 1$ and the previous position must have had a holding time of Δt_1. Therefore the transition probability is

$$d^1_{\Delta t_1}(v) = \delta_{v,\Delta t_1}(\Delta t_1). \tag{3}$$

2) If a transition does not occur at t, then $\Delta t_2 = \Delta t_1 + 1$. As all other possible values of Δt_2 have a probability of 0 and $d^1_{\Delta t_1}(v) + d^{\Delta t_1+1}_{\Delta t_1}(v) = 1$ so

$$d^{\Delta t_1+1}_{\Delta t_1}(v) = 1 - \delta_{v,\Delta t_1}(\Delta t_1). \tag{4}$$

Therefore $\boldsymbol{D}(v)$ depends only on $\delta_{v,1}(1), \ldots, \delta_{v,T}(T)$ which we will refer to as $d_1, \ldots, d_T$. Then

$$\boldsymbol{D}(v) = \begin{bmatrix} d_1 & d_2 & d_3 & \ldots & d_T \\ 1-d_1 & 0 & 0 & \ldots & 0 \\ 0 & 1-d_2 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1-d_{T-1} & 0 \end{bmatrix}. \tag{5}$$

The top row of $\boldsymbol{D}(v)$ represents transitions and the bottom $T-1$ rows represent holds. We can obtain the matrices that represent these two parts by left multiplication with

$$\boldsymbol{E}_{\mathrm{t}} = \boldsymbol{e}^1_1 \qquad \text{and} \qquad \boldsymbol{E}_{\mathrm{h}} = \sum_{\Delta t=2}^{T} \boldsymbol{e}^{\Delta t}_{\Delta t}.$$

Then $\boldsymbol{E}_{\mathrm{t}}\boldsymbol{D}(v)$ has the same first row of $\boldsymbol{D}(v)$ and all other rows are zero whereas $\boldsymbol{E}_{\mathrm{h}}\boldsymbol{D}(v)$ is equal to the last $T-1$ rows of $\boldsymbol{D}(v)$ with the first row equal to zero.

We can create $(2,2)$-tensors which include information on how the state changes by taking the tensor product of $\boldsymbol{E}_{\mathrm{t}}\boldsymbol{D}(v)$ and $\boldsymbol{E}_{\mathrm{h}}\boldsymbol{D}(v)$ and tensors in $\mathrm{Hom}(\mathbb{R}^{n_{\mathrm{a}}})$. When transitions occur, we can use $\boldsymbol{M}$; when transitions do not occur, we can use the identity tensor $\boldsymbol{I} \in \mathrm{Hom}(\mathbb{R}^{n_{\mathrm{a}}})$. As $\boldsymbol{D}(v)$ depends on v, we must sum $\boldsymbol{E}_{\mathrm{t}}\boldsymbol{D}(v)$ and $\boldsymbol{E}_{\mathrm{h}}\boldsymbol{D}(v)$ over all $v \in \mathcal{V}_{\mathrm{a}}$. We take the tensor product with individual columns of $\boldsymbol{M}$ which can be obtained by right multiplication with $\boldsymbol{E}(v) = \boldsymbol{e}^v \otimes \boldsymbol{e}_v$. Then all transitions when the robot moves can be described by

$$\mathbf{F}_{\mathrm{t}} = \sum_{v \in \mathcal{V}_{\mathrm{a}}} \left(\boldsymbol{M}\boldsymbol{E}(v)\right) \otimes \left(\boldsymbol{E}_{\mathrm{t}}\boldsymbol{D}(v)\right).$$

Similarly, all transitions when the robot stays still are described by

$$\mathbf{F}_{\mathrm{h}} = \sum_{v \in \mathcal{V}_{\mathrm{a}}} \left(\boldsymbol{I}\boldsymbol{E}(v)\right) \otimes \left(\boldsymbol{E}_{\mathrm{h}}\boldsymbol{D}(v)\right).$$

The overall tensor is $\mathbf{F} = \mathbf{F}_{\mathrm{t}} + \mathbf{F}_{\mathrm{h}}$. Robot i can use $\mathbf{F}$ and its current belief to compute where it believes the other robot will be at the next time step by

$$\widehat{\mathbf{Q}}[t+1] = \mathbf{F}\widehat{\mathbf{Q}}[t]. \tag{6}$$

As each slice of $\widehat{\mathbf{Q}}[t]$ is a probability tensor, its elements sum to 1. The way that $\mathbf{F}$ is defined ensures that $\widehat{\mathbf{Q}}[t+1]$'s elements also sum to 1 so it is a valid probability tensor.

The semi-Markov model only depends on $\boldsymbol{M}$ and $\boldsymbol{D}(v)$, as $\boldsymbol{E}(v)$, $\boldsymbol{E}_{\mathrm{t}}$, and $\boldsymbol{E}_{\mathrm{h}}$ are constant matrices that do not depend on robot behavior. $\boldsymbol{M}$ is constructed using the transition probabilities $\mu_v(w)$. $\boldsymbol{D}(v)$ is computed from (2), (3), (4), and (5) and is completely determined by $\gamma_v(\Delta t)$. Therefore, the entire semi-Markov model can be created just using $\mu_v(w)$ and $\gamma_v(\Delta t)$, which characterize where a robot will go after leaving v and how long it will stay at v.

The value of $\mu_v(w)$ is zero if (v,w) is not an edge of $\mathcal{G}_{\mathrm{a}}$. The value of a non-zero $\mu_v(w)$ depends on the task that the robots are performing. For example, in coverage, each robot has a planned path and we use a large $\mu_v(w)$ if w follows v in a path and a small $\mu_v(w)$ for other neighbors of v which do not follow it on the path. These small values of $\mu_v(w)$ represent the possibility that the robot changes its planned path. Alternatively, if the robots are performing a repetitive task, such as surveillance, historic path data will be available. Then $\mu_v(w)$ equals the fraction of times where a robot at v went to w next. The simplest model is $\mu_v(w) = \frac{1}{\deg(v)}$ for every neighbor, w, of v which assumes a robot is equally likely to move to any of its neighbors. While this model is often unrealistic, by simply relying on its observations and communication with other robots, it is possible to still have good performance.

The values of $\gamma_v(\Delta t)$ depend on the robot's speed and the decomposition of the environment. Each vertex corresponds to a cell of the environmental decomposition which has a defined geometry. We can use the cell diameter and robot speed to compute minimum, mean, and maximum times for the robot to traverse the cell. These values can be used to construct a normal or uniform distribution of times that a robot spends in one cell. If robots are performing specific tasks at each vertex, $\gamma_v(\Delta t)$ can be determined by the distribution of times that the task at v takes.

If the searcher does not know if the target is cooperative or neutral, (6) can be modified to account for this uncertainty. As cooperative and neutral targets behave differently, different semi-Markov models are needed for each behavior. Suppose a robot behaves according to $\mathbf{F}_{\mathrm{c}}$ when it is cooperative and $\mathbf{F}_{\mathrm{n}}$ when it is neutral. Then if a searcher doesn't know a target's behavior, it can use the update tensor, $c[t]\mathbf{F}_{\mathrm{c}} + (1 - c[t])\mathbf{F}_{\mathrm{n}}$ where $c[t] \in [0,1]$ is the searcher's belief that the target is cooperative. As robots are more likely to want to reconnect the longer they have been disconnected, $c[t]$ increases with time since the two robots were most recently connected. This approach is similar to the rendezvous-evasion approach of Alpern and Gal [18].

The semi-Markov model represents robot i's belief about how robot j moves; it does not need to reflect how robot j actually moves. As long as the model accurately describes ways that robot j could move, the resulting belief will still be useful in the sense that robot i always has a non-zero belief that robot j is in its actual position. For this reason, it is possible to use a simple semi-Markov model

based only on robot j's possible speeds and the topology of the environment. The belief vector will also be updated based on negative observations made by robot i which will significantly improve its belief despite little knowledge of how robot j decides where it will move.

C. Effects of observations

Two robots located at adjacent vertices in the communication graph $\mathcal{G}_c$ can communicate. Suppose that robot i cannot communicate with robot j at time t. Then robot j is not located at any of robot i's neighbor vertices in $\mathcal{G}_{\mathrm{c}}$. Therefore, all elements of $\widehat{\mathbf{Q}}_i \boldsymbol{e}_j$ corresponding to robot i's neighbors should be set to zero.

This update can be performed using $\mathbf{R}(v) \in \mathrm{Hom}(\mathbb{R}^{n_{\mathrm{a}}} \otimes \mathbb{R}^T)$, a $(2,2)$-tensor which sets elements corresponding to vertices visible to v equal to zero. The observation tensor, $\mathbf{R}(v)$ can be decomposed as $\boldsymbol{R}(v) \otimes \boldsymbol{I}_T$ where $\boldsymbol{R}(v) \in \mathrm{Hom}(\mathbb{R}^{n_{\mathrm{a}}})$ is diagonal and $\boldsymbol{I}_T \in \mathrm{Hom}(\mathbb{R}^T)$ is the identity matrix. The diagonal elements of $\boldsymbol{R}(v)$ are zero for neighbors of v and one otherwise. Using $\mathbf{R}(v)$, the update law is

$$\widehat{\mathbf{Q}}_j[t+1] = \eta \mathbf{R}(q_i[t+1]) \mathbf{F} \widehat{\mathbf{Q}}_j[t], \tag{7}$$

where η is a normalization term.

If multiple searchers can communicate, they can share information about where another robot is not presently located. Then they can update their probability tensors using (7) with $\mathbf{R}(v)$ replaced by the product of each connected robot's observation tensor.

D. Combining beliefs

When robots become connected, they can combine their belief of another robot's position into a shared, more accurate belief. The shared belief will then be based on both robots' most recent observations of the third robot and of vacant locations that they have seen since. To combine beliefs, robots will need to send their versions of $\widehat{\mathbf{Q}}$ across the network. A conservative method of combining multiple distributions into a more accurate one is to take the element-wise minimum of the distributions [17]. While this approach has some nice properties, simply using the minimum ignores information from all but one of the distributions.

Suppose robots i and k both have belief of q_j which we denote by $\mathbb{P}(q_j \mid \mathcal{I}_i)$ and $\mathbb{P}(q_j \mid \mathcal{I}_k)$. Using these beliefs, when the robots communicate, they would like to compute $\mathbb{P}(q_j \mid \mathcal{I}_i, \mathcal{I}_k)$. As the robots were not previously connected, they did not iteratively compute $\mathbb{P}(q_j \mid \mathcal{I}_i, \mathcal{I}_k)$ using (7) and must compute it using only their existing beliefs $\mathbb{P}(q_j, \mathcal{I}_i)$ and $\mathbb{P}(q_j, \mathcal{I}_k)$. Using Bayes' theorem, we can write the conditional probability as

$$\mathbb{P}(q_j \mid \mathcal{I}_i, \mathcal{I}_k) = \frac{\mathbb{P}(\mathcal{I}_k \mid q_j, \mathcal{I}_i)\mathbb{P}(q_j \mid \mathcal{I}_i)}{\mathbb{P}(\mathcal{I}_k \mid \mathcal{I}_i)} \tag{8}$$

$$= \frac{\mathbb{P}(\mathcal{I}_i \mid q_j, \mathcal{I}_k)\mathbb{P}(q_j \mid \mathcal{I}_k)}{\mathbb{P}(\mathcal{I}_i \mid \mathcal{I}_k)}. \tag{9}$$

Multiplying (8) by (9) and taking the square root yields

$$\mathbb{P}(q_j \mid \mathcal{I}_i, \mathcal{I}_k) = \eta_{j,i,k} \sqrt{\mathbb{P}(q_j \mid \mathcal{I}_i)\mathbb{P}(q_j \mid \mathcal{I}_k)}$$

where

$$\eta_{j,i,k} = \sqrt{\frac{\mathbb{P}(\mathcal{I}_k \mid q_j, \mathcal{I}_i)\mathbb{P}(\mathcal{I}_i \mid q_j, \mathcal{I}_k)}{\mathbb{P}(\mathcal{I}_k \mid \mathcal{I}_i)\mathbb{P}(\mathcal{I}_i \mid \mathcal{I}_k)}}.$$

The denominator of $\eta_{j,i,k}$ is a normalization term because it does not depend on q_j. The terms in the numerator of $\eta_{j,i,k}$ are the probabilities that one robot wouldn't have seen the target given the target's current position and the fact that the other robot has also not seen the target. We will assume that there is a similarly probability of not observing the target for any position in the environment and therefore $\eta_{j,i,k}$ is approximately constant.

Under this assumption, we will therefore use the element-wise geometric mean of two distributions as their fused distribution:

$$\mathrm{merge}\left(\widehat{\mathbf{Q}}_i, \widehat{\mathbf{Q}}_k\right) \propto \sqrt{\widehat{\mathbf{Q}}_i \circ \widehat{\mathbf{Q}}_k}$$

where $\circ$ denotes the element-wise product. This operation can be generalized to merging $M \geq 2$ distributions by multiplying them all together element-wise and then taking the M^{th} root. Similar to the element-wise minimum merge operation [17], this merge operation has the properties that $\mathrm{merge}(A, B)$ is 0 at a vertex if and only if A or B was 0 at that vertex and that $\mathrm{merge}(A, A) = A$. Furthermore, it uses information from all distributions which results in a more accurate merged distribution.

IV. OPTIMAL SEARCH TRAJECTORIES

When a robot decides it needs to find another robot, it can use its position belief tensor, $\widehat{\mathbf{Q}}$, to plan an optimal path for reconnection. This path is the one which maximizes the probability of connecting with a target over a prediction horizon, T_{p}. Let $\mathcal{P}(v_0, T_{\mathrm{p}})$ be the set of paths starting at v_0 with length T_{p}. The optimal path is

$$p^* = \underset{p \in \mathcal{P}(v_0, T_{\mathrm{p}})}{\arg\max} \left\{\mathbb{P}(c(0, T_{\mathrm{p}}) = 1 \mid p, \widehat{\mathbf{Q}}_i)\right\}$$

where $c : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ is an indicator with $c(a, b) = 1$ if the searcher and one of its targets are connected at some $t \in \{a, \ldots, b\}$ and 0 otherwise. Once the robot computes p^*, it implements the first move of p^* and then computes a new optimal path. As it is not always possible to find a path that is guaranteed to find a target during the time horizon, p^* may have $\mathbb{P}(c(0, T_{\mathrm{p}}) = 1 \mid p, \widehat{\mathbf{Q}}_i) < 1$.

A. One searcher and one target

Suppose there is one searcher and one target, robot j. $C_t(p)$ is the probability that the target and searcher will be connected at some point given the searcher's belief of the target's position and its planned path p. A related function, $C'_t(p) = 1 - C_t(p)$, is the probability that the searcher and target are never connected.

The searcher's belief of the target's state at $t = 0$ is $\widehat{\mathbf{q}}_0 = \widehat{\mathbf{Q}}_i \boldsymbol{e}_j$. Let $\mathbf{S}^* \in (\mathbb{R}^{n_{\mathrm{a}}} \otimes \mathbb{R}^T)$ be the $(0,2)$-tensor consisting entirely of ones. This tensor sums the elements of a $(2,0)$-tensor such as $\widehat{\mathbf{Q}}_i \boldsymbol{e}_j$. As $\widehat{\mathbf{q}}_0$ is a probability

distribution, $\mathbf{S}^*\widehat{\mathbf{q}}_0 = 1$. Moreover, at $t = 0$ the searcher and target are guaranteed to be disconnected so $C'_0(p) = 1$.

At $t = 1$, the searcher's belief of the target's new position is $\mathbf{F}\widehat{\mathbf{q}}_0$. Its $(v, \Delta t)^{\text{th}}$ element is the probability that the target will be in state $(v, \Delta t)$. Let $\mathbf{R}(v_1)$ be the observation tensor at v_1, the first vertex of p. If the target is visible from v_1, multiplication by $\mathbf{R}(v_1)$ will set the corresponding element of $\mathbf{F}\widehat{\mathbf{q}}_0$ to zero. Therefore, the $(v, \Delta t)^{\text{th}}$ element of $\widehat{\mathbf{q}}_1 = \mathbf{R}(v_1)\mathbf{F}\widehat{\mathbf{q}}_0$ is the probability that the target will be in state $(v, \Delta t)$ and the target and searcher will be disconnected. By summing over all states, the probability that the target and searcher have never been connected is

$$C'_1(p) = \mathbf{S}^*\widehat{\mathbf{q}}_1.$$

Continuing in this way, the elements of $\widehat{\mathbf{q}}_2 = \mathbf{R}(v_2)\mathbf{F}\widehat{\mathbf{q}}_1$ are the probability that the target is in a given state and has not been connected up to $t = 2$. Summing over all states again, we obtain $C'_2(p) = \mathbf{S}^*\widehat{\mathbf{q}}_2$. By induction, we can conclude that $C'_t(p) = \mathbf{S}^*\widehat{\mathbf{q}}_t$ where $\widehat{\mathbf{q}}_t = \mathbf{R}(v_t)\mathbf{F}\widehat{\mathbf{q}}_{t-1}$ and $\widehat{\mathbf{q}}_0 = \widehat{\mathbf{Q}}_i\boldsymbol{e}_j$. Rewriting this cost function, the cost of any path can be computed as

$$C_{T_\text{p}}(p) = 1 - \mathbf{S}\mathbf{F}_p\widehat{\mathbf{Q}}_i(0)\mathbf{e}_j \tag{10}$$

where $\mathbf{F}_p = \left[\mathbf{R}(v_{T_\text{p}})\mathbf{F}\right] \times \cdots \times \left[\mathbf{R}(v_2)\mathbf{F}\right]\left[\mathbf{R}(v_1)\mathbf{F}\right]$. Note that $\mathbf{F}_p$ is independent of $\widehat{\mathbf{Q}}_i$ and only depends on p.

B. One searcher and multiple targets

If one searcher has several targets in the set $\mathcal{R}_\text{t}$, the goal is to maximize the probability of connecting with at least one target. The probability of not connecting with a particular target $j \in \mathcal{R}_\text{t}$ is

$$C'_{T_\text{p},j}(p) = \mathbf{S}\mathbf{F}_p\widehat{\mathbf{Q}}_i(0)\mathbf{e}_j.$$

The probability of not connecting with any targets in j is $\prod_{j\in\mathcal{R}_\text{t}} C'_{T_\text{p},j}(p)$. The probability of connecting with at least one target is

$$C_{T_\text{p}}(p) = 1 - \prod_{j\in\mathcal{R}} \left(\mathbf{S}\mathbf{F}_p\right)\left(\widehat{\mathbf{Q}}_i(0)\boldsymbol{e}_j\right).$$

Solving this function does not require significantly more computation than (10). The most expensive step is the computation of $\mathbf{S}\mathbf{F}_p$ which involves multiplying many $(2,2)$-tensors together. As $\mathbf{S}\mathbf{F}_p$ only depends on the path, it only needs to be computed once. After computing $\mathbf{S}\mathbf{F}_p$, it can be multiplied by $\widehat{\mathbf{Q}}_i(0)\mathbf{e}_j$ for each target robot, which is faster as we are multiplying a $(0,2)$-tensor by a $(2,0)$-tensor. As these computations are cheaper than computing $\mathbf{S}\mathbf{F}_p$, the overall complexity does not increase.

C. Multiple searchers

If multiple robots are cooperatively searching for the same targets, the searchers stay connected and plan their paths to maximize the collective probability of finding their target. These searchers are searching for the tuple of paths, $\boldsymbol{p}^*$ which optimizes the probability that the team finds one of its targets subject to a connectivity constraint. This optimization can be performed by a single robot which plans all of the paths and communicates the optimal paths to the other robots or using a distributed optimization approach.

To account for observations from multiple robots, $\boldsymbol{F_p}$ must be defined with $\prod \mathbf{R}(v_{t,i})$, where $v_{t,i}$ is the t^{th} vertex of robot i's path, replacing the single $\mathbf{R}(v_t)$ that defines $\boldsymbol{F}_p$. The probability of finding a robot in the set of targets, $\mathcal{R}_\text{t}$, when the searchers execute a tuple of paths, $\boldsymbol{p}$, is

$$C_{T_\text{p}}(\boldsymbol{p}) = 1 - \prod_{j\in\mathcal{R}_\text{t}} \left(\mathbf{S}\mathbf{F}_{\boldsymbol{p}}\right)\left(\widehat{\mathbf{Q}}_\text{c}(0)\boldsymbol{e}_j\right). \tag{11}$$

The optimal tuple of paths maximizes C_{T_p} subject to the constraint that the searchers remain connected during the entire search. At any time, we can check for connectivity by computing

$$\Lambda_2(\boldsymbol{p}) = \prod_{i=1}^{T_\text{p}} \lambda_2(\boldsymbol{p}, t) \tag{12}$$

where $\lambda_2(\boldsymbol{p}, t)$ is the second smallest eigenvalue of the Laplacian of the ad-hoc network formed by the searchers at time t if they execute paths $\boldsymbol{p}$. To ensure connectivity at all times, we need $\Lambda_2(\boldsymbol{p}) > 0$.

Using (11) and (12), the optimal tuple of paths is:

$$\boldsymbol{p}^* = \underset{\boldsymbol{p}\in\mathcal{P}(\mathcal{R}_\text{s},T_\text{p})}{\arg\max} \left\{C_{T_p}(\boldsymbol{p}) \,\middle|\, \Lambda_2(\boldsymbol{p}) > 0\right\}.$$

Once we have found $\boldsymbol{p}^*$, each searcher should implement the first move of its path in $\boldsymbol{p}^*$.

A brute force approach to optimizing this cost function would involve finding paths for k_c robots which is $\mathcal{O}(T_p D^{k_c T_p})$ and then evaluating $\Lambda_2(\boldsymbol{p})$ which is $\mathcal{O}(k_c^3)$ and $C_{T_p}(\boldsymbol{p})$ which is $\mathcal{O}((n+2N)^2T^3)$. The overall algorithm is $\mathcal{O}(T_p D^{k_c T_p}(k_c^3+(n+2N)^2T^3))$ which is exponential in the number of connected robots and is generally not feasible for more than 2 connected robots.

The computation speed can be drastically improved using a branch and bound technique. The paths of $\boldsymbol{p}$ are generated iteratively and grow longer as new vertices are added to the shortest path. Once all robots have been assigned a vertex at a time step, the connectivity of the network at that time can be checked to see if that tuple of partial path is valid. If the network is not connected, all paths beginning with that partial path are not valid. By deleting the disconnected tuple of partial path, none of these tuples of paths will be considered. This approach bounds the number of paths as branching happens, drastically reducing the number of paths that $C_{T_p}(\boldsymbol{p})$ has to be calculated for. In general, the team of robots stays close together, resulting in a connected component which follows a single robot while maintaining one of a few relative configurations. With this modification, the number of connected paths for multiple robots is of the same order as the number of paths for a single robot so the algorithm is $\mathcal{O}(T_p D^{T_p}(k_c^3 + (n+2N)^2T^3))$.

V. COMPARISON WITH OTHER APPROACHES

The reconnection algorithm described in this paper can be used to reconnect robots who are performing a variety of tasks, such as search, coverage, surveillance, or delivery.

The approach is general in the sense that it can be used with different types of robot motion as long as a model is available. As this paper is not about a specific application, we have applied the reconnection algorithm to the simple problem of establishing a connected ad-hoc wireless network in a team of robots that are initially disconnected.

The overall team objective is to create a connected network containing all the robots as quickly as possible. A single robot's objective is to search for any disconnected robot. If a robot's connected component contains more than one robot, they plan their paths cooperatively with the constraint of maintaining connectivity in that connected component. By planning cooperatively, the robots in the connected component can spread out as much as the connectivity constraint allows to improve their probability of finding another connected component instead of following exactly the same path. As robots stay together once connected, all robots will eventually be connected achieving the overall objective.

In this example, all robots search for each other simultaneously and choose their next vertex based on their own belief of other robots' positions. The semi-Markov motion model represents the searcher's belief of where its target will move. As the searcher does not have access to its target's belief of other robots' positions, we use a semi-Markov model with broad distributions which is still useful for any belief vector. In this simple model, the searchers believe that their target robot has equal probability to move to any of its neighboring vertices and takes a time uniformly chosen between 50% and 150% of the edge's travel time which is proportional to its length. This model is valid in the sense that each robot always has a non-zero belief that any other robot is in its actual position. When combined with the effect of observations, this model results in acceptable performance despite its simplicity.

As this exact problem has not been considered in the literature, we compared our approach against two simple approaches:

1) Random: robots randomly choose to move to one of their neighbor vertices whenever they are able to move. Once two robots find each other, these choices are constrained to ensure connectivity.
2) Persistent: robots randomly choose between neighbor vertices which have not been visited recently which causes the robots to move quickly through the environment to new locations.

In each approach all robots are searching for each other simultaneously and implement identical algorithms. Each approach was evaluated using 50 simulations of 2, 3, 4, and 5 robots.

In each simulation, the environment had 150 vertices and was randomly generated (Figure 2). The robots started far apart with the first four robots placed in different corners of the environment and the fifth placed in the center.

The robots know all other robots' starting locations but move randomly for the first 50 time steps so that they do not have exact knowledge of each others' positions when they begin searching and must rely on their belief. After this initial random motion, all robots start searching and use a time horizon of 12 time steps to compute their paths. In all cases, communication is limited to nearby vertices with robots only able to communicate if they are at adjacent vertices.

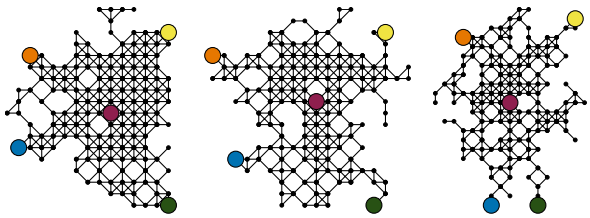

Fig. 2. Examples of randomly generated environments and robot starting locations.

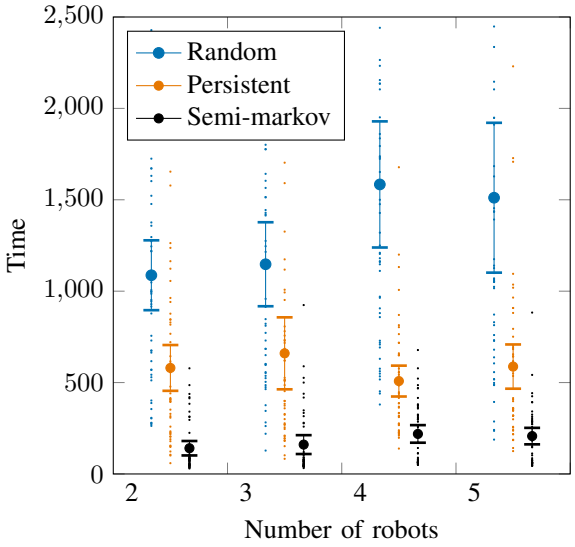

Fig. 3. Completion times and means for re-establishing communication using random (red), persistent (blue), and semi-Markov based (black) searches. Individual reconnection times are shown by small dots. Mean reconnection times are shown by large dots with 95% confidence intervals indicated using error bars.

To compare the results, we computed the average time taken for the entire network to be connected for each algorithm and number of robots (Figure 3). For each number of robots, the completion times are much faster for the semi-Markov based search than for either of the other two searches. Using a Welch's t-test, we reject the hypotheses that the mean reconnection times for different algorithms are equal at the 99% confidence level (Table I). Therefore, the semi-Markov search presented in this paper is more effective at reconnecting a team of mobile robots than the random or persistent searches.

The accompanying video shows animations of the fastest, median, and slowest trials for 5 robots re-establishing communication using this algorithm [23]. In these animations, translucent circles represent the probability distributions for the red robot's belief of other robots' positions. The size of the circle corresponds to the red robot's belief that the robot of the corresponding color is located at that vertex.

TABLE I

WELCH'S t-TEST RESULTS FOR THE HYPOTHESIS THAT THE MEAN COMPLETION TIME FOR SEMI-MARKOV BASED SEARCH IS EQUAL TO THE MEANS FOR THE RANDOM AND PERSISTENT SEARCHES.

p	$\mathbb{P}(\mu_{\mathrm{r}} = \mu_{\mathrm{s}})$	$\mathbb{P}(\mu_{\mathrm{p}} = \mu_{\mathrm{s}})$
2	1.63×10^{-13}	7.68×10^{-9}
3	1.84×10^{-11}	7.57×10^{-6}
4	2.06×10^{-10}	6.17×10^{-8}
5	5.65×10^{-8}	1.44×10^{-7}

VI. CONCLUSIONS

When robots cannot communicate over long ranges, a team of robots may need to split up into multiple smaller disconnected teams while completing their tasks. If the tasks take variable lengths of time, it can be difficult to plan a rendezvous time and place when they separate. Instead, they can simply search for each other when they have information to share and need to communicate.

In this paper, we presented an algorithm that disconnected robots can use to find each other without making an explicit plan for reconnection. Robots update their belief of the positions of disconnected robots using a semi-Markov model which incorporates variable task and transit times into a Markov model. When two robots encounter each other, they can update their beliefs of a third robot's state using a merging algorithm that is based on an element-wise geometric mean. Using the belief of a disconnected robot's position, small teams of robots can search for their target by solving a constrained optimization problem. The objective function is the probability that they will find their target and the constraints ensure the team of searchers remains connected during the search. A branch and bound technique is used to reduce the computational complexity of the optimization problem and ensure that the problem is tractable for teams of multiple robots.

We compared our approach with two other algorithms—a random search and a persistent search—for reconnecting teams of 2, 3, 4, and 5 robots. In these simulations the robots' only objective was to establish a connected ad-hoc network amongst all the robots. Our approach had better average reconnection times than the other two approaches at the 99% confidence level. It can therefore be used as a component of other multi-robot algorithms where the robots are not necessarily connected all the time.

REFERENCES

[1] Y. Mostofi, M. Malmirchegini, and A. Ghaffarkhah, "Estimation of communication signal strength in robotic networks," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 1946–1951.

[2] E. Stump, A. Jadbabaie, and V. Kumar, "Connectivity management in mobile robot teams," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2008, pp. 1525–1530.

[3] L. Sabattini, C. Secchi, N. Chopra, and A. Gasparri, "Distributed control of multirobot systems with global connectivity maintenance," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1326–1332, 2013.

[4] M. C. De Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *45th Conference on Decision and Control (CDC)*. IEEE, 2006, pp. 3628–3633.

[5] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, "Efficient boustrophedon multi-robot coverage: An algorithmic approach," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2, pp. 109–142, 2008.

[6] Y. Pei, M. W. Mutka, and N. Xi, "Coordinated multi-robot real-time exploration with connectivity and bandwidth awareness," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 5460–5465.

[7] G. A. Hollinger and S. Singh, "Multirobot coordination with periodic connectivity: Theory and experiments," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 967–973, 2012.

[8] Y. Kantaros and M. M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3109–3121, 2017.

[9] E. J. Anderson and R. Weber, "The rendezvous problem on discrete locations," *Journal of Applied Probability*, vol. 27, no. 04, pp. 839–851, 1990.

[10] A. Dessmark, P. Fraigniaud, and A. Pelc, "Deterministic rendezvous in graphs," in *European Symposium on Algorithms*. Springer, 2003, pp. 184–195.

[11] J. Chalopin, S. Das, and P. Widmayer, "Deterministic symmetric rendezvous in arbitrary graphs: Overcoming anonymity, failures and uncertainty," in *Search Theory*. Springer, 2013, pp. 175–195.

[12] G. Dudek and N. Roy, "Multi-robot rendezvous in unknown environments, or, what to do when you're lost at the zoo," in *National Conference Workshop on Online Search*. AAAI, 1997.

[13] H. Lau, S. Huang, and G. Dissanayake, "Optimal search for multiple targets in a built environment," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2005, pp. 3740–3745.

[14] A. Jotshi and R. Batta, "Search for an immobile entity on a network," *European Journal of Operational Research*, vol. 191, no. 2, pp. 347–359, 2008.

[15] H. Lau, S. Huang, and G. Dissanayake, "Probabilistic search for a moving target in an indoor environment," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2006, pp. 3393–3398.

[16] G. A. Hollinger, S. Singh, J. Djugash, and A. Kehagias, "Efficient multi-robot search for a moving target," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 201–219, 2009.

[17] G. A. Hollinger, S. Yerramalli, S. Singh, U. Mitra, and G. S. Sukhatme, "Distributed data fusion for multirobot search," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 55–66, 2015.

[18] S. Alpern and S. Gal, "Searching for an agent who may or may not want to be found," *Operations Research*, vol. 50, no. 2, pp. 311–323, 2002.

[19] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 113–126, 2001.

[20] Y. Mostofi, A. Gonzalez-Ruiz, A. Gaffarkhah, and D. Li, "Characterization and modeling of wireless channels for networked robotic and control systems-a comprehensive overview," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 4849–4854.

[21] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[22] T. Nishizeki and N. Chiba, *Planar graphs: Theory and algorithms*. Elsevier, 1988, vol. 32.

[23] I. Vandermeulen, "Establishing connection between disconnected robots using a semi-markov model," Jul 2018, doi:10.6084/m9.figshare.6854981.