Global Planning for Multi-Robot Communication Networks in Complex Environments

Yiannis Kantaros, *Student Member, IEEE,* and Michael M. Zavlanos, *Member, IEEE*

***Abstract*—In this paper, we consider networks of mobile robots responsible for servicing a collection of tasks in complex environments, while ensuring end-to-end connectivity with a fixed infrastructure of access points. Tasks are associated with specific locations in the environment, they are announced sequentially, and they are not assigned *a priori* to any robots. Information generated at the tasks is propagated to the access points via a multi-hop communication network. We propose a distributed, hybrid control scheme that dynamically grows tree networks, rooted at the access points, with branches that connect robots that service individual tasks to the main network structure. To achieve this goal, the robots switch between different roles related to their functionality in the network. The switching process is tightly integrated with distributed optimization of the communication variables and motion planning in complex environments, giving rise to the proposed distributed hybrid system. Our proposed scheme results in an efficient use of the available robots and also allows for global planning by construction, a task that is particularly challenging in complex environments.**

***Index Terms*—Multi-robot networks, communication networks, distributed control, distributed optimization, global motion planning.**

I. Introduction

MOBILE robot networks have received considerable attention in the recent years due to the effect they can have in efficiently accomplishing a number of tasks involving area coverage [1], environmental monitoring [2], and search and rescue missions [3]. In principle, these tasks are difficult to carry out using a single robot and, therefore, properly organized robot teams are needed for this purpose. Successful accomplishment of such complex tasks requires the existence of valid communication paths for information and coordination among the robots during the network deployment.

Recent methods for communication control of mobile robot networks typically rely on proximity graphs to model the exchange of information among the robots and, therefore, the communication problem becomes equivalent to preserving graph connectivity. Methods to control graph connectivity typically rely on controlling the Fiedler value of the underlying graph. One possible way of doing so is maximizing the Fiedler value in a centralized [4] or distributed [5] fashion. Alternatively, potential fields that model loss of connectivity as an obstacle in the free space can be employed for this purpose, as shown in [6]. A distributed hybrid approach to connectivity control is presented in [7] whereby communication links are efficiently manipulated using an approach that decouples the continuous robot motion from the control of the discrete graph. Further distributed algorithms for graph connectivity maintenance have been implemented in [8], [9]. A comprehensive survey of this literature can be found in [10].

A more realistic communication model for mobile networks compared to the above graph-theoretic models is proposed in [11], [12] that takes into account the routing of packets as well as desired bounds on the transmitted rates. In this model, a weighted graph is employed to capture the inter-robot communication with weights that are associated with the packet error probability. A conceptually similar communication model is proposed in [13] that models the communication rates among robots as random variables, while the routing of information is performed so that the uncertainty in the link rates is reduced. A communication model that accounts for multi-path fading of channels is proposed in [14], where robot mobility is exploited in order to increase the throughput. Multi-path fading, shadow fading, and path loss have also been used to model channels in [15], [16]. In these works, a probabilistic framework for channel prediction is developed based on a small number of measurements of the received signal power. The integration of the latter communication models with robot mobility is described in [17]. A related approach pertaining to the online evaluation of wireless channels is presented in [18] based on a sampling scheme for the link capacities.

In this paper, we assume a mobile robotic network residing in a complex environment responsible for servicing a collection of tasks with the additional requirement of ensuring reliable transmission of information to a fixed infrastructure of access points (APs). Tasks are associated with specific locations in the environment, which are announced sequentially and are not assigned *a priori* to robots. Servicing a task means that a robot is physically located in the vicinity of that task. To model the exchange of information among the robots, we employ the communication model presented in [11], [12], where information is propagated to the APs through a multi-hop network whose links model the probability that packets are correctly decoded at their intended destinations.

To address this problem, we propose a hybrid, distributed control scheme that achieves global planning in complex environments. Our approach relies on dynamically growing tree networks that connect leaf nodes/robots responsible for servicing targets to the main network structure. Specifically, when a new target is announced, a new branch is formed in the existing network that is rooted at a branch junction. As this new branch is grown the robots adopt roles associated with specific locations that they need to visit/track in the workspace and switch between these roles to facilitate the design of a network that can best service the new assigned task. The above

This work is supported by NSF under grants CNS #1261828 and CNS #1302284.

Yiannis Kantaros and Michael M. Zavlanos are with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708, USA. {yiannis.kantaros,michael.zavlanos}@duke.edu

coordination scheme for network design is integrated with robot mobility and control of the communication variables to allow the robots to move towards their assigned tasks while ensuring reliable communication with a fixed infrastructure of access points. Particularly, the communication variables are updated periodically via a distributed subgradient algorithm in the dual domain. In between communication updates, the robots move to accomplish their tasks dictated by their roles in the network. Motion planning takes place along obstacle-free geodesic paths and depends on the solution of distributed sequential convex programs that can handle the non-linear coupling of the robots' positions on the communication constraints. The distributed optimization of the communication variables and motion control of the robots are tightly integrated with the dynamical process that determines the roles of the robots in the network giving rise to the proposed distributed hybrid algorithm.

A. Contribution

Related problems that address motion control of multi-robot systems are presented in [19]–[21] assuming obstacle-free environments. Complex environments are considered in [22], [23], where the task assignment problem aims at generating collision-free trajectories that connect every robot to its assigned task. Navigation functions for driving a mobile network to a desired configuration while avoiding collisions with the environment have also been employed [24]. Common in these works is that maintenance of reliable communication among the robots during the network evolution is typically ignored. To the best of our knowledge, the most relevant literature to the work presented in this paper is [25], [26], where communication-aware deployment algorithms are developed for mobile networks residing in complex environments. Specifically, in [25], [26], to attain global planning, a rapidly exploring random tree (RRT) algorithm [27] is employed that requires every robot to be *a priori* aware of a general region where it should lie in the final network configuration. In other words, an estimate of the final network configuration is necessary in this method. To the contrary, our proposed algorithm is more flexible as the robots can determine autonomously and in a distributed way a network configuration that accomplishes a desired task; a subset of leader robots, determined dynamically and autonomously, pursue specific locations in space associated with the assigned task, while all other robots move to facilitate the leaders' motion. Additionally, our method allows for tasks to be announced and serviced dynamically and accounts for an efficient use of the available robots through constructing tree network structures and appropriately selecting branch junctions, avoiding cycles and redundant nodes.

The rest of this paper is organized as follows. In section II, we define the problem under consideration. Section III presents the distributed algorithm for the update of the communication variables, which is integrated with the robots' mobility in Section IV. Section V describes the proposed distributed coordination scheme that allows robots to switch between different roles and is critical in achieving global planning. Simulation studies demonstrating the efficiency of our control scheme are provided in Section VI and concluding remarks are presented in the last section.

II. Problem Formulation

Consider N mobile robots with wireless communication capabilities and an infrastructure of K access points[1] (APs) that are fixed in number and remain stationary for all time. Robots and access points are located in a complex polygonal environment denoted by $\mathcal{W} \subset \mathbb{R}^2$. The positions of all nodes are stacked in the vector $\mathbf{x} = [\mathbf{x}_1^T, \ldots, \mathbf{x}_i^T, \ldots, \mathbf{x}_{N+K}^T]^T$, where it holds that $i \in \{1, \ldots, N\}$ for the robots and $i \in \{N+1, \ldots, N+K\}$ for the APs. Furthermore, the robots are assumed to evolve in $\mathcal{W}$ according to the following first-order differential equation

$$\dot{\mathbf{x}}_i(t) = \mathbf{u}_i(t), \ \forall i \in \{1, \ldots, N\} \tag{1}$$

where $\mathbf{u}_i(t) \in \mathbb{R}^2$ stands for the i-th control input.

The robots are responsible for servicing a collection of tasks associated with specific locations $\mathbf{q}_v \in \mathcal{W}$, $v = 1, \ldots, m$ in the environment. We assume that the tasks are announced sequentially[2] but they are not assigned *a priori* to robots and we make no assumptions on the spacial distribution of the targets in the workspace. Every new task is announced when all previous tasks are being serviced, and it is assigned for service to a unique robot called the leader of the network through an on-line distributed process. Given an announced task located at $\mathbf{q}_v$, servicing that task means that there is a future time instant t_v so that for all time $t > t_v$ there exists a robot i that is always physically in the vicinity of that task, i.e.,

$$\|\mathbf{x}_i(t) - \mathbf{q}_v\| \leq \delta, \ \forall t \geq t_v \tag{2}$$

where $\|\cdot\|$ represents the Euclidean norm, $\delta > 0$ is an arbitrarily small positive constant, and t refers to the current time.

Along with servicing tasks, the robots also need to ensure reliable communication with the APs and, for this purpose, we employ the routing model presented in [11]. According to this model, the communication channel between the i-th robot and the j-th node (robot or AP) is captured by a link reliability metric $R_{ij}(t) \triangleq R(\mathbf{x}_i(t), \mathbf{x}_j(t))$ denoting the probability that a packet transmitted by the robot at position $\mathbf{x}_i$ is correctly decoded by the node located at $\mathbf{x}_j$.[3] The effective transmission rate from i to j is equal to $R_0 R_{ij}(t)$, where R_0 is the transmission rate of the robots' radios. Additionally, let $r_i^{\min} \in [0, 1]$ denote the normalized average rate at which

[1] Access points are information sinks responsible for collecting and processing information generated individually by mobile robots.

[2] In practice, tasks can be announced at any rate and among the announced tasks, the task to be serviced can be chosen based on various criteria, e.g., proximity-based or priority based criterion.

[3] The link reliability $R(\mathbf{x}_i, \mathbf{x}_j)$ depends on path loss that is a function of the distance between the source and the receiver, shadowing due to the existence of obstacles in the propagation path, and multi-path fading due to reflections, which are difficult to predetermine. A common model for $R(\mathbf{x}_i, \mathbf{x}_j)$ is to define it by a decreasing function of the distance between nodes i and j. This model is often used in practice to address situations that are dominated by path loss, see, e.g., [25].

information is generated by the i-th robot in packets per unit time, which can be routed to the set of APs either through a multi-hop path or directly if $R_{ij}(t) > 0$ for some $j \in \{N+1, \ldots, N+K\}$. Routing of information is modeled using routing variables $T_{ij}(t) \in [0,1]$ denoting the fraction of time that nodes i and j communicate. Assuming that routing of a packet and its successful decoding by another node are two independent processes, we obtain that the normalized rate at which information is sent from i to j is $T_{ij}(t)R_0R_{ij}(t)$.

The proposed communication model assumes that each robot stores packets of information in a queue until they are transmitted and successfully decoded by their intended destinations. The average rate at which information leaves the i-th queue is

$$r^{out}_{\mathbf{x},i}(t) = \sum_{j=1}^{N+K} T_{ij}(t)R_0R(\mathbf{x}_i(t), \mathbf{x}_j(t)). \tag{3}$$

Similarly, the average rate at which packets arrive at the i-th queue is

$$r^{in}_{\mathbf{x},i}(t) = r^{\min}_i(t) + \sum_{j=1}^{N} T_{ji}(t)R_0R(\mathbf{x}_j(t), \mathbf{x}_i(t)). \tag{4}$$

Note that the APs can only receive information, which explains the upper limits in the sums of (3) and (4). In order to guarantee reliable communication with the set of APs, the i-th queue should empty infinitely often with probability one, i.e.,

$$c_i(\mathbf{x}(t), \mathbf{T}) \triangleq r^{out}_{\mathbf{x},i}(t) - r^{in}_{\mathbf{x},i}(t) \geq 0, \tag{5}$$

for all robots $i \in \{1, \ldots, N\}$ and for all time $t \geq 0$, where $\mathbf{T} \in \mathbb{R}^{N(N+K)}$ is a vector that stacks the routing decisions T_{ij} of all robots. In what follows, for simplicity of notation and without loss of generality, we assume that $R_0 = 1$ for all robots.

Based on the above formulation, the problem that we address in this paper can be stated as follows:

Problem 1: Given $m < N$ tasks announced sequentially at locations $\mathbf{q}_v \in \mathcal{W}$, $v \in \{1, \ldots, m\}$, determine robot trajectories $\mathbf{x}_i(t)$ and routing variables $T_{ij}(t)$ so that all tasks are eventually serviced as defined in (2), the communication constraints (5) are satisfied for all robots and all time t, and collisions between robots and between robots and the environment $\mathcal{W}$ are avoided during deployment of the network.

To solve Problem 1 we develop a distributed, hybrid control scheme that can dynamically grow tree networks rooted at the APs that connect leaf nodes, i.e., robots that service a task, to the main network structure. When a task is being serviced and a new one is announced, a new branch junction is determined from where an additional branch is grown in a distributed way connecting a new leaf node to the network. To achieve this goal, two leader teams are formed, namely, *the primary leader team* and *the secondary leader team*. The primary leader team is assembled when a new task is announced and all the previously announced tasks are being serviced, and its goal is to help the leader reach its destination. If the primary leader team gets trapped at a local stationary point, then a secondary leader team is assembled with the goal to assist the primary leader team to "escape" from that stationary point. To this end, robots in either leader team adopt roles that drive them to specific locations in space, hereafter denoted by $\boldsymbol{\psi}_i(t) \in \mathcal{W}$ defined in Section V. Navigation of robots towards $\boldsymbol{\psi}_i(t)$ is based on the solution of a constrained convex optimization program constructed in Section IV. The possible roles that the robots of the primary leader team can have are either a leader or a node, i.e., a robot that belongs to a branch of the tree network. On the other hand, the robots that belong to the secondary leader team can be leaf nodes, i.e., previously elected leaders that are currently servicing past tasks, junctions nodes located at branch junctions, simple nodes as in the primary leader team, or recruits, i.e., previously redundant nodes that are recruited to help the primary leader team escape from a local stationary point. The robots switch between these roles depending on the current stage of the task updating at the same time the targets $\boldsymbol{\psi}_i(t)$. A schematic that illustrates the sought network behavior is shown in Figure 1. The integration of the role assignment process along with the optimization of the routing variables and motion planning gives rise to the proposed distributed hybrid control scheme.

Assumption 1: Throughout the rest of the paper we assume that there is a sufficient number of redundant robots, i.e., robots without an assigned task, that can be recruited to facilitate the leader in servicing its task.

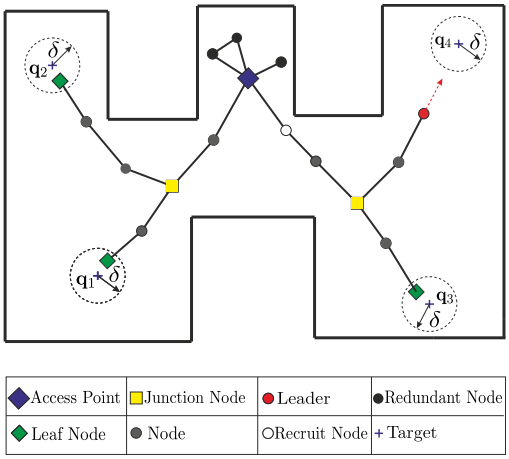


Fig. 1. An illustration of our problem formulation and proposed solution. A tree network that is rooted at the AP is constructed that connects leaf nodes servicing tasks to the main network structure. At the same time, a new branch rooted at a branch junction is growing so that the leader of the network can reach an announced target.

III. Distributed Communication

In this section, we develop a distributed algorithm to compute the routing variables T_{ij} that satisfy the communication constraints (5). Initially, we assume a given static network configuration $\mathbf{x}$, and in Section IV we extend this framework to account for node mobility. Note that for a given spatial configuration $\mathbf{x}$, there may be various routing decisions T_{ij} satisfying (5). To ensure uniqueness of those routing decisions T_{ij} we introduce a strictly convex objective function $V_{ij}(T_{ij})$ associated with the variables T_{ij}. In particular, we solve the

following constrained optimization problem for the computation of the routing variables

$$\begin{aligned} \underset{\mathbf{T}}{\text{minimize}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N+K} V_{ij}(T_{ij}) \\ \text{subject to} \quad & c_i(\mathbf{x},\mathbf{T}) \geq 0, \\ & \sum_{j=1}^{N+K} T_{ij} \leq 1, \quad 0 \leq T_{ij} \leq 1, \ \ \forall j, \end{aligned} \tag{6}$$

which for fixed robot positions $\mathbf{x}$ obtains a simple convex form. In (6), the constraints hold for all robots i. We also introduce the constraint $\sum_{j=1}^{N+K} T_{ij} \leq 1$ to ensure that the sum of time shares at node i does not exceed 1. Finally, for the strictly convex function $V_{ij}(T_{ij})$, we select $V_{ij} = w_{ij}T_{ij}^2$, $w_{ij} > 0$, in order to encourage the distribution of the packets over different links increasing in this way the robustness to link failures [11].

Solving (6) in a centralized way can incur large communication cost and delays due to the need for identifying the network topology and communicating it to the robots. Therefore, a distributed solution is preferred, where (6) is solved locally across the group of nodes. For this purpose, following the steps in [11], we define the Lagrangian of (6) by

$$\begin{aligned} \mathcal{L}_{\mathbf{x}}(\boldsymbol{\lambda},\mathbf{T}) &= \sum_{i=1}^{N}\sum_{j=1}^{N+K} V_{ij}(T_{ij}) \\ &+ \sum_{i=1}^{N}\boldsymbol{\lambda}_i\Big[\sum_{j=1}^{N+K} T_{ij}R(\mathbf{x}_i,\mathbf{x}_j) - \sum_{j=1}^{N} T_{ji}R(\mathbf{x}_j,\mathbf{x}_i) - r_i\Big], \end{aligned} \tag{7}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^N$ is a column vector of the Lagrange multipliers. Then, the dual function is

$$g_{\mathbf{x}}(\boldsymbol{\lambda}) = \min_{\sum_{j=1}^{N+K} T_{ij} \leq 1, \ \forall i \in 1,\ldots,N} \mathcal{L}_{\mathbf{x}}(\boldsymbol{\lambda},\mathbf{T}) \tag{8}$$

and the dual problem becomes

$$D_{\mathbf{x}} = \max_{\boldsymbol{\lambda}\geq 0} g_{\mathbf{x}}(\boldsymbol{\lambda}).$$

Since the optimization problem (6) is convex for fixed robot positions, it holds that $P_{\mathbf{x}} = D_{\mathbf{x}}$, where $P_{\mathbf{x}}$ is the solution of (6) for a network configuration $\mathbf{x}$. Therefore, we can equivalently work in the dual domain.

To implement a gradient ascent algorithm in the dual domain, we need to compute the gradient of the dual function (8). For this, define first the primal Lagrangian maximizers by

$$\{T_{\mathbf{x},ij}(\boldsymbol{\lambda})\}_{\forall i,j} = \underset{\sum_{j=1}^{N+K} T_{ij} \leq 1}{\operatorname{argmin}} \mathcal{L}_{\mathbf{x}}(\boldsymbol{\lambda},\mathbf{T}). \tag{9}$$

Then, the i-th component of the gradient of the dual function is given by

$$\begin{aligned} [\nabla g_{\mathbf{x}}(\boldsymbol{\lambda})]_i = &\sum_{j=1}^{N+K} T_{\mathbf{x},ij}R(\mathbf{x}_i,\mathbf{x}_j) \\ &- \sum_{j=1}^{N} T_{\mathbf{x},ji}R(\mathbf{x}_j,\mathbf{x}_i) - r_i. \end{aligned} \tag{10}$$

Note that the Lagrangian defined in (7) can be expressed as a sum of local Lagrangians $\mathcal{L}_{\mathbf{x},i}$ through reordering of its terms, i.e.,

$$\mathcal{L}_{\mathbf{x}}(\boldsymbol{\lambda},\mathbf{T}) = \sum\nolimits_{i=1}^{N} \mathcal{L}_{\mathbf{x},i}(\boldsymbol{\lambda},\mathbf{T}),$$

where

$$\begin{aligned} \mathcal{L}_{\mathbf{x},i}(\boldsymbol{\lambda},\mathbf{T}) = &-\lambda_i r_i \\ &+ \sum_{j=1}^{N}\left[V_{ij}(T_{ij}) + T_{ij}R(\mathbf{x}_i,\mathbf{x}_j)(\lambda_i - \lambda_j)\right] \\ &+ \sum_{j=N+1}^{N+K}\left[V_{ij}(T_{ij}) + \lambda_i T_{ij}R(\mathbf{x}_i,\mathbf{x}_j)\right]. \end{aligned} \tag{11}$$

Since the variables $\{T_{ij}\}_{j=1}^{N}$ appear only in $\mathcal{L}_{\mathbf{x},i}$, instead of minimizing the global Lagrangian, we can equivalently compute the minimizers of the local Lagrangians defined in (11), i.e.,

$$\{T_{\mathbf{x},ij}(\boldsymbol{\lambda})\}_{j=1}^{N+K} = \underset{\sum_{j=1}^{N+K} T_{ij} \leq 1}{\operatorname{argmin}} \mathcal{L}_{\mathbf{x},i}(\boldsymbol{\lambda},\mathbf{T}). \tag{12}$$

Finally, introducing an iteration index k and denoting by t_k the time instants at which the routing variables are updated, we obtain the following distributed gradient ascent algorithm in the dual domain:

Primal Iteration: For a given spatial configuration $\mathbf{x}(t_k)$ and Lagrange multiplies $\boldsymbol{\lambda}(t_k)$, compute the Lagrangian maximizers $\{T_{\mathbf{x}(t_k),ij}\}_{j=1}^{N+K}$ as:

$$\{T_{\mathbf{x},ij}(t_k)\}_{j=1}^{N+K} = \underset{\sum_{j=1}^{N+K} T_{ij} \leq 1}{\operatorname{argmin}} \mathcal{L}_{\mathbf{x}(t_k),i}(\boldsymbol{\lambda}(t_k),\mathbf{T}). \tag{13}$$

Dual Iteration: Given the primal variables $\{T_{\mathbf{x},ij}(t_k)\}_{j=1}^{N+K}$ from (13), update the dual variables as:

$$\begin{aligned} \lambda_i(t_{k+1}) = \mathbb{P}\Big[\lambda_i(t_k) + \epsilon\Big(&\sum_{j=1}^{N+K} T_{ij}(t_k)R(\mathbf{x}_i(t_k),\mathbf{x}_j(t_k)) \\ &- \sum_{j=1}^{N} T_{ji}(t_k)R(\mathbf{x}_j(t_k),\mathbf{x}_i(t_k)) - r_i\Big)\Big], \end{aligned} \tag{14}$$

where $\mathbb{P}$ denotes the projection to the non-negative orthant.

Note, that the algorithm in (13)-(14) is distributed, since it requires only the Lagrange multipliers λ_j [cf. (13)] and the routing variables T_{ji} [cf. (14)] from robots for which $R_{ij} \neq 0$.

Remark 3.1 (Primal-Dual Decomposition): In the above analysis, the dual subgradient method [28] was implemented in order to compute the optimal routing decisions T_{ij} for a given network configuration $\mathbf{x}$. More sophisticated primal-dual decomposition algorithms, e.g., the *Alternating Direction Method of Multipliers* (ADMM) [29], or the *Accelerated Distributed Augmented Lagrangian* (ADAL) [30], can also be used in lieu of the existing one, which enjoy faster convergence rates.

IV. Robot Navigation

As discussed in Section II, we propose a method to solve Problem 1 where the robots adopt specific roles associated with visiting or tracking a sequence of possibly temporary targets located at $\boldsymbol{\psi}_i(t_k) \in \mathcal{W}$ and determined in a way that allows the network to accomplish its assigned task. The selection of this sequence of targets $\boldsymbol{\psi}_i(t_k)$ depends on coordination between the robots and is discussed in Section V. In this Section, assuming that such a sequence of targets $\boldsymbol{\psi}_i(t_k)$ is available, we discuss how to design robot trajectories so that the communication constraints (5) are satisfied and collisions with the environment and between robots are avoided while the robots track their assigned targets $\boldsymbol{\psi}_i(t_k)$.

To jointly address robot communication and mobility, we propose a distributed control scheme that decouples these two objectives and alternates between the optimization of the two. In particular, assuming fixed robot positions $\mathbf{x}(t_k)$, every robot updates its routing variables at the time instants t_k via the distributed primal-dual algorithm (13)-(14). Then, using the routing variables $\mathbf{T}(t_k)$ obtained from the previous step, every robot moves during the time interval (t_k, t_{k+1}) towards a position $\mathbf{x}_i(t_{k+1})$ that minimizes the distance from its respective target $\boldsymbol{\psi}_i(t_k)$. Note that the update (13)-(14) ensures feasibility of the primal variables for a static network as $k \to \infty$. However, for an arbitrary finite iteration index k, the primal variables $\{T_{ij}(t_k)\}_{j=1}^{N+K}$ computed via (13)-(14) are not necessarily feasible. This situation is more pronounced in the case of mobile networks, where due to mobility the optimal solution of (6) drifts, and the primal-dual iteration (13)-(14) tries to catch up. As a result, the communication constraints $c_i(\mathbf{x}(t), \mathbf{T}) \geq 0$ may become violated as the robots move from $\mathbf{x}_i(t_k)$ to $\mathbf{x}_i(t_{k+1})$. To minimize constraint violations and ensure that an acceptable quality of communication is maintained, we require that every robot checks feasibility of its local routing variables after every communication update. [4] Robots with infeasible routing variables remain stationary until feasible routes are acquired through the iteration (13)-(14). When this happens, those robots compute the next positions $\mathbf{x}_i(t_{k+1})$ in a direction that minimizes the distance from their respective targets $\boldsymbol{\psi}_i(t_k)$ and then start moving towards these positions. In what follows, we discuss how the robots compute their next positions $\mathbf{x}_i(t_{k+1})$ so that collision avoidance between them and with the workspace boundary is guaranteed, respecting at the same time the communication constraints (5).

A. Obstacle Avoidance

To avoid collisions with the boundary of the workspace $\mathcal{W}$, denoted by $\partial\mathcal{W}$, we need to exclude this polygonal boundary from the free-space in which the robots are allowed to move. Specifically, we define an obstacle region $\mathcal{W}_o \subset \mathcal{W}$ where collisions with the workspace boundary can occur by the set $\mathcal{W}_o = \{\mathbf{q} \in \mathcal{W} \mid \|\mathbf{q} - \mathbf{q}_b\| \leq \rho,\ \mathbf{q}_b \in \partial\mathcal{W}\}$, containing all points $\mathbf{q} \in \mathcal{W}$ whose distance from the boundary $\partial\mathcal{W}$ is less than a small positive constant $\rho > 0$. Then, the free space

[4] In practice, we only require that the constraints (5) satisfy $c_i(\mathbf{x}, \mathbf{T}) > -c_i^{\min}$ for some small positive constant $c_i^{\min} > 0$.

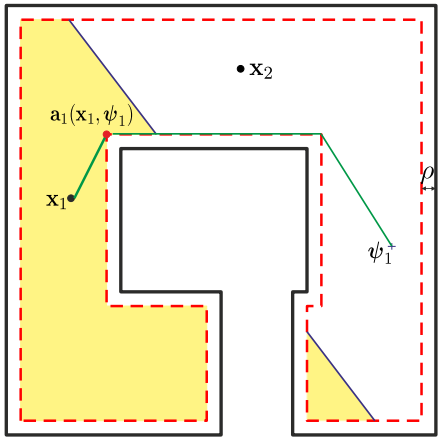


Fig. 2. Graphical example of the Voronoi partitioning of the free space $\mathcal{W}_f$ generated by two nodes. The boundary of the free-space $\mathcal{W}_f \subset \mathcal{W}$ is represented by the red dashed line. The green line stands for the geodesic path $s(\mathbf{x}_1(t_k), \boldsymbol{\psi}_1(t_k))$ computed on $\mathcal{W}_f$.

$\mathcal{W}_f$ is defined as $\mathcal{W}_f = \mathcal{W} \backslash \mathcal{W}_o$; see Figure 2. We defer the detailed description of the construction of the free-space to Appendix A. To enforce the constraint $\mathbf{x}_i(t_{k+1}) \in \mathcal{W}_f$, robot navigation is performed along *geodesic paths* computed over the space $\mathcal{W}_f$ defined as follows:

Definition 4.1 (Geodesic Path): The *geodesic path* $s(\mathbf{x}_i, \boldsymbol{\psi}_i)$ between two points $\mathbf{x}_i$ and $\boldsymbol{\psi}_i$ residing in a polygonal environment $\mathcal{W}_f$ can be uniquely defined as the shortest path between them entirely contained in $\mathcal{W}_f$, i.e.,

$$\begin{aligned} s(\mathbf{x}_i, \boldsymbol{\psi}_i) = \{&[\mathbf{x}_i, \mathbf{a}_1(\mathbf{x}_i, \boldsymbol{\psi}_i)], [\mathbf{a}_1(\mathbf{x}_i, \boldsymbol{\psi}_i), \mathbf{a}_2(\mathbf{x}_i, \boldsymbol{\psi}_i)], \dots, \\ &[\mathbf{a}_{m-1}(\mathbf{x}_i, \boldsymbol{\psi}_i), \mathbf{a}_m(\mathbf{x}_i, \boldsymbol{\psi}_i)], [\mathbf{a}_m(\mathbf{x}_i, \boldsymbol{\psi}_i), \boldsymbol{\psi}_i]\}, \end{aligned} \tag{15}$$

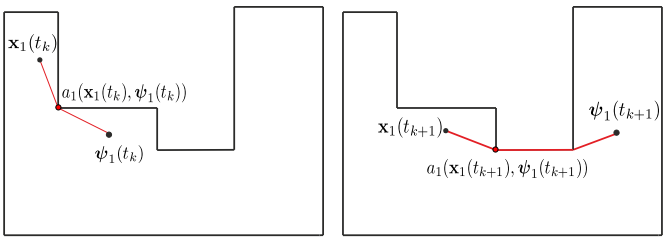
where $[\mathbf{a}_{l-1}(\mathbf{x}_i, \boldsymbol{\psi}_i), \mathbf{a}_l(\mathbf{x}_i, \boldsymbol{\psi}_i)]$ stands for the line segment that connects the reflex vertices $\mathbf{a}_{l-1}(\mathbf{x}_i, \boldsymbol{\psi}_i)$ and $\mathbf{a}_l(\mathbf{x}_i, \boldsymbol{\psi}_i)$ of the polygonal boundary of $\mathcal{W}_f$.

As the robots switch targets $\boldsymbol{\psi}_i(t_k)$ at time instants t_k, the geodesic paths need to be updated at every time instant t_k. Therefore, to reach the targets $\boldsymbol{\psi}_i(t_k)$, the robots need to track the reflex vertices $\mathbf{a}_1(\mathbf{x}_i(t_k), \boldsymbol{\psi}_i(t_k))$ as defined in Definition (4.1); see Figure 3. This gives rise to the optimization problem for the new position $\mathbf{x}_i(t_{k+1})$

$$\mathbf{x}_i(t_{k+1}) = \underset{\mathbf{x}_i \in \mathcal{W}_f}{\operatorname{argmin}} \|\mathbf{x}_i - \mathbf{a}_1(\mathbf{x}_i(t_k), \boldsymbol{\psi}_i(t_k))\|^2 . \tag{16}$$

B. Collision Avoidance

In what follows, we extend the solution of (16) to also account for collision avoidance between neighboring robots. For this, we decompose the free space $\mathcal{W}_f$ into disjoint cells, so that each cell is assigned to a unique robot. Requiring the robots always move in their assigned cells ensures collision avoidance between them. By dynamically updating those cells in a distributed way, we can guarantee that the robots are able to eventually reach their targets $\boldsymbol{\psi}_i(t_k)$. To decompose the free-space $\mathcal{W}_f$ we employ the notion of the *Voronoi diagram* defined as follows:



(a) (b)

Fig. 3. Graphical depiction of the geodesic path connecting a robot located at $\mathbf{x}_i(t)$ to its respective target $\boldsymbol{\psi}_i(t)$ residing in the free-space $\mathcal{W}_f$ at time instants t_k (Figure 3(a)) and t_{k+1} (Figure 3(b)). As the target $\boldsymbol{\psi}_i(t)$ moves, the robot updates the geodesic path and tracks the reflex vertex $\mathbf{a}_1(\mathbf{x}_1(t), \boldsymbol{\psi}_1(t))$.

Definition 4.2 ([31]): The *Voronoi diagram* generated by a set of points located at $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is the set $\mathcal{V} = \{\mathcal{V}_1, ..., \mathcal{V}_N\}$, where $\mathcal{V}_i$ is called the *Voronoi cell* of node i that contains all points that are closer to node i than to any other node, according to the Euclidean distance metric, i.e.,

$$\mathcal{V}_i = \{\mathbf{q} \in \mathcal{W}_f \mid \|\mathbf{q} - \mathbf{x}_i\| \leq \|\mathbf{q} - \mathbf{x}_j\|, \ \forall j \neq i\}.$$

In view of the above definition, it is clear that the Voronoi cells $\mathcal{V}_i$ are disjoint sets except at their boundary $\partial\mathcal{V}_i$. Particularly, the polygonal boundary $\partial\mathcal{V}_i$ consists of edges that either lie on the boundary $\partial\mathcal{W}_f$ or are shared with $\partial\mathcal{V}_j$, for some $j \neq i$, as depicted in Figure 2. Therefore, in order to avoid collisions among the robots, we confine the motion of the i-th robot inside its respective Voronoi cell excluding the edges of $\partial\mathcal{V}_i$ that are shared with other robots. Note that since the Voronoi partitioning of a non-convex environment may contain disconnected cells, as in Figure 2, we need to discard those disconnected components of $\mathcal{V}_i$ that do not contain the current robot position $\mathbf{x}_i(t_k)$. In this way, we obtain connected and disjoint subsets of the free space in which the robots can move. These cells are, in general non-convex which can result in con-convex constraints being added to (16).

To obtain convex collision avoidance constraints we construct convex subsets of the above disjoint non-convex cells, in which we now restrict the motion of the robots. In particular, we first construct an arbitrary convex polygonal set denoted by $\mathcal{P}_i(t_k) \subseteq \mathcal{W}_f$, that contains the part of the line segment $[\mathbf{x}_i(t_k), \mathbf{a}_1(\mathbf{x}_i(t_k), \boldsymbol{\psi}_i(t_k))]$ that is contained in the non-convex cell in which robot i is allowed to move.[5] Next we define the half-space

$$\mathcal{H}_i^e(t_k) = \{\mathbf{q} \in \mathbb{R}^2 \mid \mathbf{a}_e^i(t_k)^T\mathbf{q} \leq b_e^i(t_k) + \rho_e^i(t_k)\}, \quad (17)$$

that points inside $\mathcal{V}_i(t_k)$. In (17), $\mathbf{a}_e^i(t_k)^T\mathbf{q} = b_e^i(t_k)$ represents the line equation of the e-th edge of the boundary of the previously defined non-convex set that lies on $\partial\mathcal{V}_i \cap \partial\mathcal{V}_j$. Moreover, $\rho_e^i(t_k)$ are constants used to translate the e-th edge so that the distance between the lines $\mathbf{a}_e^i(t_k)^T\mathbf{q} = b_e^i(t_k)$ and $\mathbf{a}_e^i(t_k)^T\mathbf{q} = b_e^i(t_k) + \rho_e^i(t_k)$ is equal to $\rho > 0$, for all edges

[5]Note that the line segment $[\mathbf{x}_i(t_k), \mathbf{a}_1(\mathbf{x}_i(t_k), \boldsymbol{\psi}_i(t_k))]$ is not necessarily contained entirely in $\mathcal{P}_i$. This is, e.g., the case when the reflex vertex $\mathbf{a}_1$ is located in the Voronoi cell of robot $\mathbf{x}_j$ for $j \neq i$.

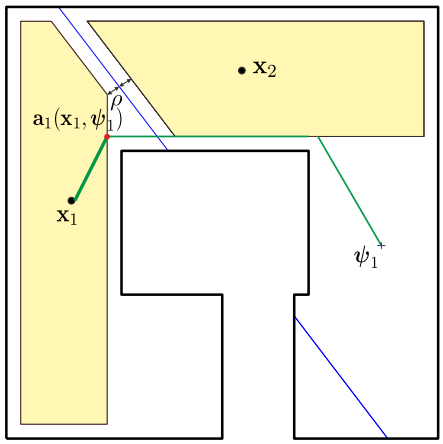


Fig. 4. Graphical example of the spaces that guarantee collision avoidance for a network of two robots. Blue lines determine the Voronoi cells for each robot, the yellow colored polygonal areas stand for the sets $\mathcal{C}_i$ as defined in (18) and the green line stands for the geodesic path $s(\mathbf{x}_1(t_k), \boldsymbol{\psi}_1(t_k))$.

that lie on $\partial\mathcal{V}_i(t_k) \cap \partial\mathcal{V}_j(t_k)$, $i \neq j$, $\forall i$. Then, the convex polygonal space $\mathcal{C}_i(t_k) \subseteq \mathcal{V}_i(t_k)$ in which robot i is confined to move is defined by the intersection of the set $\mathcal{P}_i(t_k)$ with the half-spaces $\mathcal{H}_i^e(t_k)$, i.e.,

$$\mathcal{C}_i(t_k) = \mathcal{P}_i(t_k) \cap \left(\bigcap_{e=1}^{E_i(t_k)} \mathcal{H}_i^e(t_k) \right), \quad (18)$$

where $E_i(t_k)$ is the number of the half-spaces $\mathcal{H}_i^e(t_k)$ defined in (17). Taking the intersection of the set $\mathcal{P}_i$ with the half-spaces $\mathcal{H}_i^e$ ensures that the boundary $\partial\mathcal{V}_i \cap \partial\mathcal{V}_j$ is removed from the collision free cell in which robot i can move. Requiring that

$$\mathbf{x}_i(t_{k+1}) \in \mathcal{C}_i(t_k), \quad (19)$$

ensures collision avoidance among the robots, since the inter-robot distance will always be greater than or equal to $2\rho > 0$. The resulting convex sets $\mathcal{C}_i(t_k)$ for the robot network of Figure 2 are depicted in Figure 4. Note that as the robots move, the Voronoi cells $\mathcal{V}_i$ change, and so do the subsets $\mathcal{C}_i$.

Notice that the proposed method employs a convex constraint (19) to ensure collision-free trajectories in complex environments, instead of using standard non-convex constraints of the form $\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|^2 > 0$ as in [19]. Note also that the construction of constraint in (19) requires only information acquired by the set of Delaunay neighbors of robot i denoted by $\mathcal{D}_i$, where $\mathcal{D}_i = \{j \neq i \mid \mathcal{V}_i \cap \mathcal{V}_j \neq \varnothing \text{ (non-singleton)}\}$.[6]

Remark 4.3 (Non-point Robots): Throughout the paper, for the sake of simplicity we consider point robots. However, our proposed collision avoidance scheme can also account for realistic non-point robots. For example, for robots that are modeled by a disc of radius Δ, as in [19], [22], [23] it suffices to choose the parameter ρ so that it satisfies $\rho > \Delta$.

[6]Note that robot i is aware only of the Delaunay neighbors $j \in \mathcal{D}_i$ for which it holds $R_{ij} > 0$, which implies that robot i may not know all its Delaunay neighbors. Although this will lead to a wrong evaluation of the respective Voronoi cell, it does not compromise the collision avoidance among the robots. The reason is that, since R_{ij} is associated with the inter-robot distance, then $R_{ij} = 0$ entails that the mobility of robots i and j cannot result in their collision.

C. Motion Planning

As discussed before, the robots move during the time intervals (t_k, t_{k+1}), between updates of the communication variables. Incorporating the collision avoidance constraint (19), and the communication constraint (5) into the optimization problem (16), gives rise to the following constrained optimization problem for the new position $\mathbf{x}_i(t_{k+1})$ of robot i:

$$\begin{aligned} \underset{\mathbf{x}_i}{\text{minimize}} \quad & \left\|\mathbf{x}_i - \mathbf{a}_1(\mathbf{x}_i(t_k), \boldsymbol{\psi_i}(t_k))\right\|^2 \\ \text{subject to} \quad & c_i(\mathbf{x}, \mathbf{T}(t_k)) \geq 0, \\ & \mathbf{x}_i \in \mathcal{C}_i(t_k). \end{aligned} \tag{20}$$

While the collision avoidance constraint in (20) can be expressed as a set of linear constraints, since the set $\mathcal{C}_i(t_k)$ is a convex polygonal set, the communication constraint is nonlinear due to the nonlinear dependence of the link reliabilities R_{ij} on the robot positions. To handle those nonlinearities in (20) we employ a sequential convex programming approach to solve the motion planning problem. Specifically, assuming that all the other robots are fixed at positions $\mathbf{x}_j(t_k)$, for $j \neq i$ and given routing decisions $\mathbf{T}(t_k)$, every robot i solves the following convex problem:

$$\begin{aligned} \underset{\mathbf{x}_i}{\text{minimize}} \quad & \left\|\mathbf{x}_i - \mathbf{a}_1(\mathbf{x}_i(t_k), \boldsymbol{\psi_i}(t_k))\right\|^2 \\ \text{subject to} \quad & \tilde{c}_i(\mathbf{x}_i, \{\mathbf{x}_j(t_k)\}_{j\neq i}, \mathbf{T}(t_k)) \geq 0, \\ & \|\mathbf{x}_i - \mathbf{x}_i(t_k)\| \leq \sigma, \\ & \mathbf{x}_i \in \mathcal{C}_i(t_k), \end{aligned} \tag{21}$$

for the updated position $\mathbf{x}_i(t_{k+1})$, where $\{\mathbf{x}_j(t_k)\}_{j\neq i}$ are the positions at time t_k of robots $j \neq i$. Also, $\tilde{c}_i$ stands for a linear approximation of c_i defined as:

$$\begin{aligned} &\tilde{c}_i(\mathbf{x}_i, \{\mathbf{x}_j(t_k)\}_{j\neq i}, \mathbf{T}) \\ &= c_i(\mathbf{x}(t_k), \mathbf{T}) + (\nabla_{\mathbf{x}_i} c_i(\mathbf{x}(t_k), \mathbf{T}))^T (\mathbf{x}_i(t_{k+1}) - \mathbf{x}_i(t_k)). \end{aligned} \tag{22}$$

In (21) we have introduced a trust-region constraint for some $\sigma > 0$, that defines a region where $\tilde{c}_i(\mathbf{x}_i, \{\mathbf{x}_j(t_k)\}_{j\neq i}, \mathbf{T}(t_k))$ is an acceptable approximation of $c_i(\mathbf{x}(t_k), \mathbf{T}(t_k))$. In general, the smaller the size of the trust region is, the more accurate these approximations are. Solving (21) for the new robot positions we obtain the controller $\mathbf{u}_i(t)$ for robot i

$$\mathbf{u}_i(t) = \frac{\mathbf{x}_i(t_{k+1}) - \mathbf{x}_i(t_k)}{\Delta t}, \ \forall t \in (t_k, t_{k+1}), \tag{23}$$

s which is a discrete-time version of the model discussed in (1).

Remark 4.4 (Collision Avoidance): In the analysis provided above, we chose not to model collision avoidance using the standard nonlinear constraints $\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|^2 > 0$, which we could then linearize, as we did with the nonlinear communication constraints (5). The reason behind this approach is that, in general, linearization of the constraints is an approximation that may introduce constraint violation. Although violation of the communication constraints can be handled efficiently ensuring reliable communication among the robots, as discussed at the beginning of Section IV, this is not the case for the collision avoidance constraints which cannot be 'recovered' once they are violated.

V. Distributed Coordination

In this section, we develop a distributed coordination scheme, which combined with the communication and navigation controllers discussed in sections III and IV, dynamically grows a tree network structure, rooted at the access points, with branches that connect leaf nodes responsible for servicing individual tasks to the main network structure. Specifically, each time a new task is announced by a user, a new branch is grown in the network rooted at a branch junction that maintains connectivity with the rest of the network structure. To achieve this goal, two leader teams are formulated, namely, the *primary* and the *secondary leader team*, which we discuss next. The result of this coordination mechanism are sequences of targets $\boldsymbol{\psi}_i(t_k)$ that the robots need to track in order to construct those tree networks that will allow the team of robots to achieve their goal. These sequences of targets constitute the input to the motion controller that was presented in Section IV.

A. Primary Leader Team

1) Leader Election: Assume that a tree network already exists, as in Figure 1, and that a new task is announced at position $\mathbf{q}_v$ in the environment. Then, the team of robots coordinates to elect a *leader* robot that will be responsible for servicing that task. The leader election process is a distributed process that is shown in Algorithm 1. During this phase of coordination, every robot i in the network initializes a vector of bids $\mathbf{d}_i = [0, \ldots, [\mathbf{d}_i]_i, \ldots, 0] \in \mathbb{R}^N$ with all entries equal to zero except for the i-th entry, denoted by $[\mathbf{d}_i]_i$, that contains the bid of robot i. The bid $[\mathbf{d}_i]_i$ can be associated with the geodesic distance from the target.[7] Along with the vector of bids, robot i also initializes a vector of tokens as $\boldsymbol{\phi}_i = [0, \ldots, 1, \ldots, 0] \in \{0, 1\}^N$, so that the i-th entry of this vector $[\boldsymbol{\phi}_i]_i = 1$ indicates that the robot has placed a bid, while entries $[\boldsymbol{\phi}_i]_j$ that are zero indicate that robot i is unaware of bids having been placed by other robots $j \neq i$. The leader election process depends on setting up a distributed auction, where there is no central auctioneer, so that bids placed by the robots are compared against each other. Specifically, the i-th robot communicates its vector of bids $\mathbf{d}_i \in \mathbb{R}^N$ and tokens $\boldsymbol{\phi}_i$ [line 2] to its neighboring robots $j \in \mathcal{N}_i = \{j | R_{ij} > 0\}$ and updates those vectors through a max consensus process [lines 3-4] every time a new communication message is received. When every robot has collected the bids from all robots, i.e., when $\min_j\{[\boldsymbol{\phi}_i]_j\} = 1$, the robot that has placed the maximum bid, i.e., the robot with index $\text{argmax}_j\{[\mathbf{d}_i]_i\}$ will become the leader denoted by $\ell(t)$ [line 6]. In case of ties in the bids, the robot with the highest index will become the leader, i.e., the robot with index $\max\{\text{argmax}_j\{[\mathbf{d}_i]_i\}\}$.

[7]For instance, bids can be associated with the reciprocal of the geodesic distance to the announced task. In this way, the closest robot to the announced task will eventually be elected.

Algorithm 1 Leader Election

Require: $\mathbf{d}_i$ and $\boldsymbol{\phi}_i$;
1: **if** $\min_j\{[\boldsymbol{\phi}_i]_j\} = 0$ **then**
2: Propagate bids and tokens;
3: $\boldsymbol{\phi}_i := \max_{j\in\mathcal{N}_i}\{\boldsymbol{\phi}_i, \boldsymbol{\phi}_j\}$;
4: $\mathbf{d}_i := \max_{j\in\mathcal{N}_i}\{\mathbf{d}_i, \mathbf{d}_j\}$;
5: **else**
6: $\ell(t) := \max\{\text{argmax}_j\{[\mathbf{d}_i]_i\}\}$;
7: **end if**

2) Local Stationary Points: Now, consider that a leader has already been elected and begins to move towards its assigned task at position $\mathbf{q}_v$. Since, it might not always be possible for the leader to service its assigned task due to the imposed communication constraints (5), a *primary leader team* needs to be assembled that will facilitate the leader to accomplish its goal. We denote this team of robots by $\mathcal{P}(t)$. Initially, the primary leader team consists only of the leader, i.e., $\mathcal{P}(t) = \{\ell(t)\}$. While the local communication constraint associated with the leader is satisfied, the leader moves towards its assigned target. However, when the local communication constraint tends to become violated, the leader stops. This situation corresponds to a *local stationary point* of the networked system. In this case a new member should join the primary leader team, without violating connectivity of the network, in order to release the leader from this local minimum and allow it to move further. This is achieved via a *recruit election* process that allows the leader to recruit a robot from a set of possible recruits for assistance.[8] The recruit election process is conceptually similar to a leader election. I.e., it involves bids placed by potential recruits and local auctions to compare those bids. What is different is the set of robots that can participate in a recruit election. Specifically, only *redundant* robots can participate in a recruit election, i.e., robots without an assigned task. In particular, we define *redundant* robots to be those robots that have never won a recruit or a leader election.[9] The set of redundant robots is denoted by $\mathcal{R}(t)$. In a recruit election process, all robots $i \in \mathcal{R}(t)$ place a bid in order to become a recruit that is associated with the residuals $c_i(t_h)$ so that large bids correspond to large residuals. Here t_h denotes the time instant at which the most recent recruit election process was triggered. In this way, the robot with the largest residual at the time t_h at which the network trapped at local stationary point, will eventually be elected. The reason for defining the recruit election bids in this way is that large residuals imply more free space for a robot to move. The elected redundant robot is called the *recruit* and is denoted by h.

Upon the election of the recruit, the robots should cooperate so that the leader gets untrapped from its local stationary point. The end goal is to add a new member to the primary leader team $\mathcal{P}(t)$ that will allow the leader to continue to move towards its assigned task. This should happen without violating end-to-end connectivity of the network. The coordination process that results in a new member joining the primary leader team is discussed in Section V-B. As the primary leader team continues to move towards the announced task, it either reaches this task or it stops at another *local stationary point* defined as the situation where any motion of the last node that joined the primary leader team will violate connectivity of the network. In this case, a new recruit election takes place and a new member is eventually added to the primary leader team.

Local stationary points are not only due to violation of the communication constraints by any motion of the primary leader team, but also due to situations where the newly elected leader has another role in the network that is critical for the mission. For example, the newly elected leader can hold a position in space where the presence of a node is necessary to ensure network connectivity, or the newly elected leader could have also been a leader in the past, in which case its current role is to remain in close proximity to its previous assigned task. In such situations, the new leader cannot move at all, until it is released from its past duties. To resolve the conflict in the roles of the elected leader, a recruit election is triggered directly after the leader election and the robots in the network coordinate to release the leader from its past duties. The end goal in this coordination process is to physically replace the leader in the workspace, so that the leader is released from its past duties; see Section V-B. When this happens, the leader continues to move towards the announced task and it either reaches it or stops at a local stationary point due to the communication constraints. The latter case results in adding a new member to the primary leader team, as discussed previously.

3) Coordination within the Primary Leader Team: The process of adding new members to the primary leader team results in a new branch growing from the network with the leaf node corresponding to the leader. Branches are connected to the rest of the network at nodes called the *junction nodes*. All nodes belonging to the new branch, excluding the junction node, constitute the primary leader team. The ordered set $\mathcal{P}(t)$ is constructed so that the first robot that joins the primary leader team is always the last entry of $\mathcal{P}(t)$ and correspondingly, the last robot that joins this team is the first entry of $\mathcal{P}(t)$. Consequently, the last entry of $\mathcal{P}(t)$ is always the leader, since this is the first robot that joins $\mathcal{P}(t)$ upon its election; see Figure 5(a). Denoting by $p_i(t)$ the i-th member of $\mathcal{P}(t)$ we conclude that a recruit election is triggered by robot $p_1(t)$, which is the last robot that joined the primary leader team, when it is stuck due to the violation of communication constraint $c_{p_1}(t) \geq 0$.

The robots in the primary leader team move as follows. Every member of the primary leader team follows the next robot in $\mathcal{P}(t)$ except for the leader, which moves towards the task located at $\mathbf{q}_v$, i.e.,

$$\boldsymbol{\psi}_{p_i}(t) = \begin{cases} \mathbf{q}_v & \text{if } p_i(t) = \ell(t) \\ \mathbf{x}_{p_{i+1}}(t) & \text{otherwise} \end{cases}. \tag{24}$$

[8]Note that potential recruits can be physically located far away from the leader, therefore, it might not be easy to recruit them without a specialized procedure.

[9]Initially, when no tasks are announced, all robots are considered redundant, since there is no task assigned to them. When a robot wins a leader or a recruit election, it is no more considered redundant as at that point a task is assigned to the elected robot.

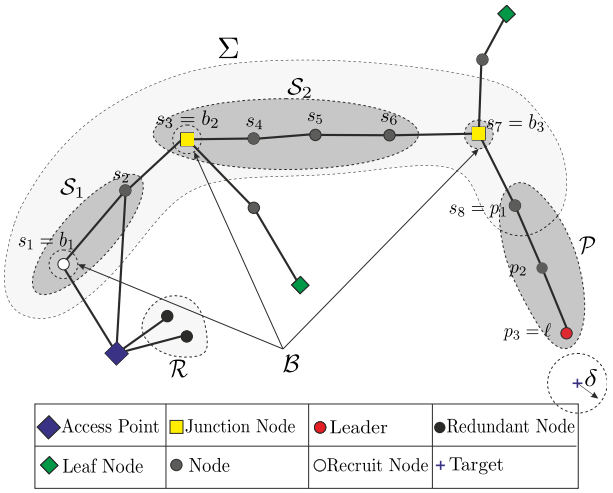
The primary leader team is said to have accomplished its goal when the leader is servicing the announced task, i.e., when (2) holds. Note that the tree network constructed by the above process is only a subgraph of the actual communication network between the nodes, and other communication links that do not belong to the tree network can exist due to the proximity between nodes; see, e.g., Figure 6. The communication network and the tree network are defined as follows:

Definition 5.1 (Communication Network): The communication network is defined as a dynamic directed graph $\mathcal{G}_c(t) = (\mathcal{V}_c, \mathcal{E}_c(t))$, where $\mathcal{V}_c = \{1, 2, \ldots, N, \ldots, N+K\}$ and $\mathcal{E}_c(t) = \{(i,j) | i, j \in \mathcal{V}\}$, where a communication link between i and j exists if and only if $T_{ij} R_{ij} > 0$.

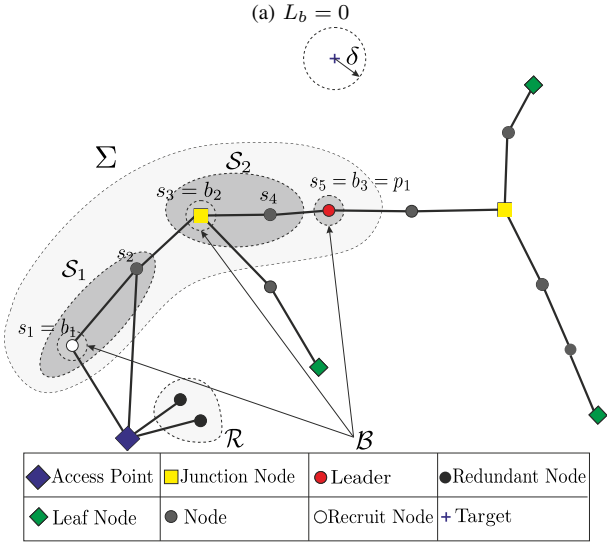
Definition 5.2 (Tree Network): The tree network is defined as a dynamic directed graph $\mathcal{G}_t(t) = (\mathcal{V}_t, \mathcal{E}_t(t))$, which is constructed by the coordination process presented in Section V. The tree network is a subgraph of the actual communication network, i.e., $\mathcal{V}_t = \mathcal{V}_c$ and $\mathcal{E}_t(t) \subseteq \mathcal{E}_c(t)$.

Remark 5.3 (Leader election): Allowing every robot in the network to participate in the leader election entails that the closest robot to the announced task will eventually be elected. In doing so, we achieve a more efficient utilization of available resources, i.e., nodes, since we avoid situations where new branches are grown from locations far away from the new task and run in parallel with the existing network structure.

Remark 5.4 (Recruit election): When the primary leader team needs to trigger a recruit election, robot p_1 transmits a message to its neighboring robots $j \in \mathcal{N}_{p_1}$. This message is propagated in the network until all the redundant robots that are connected through a multi-hop path to the leader are aware that a new recruit is needed. When this happens, a recruit election follows. Then, the elected recruit h transmits a message that eventually reaches robot p_1 to inform it about the recruit election result.



(a) $L_b = 0$



(b) $L_b = 1$

Fig. 5. Graphical depiction of breaking the group of robots in Σ into subgroups when $L_b = 0$ (Figure 5(a)) and $L_b = 1$ (Figure 5(b)). In both cases, the leader team $\mathcal{S}_1$ moves first and upon its convergence the leader team $\mathcal{S}_2$ starts moving. When all the members of the latter converge to their destinations, the primary leader team has received the required assistance and can start moving. Specifically, in Figure 5(a), the robot s_7 will join the primary leader team and will start following the robot p_1 that triggered a recruit election. As for Figure 5(b), the robot s_4 releases the leader from its past role through replacing it in the workspace and then the leader can start moving.

B. Secondary Leader Team

The *secondary leader team* is a team of robots that facilitates the primary leader team to move towards its task when the latter is trapped at a local stationary point. Essentially, the secondary leader team is a team of robots that collaborate to transfer the assistance that the recruit can provide to the primary leader team, while ensuring that network connectivity is preserved.

1) Coordination within the Secondary Leader Team: Assume that the primary leader team is trapped at a local stationary point. Assume also that a recruit has already been elected. Then by definition the robots that can assist in transferring the recruit's help to the primary leader team are the ones that belong to the shortest path in the tree network that connects the recruit h to robot $p_1(t) \in \mathcal{P}(t)$.[10] The indices of these robots are collected in an ordered set denoted by $\Sigma(t)$. The order is determined as follows. Denoting by $s_i(t)$ the i-th entry of $\Sigma(t)$, we assume that $s_1(t)$ is the recruit and for every i, robot $s_{i+1}(t)$ is the next-hop robot following $s_i(t)$ in the aforementioned path, until reaching robot $p_1(t)$. The construction of $\Sigma(t)$ is also illustrated in Figure 5. Then, the *secondary leader team* is defined as $\mathcal{S}(t) = \Sigma(t) \setminus \{p_1(t)\}$, where $p_1(t) \in \mathcal{P}(t)$.

The robots of the secondary leader team, i.e., the robots in $\Sigma(t)$ move towards the next robot in the set $\Sigma(t)$, in a similar way as those in $\mathcal{P}(t)$. Specifically, robot $s_i \in \Sigma(t)$ moves towards the position occupied by robot s_{i+1} at time instant t_h and in doing so, eventually, the primary leader team will receive the required assistance. Thus, the target $\boldsymbol{\psi}_{s_i}$ for robot $s_i \in \mathcal{S}(t)$ is given by

$$\boldsymbol{\psi}_{s_i} = \mathbf{x}_{s_{i+1}}(t_h), \tag{25}$$

[10]Since we refer to the tree network, the path that connects the recruit and robot $p_1(t) \in \mathcal{P}(t)$ is unique and can be computed through applying a distributed shortest path algorithm.

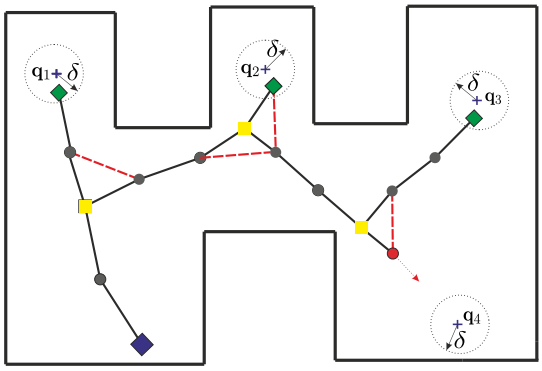


Fig. 6. Graphical example of a tree network structure (gray edges) and the respective underlying communication graph (gray and red edges).

where $\mathbf{x}_{s_{i+1}}$ is the location of robot s_{i+1}. The reason that we use the time instant t_h, i.e., the time instant at which the most recent recruit election process was triggered, in (25) is because at this time instant all robots in $\Sigma(t)$ are in a feasible configuration meaning that the communication constraints are satisfied.

2) Decomposition of Secondary Leader Team: The secondary leader team may include leaf nodes, i.e., prior leaders, that service tasks in the workspace and/or junctions nodes that connect branches to the main network structure; see Figure 5. To avoid violating end-to-end connectivity and interrupting the service of a task, which could happen if a junction node or a leaf node moved without having been replaced first by another robot, we further decompose the secondary leader team into subgroups of robots. Among those subgroups only one can move at a time and in doing so, we guarantee end-to-end connectivity for all time and uninterrupted service of tasks. The decomposition of $\Sigma(t)$ into subsets is based on identifying *break points* in $\Sigma(t)$, i.e., robots that cannot move without having been replaced first by another robot. These break points consist of the junction nodes and leaf nodes contained in $\Sigma(t)$. Also, in case a recruit election has been triggered due to a conflict in the roles of the elected leader (see Section V-A), then, the leader should also be a break point in the path $\Sigma(t)$. The reason is that in this case, similar to a junction and leaf node, the leader needs to be 'unlocked' before it starts moving, as discussed in Section V-A; such a case is depicted in Figure 5(b). In order to determine whether the leader should be a break point or not, we introduce a binary variable $L_b \in \{0, 1\}$. In particular, we set $L_b = 1$ if a recruit election was triggered due to a conflict in the roles of the elected leader and $L_b = 0$, if a recruit election was triggered by robot p_1 due to violation of its local communication constraint.

The set $\mathcal{B}(t) = [b_1(t), \ldots, b_v(t)]$ of all break nodes in $\Sigma(t)$ determines the starting and the end nodes of each subgroup of robots in $\mathcal{S}$. Specifically, we select $b_1(t) = s_1 = h$, and the rest of the entries in $\mathcal{B}$ are occupied by the junction and the leaf nodes in the order that they appear in $\Sigma(t)$. The last entry is occupied by the leader if $L_b = 1$. Given the set $\mathcal{B}(t)$, we have that the β-th subgroup of the secondary leader team $\mathcal{S}(t)$, is a sequence of robots starting from robot $s_i(t) = b_\beta(t)$ and ending at robot $s_{j-1}(t)$, where $s_j(t) = b_{\beta+1}(t)$. We denote the β-th subgroup by $\mathcal{S}_\beta(t) = [b_\beta(t), b_{\beta+1}(t))_\Sigma$, where the subscript Σ means that the sequence of nodes starting at $b_\beta(t)$ and ending with $b_{\beta+1}(t)$ is taken from $\Sigma(t)$ and the right open interval means that $b_{\beta+1}(t)$ is not included in $\mathcal{S}_\beta(t)$. Therefore, the total number of subgroups $\mathcal{S}_\beta(t)$ is $|\mathcal{B}(t)| - 1$, where $|\cdot|$ stands for the cardinality of a set. Note that the last node in $\mathcal{B}(t)$ is not a member of any subgroup $\mathcal{S}_\beta(t)$; see Figure 5. As it will become clear in the following section, this robot either joins the primary leader team in order to help it move further or is the leader that needs to be replaced due to a conflict in its assigned roles.

C. Active Leader Team

As discussed in Section V-B, given the available subgroups $\mathcal{S}_\beta(t)$, we require that only one of them can move at a time. This will ensure end-to-end connectivity and uninterrupted service of tasks. Specifically, the robot group $\mathcal{S}_{\beta+1}(t)$ is allowed to move as soon as all the members of $\mathcal{S}_\beta(t)$ have reached their goals, i.e.,

$$\left\|\mathbf{x}_{s_i}(t) - \boldsymbol{\psi}_{s_i}\right\| \leq \delta, \forall s_i(t) \in \mathcal{S}_\beta(t), \tag{26}$$

for a sufficiently small $\delta > 0$. When the last subgroup of robots of the secondary leader team has reached its goal, the primary leader team receives the required assistance to move further.

1) Escaping from Local Stationary Points: To understand how the primary leader team is eventually assisted to move further, we examine the two cases for which a recruit election is triggered. First, assume that a recruit election is triggered because robot p_1 cannot move further due to the communication constraints, i.e., $L_b = 0$. Such a case is depicted in Figure 5(a). As soon as a recruit is elected, the first subgroup $\mathcal{S}_1 = [b_1, b_2)_\Sigma$ of the secondary leader team is constructed and begins moving towards targets defined by (25). When the robots in this subgroup have reached their goals, the next subgroup $\mathcal{S}_2 = [b_2, b_3)_\Sigma$ is assembled and begins its motion. This procedure is repeated until the last subgroup of the secondary leader team converges to its desired configuration. When this happens, the last node in $\mathcal{B}$ joins the primary leader team $\mathcal{P}$ with the task to start following the robot p_1 that triggered a recruit election as dictated by (24).[11] A similar reasoning applies also when a recruit election is triggered because of a conflict in the roles of the elected leader, i.e., $L_b = 1$; see Figure 5(b). The only difference lies in the fact that when all robots in the last subgroup have reached their destinations, the last robot in this team will occupy the position of the leader releasing it so that the leader can move freely towards its goal. In our proposed algorithm, at any time, either a subgroup of the secondary leader team or the primary leader team is allowed to move. The team of robots that moves at time t is called the *active leader team* denoted by $\mathcal{A}(t)$. The active leader team is determined by Algorithm 3, while Algorithm 2 summarizes the target points assigned to every robot in $\mathcal{A}(t)$, as per Sections V-A and V-B.

[11] When this happens, the primary leader team $\mathcal{P}(t)$ is updated and, as a result, the robot that just joined the primary leader team is now the robot with index p_1. Accordingly, the other members of $\mathcal{P}(t)$ update their indices p_i.

Algorithm 2 Selection of target $\boldsymbol{\psi}_i(t)$

if $\mathcal{A}(t) \equiv \mathcal{P}(t)$ **then**
  if $p_i(t) = \ell(t)$ **then**
    $\boldsymbol{\psi}_{p_i} = \mathbf{q}_u$;
  else
    $\boldsymbol{\psi}_{p_i}(t) = \mathbf{x}_{p_{i+1}}(t)$;
  end if
else if $\mathcal{A} \equiv \mathcal{S}_\beta$ **then**
  $\boldsymbol{\psi}_{s_i} = \mathbf{x}_{s_{i+1}}(t_h)$;
end if

2) Distributed Construction of $\mathcal{A}(t)$: In the rest of this section, we focus on how a robot can determine if it belongs to the active leader team $\mathcal{A}(t)$ in a distributed way. This procedure consists of two main phases. In the first phase, every robot has to determine if it belongs to the secondary leader team. All robots that belong to the secondary leader team compute the subgroup $\mathcal{S}_\beta(t)$, in which they belong and the second phase follows. In the second phase, all robots $i \in \Sigma(t) \cup \mathcal{P}(t)$ check if $\mathcal{A}(t)$ coincides with the primary leader team or a subgroup of the secondary leader team; see Algorithm 3. These two phases are discussed next in detail.

At the beginning of the first phase, robot $p_1(t) \in \mathcal{P}(t)$ that triggered a recruit election computes the set $\Sigma(t)$, i.e., the path in the tree network that connects itself to the recruit. Then, it transmits the set $\Sigma(t)$ to the neighboring robots that belong to $\Sigma(t)$, i.e., to robots $j \in \mathcal{N}_{p_1}(t) \cap \Sigma(t)$. This process is repeated by the last robot that receives the set $\Sigma(t)$ until the communication message is received by the elected recruit h. In this way, every robot knows if it belongs to $\Sigma(t)$. Next, every robot $s_i(t) \in \Sigma(t)$ determines the subgroup $\mathcal{S}_\beta(t)$ in which it belongs. To achieve this, the robots in $\Sigma(t)$ need to determine the set of break points in $\mathcal{B}(t)$. This is done in a distributed way using a max-consensus algorithm. In particular, every robot $s_i(t)$ transmits to its neighboring robots in $\Sigma(t)$ a vector $\mathbf{r}_{s_i}$, whose entries are initially all negative (any number) except for the i-th entry, denoted by $[\mathbf{r}_{s_i}]_i \in \{0, 1\}$, that determines whether robot s_i is a break point or not. Specifically, we assume that $[\mathbf{r}_{s_i}]_i = 0$, if robot s_i is not a break point and $[\mathbf{r}_{s_i}]_i = 1$ otherwise. Then, every robot s_i updates its associated vector $\mathbf{r}_{s_i}$ through a max-consensus process, i.e., $\mathbf{r}_{s_i} = \max\{\mathbf{r}_{s_i}, \mathbf{r}_{s_j}\}$, $s_j \in \Sigma(t) \cap \mathcal{N}_{s_i}(t)$ until all its entries have non-negative values. When this happens, every robot $s_i \in \Sigma(t)$ is aware of the break points $\mathcal{B}$ and the subgroup $\mathcal{S}_\beta(t)$ to which it belongs.

The second phase of the process involves the robots determining the active leader team and updating it in a distributed fashion when needed. The active leader team is initialized to be the primary leader team upon the election of leader [line 15, Alg. 3]. As soon as a recruit election is triggered, the active leader team becomes the first subgroup of the secondary leader team [lines 1-13, Alg. 3]. Once the robots in this subgroup reach their destinations, the active leader team is updated to the next subgroup and the process continues until the robots in the last subgroup have reached their destinations [lines 11-13, Alg. 3]. At that point, the active leader team switches back

Algorithm 3 Computation of the Active leader team $\mathcal{A}(t)$

Require: $\Sigma(t)$, $\mathcal{B}(t)$, $\mathcal{P}(t)$;
1: **if** recruit Election **then**
2: **if** $\wedge_{s_j \in \mathcal{S}_{|\mathcal{B}|-1}}([\mathbf{t}_{s_j}]_j = 1)$ **then**
3: **if** $L_b = 0$ **then**
4: $\mathcal{P}(t) = [b_{\beta+1}(t); \mathcal{P}(t)]$;
5: **end if**
6: Go to 15;
7: **else**
8: $\mathcal{S}_\beta(t) = [b_\beta(t), b_{\beta+1}(t))_\Sigma$;
9: **end if**
10: $\mathcal{A}(t) := \mathcal{S}_\beta(t)$;
11: **if** $\wedge_{s_j \in \mathcal{A}}([\mathbf{t}_{s_j}]_j = 1)$ **then**
12: $\beta := \beta + 1$;
13: **end if**
14: **else**
15: $\mathcal{A}(t) := \mathcal{P}(t)$;
16: $\beta := 1$
17: **end if**

to the primary leader team [lines 2-9, Alg. 3]. To determine when a subgroup $\mathcal{S}_\beta$ has converged, we again employ a max-consensus algorithm. Particularly, each robot $s_i(t) \in \mathcal{A}(t)$ is associated with a vector $\mathbf{t}_{s_i} \in \{0, 1\}^{|\mathcal{A}|}$ with zero entries initially. When the s_i-th robot has accomplished its goal, i.e., when it has reached its destination, it updates the i-th entry of $\mathbf{t}_{s_i}$, denoted by $[\mathbf{t}_{s_i}]_i$, from 0 to 1. At the same time, the robots in $\mathcal{A}$ communicate and update their respective vectors $\mathbf{t}_{s_i}$ through a max-consensus process, i.e., $\mathbf{t}_{s_i} = \max\{\mathbf{t}_{s_i}, \mathbf{t}_{s_j}\}$, $s_j \in \mathcal{A}(t) \cap \mathcal{N}_{s_i}(t)$. When all entries of $\mathbf{t}_{s_i}$ have become equal to 1 for all robots in $\mathcal{A}(t)$, then the active leader team is updated. In this way, Algorithm 3 can be run in a distributed fashion across the robots of the network.

Remark 5.5 (Relationship between δ and ρ): To ensure both satisfaction of the collision avoidance constraint defined in (19) and task accomplishment defined in (2), we need to choose the problem parameters δ and ρ so that $\delta \geq \max\{2\rho, \rho\} = 2\rho$. To see this, recall first that the robots move in the free space $\mathcal{W}_f$. Thus, the distance between any robot and any point $\mathbf{q} \in \partial\mathcal{W}$ is always greater than or equal to ρ. Thus, if $\boldsymbol{\psi}_i \in \partial\mathcal{W}$, then we need $\delta \geq \rho$ in order to ensure task accomplishment. Also, recall that the collision avoidance constraint (19) ensures that the inter-robot distance is always greater than or equal to 2ρ. Following a similar reasoning as previously, in case a robot needs to replace another robot in the workspace, we need $\delta \geq 2\rho$. Thus, $\delta \geq \max\{2\rho, \rho\} = 2\rho$ should hold in order to ensure successful task accomplishment respecting at the same time the collision avoidance constraints. Also, the problem parameters δ and ρ can be selected to be arbitrarily small as long as it holds $\delta \geq 2\rho$ in order to ensure that robots approach sufficiently close their respective targets.

D. Switching of Robot Roles within the Active Leader Team

In this section, we discuss how the robots in the active leader team switch between different roles. These roles are, namely,

a junction node, a leaf node, a redundant node, a node, and a leader. First, we examine the case where the active leader team is a part of the secondary leader team and then we discuss the case where the active leader team is the primary leader team.

Assume that the active leader team is a subgroup of the secondary leader team and consider that $L_b = 0$. Then every robot $s_i \in \mathcal{A}(t)$ moves towards the target $\boldsymbol{\psi}_{s_i} = \mathbf{x}_{s_{i+1}}(t_h)$ as per equation (25). Once the robot s_i reaches its $\boldsymbol{\psi}_{s_i}$, it adopts the role that node s_{i+1} had at the time instant t_h [line 4, Algorithm 4]. When this happens, robot s_{i+1} is released from its past role and its new duties are dictated by whether it still belongs to the active leader team, i.e., whether $s_{i+1} \in \mathcal{A}(t)$ [line 6, Algorithm 4]. In this way, it is guaranteed that there is always a robot present at the branch junction locations ensuring that connectivity in the network is preserved. To illustrate this fact, consider Figure 5(a) and assume that $\mathcal{A} = \mathcal{S}_2$. Once robot s_6 reaches the location of robot s_7, it will become the new junction node, releasing s_7 from its past role as the junction node. Once released, robot s_7 joins the primary leader team and obtains a new role to track a target determined by (24); see Section V-C.

Now, assume that $L_b = 1$, i.e., that the new leader has a conflict with its past role in the network. In this case, the new leader belongs to the set $\mathcal{S}$. Assume that the leader is robot s_{i+1}. Then, robot s_i moves towards the leader s_{i+1}. Once robot s_i reaches the leader, it does not adopt the role of a leader. Instead, it becomes a junction node from where a new branch will grow that will assist the leader in accomplishing its goal [line 12, Algorithm 4]. Once s_i becomes a junction node, the leader is released from its prior duties and begins moving towards its goal [line 13, Algorithm 4]. The other members of the secondary leader team update their roles as previously discussed [lines 15-18, Algorithm 4]. To illustrate this behavior, consider Figure 5(b) and assume that the active leader team $\mathcal{A}$ is the set $\mathcal{S}_2$. When the robots in $\mathcal{A}$ have reached their destinations, the robot s_4 will become a junction node located at the position where the leader was originally present.

In case the active leader team is the primary leader team, all robots $p_i \in \mathcal{A}(t)$ except for the leader retain their respective roles, i.e., the role of a node for all time [line 27, Algorithm 4]. Regarding the leader, as soon as it accomplishes its task as defined by (2), it becomes a leaf node in the tree network [line 25, Algorithm 4]. The integrated distributed hybrid system resulting from the combination of motion planning, communication control, and distributed coordination, is illustrated in Algorithm 5.

Algorithm 4 Update of role for robot $i \in \mathcal{A}(t)$

```
1:  if A(t) ≡ S_β(t) then
2:    if L_b = 0 then
3:      if ||x_{s_i} − ψ_{s_i}|| ≤ δ then
4:        s_i adopts the role that robot s_{i+1} had at t_h;
5:        if s_{i+1} is junction node then
6:          s_{i+1} becomes a node;
7:        end if
8:      end if
9:    else if L_b = 1 then
10:     if ||x_{s_i}(t) − ψ_{s_i}|| ≤ δ then
11:       if s_{i+1}(t) ≡ ℓ(t) then
12:         s_i(t) becomes a junction node;
13:         ℓ is released from past duties;
14:       else
15:         s_i adopts the role that robot s_{i+1} had at t_h;
16:         if s_{i+1} is junction node then
17:           s_{i+1} becomes a node;
18:         end if
19:       end if
20:     end if
21:   end if
22: end if
23: if A(t) ≡ P(t) then
24:   if p_i = ℓ then
25:     p_i becomes a leaf node if task is being serviced;
26:   else
27:     p_i keeps being a node in P(t);
28:   end if
29: end if
```

Remark 5.6 (Switching between redundant and simple nodes). We consider the following two cases: (i) It is possible that redundant robots are critical in routing the information back to the APs. This case occurs when the tree network is not directly connected to the AP, i.e., when redundant robots are necessary for establishing a multi-hop communication path between the tree structure and the AP. Such a scenario is depicted in Figure 7. In this case, the robots that belong to this communication path become nodes. In case there are more than one such communication paths, as in Figure 7, we choose the one that contains the most reliable links. (ii) Additionally, it is possible that there are nodes that are eventually not critical in the end-to-end network connectivity, i.e., information is not routed through them to APs. To illustrate this point, consider a leaf node i and a node j that both belong to the same branch of the tree network and assume that communication of leaf node i with an AP is attained through the node j, i.e., $R_{ij}T_{ij} \neq 0$. Assume also that due to the evolving network topology the optimal routing decision T_{ij} computed by (13) may eventually become 0, i.e., the leaf node i may decide to route information back to the APs through another node that belongs to a new branch. In this case, this means that node j can now be considered redundant.

E. Correctness of Proposed Distributed Control Scheme

Correctness of the proposed distributed control scheme is guaranteed by its construction, as presented in the previous sections. Specifically, assume N mobile robots in a complex environment $\mathcal{W}$, which move as per Algorithm 5 to service tasks that are announced sequentially. Then, provided a solution to the problem exists, i.e., provided there is a sufficient number of robots that can service the tasks, these tasks will be completed (local stationary points will be avoided), collisions between robots or between robots and the environment will be avoided, and reliable communication with the infrastructure of access points will be maintained.

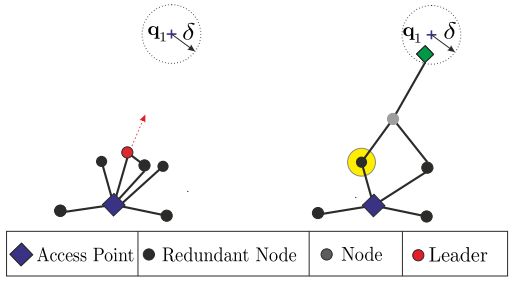


Fig. 7. An illustration of a case where a redundant robot needs to switch its role to a node. Due to the leader's mobility, a tree network structure is developed consisting of a node and a leaf node, whose connection to the AP is attained through redundant robots. In this case, the redundant robot that is marked with yellow color switches its role to a node.

Algorithm 5 Distributed Hybrid Control at Robot i

for $k = 0$ to ∞ **do**
  Compute the routing variables $\{T_{ij}\}_{j=1}^{N+K}$ via the primal-dual iteration algorithm [13]-[14];
  if $i \in \mathcal{A}(t)$ [Algorithm 3] **then**
    if $c_i(\mathbf{x}(t_k), \mathbf{T}(t_k)) \geq 0$ **then**
      Select target $\boldsymbol{\psi}_i$ [Algorithm 2];
      Compute geodesic path to target $\boldsymbol{\psi}_i$;
      Compute next position $\mathbf{x}_i(t_{k+1})$ via the optimization problem (21);
      Move towards $\mathbf{x}_i(t_{k+1})$ according to (23);
      Update the role [Algorithm 4];
    else
      Stay motionless;
    end if
  else
    Stay motionless;
  end if
end for

To show this result, assume that a new unserviced target has been announced and that a leader has been elected via Algorithm 1 giving rise to the formation of a primary leader team defined in sections V-A2 and V-A3. When the primary leader team is trapped at a local stationary point due to the communication constraints, as discussed in Section V-A2, a recruit election is triggered that results in a new member joining the primary leader team that in turns provides room for the leader to move towards its goal, as discussed in Section V-C1. If there is a sufficient number of redundant nodes, then a sufficient number of recruits will be elected to help the leader accomplish its task as per the coordination scheme presented in sections V-B and V-C. Otherwise, the leader will not service the task because of insufficient number of available robots. Since the proposed scheme generates tree networks with branch junctions located the closest to the new announced tasks, cycles are avoided and so are long branches that run in parallel to the existing network structure, resulting in an efficient utilization of resources.

Moreover, during evolution of the network, communication of all robots with the APs is guaranteed for all time either via a multi-hop path or directly. The reason is that the communication constraints guarantee connectivity within the active leader team while switching of the active leader team is performed so that there is no disconnection at the junction nodes; see Section V-B2. Therefore, communication with the APs is always guaranteed. Furthermore, as mobility respects the communication constraints an acceptable quality of communication is maintained during evolution of the system. Finally, collisions between robots and the environment, or, collisions between robots is guaranteed by construction of the collision avoidance frameworks presented in sections IV-A and IV-B and the use of geodesic paths for motion planning, that locally 'convexify' planning in the vicinity of every robot.

VI. Simulation Studies

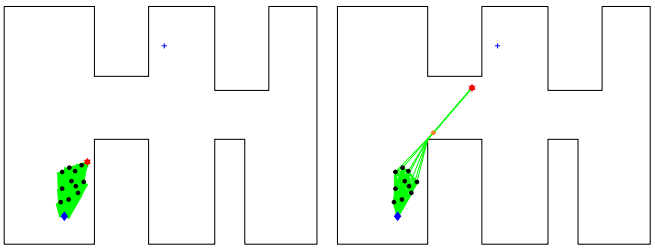
In this section, two simulation studies are presented to illustrate our proposed method. All optimization problems are solved in Matlab using the CVX toolbox [32]. In both simulation studies, the channel reliability $R(\mathbf{x}_i, \mathbf{x}_j)$ is modeled as a decreasing function of the distance between nodes i and j, i.e.,

$$R(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & \text{if } \|\mathbf{x}_{ij}\| < l \\ \sum_{p=0}^{3} a_p \|\mathbf{x}_{ij}\|^p & \text{if } l < \|\mathbf{x}_{ij}\| \leq u \\ 0 & \text{if } \|\mathbf{x}_{ij}\| > u \end{cases}, \quad (27)$$

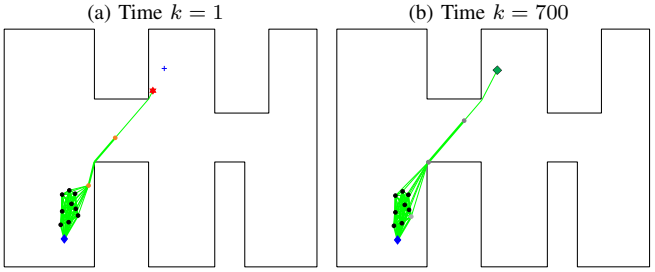
where $\|x_{ij}\| = \|\mathbf{x}_i - \mathbf{x}_j\|$ and the constants a_p, $p = 0, \ldots, 3$ are chosen so that $R(\mathbf{x}_i, \mathbf{x}_j)$ is a differentiable function. The model in (27) is a polynomial fitting of experimental curves found in the literature [25]. In practice, an accurate estimation of the channel reliability is hard to obtain, as it depends on path loss that is a function of the distance between the transmitter and the receiver, shadowing effects due to the existence of obstacles in the propagation path, and multi-path fading effects due to reflections and refractions of the electromagnetic waves, which are difficult to predetermine. It is shown in [25] that on the average $R(\mathbf{x}_i, \mathbf{x}_j)$ is a decreasing function of the distance between nodes, which validates the model (27) used here. Also, the rates r_i are assumed to be common for all robots and equal to 0.075.

The first simulation study concerns a mobile robot network consisting of $N = 13$ robots and $K = 1$ AP residing in a complex environment whose convex hull has diameter equal to 3.1 units. The parameters l and u in (27) are selected to be 0.4 and 0.5 units, respectively . The position of the AP is $\mathbf{x}_{14} = [0.5, 0.2]$ and the robots are initially deployed as shown in Figure 8(a). Figures 8, 14 and 10 show the evolution of the network when the first, second, and third target has been announced, respectively.

Once the first target is announced at position $\mathbf{q}_1 = (1.33, 1.42)$, a leader is elected and starts moving towards the target see (Figure 8(a)). When the leader is trapped at a local stationary point, a recruit election is triggered by the leader and as a result, a new member joins the primary leader team, as shown in Figure 8(b). At a later time instant, the primary leader team is trapped again due to the presence of communication constraints. As before, a recruit election is triggered and a new robot joins the primary leader team; see Figure 8(c). Eventually, the target is serviced as shown in Figure 8(d).

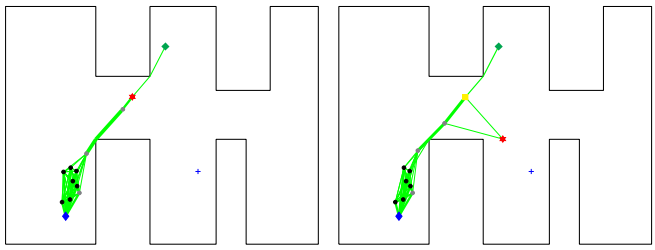


(a) Time $k = 1$ (b) Time $k = 700$

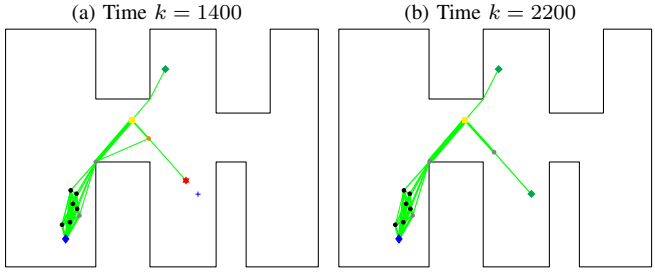


(c) Time $k = 900$ (d) Time $k = 1084$

Fig. 8. Simulation Study I: Evolution of the communication network when the first target (blue cross) is announced. Figures 8(a) through 8(d) show the evolution of the system at different time instants. Green lines represent the communication links among the robots while red lines depict the constructed tree network. Their thickness depends on the value of $T_{ij}R(\mathbf{x}_i, \mathbf{x}_j)$, i.e., thicker lines capture higher values. The blue rhombus represents the AP, the yellow square illustrates a junction node, the red star stands for the leader and the orange dots for the rest members of the primary leader team. The redundant robots are depicted by black dots and the green rhombus represents a leaf node, i.e., a robot that services a target.



(a) Time $k = 1400$ (b) Time $k = 2200$



(c) Time $k = 2700$ (d) Time $k = 2840$

Fig. 9. Simulation Study I: Evolution of the communication network when the second target (blue cross) is announced. Figures 9(a) through 9(d) show the evolution of the system at different time instants. Green lines represent the communication links among the robots. Red lines depict the constructed tree network across the edges of which robots in the secondary leader team move as described in Section V-B; see Figures 9(a)-9(b). Their thickness depends on the value of $T_{ij}R(\mathbf{x}_i, \mathbf{x}_j)$, i.e., thicker lines capture higher values. The blue rhombus represents the AP, the yellow squares illustrate a junction node, the red star stands for the leader and the orange dots for the rest members of the primary leader team. The redundant robots are depicted by black dots and the green rhombus represents a leaf node, i.e., a robot that services a target.

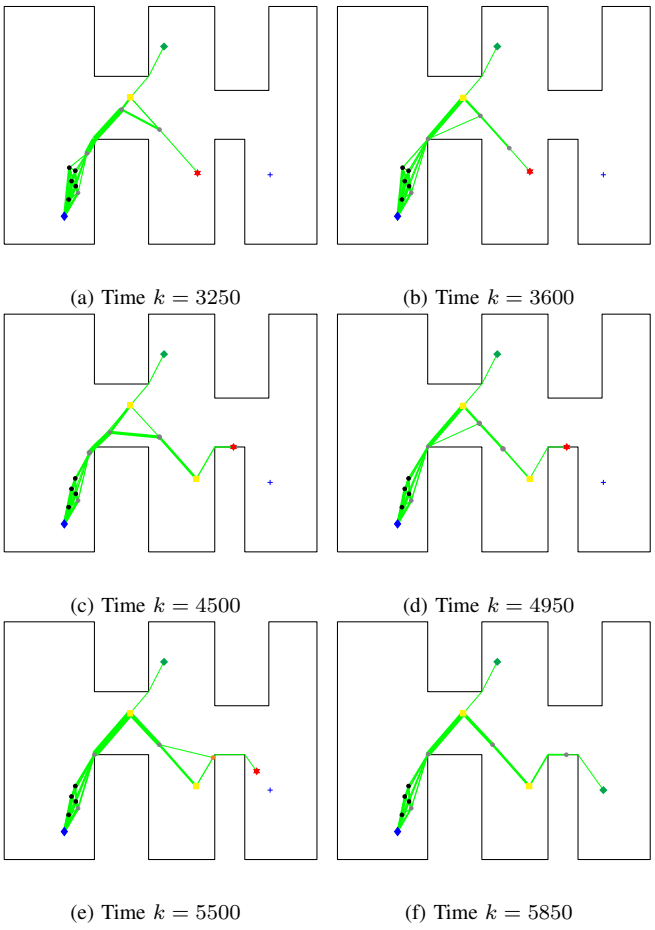
Notice also in Figure 8(d) that a redundant robot has switched its role to a node, as discussed in Remark 5.6.

When the first target has been serviced, a second target is announced at position $\mathbf{q}_2 = (1.6, 0.52)$ and is followed by a new leader election; see Figure 14. Observe in Figure 8(d) that there is a conflict in the roles of the elected robot, since before its election, it belonged to a branch of the tree network. Therefore, the leader triggers a recruit election and, as a result, a secondary leader team is formed that aims to 'unlock' the leader; see Figure 9(a). When the leader is released, a junction node is developed from which a new branch is grown. Next, the leader starts moving towards its target until its communication constraint tend to become violated. At that point, a recruit election is triggered again by the leader. When the robots in the secondary leader team have reached their destinations, a new robot joins the primary leader team as shown in Figure 9(c) and the leader is now free to reach its target; see Figure 9(d).

Finally, a third target is announced at position $\mathbf{q}_3 = (2.21, 0.5)$; see Figure 10. When this happens, a new leader election occurs and the elected leader is a leaf node in the existing tree network. In order to resolve the conflict in the roles of the elected leader, a recruit election is triggered and two subgroups of the secondary leader teams are assembled that coordinate as per the methods developed in Section V to release the leader from its past role; see Figures 10(a)-10(b). When the leader is released, it moves towards its announced task until it gets trapped at a local stationary point. This triggers a new recruit election so that a new robot eventually joins the primary leader team as shown in Figures 10(c) through 10(e)) assisting the leader to reach its associated target (Figure 10(f)).

Note that in order to service all the announced targets, multi-hop paths have been created from the leaf nodes of the tree network to the APs, that are due to the robots' restricted communication capabilities. In Figure 11, the minimum inter-robot distance at each iteration is presented, which is always greater than 0, implying collision-free robot trajectories. Also, in Figure 12, the quantity $r^{out}_{\mathbf{x},i} - r^{in}_{\mathbf{x},i}$ for all robots is illustrated showing that an acceptable quality of communication among the robots is maintained. Finally, the convergence of the Lagrange multipliers is depicted in Figure 13.

Next we consider a network of $N = 12$ robots and $K = 2$ APs residing in a complex environment whose convex hull has diameter equal to 17 units while the parameters l and u in (27) are selected to be 2.7 and 3.4 units, respectively. The two APs are located at $\mathbf{x}_{13} = [1, 4]$ and $\mathbf{x}_{14} = [1.5, 8]$ and the robots are initially deployed in the left part of the environment $\mathcal{W}$ between the two APs. Figures 14(a) through 14(d) show that a multi-hop communication path is established between leaf nodes and APs when a new task is serviced, in a similar way as in the previous simulation study. The evolution of the communication constraints $c_i = r^{out}_{\mathbf{x},i} - r^{in}_{\mathbf{x},i}$ over time is depicted in Figure 15 showing that robots are able to maintain network integrity, as defined in equation (5).



(a) Time $k = 3250$ (b) Time $k = 3600$

(c) Time $k = 4500$ (d) Time $k = 4950$

(e) Time $k = 5500$ (f) Time $k = 5850$

Fig. 10. Simulation Study I: Evolution of the communication network when the third target (blue cross) is announced. Figures 10(a) through 10(f) show the evolution of the system at different time instants. Green and red lines represent the communication links among the robots. Red lines depict the constructed tree network across the edges of which robots in the secondary leader team move as described in Section V-B; see Figures 10(a)-10(d). Their thickness depends on the value of $T_{ij}R(\mathbf{x}_i, \mathbf{x}_j)$, i.e., thicker lines capture higher values. The blue rhombus represents the AP, the yellow squares illustrate a junction node, the red star stands for the leader and the orange dots for the rest members of the primary leader team. The redundant robots are depicted by black dots and the green rhombus represents a leaf node, i.e., a robot that services a target.

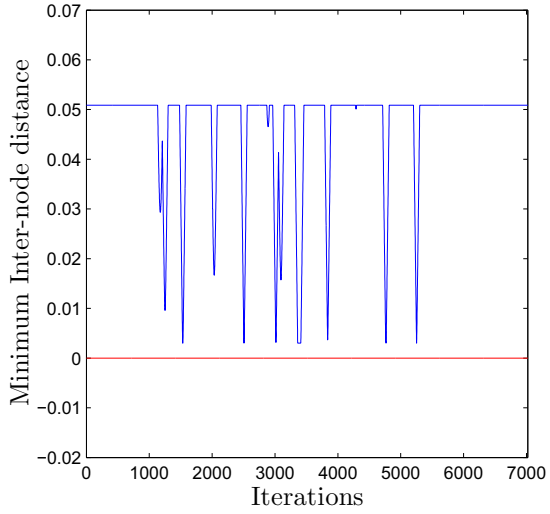


Fig. 11. Simulation Study I: Graphical depiction of the minimum inter-node distance during network evolution. Collision avoidance constraints are satisfied as the robots move.

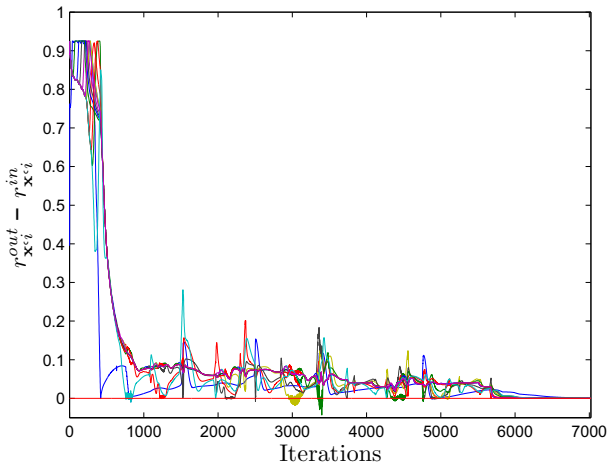


Fig. 12. Simulation Study I: Graphical depiction of the difference $r^{out}_{\mathbf{x},i} - r^{in}_{\mathbf{x},i}$ for all robots of the network.

VII. Conclusions

In this paper, we addressed the problem of servicing a collection of tasks in complex environments by a mobile robot network, while ensuring end-to-end connectivity with a fixed infrastructure of access points. Tasks were associated with specific locations in the environment, they were announced sequentially, and they were not assigned *a priori* to any robots. Communication with the access points was modeled through a routing model where the communication links captured the rate of information that can be transmitted between two nodes. A distributed, hybrid control scheme was proposed that dynamically grew tree networks, rooted at the access points, with branches that connect dedicated leaf nodes that serviced individual tasks to the main network structure. To achieve this goal, the robots switched between different roles that were related to their functionality in the network which, along with the communication optimization and motion planning, gave rise to the proposed distributed hybrid system. Construction of tree networks along with an appropriate selection of branch junctions resulted in an efficient use of the available robots. Our proposed scheme achieves global planning by construction verified also by numerical simulations.

Appendix A
Computation of free-space $\mathcal{W}_f$

The construction of the free-space $\mathcal{W}_f$ is presented in Algorithm 6. Initially, the free-space is assumed to be the whole workspace $\mathcal{W}$ [line 1]. Then, in lines 2-6 of the algorithm, the points $\mathbf{q} \in \mathcal{W}$ whose distance from each edge e of the boundary $\partial\mathcal{W}$ is less than or equal to some small number $\rho > 0$ are removed from the free-space, so that $\mathcal{W}_f$ is finally obtained. To achieve this, we first define the half-space determined by the e-th edge of $\partial\mathcal{W}$ and pointing inside $\mathcal{W}$, as $\mathcal{H}^e = \left\{\mathbf{q} \in \mathbb{R}^2 \mid \mathbf{a}_e^T\mathbf{q} \leq b_e\right\}$.

Then, we translate the half-space $\mathcal{H}^e$ in oder to discard the points $\mathbf{q} \in \mathcal{W}$ for which it holds that $\|\mathbf{q} - \mathbf{q}_b\| \leq$

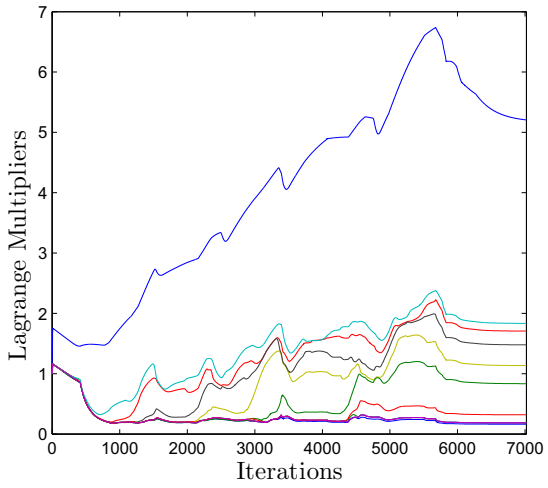


Fig. 13. Simulation Study I: Evolution of Lagrange multipliers λ_i for all robots of the network.

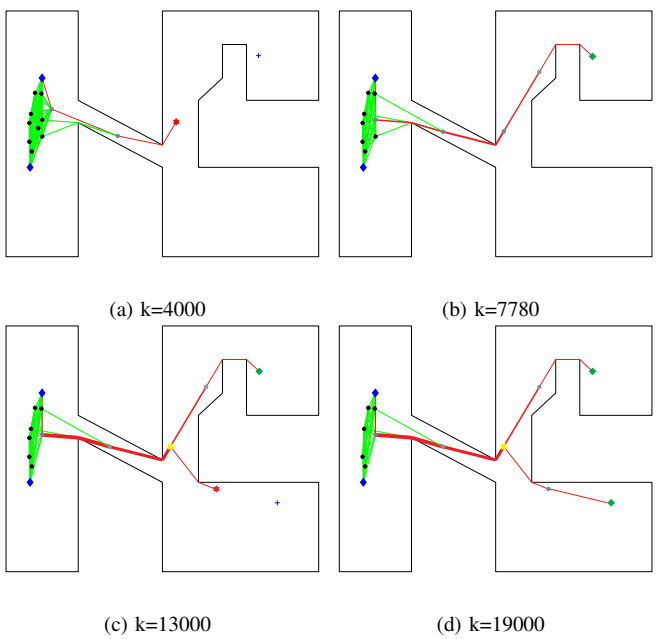


Fig. 14. Simulation Study II: Figures 14(a)-14(b) and 14(c)-14(d) show the network configuration until the first and the second task are serviced, respectively. Green and red lines represent the communication links among the robots. Red lines depict the constructed tree network across the edges of which robots in the secondary leader team move as described in Section V-B. Their thickness depends on the value of $T_{ij}R(\mathbf{x}_i, \mathbf{x}_j)$, i.e., thicker lines capture higher values. The blue rhombus represents the AP, the yellow squares illustrate a junction node, the red star stands for the leader and the orange dots for the rest members of the primary leader team. The redundant robots are depicted by black dots and the green rhombus represents a leaf node, i.e., a robot that services a target.

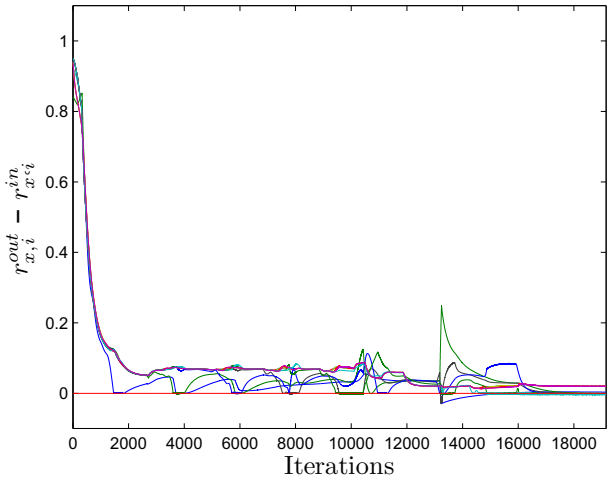
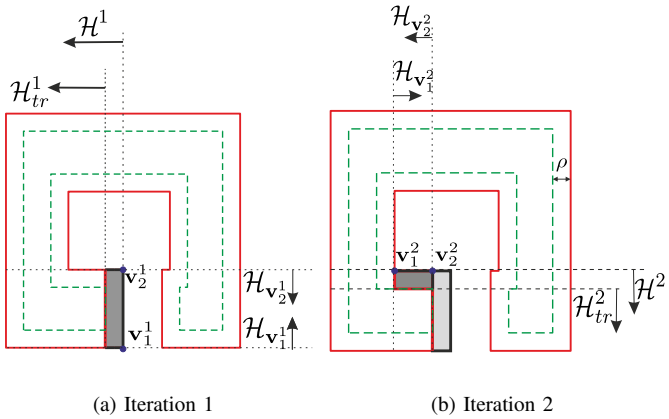


Fig. 15. Simulation Study II: Graphical depiction of the difference $r^{out}_{\mathbf{x},i} - r^{in}_{\mathbf{x},i}$ for all robots of the network.



(a) Iteration 1 (b) Iteration 2

Fig. 16. An illustrative explanation of Algorithm 6. The first 2 iterations of Algorithm 6 are shown in Figures 16(a) and 16(b). The red polygonal line stands for the boundary of the free-space $\mathcal{W}_f$ at the end of every iteration. The dark gray area represents the region $\mathcal{W}^e_o$ that is subtracted from $\mathcal{W}_f$ at the current iteration, while the light gray area is the region that was subtracted from $\mathcal{W}_f$ at previous iterations. The green dashed line depicts the boundary of the resulting $\mathcal{W}_f$ when Algorithm 6 terminates.

ρ, $\mathbf{q}_b \in \partial\mathcal{W}$. The translated half-space is defined as $\mathcal{H}^e_{tr} = \left\{\mathbf{q} \in \mathbb{R}^2 \mid \mathbf{a}_e^T\mathbf{q} \leq b_e + \rho_e\right\}$, where ρ_e is a constant used to translate the half-space $\mathcal{H}^e$ so that for every edge e of $\partial\mathcal{W}$ the distance between the lines $\mathbf{a}_e^T\mathbf{q} = b_e$ and $\mathbf{a}_e^T\mathbf{q} = b_e + \rho_e$ is equal to $\rho > 0$, $\forall e$. A graphical example of these half-spaces is depicted in Figure 16.

Next, we denote by $\mathbf{v}^e_1$ and $\mathbf{v}^e_2$ the vertices of the polygonal boundary $\partial\mathcal{W}_f$ that constitute the endpoints of the e-th edge of $\partial\mathcal{W}_f$. With slight abuse of notation, we denote by $\mathcal{H}_{\mathbf{v}^e_1}$ the half-space determined by the edge of $\partial\mathcal{W}_f$ that is incident to the point $\mathbf{v}^e_1$ and does not pass through $\mathbf{v}^e_2$ and points towards $\mathbf{v}^e_2$. Accordingly, we define the half-space $\mathcal{H}_{\mathbf{v}^e_2}$. Such half-spaces are illustrated in Figure 16. Then, we exclude from the free-space $\mathcal{W}_f$ the points $\mathbf{q} \in \mathcal{W}^e_o$ [lines 4-5] defined as $\mathcal{W}^e_o = \mathcal{H}^e \cap (\mathbb{R}^2 \setminus \mathcal{H}^e_{tr}) \cap \mathcal{H}_{\mathbf{v}^e_1} \cap \mathcal{H}_{\mathbf{v}^e_2}$.

An illustrative explanation of Algorithm 6 is depicted in Figure 16. Note that the application of Algorithm 6 results in excluding all points $\mathbf{q} \in \mathcal{W}$ that are arbitrarily close to the polygonal boundary $\partial\mathcal{W}$ through controlling the parameter $\rho > 0$.

REFERENCES

[1] J. Cortés, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[2] S. He, J. Chen, and Y. Sun, "Coverage and connectivity in duty-cycled wireless sensor networks for event monitoring," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 475–482, 2012.

Algorithm 6 Computation of the free-space $\mathcal{W}_f$

Require: $\mathcal{W}$ and $\rho > 0$;

$\mathcal{W}_f := \mathcal{W}$;
for $e = 1 : E$ **do**
  Compute the half-spaces $\mathcal{H}^e$, $\mathcal{H}^e_{tr}$, $\mathcal{H}_{\mathbf{v}_1^e}$ and $\mathcal{H}_{\mathbf{v}_2^e}$;
  $\mathcal{W}_o^e = \mathcal{H}^e \cap (\mathbb{R}^2 \setminus \mathcal{H}^e_{tr}) \cap \mathcal{H}_{\mathbf{v}_1^e} \cap \mathcal{H}_{\mathbf{v}_2^e}$;
  $\mathcal{W}_f := \mathcal{W}_f \setminus \mathcal{W}_o^e$;
end for

[3] I. Nourbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion, "Human-robot teaming for search and rescue," *IEEE Pervasive Computing*, vol. 4, no. 1, pp. 72–79, 2005.

[4] Y. Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph laplacian," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 116–120, 2006.

[5] M. C. DeGennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *45th IEEE Conference on Decision and Control*, San Diego, CA, USA, December 2006, pp. 3628–3633.

[6] M. M. Zavlanos and G. J. Pappas, "Potential fields for maintaining connectivity of mobile networks," *IEEE Transactions on Robotics,*, vol. 23, no. 4, pp. 812–816, 2007.

[7] M. Zavlanos and G. Pappas, "Distributed connectivity control of mobile networks," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1416–1428, 2008.

[8] G. Notarstefano, K. Savla, F. Bullo, and A. Jadbabaie, "Maintaining limited-range connectivity among second-order agents," in *IEEE American Control Conference*, Minneapolis, MN,USA, June 2006, pp. 6–pp.

[9] M. Ji and M. B. Egerstedt, "Distributed coordination control of multiagent systems while preserving connectedness." *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693–703, August 2007.

[10] M. Zavlanos, M. Egerstedt, and G. Pappas, "Graph theoretic connectivity control of mobile robot networks," *Proc. of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[11] M. M. Zavlanos, A. Ribeiro, and G. J. Pappas, "Network integrity in mobile robotic networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 3–18, 2013.

[12] ——, "Mobility & routing control in networks of robots," in *49th IEEE Conference on Decision and Control*, Atlanta, GA, USA, December 2010, pp. 7545–7550.

[13] J. Fink, A. Ribeiro, V. Kumar, and B. M. Sadler, "Optimal robust multihop routing for wireless networks of mobile micro autonomous systems," in *IEEE Military Communications Conference, (MILCOM)*, San Jose, CA, November 2010, pp. 1268–1273.

[14] M. Lindhé and K. H. Johansson, "Adaptive exploitation of multipath fading for mobile sensors," in *IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, May 2010, pp. 1934–1939.

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

[16] M. Malmirchegini and Y. Mostofi, "On the spatial predictability of communication channels," *IEEE Transactions on Wireless Communications*, vol. 11, no. 3, pp. 964–978, 2012.

[17] A. Ghaffarkhah and Y. Mostofi, "Channel learning and communication-aware motion planning in mobile networks," in *IEEE American Control Conference*, Baltimore, Maryland, USA, 2010, pp. 5413–5420.

[18] J. Le Ny, A. Ribeiro, and G. J. Pappas, "Adaptive communication-constrained deployment of unmanned vehicle systems," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 923–934, 2012.

[19] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

[20] D. Panagou, M. Turpin, and V. Kumar, "Decentralized goal assignment and trajectory generation in multi-robot networks: A multiple lyapunov functions approach," *arXiv preprint arXiv:1402.3735*, 2014.

[21] M. M. Zavlanos and G. J. Pappas, "Dynamic assignment in distributed motion planning with local coordination," *Robotics, IEEE Transactions on*, vol. 24, no. 1, pp. 232–242, 2008.

[22] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of aerial robots." in *Robotics: Science and Systems*, 2013.

[23] ——, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Autonomous Robots*, vol. 37, no. 4, pp. 401–415, 2014.

[24] S. G. Loizou and K. J. Kyriakopoulos, "Navigation of multiple kinematically constrained robots," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 221–231, 2008.

[25] J. Fink, A. Ribeiro, and V. Kumar, "Robust control for mobility and wireless communication in cyber-physical systems with application to robot teams," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 164–178, 2012.

[26] J. Stephan, J. Fink, B. Charrow, A. Ribeiro, and V. Kumar, "Robust routing and multi-confirmation transmission protocol for connectivity management of mobile robotic teams," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, September 2014, pp. 3753–3760.

[27] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, vol. 2, San Fransisco, CA, USA, April 2000, pp. 995–1001.

[28] A. Ruszczynski, *Nonlinear optimization*. Princeton University Press, 2011, vol. 13.

[29] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[30] N. Chatzipanagiotis, D. Dentcheva, and M. M. Zavlanos, "An augmented lagrangian method for distributed optimization," *Mathematical Programming, Ser.A*, vol. 152, no. 1-2, pp. 405–434, August 2015.

[31] F. Aurenhammer and R. Klein, *Handbook of Computational Geometry*. Elsevier Publishing House, 1999, ch. 5: *Voronoi Diagrams*, pp. 201–290.

[32] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

Yiannis Kantaros received the Diploma in Electrical and Computer Engineering in 2012 from the University of Patras, Patras, Greece. He is currently working toward the Ph.D. degree in the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC, USA. His current research interests include distributed control, distributed optimization, multi-agent systems and robotics. He received the Best Student Paper Award at the 2nd IEEE Global Conference on Signal and Information Processing in 2014.

Michael M. Zavlanos received the Diploma in Mechanical Engineering from the National Technical University of Athens (NTUA), Athens, Greece, in 2002, and the M.S.E. and Ph.D. degrees in Electrical and Systems Engineering from the University of Pennsylvania, Philadelphia, PA, in 2005 and 2008, respectively.

From 2008 to 2009 he was a Post-Doctoral Researcher in the Department of Electrical and Systems Engineering at the University of Pennsylvania, Philadelphia. He then joined the Stevens Institute of Technology, Hoboken, NJ, as an Assistant Professor of Mechanical Engineering, where he remained until 2012. Currently, he is an assistant professor of mechanical engineering and materials science at Duke University, Durham, NC. He also holds a secondary appointment in the department of electrical and computer engineering. His research interests include a wide range of topics in the emerging discipline of networked systems, with applications in robotic, sensor, communication, and biomolecular networks. He is particularly interested in hybrid solution techniques, on the interface of control theory, distributed optimization, estimation, and networking.

He is a recipient of the 2014 Office of Naval Research Young Investigator Program (YIP) Award and the 2011 National Science Foundation Faculty Early Career Development (CAREER) Award. He was also a finalist for the best student paper award at CDC 2006.