J Intell Robot Syst
DOI 10.1007/s10846-017-0471-3

Multi-robot Deployment using Topological Maps

Reza Javanmard Alitappeh · Guilherme A. S. Pereira · Arthur R. Araújo · Luciano C. A. Pimenta

Received: 12 May 2016 / Accepted: 5 January 2017

Abstract This paper proposes an efficient and distributed deployment strategy to optimally distribute teams of robots in environments that can be represented by topological maps. Among the several applications of our solution are sensing and coverage of large corridor-based buildings, such as hospitals and schools, and the optimal placement of service vehicles in the streets of a big city. The representation of the environment as a topological map transforms the original two or three-dimensional problem into a one-dimensional, simplified problem, thus reducing the computational cost of the solution. Moreover, each robot can reach its final position by simply following a sequence of intuitive, human-like commands, without the need for global metric localization, which also simplifies robot control. Besides presenting convergence proofs for our method, the paper also presents simulated and real world experiments that illustrate and validate our approach.

G. A. S. Pereira · A. R. Araújo · L. C. A. Pimenta (✉)
School of Engineering, Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, MG 31270-901, Brazil
e-mail: lucpim@cpdee.ufmg.br

R. J. Alitappeh
Electrical and Computer Engineering Department, University of Science and Technology of Mazandaran, Behshahr, Mazandaran, Iran
e-mail: Rezajavanmard64@gmail.com

Keywords Multi-robot deployment · Topological map · Multi-robot coverage

1 Introduction

Among the different research topics in the area of cooperative and distributed robotic systems, the *multi-robot deployment problem* appears as one of the most studied ones [5]. In fact, distributing a team of mobile robots in a known workspace is one of the fundamental tasks in multi-robot systems. Applications for robot deployment include sensing and coverage, where the robots are distributed in an environment and respond to the events that occur in their associated region. An example of deploying two robots by using the method proposed in this work is shown in Fig. 1. Our method does not require a precise map of the environment nor a metric localization system. Robots are distributed based on a topological representation of the input map. In Fig. 1, the different colors and patterns indicate their associated regions.

A well known distributed technique for multi-robot deployment is the one proposed by Cortes et al. [8] based on Lloyd's framework [17]. In that work, robots were deployed over a convex environment based on a continuous control law that minimized a functional representing the quality of the deployment. A density function represented the priority of points inside a region, so that higher priority points must be better covered. Coverage in the sense of [8] is illustrated

Published online: 26 January 2017

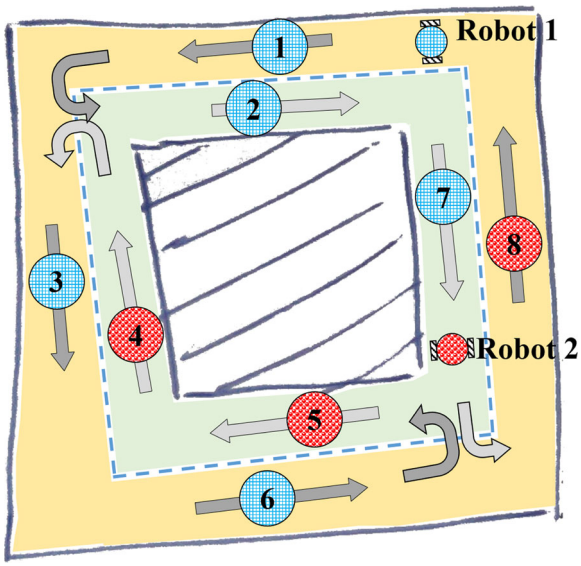


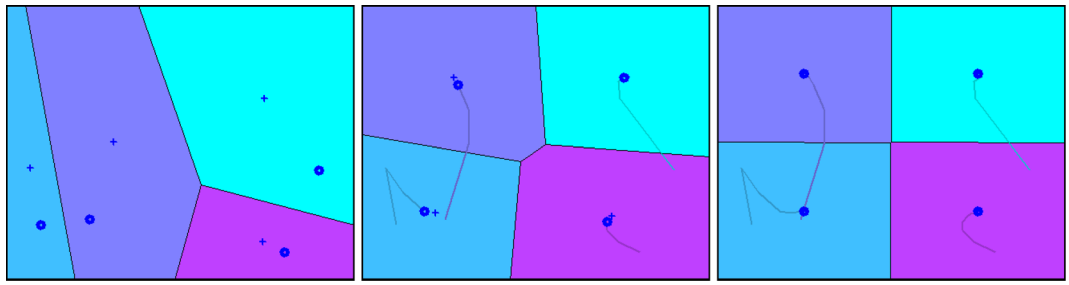
Fig. 1 An example of proposed methodology on a non-precise drawn map with an obstacle (hatched area) at the center; its topological representation, where arrows indicate the possible direction of movement; and the result of partitioning the workspace based on the proposed method is also illustrated

in Fig. 2. In this figure four robots are deployed in a convex environment. The density function in this case is uniform, so all points of the environment have identical priorities. At the beginning, the environment is divided in four Voronoi regions,[1] one for each robot. The technique proposed in [8] generates velocity vectors for each robot so that they continuously move towards the centroids of their respective Voronoi regions, which, as a consequence, change the region itself, as shown in Fig. 2. As the time evolves, it can be proved that the multi-robot system converges to a configuration where the environment is equally distributed among the robots or, in the case of non-uniform density functions, is distributed according to this function. It is important to notice that, even in convex workspaces, real world implementations of this methodology require global and precise metric localization for the robots and non-linear control strategies to follow the velocity vectors.

Different extensions for this strategy were developed by several authors. These extensions can be classified according to different aspects. As shown in Table 1, several previous solutions encountered in the literature can work in convex or non-convex environment, be distributed or centralized, work in continuous or discrete setup, and finally, be based on geometric or grid maps.

The main contribution of this work is the development of a novel, efficient and decentralized framework with mathematical guaranteed convergence to allow the deployment of multi-robot systems without the need of precise metric maps and precise localization. Our method can be used when the environment model is given by partially known or even manually sketched maps. To the best of the authors knowledge, the proposed framework is the only one with all these features. Our work is based on the use of graph based topological maps to model the robot's workspace. Independently of the dimension of the original workspace, this strategy transforms the problem into a one dimensional problem, what highly increases the computational efficiency of the method, thus allowing for the deployment of large teams of robots in very extensive workspaces. Due to the use of a topological map, in our approach the robots may be controlled using a sequence of human like commands, such as "turn right", "turn left" and "move straight". Furthermore, no global metric localization is required. Our approach was designed to be applied in structured, usually non-convex workspaces, that are suitable for topological mapping. These include metropolitan regions composed of streets and intersections, pipelines and energy distribution systems with several connections and bifurcations, and large buildings with intersecting corridors. Since in many applications the robotic group cannot be controlled by a centralized system, our method is fully distributed in a way each robot needs to communicate only with its neighbors. Moreover, as we will make it clear in the rest of the paper, our method is more efficient in terms of computational complexity and also in terms of the required communication bandwidth when compared to the most similar strategies found in the literature [12, 30]. Also, the proposed strategy is provably correct in the sense that it is guaranteed that the robot positions converge to the optimal locations in the topological map. This is also an important characteristic of this work since some of the other works found in the literature, for example [2, 3], and [14], do not provide such a guarantee for discrete algorithms. As a drawback, it is important to mention that, the locations in topological map for where the robots converge

[1] A Voronoi region for a robot is composed of all points closer to that robot than to any other robot.



(a) Time 1. (b) Time 4. (c) Final.

Fig. 2 Deployment of 4 robots in a convex environment using the approach proposed by [8]. In (**a**) and (**b**) the "+" sign indicates the center of the Voronoi regions, which represent the current goal position for each robot. Voronoi regions are highlighted with different colors

do not necessarily correspond to optimal positions in the real workspace, which makes the quality of the deployment dependent on the map discretization.

Before we present our methodology, next section will present a brief review of the multi-robot deployment area. The rest of the paper is divided as follows: problem statement is in Section 3; Section 4 is dedicated to present the proposed methodology, including the discretization technique, the topological map model, the methodology itself and the convergence proofs; Simulations and actual robot experiments are in Section 5; Finally, in Section 6 we conclude the paper and present some ideas for future research.

Table 1 Different classes of deployment approaches (abbreviation are used for convex "Conv." (non-convex, "n-Conv. "), discrete, "Dis." continuous "Cont." and heterogeneous, "Hetr.")

Methodology	workspace	setup	Hetr.	map
2004, [8]	Conv.	Con.	-	geometric
2008, [22]	n-Conv.	Con.	Hetr.	geometric
2009, [25]	Conv.	Con.	-	geometric
2010, [4]	n-Conv.	Con.	-	geometric
2011, [29]	Conv.	Con.	Hetr.	geometric
2012, [18]	n-Conv.	Con.	-	geometric
2012, [12]	n-Conv.	Dis.	Hetr.*	grid
2013, [2, 3]	n-Conv.	Con.*	-	geometric
2014, [30]	n-Conv.	Dis.	-	grid
2014, [14]	n-Conv.	Con.*	-	grid
2014, [27]	Conv.	Con.	Hetr.	geometric
2015, [21, 28]	Conv.	Con.	Hetr.	geometric
This paper	n-Conv.	Dis.	Hetr.	topologic

Con.* : A continuous setup with discrete approximation
Hetr.* : The method has the potential to work in Heterogeneous systems.

2 Literature Review

This section will survey the main works in the area of multi-robot deployment. Table 1 shows the works in a chronological order and also shows some of the main features of the discussed methods.

After the initial work by Cortes et al. [8] in the application of a locational optimization based framework in multi-robot deployment considering convex and static environments, several other works proposed different modifications of the framework to tackle different scenarios. The problem of considering time-varying distribution density function was studied in [23] and [16]. The authors of [25] developed an adaptive controller, such that the robots learn the distribution of sensory information (density function) during the deployment. Adaptive controllers were also proposed in [20] and [21] to consider heterogeneous sensing performance and actuator performance, respectively.

Multi-robot deployment with the use of multiplicative weighted Voronoi (MW-Voronoi) diagram in an obstacle free environment is discussed in [18]. In this work, the authors consider different weights for each robot in the process of constructing the Voronoi tessellation. Moreover, in the presence of obstacles, the tessellation is done by applying a visibility-aware multiplicatively weighted Voronoi (VMW-Voronoi) diagram. Therefore, it is possible to have an uncovered region, where it is not only invisible from the robots viewpoint, but also will not be considered in the Voronoi tessellation. Communication delay and sensors effectiveness variation is addressed in [27]. In this work a new partitioning technique is developed in order to address variation in sensors behavior, which

is called Guaranteed Multiplicative Weighted (GMW) Voronoi. Agents with different types of dynamics are taken into account by the same authors in [28]. They also used MW-Voronoi partitioning approach to find the corresponding region for each robot.

Pimenta et al. [22] used geodesic metric for deploying a team of heterogeneous robots in non-convex environments. Authors in [6] and [7] considered non-convex environments by constructing a diffeomorphism to convex regions with isolated obstacles in its interior, in which regular Voronoi coverage can be applied. As they mentioned, besides significant computational challenges, the generated solution may differ from the corresponding optimal coverage solution in the original space. They addressed these shortages in [6] by characterizing a set of stationary points for the Lloyd's algorithm in general regions. Another approach for deploying multiple robots in non-convex environments was presented in [4]. In order to avoid collisions with obstacles, the authors combined the deployment control law with a local planner (Tangent Bug).

While the approaches cited above focused on environments with two or three dimensions represented by geometric maps, in this paper a discrete, one-dimensional topological map is used to represent the robot's workspace. As it is usual, the topological map is represented by a graph, in our case a directed graph. Previous works have already considered multi-robot deployment in discrete spaces. For instance, in [12] the authors represent the environment using grid cells and a discrete coverage optimization algorithm is applied on robots with short-range communication. In [2] and [3] Bhattacharya et al. tried to approximate the continuous control setup with a discrete version. They discretized a non-convex environment and represented it as a graph. A standard graph search algorithm was employed in order to compute the control law. A safe deployment problem was considered by Javanmard and Pimenta in [14], where a new metric based on Generalized Voronoi Diagrams (GVD) (called Geodesic GVD) yields a safe motion for a team of robots. In their implementation, a discrete approximation with graph representation is also used.

Among all previous approaches surveyed, the most similar to the one presented in this paper is [30]. The authors computed the Voronoi partitions upon an undirected graph that topologically encodes the environment. However, the authors still use the original 2D metric map to control the robots, what, similarly to all other previous strategies, makes the method dependent on precise distance and localization. The approach proposed in the present paper improves on that point once all steps of the method execute on the topological map. This highly simplifies the actual robot implementation, as will be clear in the rest of the text. Moreover, as we are going to clarify in Section 4, in comparison to other similar methods, [30] and [12], the proposed algorithm is more efficient in terms of running time (computational complexity) and also in the amount of data exchanged by the robots. Finally, in contrast to other works such as [2, 3], and [14] we do show convergence guarantees valid for the presented discrete setup.

The main idea behind our methodology was inspired by [1], where a robot uses human-like commands to move in structured environments. In fact, humans do not need to have a precise metric localization to reach a destination. A simple sequence of directions such as "turn right", "turn left", and "go straight" may be enough for a person to reach its destination in an urban environment or office building. Our methodology uses graph search strategies to generate such sequences of commands and deploy the robot team in the environment. This can be considered as a milestone in the literature of deployment. In such a way, we do not need a specific metric for the computation (as for the humans, metric localization is also not necessary). Given a robot that is able to move without the need of localization (by following walls or driveways, for example), high speed motion and fast response can be achievable. Arguably, this method is suitable for emergency responses like chasing an evader or responding to accidents. It is important to remark that the proposed technique does not need a complete and precise map as input, since it can even be applied on partially known or even manually sketched maps (see Fig. 1). In the next section we precisely define the problem considered in this paper.

3 Problem Statements

Given an environment and a team of mobile robots we want to distribute this team in the environment in such a way that *events of interest* over the environment can be efficiently handled by the robots. We assume

that the environment is partially known and may be roughly represented by a non-proportional drawing, a sketch without metric information or even a photo (an example is shown in Fig. 3). Thus, the objective of this work is to present solutions to the two problems given below:

Problem 1 (Discrete topological map representation) Let $E \subset \mathbb{R}^n$ be an environment composed of corridors or driveways connected through intersections, which are the regions where the corridors or driveways cross each other (see for example Fig. 3). Assume a team of n mobile robots equipped with very simple sensors and reactive controllers so that they are only able to identify intersections, to move along corridors or driveways, and to perform turning motions. *Provide an efficient discrete data structure $\mathcal{G}$ to encode the environment without the use of any type of metric information so that this limited group of robots can use it to navigate.*

Problem 2 (Optimal distribution of robots) Consider the team of n mobile robots as described in Problem 1 with access to the data structure provided by the solution of Problem 1. Consider also that the robots have knowledge of a density function $\phi : E \rightarrow \mathbb{R}^+$ that defines the relative importance of locations defined in the environment. Regions with higher values of density function are more likely to have events of interest in its interior. *Provide a deployment strategy, with guaranteed convergence, that is able to optimally distribute the team without the need of metric information and precise sensors for localization.*

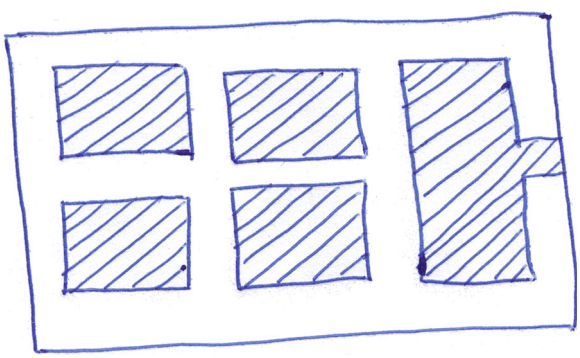


Fig. 3 A sketched map of a building containing rooms (hatched polygons) corridors and intersections, where robots can move (white spaces). Our methodology can make use of simple, non-scaled drawing like this to deploy the robots over the environment

In the next section we present solutions to the stated problems.

4 Methodology

In this section, we present a map representation that solves Problem 1 and a new distributed deployment algorithm, along with a proof of convergence, that solves Problem 2.

4.1 Topological Map Representation

In this paper, a *Graph* is used as the data structure to represent the environment as a topological, one dimensional map without the need of metric information. Thus, we obtain a topological representation of an elementary representation (i.e. sketch, drawing, or photo) of a bounded environment E by defining a mapping $E \rightarrow \mathcal{G}$, that converts the environment into a graph space. In particular, for the sake of simpler presentation and easier implementation, this work focuses on block shaped environments, such as the floors of an office building or neighborhoods of a city. Although this kind of environment cover several applications, our theoretical methodology can be easily generalized and used in other types of environments that can be represented by a topological map, specially the ones composed of corridors and intersections. For environments different from the ones illustrated in this paper, the main modifications that must be made are related to the robot command set and the robot perception system, as will become clear along the rest of the text.

Our weighted directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{C}, \mathcal{I})$ is defined by a set of m nodes $\mathcal{V}$, connected by directed edges $\mathcal{E}$ with specific costs $\mathcal{C}$, and robot commands $\mathcal{I}$. In this way, an edge $e \in \mathcal{E}$ denotes the link between two nodes, $e = \overrightarrow{xy}$ $(x, y \in \mathcal{V})$ and, $c(e) \in \mathcal{C}$ indicates the cost of robot motion given by $I(e) \in \mathcal{I}$ between x and y. Furthermore, $\mathcal{N}_{\mathcal{G}}(x)$ indicates the set of the neighbor nodes of x: $\mathcal{N}_{\mathcal{G}}(x) = \{y \in \mathcal{V} \mid \overrightarrow{xy} \in \mathcal{E}\}$.

To construct the graph $\mathcal{G}$, differently from previous methods ([12, 30]) that discretized the environment into grid cells, we model a full specific region, for example, a corridor or street, as a cell. Since we do not consider a precise metric, the size of those cells are not necessarily equal, although we assume in this paper that the cells are about the same size. Figure 4a

illustrates the division of the sketched map of Fig. 3 in several cells.

To map real world problems where the direction of movements in some regions are constrained, such as one-way and two-way streets in the big cities, we associate to each cell of our map a direction of movement. Regions that allow two-way movements are subdivided into two new regions. Each of these regions is then associated to a node in $\mathcal{V}$. Figure 4b shows an example of this process. In this case, all regions of the graph are bi-directional. Therefore, each region has been separated into two regions, thus generating two nodes in the graph. The corresponding graph for this example is shown in Fig. 5. Notice that the edges of this graph represent possible movements among the nodes. If a robot can move from a node to another, we assume that these nodes are neighbors and add a corresponding edge to $\mathcal{E}$.

Given the graph $\mathcal{G}$, a cost between two neighbor nodes $c(x, y) \in \mathcal{C}$, $x, y \in \mathcal{V}$ and a command $I(x, y) \in \mathcal{I}$ are assigned to the edge connecting x and y ($\overrightarrow{xy}$). This means that, to go from node x to y, a robot must execute a command $I(x, y)$ that will result in a cost $c(x, y)$, where $c : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$. Below are some properties assumed for c:

- $c(x, x) = 0$,
- $c(x, y) \geq 0$,
- $c(x, y) \leq c(x, z) + c(z, y)$,
- The graph might be asymmetric, *i.e.*, $c(x, y) \neq c(y, x)$.

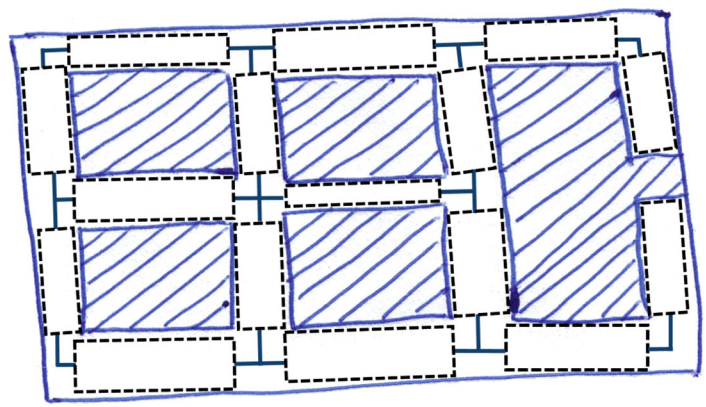
Given this, *"Path"*, *"Commands"* and cost "*d*" between two arbitrary nodes (x and z) are described respectively as:

$$Path(x, z) = \{x, \cdots, y, \cdots, z\}, \quad x, y, z \in \mathcal{V},$$

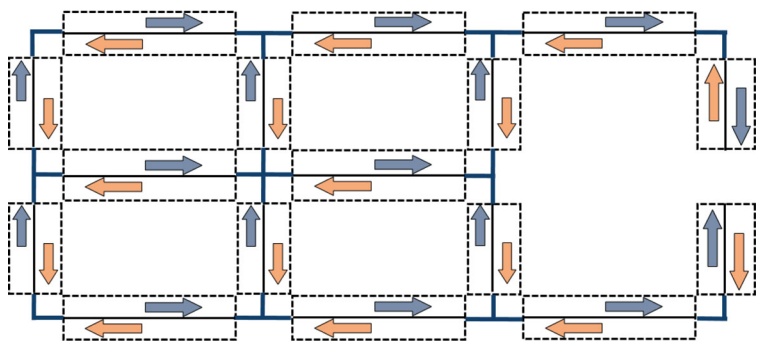
$$Commands(x, z) = \{I(x, p), \cdots, I(y, q), \cdots, I(w, z)\}$$

$$x, p, y, q, w, z \in \mathcal{V}, \text{ and } \overrightarrow{xp}, \overrightarrow{yq}, \overrightarrow{wz} \in \mathcal{E}$$

$$d(x, z) = c(x, p) + \cdots + c(y, q) + \cdots + c(w, z).$$



(a) Map discretization. Each region between two intersections is considered to be a cell.



(b) Depending on the allowed direction of movement, each original cell may be divided into two other cells. The arrows indicate the allowed motion in each region.

Fig. 4 Discretization and definition of directions on the sketch map of Fig. 3

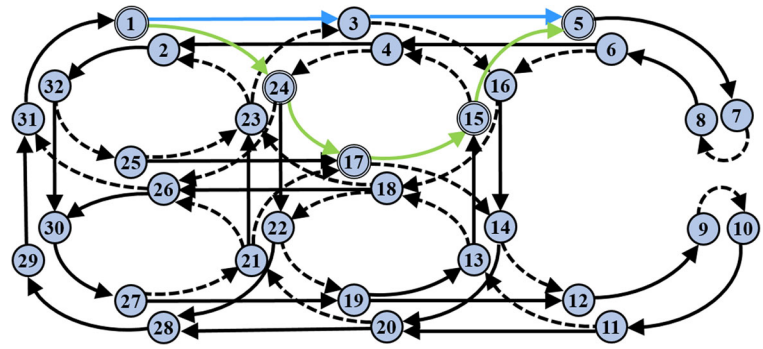


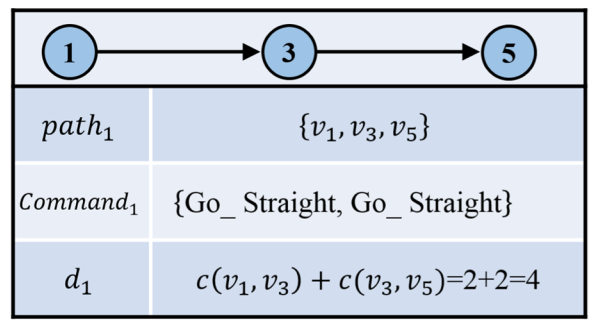
Fig. 5 The corresponding graph of the map in Fig. 4 and two possible paths from node 1 to 5 with different colors

The command set is induced based on real world vehicle or human motions. An example of human-like command set is {*Turn_left*, *Turn_right*, *Go_Straight*, *Turn_Back*}, with costs $c(x, y) \in \{4, 4, 2, 8\}$. In this example, the cost of making a turn to right or left is higher than the cost of going straight, which is realistic for several robots. Moreover, the command "turn_back" costs the maximum value for a robot. Function $d(x, z)$ is then a cost function that denotes the sum of the costs over the path from node x to y.

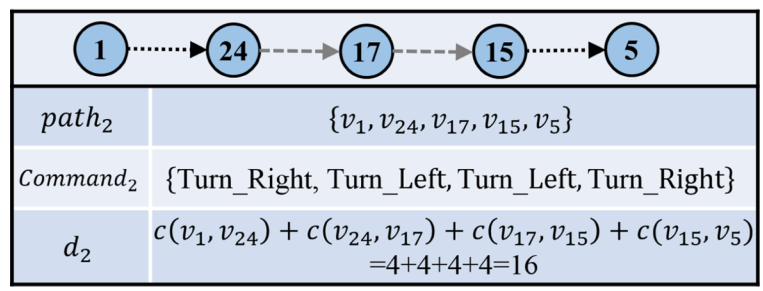
As an example, in Fig. 5 at least two paths between the left uppermost region (node 1) and the right uppermost region (node 5) exist. As is highlighted in this figure using blue and green, these paths are: {1, 3, 5} and {1, 24, 17, 15, 5}. Figure 6 presents the corresponding values of commands and costs for the two paths shown in Fig. 5. By relying on this weighting technique, the path with the smallest cost between two nodes can be found using the Dijkstra algorithm [10].

A density function might be defined over our topological map indicating cells that have priority to be serviced. Assuming a continuous density function over the original environment, this is done by assigning a higher number to the node corresponding to the center of the density function, and decreasing this value as we get far from this node (in terms of number of nodes). This function can be also defined based on

Fig. 6 Moving from a node to another one in our new representation scheme. And computing the distance between nodes



(a) A path between nodes 1 and 5.



(b) An alternative path between the same nodes.

the frequency of events of interest for each region or a global probability function. Since the density function states the priority of a region (node) to be serviced, regions with lower *events of interest* will receive a lower number. If we consider the map in Fig. 4 as an office, and the number of users as the density function, Fig. 7 indicates a function where the middle corridor has more priority. In practical applications, we can assume that an automatic system monitors the traffic of users in order to obtain the density function. In this way, the values in the nodes denote the density of users in the corridors (see Fig. 7).

Next section will show our decentralized solution for deploying robots in the environment modeled as a graph $\mathcal{G}$ with density function defined over it.

4.2 Multi-robot Deployment

Now, we show our solution to Problem 2. As previously stated, we assume a team of n mobile robots, $R = \{r_1, \cdots, r_n\}$, with access to the graph $\mathcal{G}$ and full knowledge of the density function $\phi : \mathcal{V} \to \mathbb{R}^+$. The graph node in which robot i is currently located is given by $p_i \in \mathcal{V}$, and $P = \{p_1, \cdots, p_n\}$.

Our strategy is based on the partition of the graph in such a way that, after the deployment, each robot will be responsible to respond only to the events that happen at the graph nodes assigned to that robot. Before showing our algorithm we first need to define the specific graph partitioning used in this work:

Definition 1 (Voronoi subgraph) : The Voronoi subgraph g_i in $\mathcal{G}$ is given by:

$$g_i = \{x \in \mathcal{V} \mid d(p_i, x) \leq d(p_j, x),\ \forall i \neq j\}, \qquad (1)$$

where, $d(x, y)$ is a function $d : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$ that denotes the cost of the shortest path between nodes x and y. If $d(p_i, x) = d(p_j, x)$, the node x is assigned to the robot with smaller index number as in [30].

The set of Voronoi subgraphs defines a Voronoi partitioning of the graph $\mathcal{G}$. In the partition every node will be assigned to a robot. Also the union of all subgraphs is equal to $\mathcal{V}$ and the intersection between two different subgraphs is empty.

Our approach builds upon the work in [30] in which the problem of optimally deploying the team of robots on a graph is treated as a locational optimization problem.

As in [30], we reformulate our general deployment problem as the one of minimizing the cost function:

$$\mathcal{H}(P, \mathcal{G}) = \sum_{i=1}^{n} \mathcal{H}_i(p_i, g_i)\,, \qquad (2)$$

where,

$$\mathcal{H}_i(p_i, g_i) = \sum_{q \in g_i} d(p_i, q)\phi(q). \qquad (3)$$

The discrete formulation given above is exactly the same formulation of the well known p-median problem [24], which is a NP-hard problem with several centralized previously proposed solutions. In this work we present a novel distributed solution which is used for multi-robot deployment in the same spirit of [30].

The main idea of our solution is to generate successive iterations in which the robots are relocated to different nodes in such a way $\mathcal{H}$ decreases until reaching convergence. In fact, these iterations consists of choosing a special node inside the robot subgraph and then moving the robot to this node.

Our solution is presented in the form of a distributed control algorithm in Algorithm 1, same to what is shown as Algorithm 1 in [30]. In fact, an

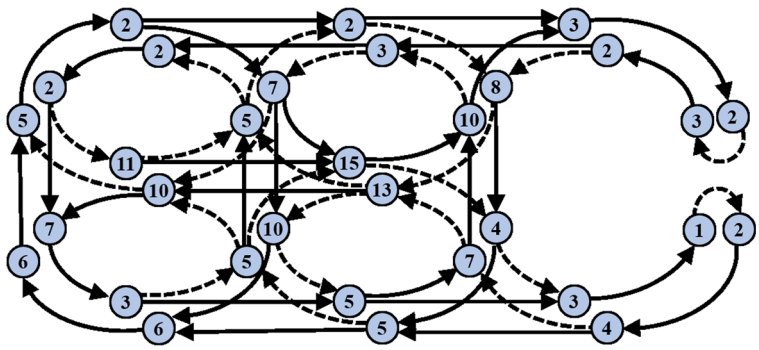


Fig. 7 The density function in this map represents the fact that the middle corridor deserves priority to be serviced. The numbers in the nodes indicate the density of users of the multi-robot system

important contribution of our work is the proposition of a more efficient algorithm (Algorithm 2) to find the next best node, where the robot should be relocated. Thus, our Algorithm 2 is used in the place of the one (Algorithm 2) defined in [30] for multi-robot deployment.

Algorithm 1 Distributed controller for robot i (same as [30]).

Input: $\mathcal{G}, p_i, p_i^*$ // where $\mathcal{G}$ is the graph, p_i is the robot location, and p_i^* is the next best node
State: Compute
foreach $j \in \mathcal{R}_i$ **do** //$\mathcal{R}_i \subseteq R$ is the set of neighbor robots of robot i
  $\mathcal{P}_i^* \leftarrow Receive\_Locations()$ // Receive locational information of neighbor robots (the current next best node of neighbor robots)
$g_i \leftarrow Compute\_Voronoi(\mathcal{G}, p_i^*, \mathcal{P}_i^*)$ // Compute Voronoi subgraph.
$p_i^* \leftarrow Find\_next\_node(g_i, p_i^*)$ // Call Algorithm 2.
if $(p_i \neq p_i^*)$ **then**
  State $\leftarrow$ **Moving** // Switch to **Moving State**.
State: Moving
$Move\_To(p_i^*)$ // Explained in Section 4.4.
if $(p_i = p_i^*)$ **then**
  State $\leftarrow$ **Compute** // Switch to **Compute State**.

In Algorithm 1, the robot is always in one of the two states; *Compute:* to compute the *next best node* (p_i^*); and *Moving:* to move to the node p_i^*. Before computing the next node, robot i must receive the current information from other robots called neighbors. A set of neighbors is defined by $\mathcal{R}_i = \{r_j \in R \mid \exists \overrightarrow{xy} \in \mathcal{E}, x \in g_i, y \in g_j, i \neq j\}$ ($\mathcal{R}_i \subseteq R$), which means robot r_j is a neighbor of robot r_i if they have vertices that are neighbors in graph $\mathcal{G}$; it should be noticed that, in contrast to the continuous setup, here we do not have a common boundary between two Voronoi regions. Instead, the vertices on the boundary of Voronoi region g_i are the neighbors of the corresponding vertices in Voronoi region g_j. We assume that robot r_i and its neighbor robots can communicate to each other whenever they need to exchange information. Most works based on the locational optimization framework rely on the same assumption. The information shared between the robots is the next best location and not the current location of the robots.

In *Compute* state, after obtaining $\mathcal{P}_i^*$ (next best neighbor robots' location of robot i), and computing the corresponding Voronoi subgraph (g_i), the next best node (p_i^*) is found by calling Algorithm 2. The state will be switched to *Moving* if the new best node differs from the current one. Robot i moves toward p_i^* and reaches this node in finite time (assuming absence of failure in the low level controller of the robot) and changes the state to *Compute* again. Note that according to Algorithm 1, function $Find\_next\_best\_node()$ is only called when $p_i = p_i^*$.

Algorithm 2 Function $Find\_next\_best\_node()$.

Input: g_i, p_i' // g_i is the Voronoi subgraph of robot r_i and p_i' corresponds to the current best node.
Output: p_i^*; // The next best node.
$\mathcal{H}_i \leftarrow \sum_{q \in g_i} d(p_i', q)\phi(q)$// Compute the cost function of the current best node of robot r_i.
foreach $a \in \mathcal{N}_{\mathcal{G}}(p_i')$ **do**
  $\mathcal{H}_a \leftarrow \sum_{q \in g_i} d(a, q)\phi(q)$ //Compute $\mathcal{H}_a$ value for all the neighbor nodes of p_i'.
$p_i^{min} \leftarrow \arg\min_{k \in \{\mathcal{N}_{\mathcal{G}}(p_i')\}} \mathcal{H}_k$ // Find the minimum $\mathcal{H}_k$ among the neighbor graph nodes
if $\mathcal{H}_k < \mathcal{H}_i$ **then** // If the minimum cost function of the neighbor nodes is less than the robot current $\mathcal{H}_i$
  $p_i^* \leftarrow p_i^{min}$ // Set the neighbor node (with minimum cost) as the next best node.
else
  $p_i^* \leftarrow p_i'$ // Otherwise the current node is the best node (there is no better node out of neighbor nodes)
Return p_i^*

In Algorithm 2, the next best node is selected based on the possibility of decreasing $\mathcal{H}$ by decreasing the component of this function related to robot r_i, $\mathcal{H}_i$. The next node is chosen to be the directed graph neighbor node which allows for the maximum decreasing of $\mathcal{H}$ considering the current Voronoi subgraph as the partition associated to robot r_i.

As an example, in Fig. 8, by assuming that the robot is located at node 15, according to lines 1 and 2 in Algorithm 2, $\mathcal{H}_i$ will be computed for the current node and its neighbors (4 and 5). Later in line 3, the node with smaller $\mathcal{H}_i$ will be selected. It should be noticed from the figure that for computing $\mathcal{H}_i$ for nodes 4 and 5, the same Voronoi subgraph that was applied to compute $\mathcal{H}_i$ for node 15, is used.

Next section presents the proof of convergence for the proposed algorithm along with its computational complexity analysis and comparison with other methods.

4.3 Analysis

The proposed algorithm works upon the following assumptions:
i) The graph $\mathcal{G}$ containing weights and commands which is created as explained in Section 4.1 to represent the environment is given to all the robots in the beginning of the task.
ii) Robots have access to the next best node of their neighbors. Whenever a robot needs to compute its next best node, the latest updated information of neighbor robots is available by means of communications.

Given these assumptions, we can now prove the convergence of the proposed multi-robot system.

4.3.1 System Convergence

We start by presenting some lemmas and definitions.

Lemma 1 *Let* $\mathcal{H}^*(P, \mathcal{G}, W) = \sum_{i=1}^{n} \mathcal{H}_i^*(p_i, w_i)$, *where* $\mathcal{H}_i^*(p_i, w_i) = \sum_{q \in w_i} d(p_i, q)\phi(q)$, w_i *is an arbitrary partition of* $\mathcal{G}$ *which is different from the Voronoi partition defined in (1), and* $\phi : \mathcal{V} \mapsto \mathbb{R}^+$. *The following inequality holds:*

$$\mathcal{H} \leq \mathcal{H}^*,$$

where $\mathcal{H}$ *is computed according to Eq. (2) using a Voronoi partition of the nodes of* $\mathcal{G}$.

Proof The proof follows the same arguments used in the proof of Proposition 3.1 in [11]. According to Eqs. (2) and (3) we have that:

$$\mathcal{H} = \sum_{i=1}^{n} \sum_{q \in g_i} d(p_i, q)\phi(q),$$

and

$$\mathcal{H}^* = \sum_{i=1}^{n} \sum_{q \in w_i} d(p_i, q)\phi(q).$$

According to Definition 1 we have that $d(p_i, q)\phi(q) \leq d(p_j, q)\phi(q)$ if q is in the Voronoi subgraph of r_i, g_i. Thus, as w_i is not the associated Voronoi subgraph the inequality holds:

$$\mathcal{H} = \sum_{i=1}^{n} \sum_{q \in g_i} d(p_i, q)\phi(q) \leq \sum_{i=1}^{n} \sum_{q \in w_i} d(p_i, q)\phi(q),$$

Thus:

$$\mathcal{H} \leq \mathcal{H}^*. \qquad \square$$

Definition 2 (Decreasing lower bounded sequence) A sequence $\{x_n\}$ is called decreasing and lower bounded, if:

i) $x_i \leq x_{i-1} \ \forall i \geq 1$
ii) $\exists B_0 \in \mathbb{R}$ such that $\forall n \ x_n \geq B_0$

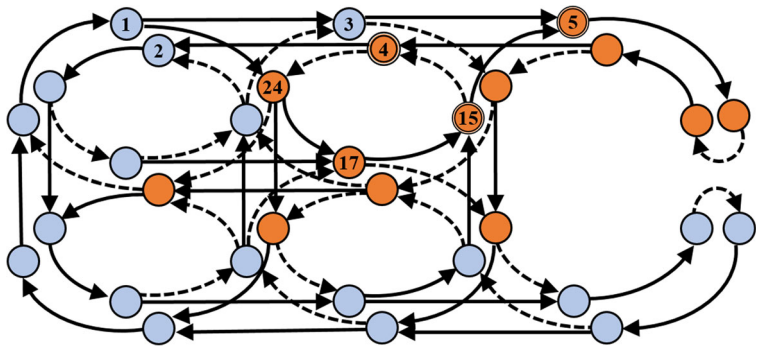


Fig. 8 Robot is placed on node 15. Based on Algorithm 2, $\mathcal{H}_i$ will be computed for nodes 15, 4 and 5, over the same Voronoi subgraph (nodes with brown color)

Lemma 2 *Every decreasing lower bounded sequence* $\{x_n\}$ *converges to the greatest lower bound of the set* $\{x_n : n \in \mathbb{N}\}$.

Proof This is a version of a well known result in real analysis. □

Definition 3 Let $\mathcal{S}$ be the set of state vectors of the multi-robot system described in Section 4.2 with robots executing Algorithms 1 and 2, where a state vector S is a vector of the next best nodes:

$$S = \begin{bmatrix} p_1^* \\ \vdots \\ p_n^* \end{bmatrix}.$$

Definition 4 A state vector S is called a local minimum state of the $\mathcal{H}$ function in Eq. (2) if :

$$\mathcal{H}(S, \mathcal{G}) = \sum_{i=1}^{n} \mathcal{H}_i(p_i^*, g_i) \leq \mathcal{H}(S', \mathcal{G}), \ \forall S',$$

where, S' is any state vector which differs from S in only one of its entries j, where the current next best node is replaced by a node which is a neighbor of p_j^*. The computation of $\mathcal{H}(S', \mathcal{G})$ is performed using the same Voronoi subgraph defined for $\mathcal{H}(S, \mathcal{G})$ as the associated graph partition.

Theorem 1 *Given a multi-robot system as described in Section 4.2 with robots executing algorithms 1 and 2, its associated state vector converges to a local minimum state of* $\mathcal{H}$ *in Eq. (2) in finite time if the following assumptions are verified: (i) the robots have access to the graph* $\mathcal{G}$*; and (ii) the robots have access to the next best nodes of their neighbors whenever they need.*

Proof The value of $\mathcal{H}$ can only change due to two reasons: 1) the computation of a new Voronoi tessellation (line 3 in Algorithm 1); and 2) the computation of a next best value (line 5 in Algorithm 2).

If the assumptions are verified then, according to Lemma 1, the computation of the new Voronoi tessellation cannot increase the value of $\mathcal{H}$. Also by inspection of Algorithm 2 we can say that if there exists a state vector S' obtained by the replacement of the entry associated with the current p_i^* with a graph neighbor of p_i^* such that $\mathcal{H}(S', \mathcal{G}) < \mathcal{H}(S, \mathcal{G})$, where S is the current state vector, then the system will evolve to a new state vector in finite time and this new state vector is so that $\mathcal{H}$ will decrease. Moreover, if such a vector S' does not exist then the system does not change the state vector.

Given these facts and also the fact that $\mathcal{H} \geq 0$, we can guarantee that $\mathcal{H}$ evolves according to a decreasing lower bounded sequence $\{\mathcal{H}_t\}$. Thus, according to Lemma 2, $\mathcal{H}$ converges to the greatest lower bound of the set $\{H_t : t \in \mathbb{N}\}$. Since the number of state vectors in $\mathcal{S}$ is finite, this convergence happens in finite time.

When $\mathcal{H}$ reaches convergence, the state vector will have converged to a local minimum state as given by Definition 4, since this vector does not change when no feasible S' capable of decreasing the value of $\mathcal{H}$ can be found. □

4.3.2 Computational Complexity

We consider the bottleneck of our algorithm: *computing the Voronoi tessellation and the cost function* $\mathcal{H}_i$ in each time step. Considering our weighted directed graph, similarly to [3] our algorithm can be implemented as a modified Dijkstra algorithm with complexity $O(|\mathcal{V}| \log |\mathcal{V}|)$ when an efficient data structure such as a heap is used, where $|\mathcal{V}|$ is the number of nodes in the graph. In fact, all the computations can be done over a single run of Dijkstra algorithm.

To compute the complexity of the algorithm precisely, we may consider its two main parts. In the first part, the cost function (also the Voronoi tessellation) is computed for p_i^* (line 1 in Algorithm 2). The second part denotes the time related to the computation of the $\mathcal{H}_i$ for the graph neighbor nodes of p_i^* (line 2 in Algorithm 2). As we mentioned, the complexity of computing $\mathcal{H}_i$ is $O(|\mathcal{V}| \log |\mathcal{V}|)$. For the types of environments that are tackled by the proposed methodology (those composed of corridors and intersections) the maximum number of graph neighbor nodes is finite. For instance, given our simplification to block-shaped environments we do not need to consider more than 4 graph neighbor nodes, $|\mathcal{N}_G(p_i^*)| \leq 4$. Thus, the complexity for all the neighbor nodes is $4 \cdot O(|\mathcal{V}| \log |\mathcal{V}|)$. However we can consider it as $O(|\mathcal{V}| \log |\mathcal{V}|)$. It should be mentioned that to compute the $\mathcal{H}_i$ for neighbor nodes, the Voronoi tessellation that was computed for p_i^* is used. Thus, the Voronoi tessellation is computed only once in each step.

Thus, we can write that the total complexity of our algorithm is $O(|\mathcal{V}| \log |\mathcal{V}|)$.

4.3.3 Comparison

In this section, we compare the complexity of our method with the most similar works in the literature, [12] and [30]. In [30], the method looks, in the set formed by the nodes of the Voronoi subgraph g_i, for the next best node to be reached by robot i. This requires $O(|\mathcal{V}|^2)$ operations since it is necessary to access pairs of nodes to quantify the impact of a given motion in the global cost function. In [12], as only short-range communication is allowed, robots move to a random destination in their associated subgraph in order to meet other robots and then be able to communicate with these other robots to compute a new optimized graph partition. When the robots meet neighbor robots, they exchange information about their current partitions and run a pairwise partitioning rule which demands $O(|\mathcal{V}|^3)$ operations.

The methods in [12] and [30] also require some time to compute the shortest path from the robot's current position to the next aimed point. This can be done by running BFS (Breast-First Search) algorithm with $O(|\mathcal{V}|)$ or Dijkstra algorithm with complexity $O(|\mathcal{V}| \log |\mathcal{V}|)$. In both cases, the computation of shortest path does not change the original complexity of $O(|\mathcal{V}|^3)$ and $O(|\mathcal{V}|^2)$, respectively.

In contrast, as show in Section 4.3.2, the complexity of our approach is $O(|\mathcal{V}| \log |\mathcal{V}|)$. Also, instead of searching for a path in the graph, as required by [12] and [30], we simply compute Eq. (3) k times, where k is a constant that defines the number of graph neighbor nodes ($k \leq 4$ for the block-shaped environments considered in this paper,) to find the next possible movement. Hence, robots are ready to move to one of their neighbor nodes in each iteration after this computation. This procedure which is applicable to any kind of graph representation, decreases substantially the computational time, and also makes the algorithm provable in the sense of convergence.

Another shortage of works [12] and [30] is the fact that they demand a network with high data transferring capacity, which is not necessary in our method. In [12], authors use short-range "gossip"[2] communication networks. Robots r_i and r_j need to transfer $O(|V_i| + |V_j|)$ data (where V_i is the Voronoi partition of robot i) through the channel when they meet each other and want to compute their pairwise Voronoi partitions. In [30] a two-hop communication is needed, since the information of neighbor robots and neighbors of neighbors are needed. Our methodology assumes a reliable network communication between neighbor robots only and we do not need high bandwidth network, since only the robots' locational information (a single integer node number) is transmitted.

Finally, in contrast to previous works, our method works without the need for a precise map. This increases the robustness of the implementation and reduces the need for on-board computation.

4.4 Robot Control

The solution for the multi-robot deployment problem shown in the previous section gives, for each robot at each time interval, the node of graph $\mathcal{G}$ where the robot must go after leaving the current node. Based on the way $\mathcal{G}$ was defined, remember that each node represents a portion of the environment, which could be a corridor or a street block, for example. Also, to each edge of the graph, there was associated a command for the robot, whose cost of execution was used in our algorithm to help deciding the best path. Therefore, assuming that the robot is currently at node $x \in \mathcal{V}$ and the next node computed by the algorithm is $y \in \mathcal{V}$, a controller needs to be designed so the robot follows edge $\overrightarrow{xy}$ by executing command $I(x, y)$. As mentioned before, the set of commands used in this work contains "human-like" instructions which are executed when the robot leaves the current node. If a robot is in an indoor environment, for example, a command will be executed when an intersection of corridors is found. Thus, two controllers are necessary: one to drive the robots inside the node and another to follow the command associated to the edge.

For the first controller, remember that the workspaces considered in this work are defined by regions similar to corridors or street blocks. These regions are usually defined by constrained areas surrounded by some sort of structure, for example, walls in a building, or the sidewalks on the streets. These structures will be a constant reference for the robot inside a node and can be used to aid its guidance and control. In this paper, we use such structures to define a vector field to guide the robot. Without loss of generality and to facilitate the explanation of our vector

[2] A short-range communication with asynchronous and unreliable communication between nearby robots

field generation method, we will call the structure that delimits the node region by *wall* in the rest of the section.

To compute the vector field, we create a reference frame on the closest point from the robot to the wall on its right side, as shown in Fig. 9. In this frame, the X axis is tangent to the wall, pointing in the forward direction, and the Y axis is orthogonal to it, pointing to the left wall. It is assumed that the robot is always in coordinate $X = 0$, so the frame moves with the robot.

To make the robot move along the corridor by keeping itself parallel to the corridor's right wall at a distance d_0 from it, a planar velocity vector field $\mathbf{u} = [u_X, u_Y]^T$ is created. While the field component along the corridor, u_X, have a constant value, the component along Y-axis, u_Y, is proportional to the error between the robot current distance to the wall, d, and the desired distance d_0. The vector is then normalized and scaled to the desired velocity of the robot. In practical implementations, to localize itself in relation to the wall by computing distance d and orientation θ (see Fig. 9), a robot could use, among other sensors, a planar laser range sensor (LADAR). With such a sensor, this can be done by simply detecting and computing the parameters of the straight line found in the sensor data, as shown in [1].

To avoid possible obstacles in the corridor, such as people and other robots, vector field **u** can be summed with a repulsion vector field designed to avoid obstacles. Although this approach seems to be simple, it can easily create local minima in the field, which would stop the robot. Since the authors believe that the definition of obstacle avoidance vector fields and the solution to the problems related to it are out of the scope of this paper, the reader is refereed to [15], which presents a vector field that allows the robot to present human-friendly behaviors in the presence of people and [1], which adds an obstacle/people avoidance solution to our vector field.

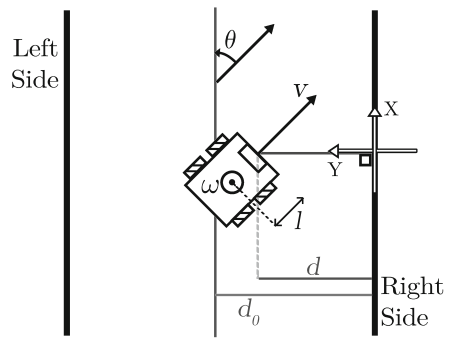


Fig. 9 Geometry involved in the vector field generation and robot control. From the reference frame, distance d_0 is the expected distance while d is the current distance of the robot. θ indicates the robot's relative orientation and l, the distance from the center of the robot to its control point

To track the vector field with a nonholonomic robot, we transform each vector into the inputs of the robot using a static feedback linearization controller [19]. Assuming that the robot inputs are linear velocity v and angular velocity ω we have:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta)/l & \cos(\theta)/l \end{bmatrix} \begin{bmatrix} u_X \\ u_Y \end{bmatrix} \quad (4)$$

where l is the distance from the center of the robot to its control point and θ is the angle between the robot and the reference frame, as shown in Fig. 9.

This previous controller drives the robot inside the nodes of the topological map until it finds itself in an intersection,[3] which represents a node change. Because of the block-shaped regions of the map, it is possible to perceive that the robot is approximating an intersection by detecting that the structures of the corridor are shaping up to the opening of an intersection. If the robot is using a laser to follow the wall, for example, it is possible to detect that the segment that represents the wall in the laser scan is decreasing in length. When an intersection is detected, the robot must follow the command associated to the edge that connects the current node with the next one computed by the planner. Since the number of possible instructions is small, one simple controller can be developed for each instruction. For example, a "Turn Left" instruction would lead to the activation of a proportional controller that would turn the robot by 90 degrees to the left, so it can move forward and enter the region of the new node, after the intersection. Once the controller finishes its task, the corridor follower, vector field based controller is switched back to move the robot inside the new node.

5 Implementation Results

In this section, the efficiency of the proposed method is investigated in both simulated and real robot experiments. Before showing the result, we will explain the

[3] The place where two or more corridors cross each other

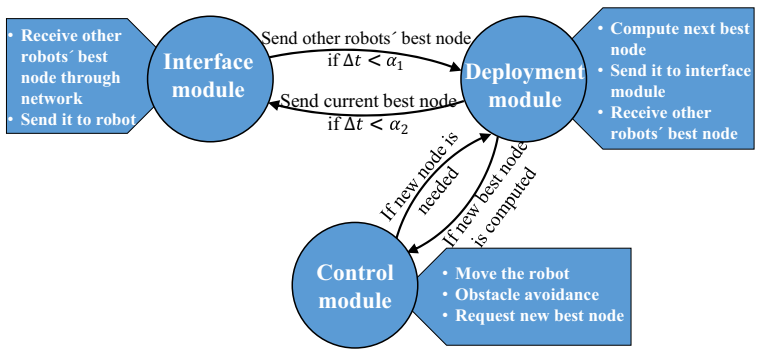
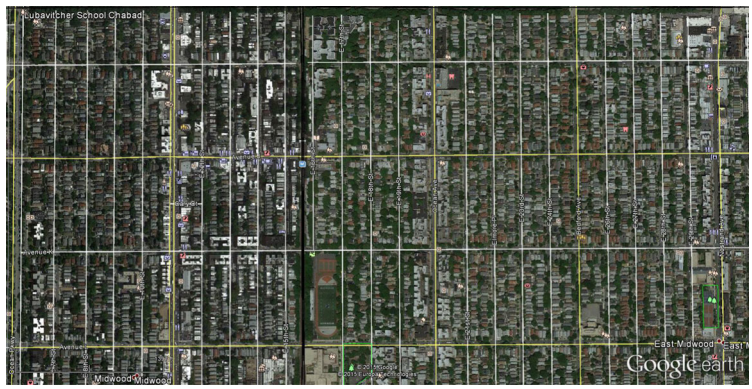
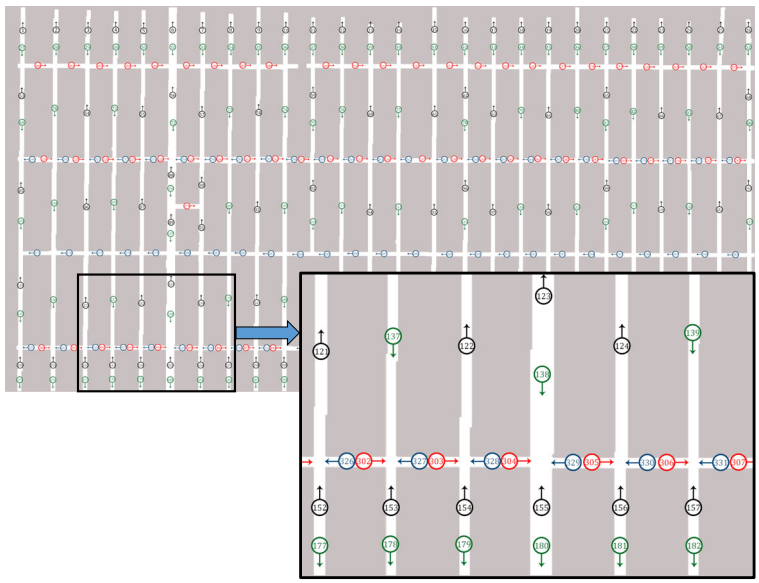


Fig. 10 The high-level system of robot i, and the relation between three main modules in a high-level description. In order to communicate between modules (Interface and Deployment), time difference Δt should be less than thresholds (α_1 or α_2)



(a) Sattelite view.



(b) Extracted streets with directions. Numbers indicate the nodes of the corresponding graph. Some of the streets have two numbers which means they are bidirectional, *i.e.* they have two nodes associated with them. Different directions of movement are painted in different colors: right and left are red and blue, as well as up and down with black and green color, respectively.

Fig. 11 The map of a neighborhood in New York grabbed from Google maps

Table 2 Set of commands and costs in the simulation

Command#	type	Cost
1	Turn_Left	1.5
2	Turn_Right	1.5
3	Straight	1
4	Turn_Back	2

developed distributed architecture. The high-level system of the robots is shown in Fig. 10, where *Control, Deployment* and *Interface* modules are our three main entities. While the control module captures data and interacts with the environment, deployment module compute the next best node by relying on data received from the control module. To keep robots communication, the interface module send/receive pose information to robot r_i to/from other robots through a wireless connection. The deployment module receives and sends the best node information from/to the interface module periodically. Thus α_1 and α_2 are thresholds for receiving and sending information from/to other robots. Also Δt refers to the time difference: $\Delta t = |t^{i+1} - t^i|,\ t > 0$.

In our implementations, we used C++ programming, ROS and Matlab for the different modules. In Fig.10, the Deployment module was implemented in Matlab, and Interface and Control modules were developed in ROS. We used ROS toolbox in Matlab to exchange information between robots.

5.1 Simulation Results

For the simulation, we selected a real outdoor map of a neighborhood in New York City from Google Maps (See Fig. 11). In this map, streets and blocks have a symmetric shape, which is important for metropolitan cities in order to facilitate distributing services and urban management *i. e.* transportation, pipeline, electricity and so on. Furthermore, this block-shape property gives us the ability to run our deployment algorithm without the need of precise localization. Thus, regardless the scale of the input map, first of all we find a topological representation of the map by extracting its streets as in Fig. 11b. This is done by applying morphological operators in the image i.e. thresholding, erosion and dilation. We simulated 6 differential drive robots able to move along the required streets, using, for example, a curb detection and following algorithm [13]. We also assume that the robots have access to perfect communication with their immediate neighbor robots.

Since the direction of the streets is available in Google Maps, a directed graph $\mathcal{G}$ may be constructed (the method was explained in section 4.1). The command set $\mathcal{I}$ and its corresponding cost $\mathcal{C}$ used in this

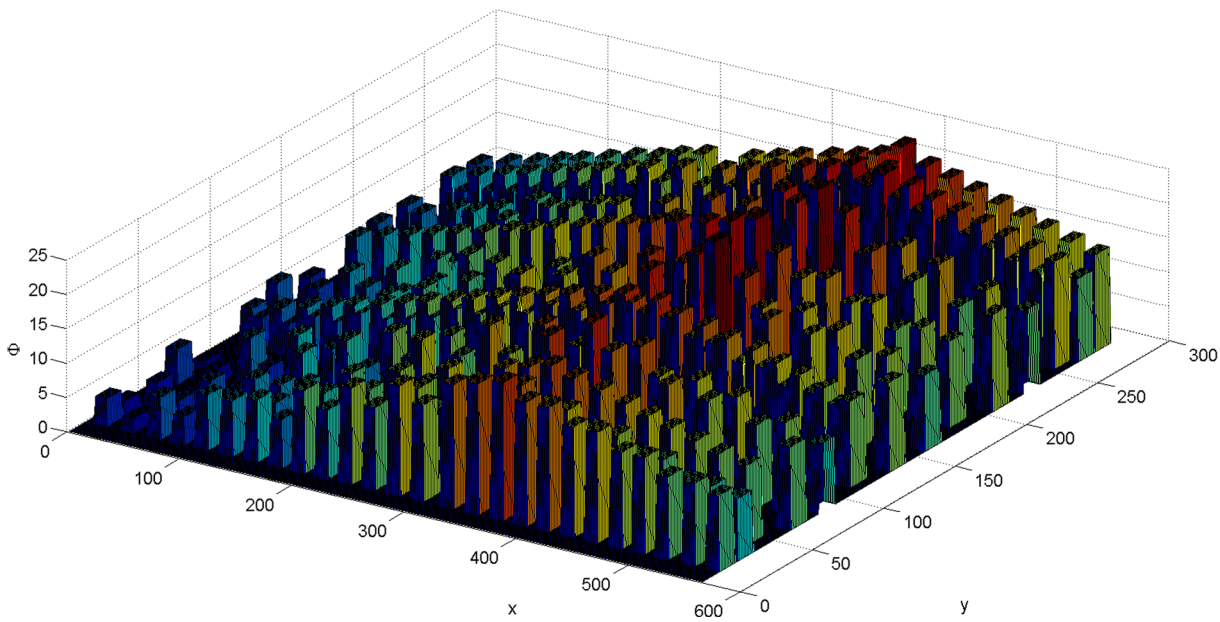
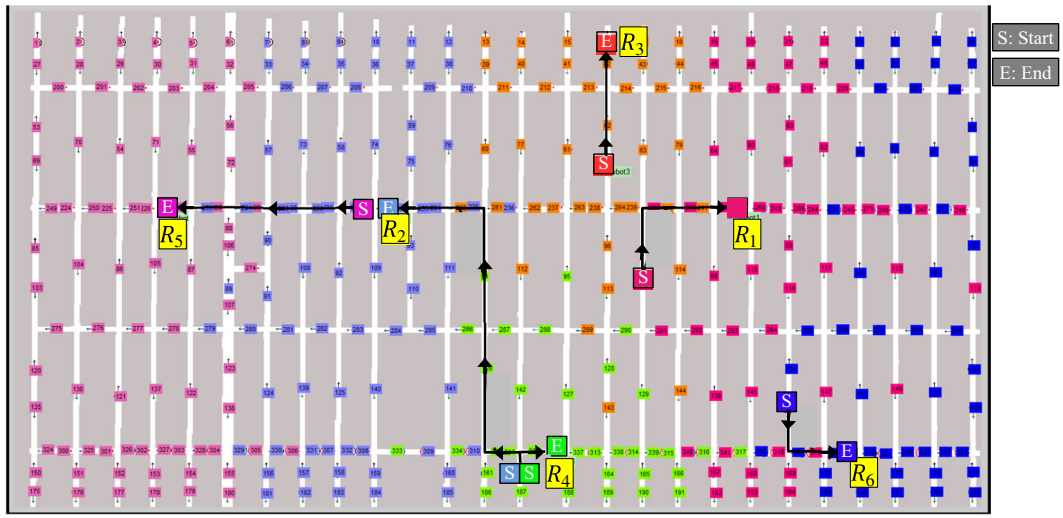
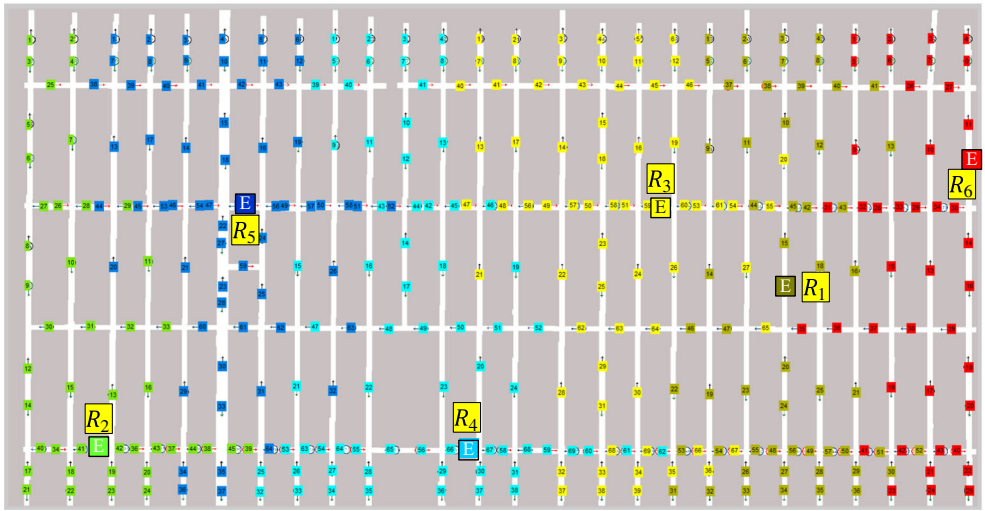


Fig. 12 The density function defined in this scenario. The center of this function is defined at (387,152) where node 114 is placed



(a) Voronoi regions and the traversed trajectory by robots.



(b) Global solution by solving p-median problem; independent from the initial position of the robots.

Fig. 13 Partitioning obtained by the proposed method and solving the MILP model of the p-median problem

simulation is shown in Table 2. We defined the center of density function on the node 114, such that Fig. 12 represents the density function on this map.

After constructing graph $\mathcal{G}$, we distribute the 6 robots randomly over the environment. After applying Algorithm 1, robots' final locations, corresponding assigned regions and traversed paths are depicted in Fig. 13a with different colors. The video of this simulation is available in the following link, https://www.youtu.be/yAyio1--viA.

In order to investigate the performance of our on-line method, we performed an off-line p-median

Table 3 Comparison of $\mathcal{H}$ and computation time in different methods

Methodology	Cost($*10^5$)	Computation Time
Global Solution	1.12686	more than a hour
Proposed algorithm	1.18786	0.6797 sec

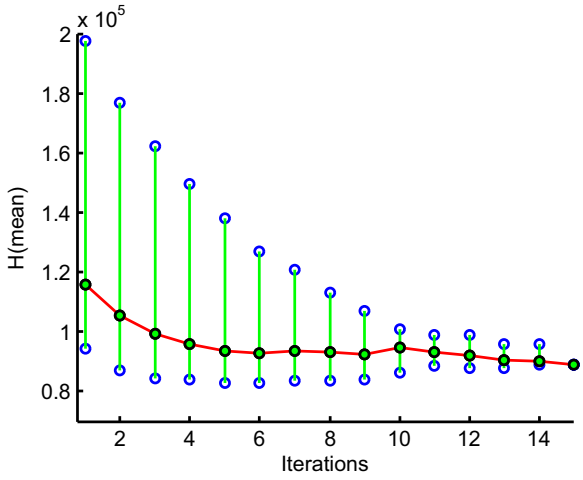


Fig. 14 Result of simulating the proposed method 100 times on New York scenario. Each column contains mean (red line), max and min of the deployment cost function

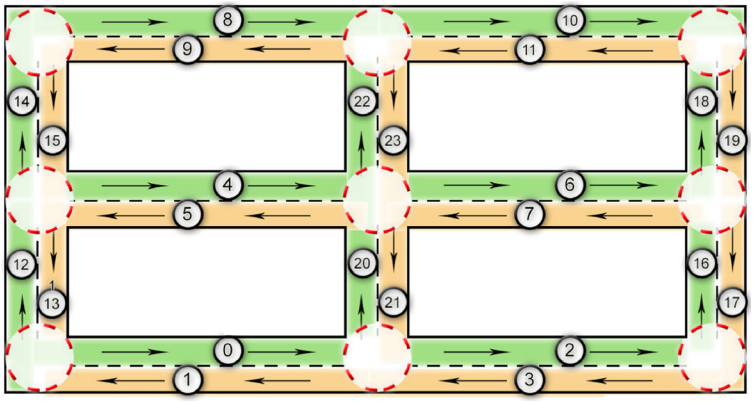
solution [9] which is applicable for similar purpose on graphs. In this problem, by considering a graph with $n \times m$ nodes, the objective is to assign n facilities to m customers. One of the methods to solve this NP-hard problem is the Mixed Integer Linear Programming (MILP) which yields the global optimum solution [26].

In this simulation, robots start their movement from the same initial locations that were used in the proposed method. After applying a solver on the MILP model which takes hours for the graph with $|\mathcal{V}| = 347$ we find a result which is shown in Fig. 13b. Moreover, the value of the $\mathcal{H}$ function and computation time of proposed technique and MILP methods are shown in Table 3.

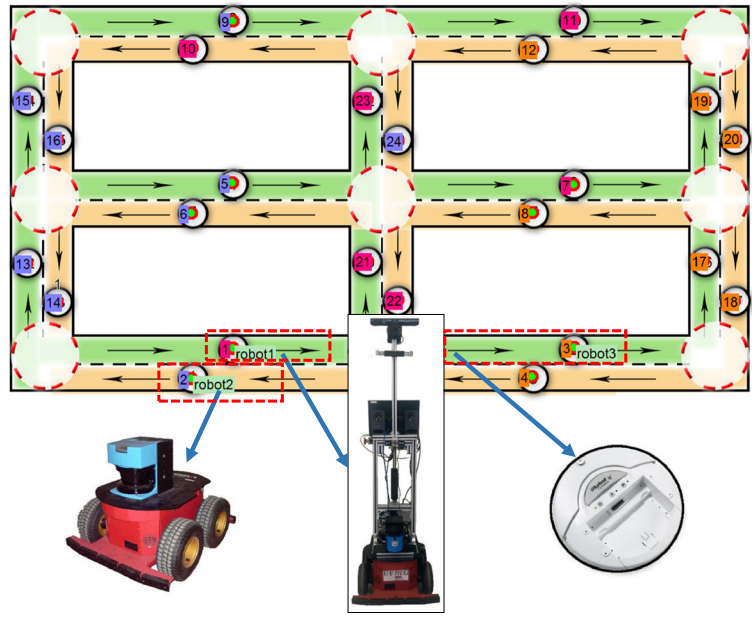
In our algorithm, function $\mathcal{H}$ decreases over time. The result of 100 trial runs of the proposed method on

Fig. 15 The map used in real robot experiment

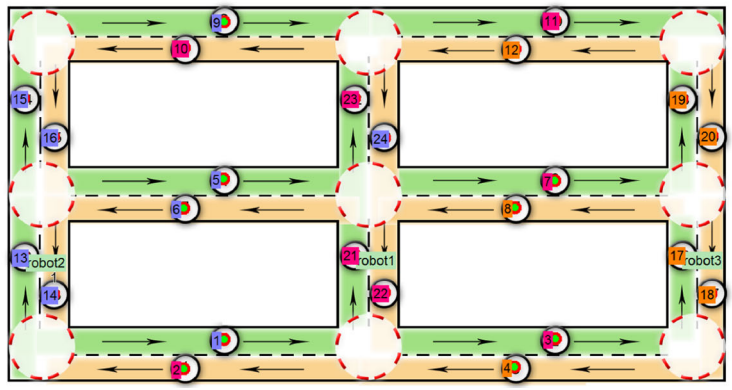
(a) Map is captured from Google map.



(b) Representing the input map as a graph.



(a) Initial position with corresponding Voronoi subgraph with different color



(b) Final Deployment

Fig. 16 Deployment of 3 robots in a real scenario

the New York map scenario is illustrated in Fig. 14. In each trial the robots are distributed randomly at initial positions, and after executing about 15 iterations, they converged to the final positions. It is important to remark that in order to move in such a big map, only a few human-like commands are enough to get from a point (or street) to another.

5.2 Real Robot Experiments

After validating the performance of the proposed method through simulations, we performed a real world experiment to verify its applicability. Three robots were chosen to cooperate in a distributed setup.

Two of the robots are based on the Pioneer P3-AT mobile base. They are non-holonomic robots with four

Table 4 Commands applied on robots in the real experiment

Robot#	command
1	Turn_Left
2	Turn_Right
3	Turn_Left

wheels, used in conjunction with a laser range sensor and odometric sensors. The first Pioneer, which we will call Robot 1, has indoors wheels and a SICK LMS100 laser range sensor, while the second robot, to be called Robot 2, has outdoor wheels and a SICK LMS291 laser range sensor. The third robot, Robot 3 is a small iRobot Create mobile base. It has also odometric sensors and a Hokuyo URG laser range sensor.

The experiments were executed in the second floor of the School of Engineering building at the Federal University of Minas Gerais. It is a symmetrical building, and is composed of many corridors and intersections (See Fig. 15a). To build the corresponding graph, the corridors were represented as nodes of the graph, one node for each direction of movement, while edges model the intersections. Each edge received a label with the direction to which the robot had to turn to reach the new corridor in the intersection.

Figure 15b shows the topological map of the environment, with 24 nodes. The corridor at the middle (nodes 4, 5, 6 and 7) is defined as the center of the density function, so the robots are expected to move toward this corridor.

The initial nodes, and the final deployment locations are depicted in Fig. 16. Robots 1, 2, and 3 started from nodes 1 to 3 (see Fig. 16a), and ended on nodes 21, 13 and 17 (see Fig. 16b), respectively. In the entire execution each robot runs a single command (See Table 4).

As we mentioned, the experiment was implemented in a decentralized fashion, so that each robot moves and decides individually. The complete video of this experiment can be found in https://youtu.be/3BUPjRI-x04.

6 Conclusions

In this work, we proposed a distributed multi-robot deployment method based on a topological representation of the environment. A practical technique was applied in order to simplify a multi-dimensional map into a single dimensional one. This technique can be used with very large maps decreasing the dimension and computation cost relatively. Once the topological model of the input map is obtained, robots use a wall following control to move , hence no precise localization is needed. This is done by using a natural scheme of navigation, such that robots move to a location by following a sequence of human like commands. We derived our strategy upon the topological framework proposed by [1]. In comparison with different techniques found in the literature, our method has a good performance in real applications because of the low computation, no need of precise localization, and no need of high bandwidth communication. These remarkable properties make our method fast enough to be executed in real world scenarios. In comparison to two similar discrete deployment methods found in the literature ([12, 30]), the proposed approach declines the computational complexity, and requires lower-bandwidth network to exchange locational information (next best node) with neighbor robots, which is usually a challenge in real experiments. As future work, we would like to perform real experiments in multi-floor maps, combining aerial and ground based robots in the same mission. Notice that even in this complex situation the problem can be considered a 1D problem.

Acknowledgments This work was supported by the Brazilian agencies, CAPES, CNPq, and FAPEMIG.

Compliance with Ethics Requirements All the authors declare that they have no conflict of interest.
Reza Javanmard Aliappeh has received research grants from Brazilian agency CNPq.
Arthur R. Araujo has received research grants from Brazilian agency CNPq.
Arthur R. Araujo is an undergraduate student at Federal University of Minas Gerais.
Guilherme A. S. Pereira and Luciano C. A. Pimenta have received research grants from Brazilian agencies CNPq and FAPEMIG.
Guilherme A. S. Pereira and Luciano C. A. Pimenta are faculty members of Federal University of Minas Gerais.
This article does not contain any studies with human or animal subjects.

References

1. Araujo, A.R., Caminhas, D.D., Pereira, G.A.S.: An architecture for navigation of service robots in human-populated office-like environments. In: 11th IFAC Symposium on Robot Control SYROCO 2015, vol. 48, pp. 189–94 (2015)
2. Bhattacharya, S., Michael, N., Kumar, V.: Distributed coverage and exploration in unknown non-convex environments. Distributed Autonomous Robotic Systems, Springer Tracts in Advanced Robotics **83**, 61–75 (2013)
3. Bhattacharya, S., Ghrist, R., Kumar, V.: Multi-robot coverage and exploration on riemannian manifolds with boundaries. Int. J. Robot. Res. **33**(1), 113–137 (2014)

4. Breitenmoser, A., Schwager, M., Metzger, J.C., Siegwart, R., Rus, D.: Voronoi coverage of non-convex environments with a group of networked robots. In: Proceedings of The IEEE International Conference on Robotics and Automation (ICRA), pp. 4982–4989 (2010)
5. Bullo, F., Cortés, J., Martínez, S.: Distributed Control of Robotic Networks. Applied Mathematics Series, Princeton University Press (2009)
6. Caicedo-Nunez, C., Zefran, M.: Performing coverage on nonconvex domains. In: Proceedings of the IEEE International Conference on Control Applications (CCA), pp. 1019–1024 (2008a)
7. Caicedo-Nunez, C.H., Zefran, M.: A coverage algorithm for a class of non-convex regions. In: Proceedings of the IEEE Conference on Decision and Control (CDC), pp. 4244–4249 (2008b)
8. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. Robot. Autom. **20**(2), 243–255 (2004)
9. Daskin, M., Maass, K.: The p-median problem. In: Location Science, pp. 21–45. Springer International Publishing (2015)
10. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(l 959), 269–271 (1959)
11. Du, Q., Faber, V., Gunzburger, M.: Centroidal voronoi tessellations: Applications and algorithms. SIAM Rev. **41**(4), 637–676 (1999)
12. Durham, J., Carli, R., Frasca, P., Bullo, F.: Discrete partitioning and coverage control for gossiping robots. IEEE Trans. Robot. **28**(2), 364 – 378 (2012)
13. Hata, A.Y., Osorio, F.S., Wolf, D.F.: Robust curb detection and vehicle localization in urban environments. In: Proceedings of the 2014 IEEE Intelligent Vehicles Symposium, pp. 1257–1262 (2014). doi:10.1109/IVS.2014.6856405
14. Javanmard Alitappeh, R., Pimenta, L.C.A.: Distributed Safe Deployment of Networked Robots. Distributed Autonomous Robotic Systems, Springer Tracts in Advanced Robotics **12**, 65–77 (2016)
15. Lam, C.P., Chou, C.T., Chiang, K.H., Fu, L.C.: Human-centered robot navigation—-towards a harmoniously human-robot coexisting environment. IEEE Trans. Robot. **27**(1), 99–112 (2011)
16. Lee, S.G., Diaz-Mercado, Y., Egerstedt, M.: Multirobot control using time-varying density functions. IEEE Trans. Robot. **31**(2), 489–493 (2015)
17. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
18. Mahboubi, H., Sharifi, F.: Distributed coordination of multi-agent systems for coverage problem in presence of obstacles. In: Proceedings of American Control Conference (ACC), pp. 5252–5257 (2012)
19. d'Andréa Novel, B., Campion, G., Bastin, G.: Control of nonholonomic wheeled mobile robots by state feedback linearization. Int. J. Robot. Res. **14**(6), 543–559 (1995)
20. Pierson, A., Schwager, M.: Adaptive inter-robot trust for robust multi-robot sensor coverage. Distributed Autonomous Robotic Systems, Springer Tracts in Advanced Robotics **114**, 167–183 (2016)
21. Pierson, A., Figueiredo, L., Pimenta, L.C.A., Schwager, M.: Adapting to performance variations in multi-robot coverage. In: Proceedings of The IEEE International Conference on Robotics and Automation (ICRA), pp. 415–420 (2015)
22. Pimenta, L.C.A., Kumar, V., Mesquita, R.C., Pereira, G.A.S.: Sensing and coverage for a network of heterogeneous robots. In: Proceedings of IEEE Conference on Decision and Control (CDC), pp. 3947–3952 (2008)
23. Pimenta, L.C.A., Schwager, M., Lindsey, Q., Kumar, V., Rus, D., Mesquita, R.C., Pereira, G.A.S.: Simultaneous coverage and tracking (scat) of moving targets with robot networks. Algorithmic Foundation of Robotics **VIII**(57), 85–99 (2010)
24. Reese, J.: Solution Methods for the p-Median Problem: An Annotated Bibliography. Networks **48**, 125–142 (2006)
25. Schwager, M., Rus, D., Slotine, J.J.: Decentralized, adaptive coverage control for networked robots. Int. J. Robot. Res. **28**(3), 357–375 (2009)
26. Senne, E., Lorena, L.A., Pereira, M.A.: A branch-and-price approach to p-median location problems. Comput. Oper. Res. **32**, 1655–1664 (2005)
27. Sharifi, F., Zhang, Y., Aghdam, A.G.: A Distributed Deployment Strategy for Multi-Agent Systems Subject to Health Degradation and Communication Delays. J. Intell. Robot. Syst. **73**(1), 623–633 (2014)
28. Sharifi, F., Chamseddine, A., Mahboubi, H., Zhang, Y., Aghdam, A.: A distributed deployment strategy for a network of cooperative autonomous vehicles. IEEE Trans. Control Syst. Technol. **23**(2), 737–745 (2015)
29. Stergiopoulos, Y., Tzes, A.: Coverage-oriented coordination of mobile heterogeneous networks. In: Proceedings of 19th Mediterranean Conference on Control & Automation (MED), pp. 175–180 (2011)
30. Yun, S.k., Rus, D.: Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning. Robotica **32**(02), 257–277 (2014)

Reza Javanmard Alitappeh obtained his BSc degree from Shomal University of Amol (SUA), and MSc degree from Azad University of Qazvin, Iran, in 2005 and 2011 respectively. From 2008 to June 2011, he was a researcher at Mechatronic Researcher Laboratory (MRL) at Azad University of Qazvin. He also finished his Ph.D. in April 2016 under the supervision of Prof. Luciano Pimenta in Universidade Federal de Minas Gerais (UFMG), Brazil. His research interest is centered on machine learning, machine vision and multi-robot systems.

Guilherme A. S. Pereira received the B.S. and M.S. degrees in electrical engineering and the Ph.D. degree in computer science from the Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil, in 1998, 2000, and 2003, respectively. Dr. Pereira received the Gold Medal Award from the Engineering School of UFMG for garnering first place among the electrical engineering students in 1998. He was, from November 2000 to May 2003, a Visiting Scientist at the General Robotics, Automation, Sensing and Perception (GRASP) Laboratory, University of Pennsylvania, Philadelphia, and from August 2015 to July 2016, a Visiting Scholar at the Robotics Institute, Carnegie Mellon University, Pittsburgh. Since July 2004, he is an Associate Professor of the Electrical Engineering Department at the Federal University of Minas Gerais (DEE/UFMG), where he is the director of the Computer Systems and Robotics (CORO) Laboratory, one of the laboratories that compose the Group for Reaserch and Development of Autonomous Vehicles (PDVA) at UFMG. His research interests include cooperative robotics, robot navigation, autonomous vehicles development, computer vision, and distributed sensing. Dr. Pereira is a Member of Sociedade Brasileira de Automática (SBA) and a Senior Member of IEEE.

Arthur R. Araújo is currently studying for his BEng degree in Electrical Engineering at the Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Brazil. From October 2015 to August 2016, he was an exchange student on the BEng Electronic Engineering with Computer Systems program at the University of Surrey, Guildford, UK. His research interests include robotics, control theory, space sciences and software design.

Luciano C. A. Pimenta received his BS, MSc, and Ph.D. degrees in electrical engineering from the Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Brazil, in 2003, 2005, and 2009, respectively. From April 2007 to June 2008, he was a visiting Ph.D. student at the General Robotics, Automation, Sensing and Perception (GRASP) Laboratory at the University of Pennsylvania, Philadelphia, USA. He is currently an assistant professor with the Department of Electronic Engineering at UFMG. His research interests include robotics, multi-robot systems, and control theory.