Article

The International Journal of Robotics Research
1–21
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0278364919882082
journals.sagepub.com/home/ijr
SAGE

Equitable persistent coverage of non-convex environments with graph-based planning

José Manuel Palacios-Gasós[1], Danilo Tardioli[2,3], Eduardo Montijano[3] and Carlos Sagüés[3]

Abstract

In this article, we tackle the problem of persistently covering a complex non-convex environment with a team of robots. We consider scenarios where the coverage quality of the environment deteriorates with time, requiring every point to be constantly revisited. As a first step, our solution finds a partition of the environment where the amount of work for each robot, weighted by the importance of each point, is equal. This is achieved using a power diagram and finding an equitable partition through a provably correct distributed control law on the power weights. Compared with other existing partitioning methods, our solution considers a continuous environment formulation with non-convex obstacles. In the second step, each robot computes a graph that gathers sweep-like paths and covers its entire partition. At each planning time, the coverage error at the graph vertices is assigned as weights of the corresponding edges. Then, our solution is capable of efficiently finding the optimal open coverage path through the graph with respect to the coverage error per distance traversed. Simulation and experimental results are presented to support our proposal.

Keywords
Persistent coverage, multi-robot system, equitable partitioning, non-convex environments, path planning

1. Introduction

Recent advances in mobile robotics and an increasing development of affordable autonomous mobile robots have motivated extensive research into multi-robot systems. The complexity of these systems resides in the design of communication, coordination, and control strategies to perform complex tasks that a single robot is not able to do by itself. A particularly interesting task is that of persistent coverage, in which a team of robots aims to maintain a desired coverage level over time in a given environment. The particularity of persistent coverage, as opposed to other known coverage problems (Cortes et al., 2004; Panagou et al., 2017), is that the coverage level deteriorates with time. This implies that the robots need to revisit every point in the environment in order to keep the coverage at the desired levels. This problem is of special interest in many applications such as vacuuming (Mackenzie and Balch, 1993), cleaning a place where dust is continuously settling (Kakalis and Ventikos, 2008), lawn mowing (Arkin et al., 2000; Sahin and Guvenc, 2007), or environmental monitoring (Paley et al., 2008; Smith et al., 2011).More recently, the apparition of useful unmanned aerial vehicles (UAVs) has encouraged the application of persistent coverage to surveillance and monitoring (Nigam, 2014; Renzaglia et al., 2012).

The solutions to this problem can be separated into two different groups, according to the division of the environment that is adopted to carry out the coverage mission. In the first group of solutions, all the robots cover the entire environment together in a cooperative manner (Atinç et al., 2014; Nigam et al., 2012),i.e., they are not restricted to particular areas. The second group is composed of partition-based solutions. They address the problem with a divide-and-conquer strategy in the sense that they partition the

[1]Firmware Group, Induction Technology, Product Division Cookers, BSH Home Appliances Group, Zaragoza, Spain
[2]Centro Universitario de la Defensa, Zaragoza, Spain
[3]Instituto de Investigación en Ingeniera de Aragón (I3A), Universidad de Zaragoza, Zaragoza, Spain

Corresponding author:
Eduardo Montijano, Instituto de Investigación en Ingeniera de Aragón (I3A), Universidad de Zaragoza, C/ Mariano Esquillor s/n, 50018 Zaragoza, Spain.
Email: emonti@unizar.es

environment and assign each partition to a single robot that becomes responsible for developing the coverage on it (Barrientos et al., 2011). The advantages of the second type are that, once the environment is partitioned, each robot can work independently, with no risk of collision and with no need for communication in the normal operation of the system. For these reasons, we propose a solution of this type and pay special attention to the partitioning and the single-robot coverage inside each region.

1.1. Environment partitioning

The most well-known partition technique for coverage in multi-robot systems is the Voronoi tessellation (Cortes et al., 2004). Although different partition strategies for coverage had been proposed previously (Jager and Nebel, 2002), the Voronoi partition has been the most widely used, especially for static coverage (Moon and Frew, 2017; Pierson et al., 2017). This type of partition has been successfully used in persistent coverage tasks for convex environments (Palacios-Gasós et al., 2016a) or environments with few obstacles (Palacios-Gasós et al., 2017) However, they present two important drawbacks. In general, these partitions do not take into account the shape of the environment nor do they account for obstacles and non-convexity. The second and most important drawback is that, since they are purely geometric, the coverage share-out is not equitable. This overloads some robots, while others have less work assigned than they can carry out.

Equitable partitions in convex environments are usually calculated in a centralized fashion. In the approach of Araujo et al. (2013), a line in the optimal sweeping direction is moved along the environment to slice it each time the area on one of the sides is the same as any required partition. A polygon decomposition was applied by Maza and Ollero (2007), with the restriction of including the robot position inside its partition, and in Peters et al. (2017) an algorithm handles sporadic communication between each robot and a central base to update the partition. Pavone et al. (2011) computed equitable partitions in convex environments using power diagrams in a distributed way.

On the other hand, the assumption of convex environments is not reasonable when it comes to ground vehicles. In non-convex environments, solutions based on Lloyd's algorithm appeared in Pimenta et al. (2008) and Breitenmoser et al. (2010b). This algorithm was also combined with the geodesic distance in Bhattacharya et al. (2014), together with a graph-based path planning for exploration. Frontier-based exploration of a non-convex environment was treated in Haumann et al. (2011). Power diagrams were considered by Thanou et al. (2013) for heterogeneous teams. Complete coverage was guaranteed in Kapoutsis et al. (2017), with a specifically tailored, optimality-preserving, technique. In the work of Sea et al. (2017), the features of online obstacle and decay learning were added to a decentralized partitioning of the target area on the basis of the robot performance. The main difference of our approach with respect to these solutions is that the resulting partition is equitable in terms of the robots' workload.

Over discrete representations and graphs, Voronoi-based load-balancing partitions were presented in Boardman et al. (2016) and in Durham et al. (2012), where it is combined with a gossip algorithm. Surfaces in three dimensions were treated by Breitenmoser et al. (2010a, 2014). Distributed vertex substitution was used by Yun and Rus (2014), with two-hop communication to guarantee convergence to the locally optimal configuration. Compared with the aforementioned works, our partitioning approach works for a continuous environment formulation.

1.2. Planning of coverage paths

There are many possibilities to define the motion of the robot inside their partitions (Galceran and Carreras, 2013; Nigam, 2014). Some solutions posed the problem as a feedback control problem, deciding at each time which is the best direction to move (Franco et al., 2013). These solutions are easy to compute individually but typically lead the robots to local minima. Our solution is also computationally light but considers a larger planning horizon.

A wide variety of planning approaches look for closed paths that each robot can follow periodically to cover its entire partition (Xu et al., 2014). Closed paths in square grids were computed by Arkin et al. (2000). Adaptive control was used by Soltero et al. (2014) and Lan and Schwager (2013) considered Rapidly-exploring Random Cycles in Gaussian random fields. Discretized decay rates by a factor of two were considered by Smith and Rus (2010) and Alamdari et al. (2014) to compute different tours for several robots based on the importance of each location. All these methods are closely related to the traveling salesman problem (TSP). Their main advantage is that they typically allow for some measurable degree of optimality with respect to the optimal NP-hard solution. Compared with these methods, our planning algorithm considers fast online re-planning of open trajectories without being limited by the importance and decay factors of the vertices, at the cost of sacrificing bounds on the optimality gap with respect to the general problem.

Finally, we can find other similar solutions that deal with the online computation of open paths at particular times considering a finite horizon. A prioritized A* algorithm together with barrier functions was used in Ma et al. (2018). An integer program applied over a receding planning horizon was solved in Ahmadzadeh et al. (2007) to find the paths that maximize spatiotemporal coverage. Graham and Cortés (2012) obtained an approximate solution for a dynamic program over a finite time horizon, whose optimization criterion is the minimum uncertainty of a field estimate. The local path planning algorithm *TangentBug* was used by Breitenmoser et al. (2010b) to plan the trajectories to the Voronoi centroids in a non-convex environment. A line strip over a triangular mesh of equally important points was used by Breitenmoser et al. (2014). Palacios-Gasós et al. (2017) used a fast marching method to compute optimal paths in terms of coverage quality to badly covered areas. The main difference of our solution with respect to these approaches is that we consider an optimization of the coverage cost per node visited.

1.3. Contributions

In this article, we propose a solution to persistently cover a complex, non-convex environment with a team of robots. Our solution works in two steps: the first allows the robots to find an equitable partition of the environment in a distributed way, while the second deals with the problem of individual online planning inside the partition using a graph representation.

The partitioning method builds upon that presented by Pavone et al. (2011). The main contribution is the extension of the method to a more general continuous environment formulation, including complex, non-convex ones, by developing the algorithm in terms of geodesic distance. This formulation allows the partition to be consistent with the shape of the environment. In addition, we present two extensions of the algorithm to improve the partitions. The first is designed to reduce the amount of disconnected partitions, in the sense that every partition is represented by a single connected component in the whole environment, avoiding the need to walk through other partitions to cover it. The second extension allows the algorithm to compute equitable partitions weighted by the different capabilities of each robot. In all cases, convergence is guaranteed.

The second part of our solution deals with the problem of individual path planning for persistent coverage inside each partition. The method first generates a graph, discretizing the partition in a grid based on the coverage capabilities of the robot that will cover it. Then, an online planner is run asynchronously by each robot, where at each planning time, the coverage error at the graph vertices is assigned as weights of the corresponding edges. The planner finds then the path that maximizes the coverage quality per node visited using a variation of the Bellman–Ford algorithm (Cormen, 2009). This allows the planning to be efficiently computed online with the particular advantages of using open paths that, combined, resemble a sweeping strategy.

The remainder of the article is structured as follows. In Section 2, we introduce the problem formulation. In Section 3, we present the equitable partitioning method. We enhance the method to reduce disconnections in the partitions and consider different coverage capabilities in Section 4. In Section 5 the strategy to plan paths and individually cover each partition is presented. Finally, we present simulation results in Section 6, experimental results in Section 7, and conclusions in Section 8.

2. Problem formulation

Let $\mathcal{Q} \subset \mathbb{R}^2$ be a known bounded environment, possibly non-convex, and $\mathcal{Q}_f \subseteq \mathcal{Q}$ the set of points in $\mathcal{Q}$ that are not obstacles. The objective of persistent coverage in this environment is to maintain the actual coverage level, represented by a time-varying field, $Z(\mathbf{q}, k) \geqslant 0$, as close as possible to a desired coverage objective, denoted as $Z^*(\mathbf{q}) > 0, \forall \mathbf{q} \in \mathcal{Q}_f$. The importance of maintaining the coverage of each point is represented by a constant weighting function $\Phi(\mathbf{q}) \in (0, 1]$.

The coverage level deteriorates over time with a constant decay rate, $d(\mathbf{q})$, with $0 < d(\mathbf{q}) < 1$, according to the following recurrence equation:

$$Z(\mathbf{q}, k) = d(\mathbf{q}) Z(\mathbf{q}, k-1) + \alpha(\mathbf{q}, k) \tag{1}$$

where $\alpha(\mathbf{q}, k)$ is the total coverage action that a team of robots applies at point $\mathbf{q}$ of the environment at time instant k in order to satisfy the coverage objective.

The team is composed of N robots, located at positions $\mathbf{p}_i(k), i \in \{1, \dots, N\}$. The robots are capable of increasing the coverage in an area $\mathbf{\Omega}_i(\mathbf{p}_i(k)) \subseteq \mathcal{Q}_f$, that we call the coverage area. This area is bounded by a circle of radius r_i^{cov}, although it is not necessarily circular, convex, or equal for all the robots. We denote by $\alpha_i(\mathbf{q}, k)$ the coverage value applied by the robot at position $\mathbf{q}$. This value only depends on the physical properties of the robot, and outside of $\mathbf{\Omega}_i(\mathbf{p}_i(k))$ is equal to zero.

In addition, we consider that the robots have the capability to adjust their production at each point of $\mathbf{\Omega}_i(\mathbf{p}_i(k))$ by a coverage gain, $0 \leqslant \rho_i(\mathbf{q}, k) \leqslant 1$. In this way, the total coverage action is determined by

$$\alpha(\mathbf{q}, k) = \sum_{i \in \{1, \dots, N\}} \rho_i(\mathbf{q}, k) \alpha_i(\mathbf{q}, k) \tag{2}$$

From now on, to simplify the notation we omit the spatial dependencies, $\mathbf{q}$, $\mathbf{p}_i(k)$, in all the functions where it can be inferred by the context, e.g., $Z(\mathbf{q}, k) \equiv Z(k)$, $\alpha(k) \equiv \alpha(\mathbf{q}, k)$, $\mathbf{\Omega}_i(\mathbf{p}_i(k)) \equiv \mathbf{\Omega}_i(k)$, etc.

In this article, we consider that the coverage gain is adjusted as in Palacios-Gasós et al. (2017):

$$\rho_i(k) = \begin{cases} 1, & \text{if } \rho_i^*(k) \geqslant 1 \\ \rho_i^*(k), & \text{if } 0 < \rho_i^*(k) < 1 \\ 0, & \text{if } \rho_i^*(k) \leqslant 0 \end{cases} \tag{3}$$

with

$$\rho_i^*(k) = \frac{\int_{\mathbf{\Omega}_i(k)} \Phi \cdot (Z^* - d\, Z(k-1)) \alpha_i(k) \mathrm{d}\mathbf{q}}{\int_{\mathbf{\Omega}_i(k)} \cdot \Phi \alpha_i(k)^2 \mathrm{d}\mathbf{q}} \tag{4}$$

Intuitively, $\rho_i^*(k)$ represents the fraction of the coverage action, $\alpha_i(k)$, required to reach the desired level, Z^*. Therefore, what this action is doing is basically to adjust the coverage production to prevent coverage values above Z^*.

Now, we are interested in finding trajectories, $\Gamma_i(k)$, for all the robots that make $Z(k)$ as close as possible to Z^* for the whole environment and every k. Addressing this problem as a whole, e.g., as an optimization problem of the coverage error over some time horizon, is only possible for small teams of robots and very coarse discretized environments, as can be seen in Palacios-Gasós et al. (2016b). The reason for this is that we need to account for the actions of multiple robots in the same environment, as well as the influence in the future coverage values of previous actions of the computed trajectories. Therefore, in this article, we adopt a divide-and-conquer strategy to address the problem in two stages.

In a first stage, described in Sections 3 and 4, we focus on dividing the environment into N disjoint regions, one for each robot, such that the work needed to cover them, in terms of coverage action, is equal. In this way, we simplify the problem of finding N paths into N individual problems of finding a single path in a smaller environment. This partition can be computed once offline and be assumed constant afterwards.

In a second stage, detailed in Section 5, we propose an online planning algorithm that each robot executes locally within its partition to obtain the trajectories $\Gamma_i(k)$ that minimize the *current* coverage error. The most important simplification we make to keep the problem tractable is to disregard in the planning algorithm the influence of previous actions of the computed trajectory into the coverage value.

3. Equitable partitioning

The first step of our strategy to solve the persistent coverage problem is to divide the environment into as many equitable regions as robots. The goal is that each partition requires an amount of work proportional to the capabilities of its assigned robot with respect to the rest of the team and to the importance of its points. To this end, we introduce the notions of power diagram and geodesic distance, define the importance-weighted workload of the agents in terms of the coverage problem and detail our distributed partitioning algorithms.

3.1. Power diagrams and geodesic distance

Let us first define a set of points, $\mathcal{G} = \{\mathbf{g}_1, \ldots, \mathbf{g}_N\}$, with $\mathbf{g}_i, \mathbf{g}_j \in \mathcal{Q}_f$ for all i and j and $\mathbf{g}_i \neq \mathbf{g}_j$ if $i \neq j$, which will act as the generators of the partitions. Our algorithm to find the equitable partitions is based on power diagrams (Aurenhammer, 1987), that are a generalization of the Voronoi diagrams. Instead of assigning the points to each partition according to the Euclidean distance to the generator, the power diagrams use the squared distance minus a certain weight. If we let $\mathbf{w} = \{w_1, \ldots, w_N\}$ be the set of power weights associated to the partitions, we can formally define the partitions obtained from the power diagram as

$$\mathcal{P}_i(\mathbf{w}) = \{\mathbf{q} \in \mathcal{Q}_f | d(\mathbf{q}, \mathbf{g}_i)^2 - w_i \leqslant d(\mathbf{q}, \mathbf{g}_j)^2 - w_j\} \tag{5}$$

In order to apply this partitioning to a non-convex environment, we make use of the geodesic distance, $d_g(\mathbf{q}, \mathbf{g})$, the length of the shortest path between two points of the environment. Assuming that the environment is composed of polygonal obstacles, the shortest path between every two points is the concatenation of a set of straight lines connecting the two points and some vertices of the obstacles, $\{\mathbf{q}, \mathbf{h}^1_{\mathbf{q},\mathbf{g}}, \mathbf{h}^2_{\mathbf{q},\mathbf{g}}, \ldots, \mathbf{h}^{l_{\mathbf{q},\mathbf{g}}}_{\mathbf{q},\mathbf{g}}, \mathbf{g}\}$, where $l_{\mathbf{q},\mathbf{g}}$ is the number of obstacle vertices that define such path. Therefore, the geodesic distance can be decomposed in the summation of the Euclidean distances between each pair of consecutive points:

$$\begin{aligned} d_g(\mathbf{q}, \mathbf{g}) = \ \| \mathbf{q} - \mathbf{h}^1_{\mathbf{q},\mathbf{g}} \| + \| \mathbf{h}^1_{\mathbf{q},\mathbf{g}} - \mathbf{h}^2_{\mathbf{q},\mathbf{g}} \| + \cdots \\ + \| \mathbf{h}^{l_{\mathbf{q},\mathbf{g}}}_{\mathbf{q},\mathbf{g}} - \mathbf{g} \| \end{aligned} \tag{6}$$

Using this metric, the boundary between two partitions is

$$\Delta_{ij} = \{\mathbf{q} \in \mathcal{Q}_f | d_g(\mathbf{q}, \mathbf{g}_i)^2 - w_i = d_g(\mathbf{q}, \mathbf{g}_j)^2 - w_j\} \tag{7}$$

According to this definition, the neighbors of a generator are

$$N_i = \{j \in \{1, \ldots, N\} \backslash i | \Delta_{ij} \neq \emptyset\} \tag{8}$$

We assume that each robot can communicate with robots of neighboring regions.

3.2. Importance-weighted workload

The partitions that are assigned to the robots have to be equitable in terms of the work that has to be carried out inside them. At the same time, they have to take into account the importance of the coverage of each point. Therefore, it is essential to define the workload in terms of the problem's variables. The definition of the work at each point of the environment that we consider is

$$\lambda(\mathbf{q}) = \Phi(1 - d)Z^* \tag{9}$$

Strictly speaking, $(1 - d)Z^*$ is the actual workload of each point. It represents the coverage that decays at each time if the point has reached the desired level, i.e., in the steady state. Nevertheless, we weight it with the importance to allow the robots to spend more time covering the most important points, at the expense of reducing the attention to less important points. This may lead to partitions in which $(1 - d)Z^*$ is not equitable but the importance of the work of each partition is. From now on, we refer to this importance-weighted workload simply as workload.

The workload inside each partition can be calculated as

$$\lambda_{\mathcal{P}_i(\mathbf{w})} = \int_{\mathcal{P}_i(\mathbf{w})} \lambda(\mathbf{q}) d\mathbf{q} \tag{10}$$

Both the importance and the decay take values between 0 and 1. In contrast, the desired coverage level is expressed in general in coverage units and, therefore, we normalize the entire workload by $\int_{Q_f} \Phi(1-d)Z^* d\mathbf{q}$ to obtain it in percentages.

3.3. Distributed algorithm for equitable partitioning

The power diagram can be seen as a relation between the weights and the regions that belong to their corresponding generators. Therefore, we minimize a cost function whose minima correspond to sets of weights that lead to an equitable partition:

$$H(\mathbf{w}) = \sum_{i=1}^{N} \frac{1}{\lambda_{\mathcal{P}_i(\mathbf{w})}} \tag{11}$$

One can see that this function reaches its minimum when the workload of all the partitions is the same. It is also important to remark that such a minimum can always be found, see Pavone et al. (2011). From now on, we omit the dependencies of H and $\mathcal{P}_i$ with $\mathbf{w}$ for the sake of clarity.

To minimize this function distributively, each robot is in charge of a generator, that we locate at the robot's position for simplicity, and its associated weight, initially set to zero. The evolution of the weight is driven to achieve equity using information from the neighbors in the following control law.

Theorem 3.1. *The power diagram generated by $\mathcal{G}$ and $\mathbf{w}$ converges to an equitable power diagram under the distributed control law*

$$\dot{w}_i = -k_w \frac{\partial H}{\partial w_i} \tag{12}$$

with k_w, a positive gain, and

$$\frac{\partial H}{\partial w_i} = \sum_{j\in N_i}\left(\frac{1}{\lambda^2_{\mathcal{P}_j}} - \frac{1}{\lambda^2_{\mathcal{P}_i}}\right)\int_{\Delta_{ij}} \frac{\lambda(\mathbf{q})}{\| n_{ij}'(\mathbf{q}) \|} d\mathbf{q} \tag{13}$$

where $n_{ij}'(\mathbf{q})$ is the outward normal to the boundary between the partitions i and j at point $\mathbf{q} \in \Delta_{ij}$.

Proof. In the first place we develop the gradient of the cost function with respect to a single weight w_i:

$$\frac{\partial H}{\partial w_i} = -\frac{1}{\lambda^2_{\mathcal{P}_i}}\frac{\partial \lambda_{\mathcal{P}_i}}{\partial w_i} - \sum_{j\in N_i}\frac{1}{\lambda^2_{\mathcal{P}_j}}\frac{\partial \lambda_{\mathcal{P}_j}}{\partial w_i} \tag{14}$$

It only depends on the neighboring partitions since a variation on w_i only affects them. The derivative of the workload in the partition of robot i is

$$\frac{\partial \lambda_{\mathcal{P}_i}}{\partial w_i} = \frac{\partial}{\partial w_i}\int_{\mathcal{P}_i} \lambda(\mathbf{q}) d\mathbf{q} \tag{15}$$

It can be transformed using the following result associated with the divergence theorem:

$$\frac{\partial}{\partial w_i}\int_{\mathcal{P}_i} \lambda(\mathbf{q}) d\mathbf{q} = \int_{\partial\mathcal{P}_i}\left(\frac{\partial \mathbf{q}}{\partial w_i}\cdot n_{\partial\mathcal{P}_i}(\mathbf{q})\right)\lambda(\mathbf{q}) d\mathbf{q} \tag{16}$$

where $n_{\partial\mathcal{P}_i}(\mathbf{q})$ represents the unit outward normal to the boundary of the partition, $\partial\mathcal{P}_i$, at point $\mathbf{q}$. It results in

$$\begin{aligned}\frac{\partial \lambda_{\mathcal{P}_i}}{\partial w_i} =& \sum_{j\in N_i}\int_{\Delta_{ij}}\left(\frac{\partial \mathbf{q}}{\partial w_i}\cdot n_{ij}(\mathbf{q})\right)\lambda(\mathbf{q}) d\mathbf{q} \\ &+ \sum_{j\in N_i}\int_{\Delta_i^{Q_f}}\left(\frac{\partial \mathbf{q}}{\partial w_i}\cdot n_{ij}(\mathbf{q})\right)\lambda(\mathbf{q}) d\mathbf{q}\end{aligned} \tag{17}$$

where $\Delta_i^{Q_f}$ is the boundary between the partition and the environment and $n_{ij}(\mathbf{q})$ is the unit normal outward this boundary or the boundary between partitions i and j, Δ_{ij}. The second term is always zero either because $\Delta_i^{Q_f} = \emptyset$ or because a variation on the weight does not affect the boundary between the partition and the environment, i.e., $\partial\mathbf{q}/\partial w_i = 0$. Applying the same procedure to $\partial\lambda_{\mathcal{P}_j}/\partial w_i$ and introducing in (14), we have

$$\begin{aligned}\frac{\partial H}{\partial w_i} =& -\frac{1}{\lambda^2_{\mathcal{P}_i}}\sum_{j\in N_i}\int_{\Delta_{ij}}\left(\frac{\partial \mathbf{q}}{\partial w_i}\cdot n_{ij}(\mathbf{q})\right)\lambda(\mathbf{q}) d\mathbf{q} \\ &-\sum_{j\in N_i}\frac{1}{\lambda^2_{\mathcal{P}_j}}\int_{\Delta_{ij}}\left(\frac{\partial \mathbf{q}}{\partial w_i}\cdot n_{ji}(\mathbf{q})\right)\lambda(q) d\mathbf{q}\end{aligned} \tag{18}$$

Noting that $n_{ji}(\mathbf{q}) = -n_{ij}(\mathbf{q})$, we only have to calculate the scalar product $\partial\mathbf{q}/\partial w_i \cdot n_{ij}(\mathbf{q})$. The first term can be obtained by deriving the condition of the boundary points (7) with respect to the power weight,

$$\frac{\partial}{\partial w_i}(d_g(\mathbf{q},\mathbf{g}_i)^2 - w_i) = \frac{\partial}{\partial w_i}(d_g(\mathbf{q},\mathbf{g}_j)^2 - w_j), \tag{19}$$

that leads to

$$2d_g(\mathbf{q},\mathbf{g}_i)\frac{\partial d_g(\mathbf{q},\mathbf{g}_i)}{\partial \mathbf{q}}\cdot\frac{\partial \mathbf{q}}{\partial w_i} - 1 = 2d_g(\mathbf{q},\mathbf{g}_j)\frac{\partial d_g(\mathbf{q},\mathbf{g}_j)}{\partial \mathbf{q}}\cdot\frac{\partial \mathbf{q}}{\partial w_i} \tag{20}$$

The partial derivative of the geodesic distance with respect to the boundary point $\mathbf{q}$ only affects the first term on the right-hand side of (6):

$$\frac{\partial d_g(\mathbf{q},\mathbf{g}_i)}{\partial \mathbf{q}} = \frac{\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i}}{\| \mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_i} \|} \tag{21}$$

Analogously for the other generator we have

$$\frac{\partial d_g(\mathbf{q},\mathbf{g}_j)}{\partial \mathbf{q}} = \frac{\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j}}{\| \mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j} \|} \tag{22}$$

Note that these derivatives have two components, x and y, and that they appear in a scalar product in (20). Therefore, introducing (21) and (22) in (20), we have

$$2d_g(\mathbf{q},\mathbf{g}_i)\left(\frac{(\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i})_x}{\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i}\|}\frac{\partial\mathbf{q}_x}{\partial w_i}+\frac{(\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i})_y}{\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i}\|}\frac{\partial\mathbf{q}_y}{\partial w_i}\right)-1$$
$$=2d_g(\mathbf{q},\mathbf{g}_j)\left(\frac{(\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j})_x}{\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j}\|}\frac{\partial\mathbf{q}_x}{\partial w_i}+\frac{(\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j})_y}{\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j}\|}\frac{\partial\mathbf{q}_y}{\partial w_i}\right) \tag{23}$$

and, regrouping the terms,

$$\left[2d_g(\mathbf{q},\mathbf{g}_i)\frac{(\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i})_x}{\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i}\|}-2d_g(\mathbf{q},\mathbf{g}_j)\frac{(\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j})_x}{\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j}\|},\right.$$
$$\left.2d_g(\mathbf{q},\mathbf{g}_i)\frac{(\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i})_y}{\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i}\|}-2d_g(\mathbf{q},\mathbf{g}_j)\frac{(\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j})_y}{\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j}\|}\right]\cdot\frac{\partial\mathbf{q}}{\partial w_i}=1 \tag{24}$$

To obtain the normal to the boundary between two partitions, $n_{ij}(\mathbf{q})$, we make use of the property of the gradient of a function, that is always perpendicular to the level curves of the function. To this end, we transform the equation that defines the boundary points (7) into a function,

$$f(\mathbf{q})=d_g(\mathbf{q},\mathbf{g}_i)^2-w_i-d_g(\mathbf{q},\mathbf{g}_j)^2+w_j \tag{25}$$

Then, we calculate the gradient at the points that satisfy $f(\mathbf{q})=0$, that is, $\mathbf{q}\in\Delta_{ij}$, for the x component,

$$\frac{\partial f(\mathbf{q})}{\partial\mathbf{q}_x}=2d_g(\mathbf{q},\mathbf{g}_i)\frac{\partial d_g(\mathbf{q},\mathbf{g}_i)}{\partial\mathbf{q}_x}-2d_g(\mathbf{q},\mathbf{g}_j)\frac{\partial d_g(\mathbf{q},\mathbf{g}_j)}{\partial\mathbf{q}_x}$$
$$=2d_g(\mathbf{q},\mathbf{g}_i)\frac{\partial\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i}\|}{\partial\mathbf{q}_x}-2d_g(\mathbf{q},\mathbf{g}_j)\frac{\partial\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j}\|}{\partial\mathbf{q}_x}$$
$$=2d_g(\mathbf{q},\mathbf{g}_i)\frac{(\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i})_x}{\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_i}\|}-2d_g(\mathbf{q},\mathbf{g}_j)\frac{(\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j})_x}{\|\mathbf{q}-h^1_{\mathbf{q},\mathbf{g}_j}\|} \tag{26}$$

and, similarly, for the y component. Therefore, the normal to the boundary is

$$n_{ij}'(\mathbf{q})=\left(\frac{\partial f(\mathbf{q})}{\partial\mathbf{q}_x},\frac{\partial f(\mathbf{q})}{\partial\mathbf{q}_y}\right) \tag{27}$$

and the unit normal is

$$n_{ij}(\mathbf{q})=\frac{n_{ij}'(\mathbf{q})}{\|n_{ij}'(\mathbf{q})\|} \tag{28}$$

Introducing the last three equations in (24), we obtain

$$\frac{\partial\mathbf{q}}{\partial w_i}\cdot n_{ij}(\mathbf{q})=\frac{1}{\|n_{ij}'(\mathbf{q})\|} \tag{29}$$

and replacing in (17) we obtain the complete formulation of the gradient stated in (13). Finally, the proof of the convergence is equivalent to Theorem 3.7 of Pavone et al. (2011), completing the proof of this theorem. □

Remark 3.2. *The main differences of our proof with respect to that presented in Pavone et al. (2011) reside in the derivation of the geodesic distance and in the calculation of the normal to the boundary points. These two key modifications generalize the algorithm to any kind of non-convex environments, yet still giving the same solution for convex ones.*

It is also important to highlight that the gradient is still distributed since each robot can update its own weight and, therefore, its own partition, by only communicating with its neighbors. In addition, note that the gradient has no physical representation in the real environment since it only affects the weights associated with the generators.

4. Partition enhancements

The partitions obtained with the algorithm described in the previous section are proven to be equitable. However, for for complex environments, they might ones, they might be disconnected, in the sense that in order to reach some points of the partition, a robot needs to go through the partitions of other robots. This may affect significantly the persistent coverage of the environment. Moreover, they do not take into account the different capabilities that each robot might have. In this section, we present two enhancements of our partitioning algorithm: one where we control the positions of the generators to reduce possible disconnections and other to account for energy and coverage power of the robots.

4.1. Connectivity-aware partitioning

Let us begin by defining the connected components, $\mathcal{P}_i^\ell(\mathbf{w})$, $\ell\in\{1,\ldots,L_i\}$, $L_i\in\mathbb{Z}+$, that form a partition:

$$\mathcal{P}_i=\bigcup_{\ell=1}^{L_i}\mathcal{P}_i^\ell \tag{30}$$

Between these connected components we select the one with the highest workload:

$$\mathcal{P}_i^c=\underset{\mathcal{P}_i^\ell}{\operatorname{argmax}}\,\lambda_{\mathcal{P}_i^\ell}. \tag{31}$$

Note that for connected partitions $\mathcal{P}_i^c$ coincides with $\mathcal{P}_i$. Then, we calculate the center of mass of $\mathcal{P}_i^c$ as

$$\mathbf{g}_i^*=\frac{1}{\lambda_{\mathcal{P}_i^c}}\int_{\mathcal{P}_i^c}\mathbf{q}\lambda(\mathbf{q})d\mathbf{q}. \tag{32}$$

This point is the best point of the environment in which the generator $\mathbf{g}_i$ could be located to favor the connectivity of its partition. However, it cannot be moved straightaway in the direction $\overline{\mathbf{g}_i\mathbf{g}_i^*}$ since there are two restrictions that must be taken into account. The first one is that the motion of the generator must not hinder the evolution of the weights towards the equitable partition. Therefore, such motion can only be executed if it contributes to the minimization of (11) or, at least, it is not detrimental. The second restriction is that in a non-convex environment it has to follow the geodesic path from its current position to the center of mass to avoid sudden changes in the shape, size and workload of the partitions. If we call γ_i the shortest path from $\mathbf{g}_i$ to $\mathbf{g}_i^*$, we want the generator to move in the direction $\gamma_i(\mathbf{g}_i)$, that is, the direction of the path at $\mathbf{g}_i$. In this context, the algorithm that we propose is presented in the following theorem.

Theorem 4.1. *The power diagram generated by $\mathcal{G}$ and $\mathbf{w}$ converges to an equitable power diagram under the distributed control law*

$$\dot{w}_i = -k_w \frac{\partial H}{\partial w_i} \tag{33a}$$

$$\dot{\mathbf{g}}_i = k_g f_{\text{sat}}\left(\overline{\mathbf{g}_i\mathbf{g}_i^*} \cdot \frac{-\partial H}{\partial \mathbf{g}_i}\right)\gamma_i(\mathbf{g}_i) \tag{33b}$$

with k_w, k_g, positive gains,

$$f_{\text{sat}}(x) = \begin{cases} 0, & \forall x \leq 0 \\ \exp\left(\frac{-1}{(k_{\text{sat}}x)^2}\right), & \forall x > 0 \end{cases} \tag{34}$$

a saturation function with $k_{\text{sat}} > 0$, and

$$\frac{\partial H}{\partial \mathbf{g}_{i,k}} = \sum_{j \in N_i}\left(\frac{1}{\lambda_{\mathcal{P}_j}^2} - \frac{1}{\lambda_{\mathcal{P}_i}^2}\right)\int_{\Delta_{ij}} \frac{(\mathbf{h}_{\mathbf{q},\mathbf{g}_i}^{l_{\mathbf{q},\mathbf{g}_i}} - \mathbf{g}_i)_k}{\| \mathbf{h}_{\mathbf{q},\mathbf{g}_i}^{l_{\mathbf{q},\mathbf{g}_i}} - \mathbf{g}_i \|}\frac{\lambda(\mathbf{q})}{\| n_{ij}'(\mathbf{q}) \|}\, d\mathbf{q} \tag{35}$$

where $n_{ij}'(\mathbf{q})$ is the outward normal to the boundary between the partitions i and j at point $\mathbf{q}$.

Proof. The convergence to an equitable power diagram is guaranteed under the first part of the control law (33) in Theorem 3.1. In this proof, we only show that the second part of the control law (34) does not counteract this convergence.

The direction of motion of the generators that maximizes the improvement of the cost function is the gradient of the cost function with respect to $\mathbf{g}_i$, i.e., $\partial H/\partial \mathbf{g}_i$. This gradient is obtained in the same way as with respect to w_i in the proof of Theorem 3.1 with only two particularities. The derivatives of the geodesic distances with respect to $\mathbf{g}_{i,x}$ are

$$\begin{aligned}\frac{\partial d_g(\mathbf{q},\mathbf{g}_i)}{\partial \mathbf{g}_{i,x}} &= \frac{\partial \| \mathbf{q} - \mathbf{h}_{\mathbf{q},\mathbf{g}_i}^1 \|}{\partial \mathbf{g}_{i,x}} + \ldots + \frac{\partial \| \mathbf{h}_{\mathbf{q},\mathbf{g}}^{l_{\mathbf{q},\mathbf{g}_i}} - \mathbf{g}_i \|}{\partial \mathbf{g}_{i,x}} \\ &= \frac{\mathbf{q} - \mathbf{h}_{\mathbf{q},\mathbf{g}_i}^1}{\| \mathbf{q} - \mathbf{h}_{\mathbf{q},\mathbf{g}_i}^1 \|}\cdot\frac{\partial \mathbf{q}}{\partial \mathbf{g}_{i,x}} - \frac{(\mathbf{h}_{\mathbf{q},\mathbf{g}}^{l_{\mathbf{q},\mathbf{g}_i}} - \mathbf{g}_i)_x}{\| \mathbf{h}_{\mathbf{q},\mathbf{g}}^{l_{\mathbf{q},\mathbf{g}_i}} - \mathbf{g}_i \|}\end{aligned} \tag{36}$$

$$\frac{\partial d_g(\mathbf{q},\mathbf{g}_j)}{\partial \mathbf{g}_{i,x}} = \frac{\partial \| \mathbf{q} - \mathbf{h}_{\mathbf{q},\mathbf{g}_j}^1 \|}{\partial \mathbf{g}_{i,x}} = \frac{\mathbf{q} - \mathbf{h}_{\mathbf{q},\mathbf{g}_j}^1}{\| \mathbf{q} - \mathbf{h}_{\mathbf{q},\mathbf{g}_j}^1 \|}\cdot\frac{\partial \mathbf{q}}{\partial \mathbf{g}_{i,x}} \tag{37}$$

and equivalent for $\mathbf{g}_{i,y}$. Note that in the first step of the first derivative all the intermediate terms represented with the dots are equal to zero since they do not depend on $\mathbf{g}_{i,x}$ or $\mathbf{q}$. The second particularity is that in the end we obtain

$$\frac{\partial \mathbf{q}}{\partial \mathbf{g}_{i,x}}\cdot n_{ij} = \frac{1}{\| n_{ij}'(\mathbf{q}) \|}\frac{(\mathbf{h}_{\mathbf{q},\mathbf{g}}^{l_{\mathbf{q},\mathbf{g}_i}} - \mathbf{g}_i)_x}{\| \mathbf{h}_{\mathbf{q},\mathbf{g}}^{l_{\mathbf{q},\mathbf{g}_i}} - \mathbf{g}_i \|} \tag{38}$$

which leads to (35).

The saturation function in (33b) compares the direction of the gradient with the direction from the generator to the centroid of the biggest connected component. If both directions are approximately aligned, i.e., the movement of the generator to the centroid favors the convergence to the equitable partition, the generator is able to follow the geodesic path to the centroid according to $\gamma_i(\mathbf{g}_i)$. Otherwise, if the movement to the centroid hinders the convergence, $f_{\text{sat}} = 0$ and the generator does not move. □

Remark 4.2. *Even though Theorem 4.1 does not provide formal guarantees on connectedness, since the movement of the generator is towards the centroid of the component with the biggest workload, we can assert that this partition methodology favors connectivity of the regions.*

Although the generators move towards their respective centroids, the convergence of the control law is to an equitable partition. Therefore, the generator may either reach the centroids or stop somewhere in the geodesic path to such point when an equitable partition is achieved.

It should also be noted that the motion of the generators in this control law is independent on the actual motion of the robots, as discussed in Section 5. This also means that, if the computed partition is not satisfactory, e.g., is still disconnected, the algorithm can be run again with different initial conditions until a connected partition is obtained. Similarly, the robots can "trade" small regions of equivalent workload so that equity is preserved, but in such a way that the partitions are improved, e.g., connected or associated with physical rooms. We provide a more detailed empirical analysis in Section 6 where we analyze how often the resulting partitions are not connected, leaving more sophisticated partition mechanisms, such as trading, for future work.

4.2. Capability-aware partitioning

The algorithm to find the equitable partition that we have introduced simply divides the environment into regions with the same workload. It is noteworthy that the cost function can be easily extended with gains that reflect different coverage capabilities for each robot. If we consider

$$H(\mathbf{w}) = \sum_{i=1}^{N} \frac{c_i}{\lambda_{\mathcal{P}_i(\mathbf{w})}} \tag{39}$$

as an alternative cost function, where c_i is a constant value. Since this term is constant with respect to w_i, we can still use the control law of the previous section. The only difference is that each robot now will have a different gain driving the evolution of its own partition weight and generator. The partitions obtained in this case will satisfy that

$$\frac{c_i}{\lambda_{\mathcal{P}_i(\mathbf{w})}} = \frac{c_j}{\lambda_{\mathcal{P}_j(\mathbf{w})}} \tag{40}$$

As an example, the constants, c_i can be chosen by

$$c_i = \int_{\mathbf{\Omega}_i(\mathbf{p})} \alpha_i(\mathbf{q}, \mathbf{p}) \mathrm{d}\mathbf{q} \tag{41}$$

where $\mathbf{p}$ is a virtual position to integrate the maximum coverage that each agent can provide. With this value, the partition of a robot that is able to produce more coverage action will be bigger than the partition of another robot that is able to produce less action.

5. Finite-horizon, graph-based coverage planning

The second part of our persistent solution is in charge of planning online coverage paths for the robots. To do this, we first define the coverage error at each location and time instant,

$$e(\mathbf{q}, k) = \Phi(\mathbf{q})(Z^*(\mathbf{q}) - Z(\mathbf{q}, k)) \tag{42}$$

where we allow positive values for under-covered points and negative values for over-covered ones. We assume that the initial coverage value for each point is equal to zero. Similarly, we define the *quadratic* coverage error over the whole environment as

$$f(k) = \int_{Q} e^2(\mathbf{q}, k) d\mathbf{q} \tag{43}$$

in this case to emphasize that it is equally bad to be above and below the desired coverage level.

Ideally, the best coverage paths would be the Γ_i that minimize $f(k)$ for the whole duration of the coverage task, e.g., minimize $\sum_k f(k)$. However, owing to the coverage deterioration over time, finding a solution to this optimization problem is computationally intractable even for small teams, as shown in Palacios-Gasós et al. (2016b). Therefore, in the following we are going to propose an online planning solution that minimizes the quadratic coverage error of a subset of points in $f(k)$ at each planning time.

In particular, we are going to generate a graph of locations that considers the coverage capabilities of each robot inside the previously computed equitable partitions. In this way, we can reduce the planning problem to that of finding the shortest path in a graph, which can be computed in polynomial time very efficiently. In addition, the construction of the graph favors the planning of sweep-like paths, that are very suitable for coverage problems. In the rest of the section, we explain in detail the whole procedure.

5.1. Graph construction based on sweep-like paths

Let us first define $\mathcal{P}'_i$ as the subset of locations from $\mathcal{P}_i$ where the robot can be located without colliding with an obstacle.

Inside the partitions we separate the construction of the path graph in two parts: grid and boundary. The aim of the first part is to define paths that allow the robot to perform a complete sweep of the partition and the second part is intended to cover the entire boundary of the partition.

For the first part, we intersect a grid of points with the partition:

$$\mathcal{V}_i^g = \{\mathbf{v} \in \mathcal{P}'_i | \mathbf{v} = (x_i^0 + k d_i, y_i^0 \pm \ell d_i), k, \ell \in \mathbb{N}_0\} \tag{44}$$

where $\mathbf{v}_i^0 = (x_i^0, y_i^0) \in \mathcal{P}'_i$ is the left-most point of $\mathcal{P}'_i$ and d_i is the optimal separation between consecutive sweeping lines. Note that the selection of $\mathbf{v}_i^0$ is arbitrary since the grid is fixed by d_i and the boundary is covered by the second part of the graph. The vertices in $\mathcal{V}_i^g$ are connected by an edge if they are at a distance equal to d_i. Formally,

$$\mathcal{E}_i^g = \{(\mathbf{v}_1, \mathbf{v}_2) | \parallel \mathbf{v}_1 - \mathbf{v}_2 \parallel = d_i, \forall \mathbf{v}_1, \mathbf{v}_2 \in \mathcal{V}_i^g\} \tag{45}$$

Figure 2a shows an example of the vertices and edges that correspond to the grid part of the path graph.

The optimal separation between two horizontal or two vertical sweeping lines, d_i, depends on the shape of the coverage area and the production function. It can be obtained by minimizing

$$\int_{-r_i^{\text{cov}}}^{r_i^{\text{cov}}} \int_{-r_i^{\text{cov}}}^{d_i + r_i^{\text{cov}}} (\overline{\alpha} - (\alpha_i((x,y),(0,0)) + \alpha_i((x,y),(0,d_i))))^2 \, dy dx \tag{46}$$

with

$$\overline{\alpha} = 2 \frac{\int_{\Omega_i(\mathbf{p}_i)} \alpha_i(\mathbf{p}_i) d\mathbf{q}}{\int_{-r_i^{\text{cov}}}^{r_i^{\text{cov}}} \int_{-r_i^{\text{cov}}}^{d_i + r_i^{\text{cov}}} dy dx} \tag{47}$$

The idea behind (46) is illustrated in Figure 1. The two circumferences represent the areas that the robot can cover when it is located at $\mathbf{p}_i^1 = (0, 0)$ and $\mathbf{p}_i^2 = (0, d_i)$, respectively. The two locations correspond to consecutive sweeps in the x direction, i.e., two vertices in $\mathcal{V}_i^g$ connected by an edge in $\mathcal{E}_i^g$. Note that $\Omega_i(\mathbf{p}_i)$ does not have to be circular, but only bounded by the circumference, and that $\alpha_i(\mathbf{p}_i)$ can be any function. The optimal distance between these parallel sweeps, d_i, is the one that optimizes the sum of the coverage provided from these two positions. In particular, the goal of (46) is that the same coverage is provided to each point that can be covered by the robot from any of the positions. This coverage is represented by $\overline{\alpha}$, that is the sum of the coverage from the two positions divided by the total area. Therefore, $\overline{\alpha}$ can also be seen as an average coverage level. Equation (46) computes the quadratic difference between the coverage provided to each point, i.e., the sum of α_i from p_i^1 and p_i^2, and this average to obtain a d_i that keeps the coverage of all points as close as possible to $\overline{\alpha}$.

For the second part of the graph, we represent the boundary as $\partial\mathcal{P}_i'$ and locate a finite number of vertices over it as follows:

$$\mathcal{V}_i^b = \{\mathbf{v} \in \partial\mathcal{P}_i' | \mathbf{v}_x = x_i^0 \pm k\,\frac{d_i}{2} \vee \mathbf{v}_y = y_i^0 \pm \ell\,\frac{d_i}{2}, k, \ell \in \mathbb{Z}\} \tag{48}$$

They are located at the intersections of a square grid of size $d_i/2$ with the boundary, to provide a sufficiently fine discretization. These vertices are linked by the edges

$$\mathcal{E}_i^b = \{(\mathbf{v}_1, \mathbf{v}_2) | \mathbf{v}_1 \text{and} \mathbf{v}_2 \text{ are consecutive over } \partial\mathcal{P}_i'\} \tag{49}$$

These edges allow the robots to follow the entire boundary of the partition as shown in Figure 2b.

Finally, we merge the two parts by including an additional set of links,

$$\mathcal{E}_i^{gb} = \{(\mathbf{v}_1, \mathbf{v}_2) || \mathbf{v}_1^x - \mathbf{v}_2^x| < d_i \wedge |\mathbf{v}_1^y - \mathbf{v}_2^y| = 0 \vee |\mathbf{v}_1^x - \mathbf{v}_2^x| = 0 \wedge |\mathbf{v}_1^y - \mathbf{v}_2^y| < d_i, \forall \mathbf{v}_1 \in \mathcal{V}_i^g, \mathbf{v}_2 \in \mathcal{V}_i^b\} \tag{50}$$

As can be seen in Figure 1, these edges link the vertices of both parts that are closer than d_i either in the x- or in the y-axis and the resulting directed path graph can be expressed as

$$\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i) \equiv (\mathcal{V}_i^g \cup \mathcal{V}_i^b, \mathcal{E}_i^g \cup \mathcal{E}_i^b \cup \mathcal{E}_i^{gb}) \tag{51}$$

In addition to the computational savings of considering a discrete subset of positions to compute the coverage paths, this simplification can also be exploited to decompose the general planning problem for N robots into N individual problems for one robot each.

Proposition 5.1. *The paths that optimize $f(k)$, subject to the constraint of each robot only following trajectories within $\mathcal{G}_i$, are the paths that optimize individually the cost function inside each partition, i.e.,*

$$\min_{\Gamma_i} f(k) = \min_{\Gamma_1} f_1(k) + \cdots + \min_{\Gamma_N} f_N(k) \tag{52}$$

with

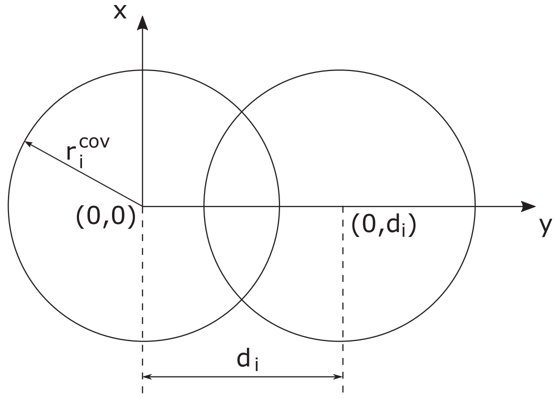


Fig. 1. Schematic of the overlap of the coverage areas for two consecutive sweeps in the x direction and the optimal separation between them.

$$f_i(k) = \int_{\mathcal{P}_i} e^2(\mathbf{q}, k)\, d\mathbf{q} \tag{53}$$

Proof. It is straightforward to see that

$$f(k) = f_1(k) + \cdots + f_N(k) \tag{54}$$

Since the path of each robot is constrained to be within the graph $\mathcal{G}_i$, the coverage contribution of such path is also restricted to $\mathcal{P}_i$, because no coverage is applied outside of it. Then, the path of each robot i only affects the value of $f_i(k)$ in (53), which demonstrates (52). □

The immediate consequence of this proposition is that each robot can plan locally its own path, inside its own partition according to $f_i(k)$, working distributively towards a global minimization of $f(k)$. This also implies that at this stage robots do not need to share any coverage value with any other robot, since they act independently in their regions. It should also be noted that Proposition 5.1 is valid for any partition of the environment, as long as the graphs $\mathcal{G}_i$ ensure that each robot can only apply coverage within its assigned partition. Likewise, the proposition can be applied to other cost functions that, similarly to $f(k)$, consider additive cost in the points of the environment. We utilize this in the following section to obtain online trajectories with a computationally light algorithm.

5.2. Optimal path planning

In order to compute the trajectories within the graph, we first transform it into a weighted digraph. We assign to each edge a weight equivalent to the amount of coverage required to reach the desired value, Z^*, in the head of the edge,

$$\omega_i(\mathbf{v}_1, \mathbf{v}_2, k) = \max(e(\mathbf{v}_2, k), 0) \tag{55}$$

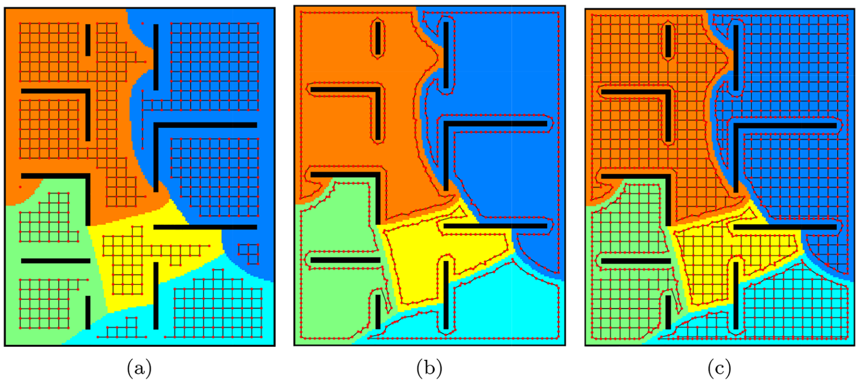


Fig. 2. Example of graph construction: (a) grid; (b) boundary; (c) complete graph.

for all $(\mathbf{v}_1, \mathbf{v}_2) \in \mathcal{E}_i$. In the second place, for a given trajectory, $\Gamma_i(k)$, through the graph $\mathcal{G}_i$ at time k, starting from the position of robot i at that time, we define the error per number of visited vertices as

$$g_i(\Gamma_i(k)) = \frac{\sum_{\ell=1}^{L_v} \omega_i(\mathbf{v}_{\ell-1}, \mathbf{v}_\ell, k)}{L_v} \tag{56}$$

where L_v is the number of traversed vertices in Γ_i and $(\mathbf{v}_{\ell-1}, \mathbf{v}_\ell) \in \mathcal{E}_i$ for all ℓ. Therefore, the optimization problem considered by our planning algorithm is

$$\underset{\Gamma_i(k)}{\text{maximize}} \quad g_i(\Gamma_i(k)) \tag{57}$$

This optimization problem aims at finding the optimal between all the paths of all possible lengths that go through the graph without repeating vertices.

The transformation of the coverage path planning into finding the optimal path through a graph allows the problem to be solved rapidly and efficiently with proven methods to find the shortest paths in graphs. In particular, we adapt the Bellman–Ford algorithm (Cormen, 2009) in two ways. Instead of the shortest distance, we look for the highest accumulated error per vertex and, therefore, we initialize our metric to $-\infty$ for all the nodes of the graph. In addition, since the aim is to maximize this metric, the path to a vertex is updated and the vertex appended to the priority queue if the previously stored value of the metric is lower than the new one. It should be noted that these modifications do not alter the optimality of the method and maintain the worst-case complexity of $\mathcal{O}(|\mathcal{V}_i||\mathcal{E}_i|)$. Each path is then fully executed by the robot until it is finished. Then, the planning process is repeated with the new values of the coverage errors in each vertex.

6. Simulation results

In this section, we present simulation results for the partitioning algorithms and the complete approach to the persistent coverage problem. For the partitioning, we consider four different, non-convex environments of 10 m × 8 m and increasing complexity. We call them *Open Rooms, Rooms, Spiral*, and *Maze*, respectively. They can be seen in Figure 4. These simulations have been carried out with $N = 5$ robots of radius $r_i = 0.1$ m, a circular coverage area Ω_i of radius $r_i^{\text{cov}} = 0.2$ m, a production $\alpha_i = 1$, $\forall \mathbf{q} \in \Omega_i$, and $\rho_i^{\max} = 20$. The objective, decay, and importance of the coverage are

Table 1. Values for k_g depending on the environment type.

Open Rooms	Rooms	Spiral	Maze
5	5	2	0.1

$$Z^* = 80 + 20\,\frac{\mathbf{q}_y}{|\mathcal{Q}|_y} \tag{58}$$

$$d = 0.9995 \tag{59}$$

$$\phi = 0.5 + 0.5\,\frac{\mathbf{q}_y}{|\mathcal{Q}|_y} \tag{60}$$

where $\mathbf{q}_y$ are the y-coordinates of point $\mathbf{q}$ and $|\mathcal{Q}|_y$ is the y-size of the environment $\mathcal{Q}$. The resulting work function $\lambda(\mathbf{q})$ can be seen in Figure 3 for the *Rooms* environment, normalized by $\int_{\mathcal{Q}} \lambda(\mathbf{q})\, d\mathbf{q}$ and multiplied by 100. The upper part of the environment requires less work because of its lower importance and objective and the central and lower parts require more work due to the peak of the decay in the center and the higher importance and objective at the bottom.

The positive constant of the saturation function (34) is set to $k_{\text{sat}} = 3$ and the gains of the control law are different for the different environments and are updated online depending on the value of $\partial H / \partial w_i$. In particular, k_g takes the values from Table 1 and k_w is assigned according to

$$k_w = \begin{cases} k_w^1, & \text{if } |\partial H/\partial w_i| > \beta_1, \\ k_w^2, & \text{if } \beta_1 \leqslant |\partial H/\partial w_i| < \beta_2, \\ k_w^3, & \text{if } \beta_2 \leqslant |\partial H/\partial w_i| < \alpha_3, \\ k_w^4, & \text{if } |\partial H/\partial w_i| \leqslant \beta_3, \end{cases} \tag{61}$$

Table 2. Values for k_w depending on the environment.

	Open Rooms	*Rooms*	*Spiral*	*Maze*
k_w^1	5	5	15	15
k_w^2	100	100	75	150
k_w^3	500	500	375	1,500
k_w^4	5,000	5,000	1,500	15,000
β_1	5×10^{-3}	5×10^{-3}	5×10^{-3}	10^{-2}
β_2	3×10^{-4}	2×10^{-4}	5×10^{-4}	10^{-3}
β_3	10^{-5}	10^{-5}	$5 \cdot 10^{-5}$	10^{-4}

with the values of the constants presented in Table 2. These parameters were manually tuned in order to speed up the convergence of the partitions. In practice, there is no need to change these parameters for different environments. On the downside, the convergence time of the equitable partitions may vary and affect some environments more than others if they are not appropriately tuned. In any case, this time is negligible when compared with the duration of the task, that can be infinite.

In our simulations the map is represented by a matrix of pixels (an image), thus, even when the partition method is continuous, there is some degree of discretization. To compute the gradients, we keep track of the partition associated to each pixel, obtaining the boundaries by checking whether all the neighbor pixels belong to the same one. Then, for the pixels in the boundary we compute the geodesic distance and $(\mathbf{q} - h^1_{\mathbf{q},\mathbf{g}_j})$ in (26). From there, we can compute (29) and the gradient in (13) by summing this value for all the points of the partition boundary. Although this procedure looks complex, it is not computationally expensive, requiring in our case 31 ms per iteration and robot.

6.1. Partitioning example

In the first place we present an example of the partitioning algorithms in the different environments. In the top row of Figure 4 we show the resulting equitable partitions under (12), i.e., when only the weights are modified and the positions of the generators remain fixed. Although in some cases such as Figure 4a the resulting partitions are connected, it can be seen that in others they are not, see, e.g., the yellow partition in Figure 4d. These disconnections are avoided under (33) as can be seen in the bottom row of Figure 4. In this case, moving the generators towards the centroid of the connected components with the highest workload at the same time as the weights are changing, allows the robots to find equitable and connected partitions.

Now we focus in the particular example of the *Spiral* environment to assess the evolution and convergence under both (12) and (33). In Figure 5 we represent the evolution of the workload, the weights, and the gradient of the cost function with respect to the weight for each robot in a different color under (12). First, in the top figure we can observe that all the workloads converge to the same

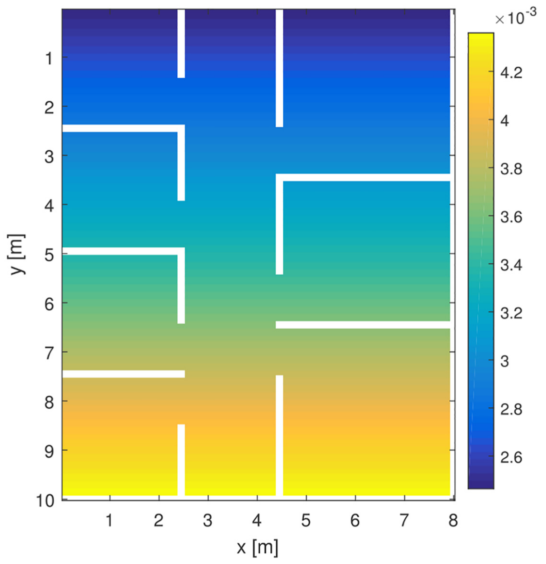


Fig. 3. Normalized work function $\lambda(\mathbf{q})$ over the *Open Rooms* environment. White areas represent obstacles.

percentage value (20%), that is, to the equity. Convergence is assumed when the maximum difference between the workload of the partitions is lower than 5%. Second, it is noteworthy that the weights associated to the partitions at the bottom of the map (black and magenta) are negative, while those at the top (red and yellow) are positive. This makes sense when we consider that the workload is distributed as in Figure 3. Since the upper part is less important, robots in that area are assigned larger partitions, and therefore positive weights. Finally, the value of the gradient is represented in the bottom plot.

Under (33), in the beginning the weights and workloads also evolve slowly as shown in Figure 6. The convergence speed of this law is similar to that in (12). The main difference appears in the bottom plot of the figure, where we show how often the motion of the generators is saturated, f_{sat} in (34). Values equal to zero mean that moving the generator goes against obtaining equity. The main consequence is that the evolution of the partitions is not as smooth as with (12) but all the resulting partitions are connected. In the following section we carry out a parametric analysis to be able to generalize the results.

6.2. Parametric analysis of the partitioning algorithms

We analyze now the evolution of the performance of the partitioning algorithm when the parameters of the problem change. In particular, we performed a Monte Carlo analysis of 10 runs for different initial positions of the generators and different work functions in all the environments. In order to represent the results concisely, we refer to the different environments by their initials as follows: *Open Rooms* as OR, *Rooms* as R, *Spiral* as S, and *Maze* as M.

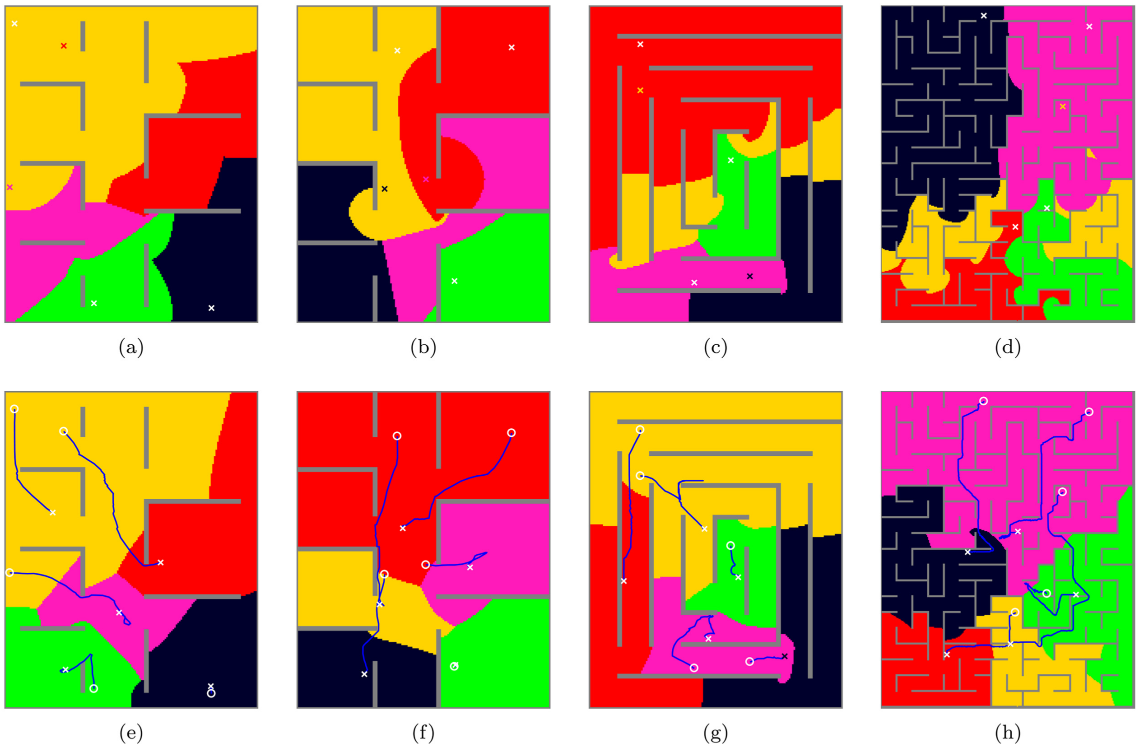


Fig. 4. Example of partitioning in the four different environments under control law (12) (top row) and under control law (33) (bottom row). The resulting partitions are shown in different colors: (a) *Open Rooms*; (b) *Rooms*; (c) *Spiral*; (d) *Maze*; (e) *Open Rooms*; (f) *Rooms*; (g) *Spiral*; (h) *Maze*. The initial locations of the generators are represented with white circumferences and the final location with white crosses. Blue lines in the bottom row represent the path of the generators until the equitable partitions are achieved. In cases where a generator ended outside its own partition, it has been plotted with the color of its partition instead of white to show the correspondences.

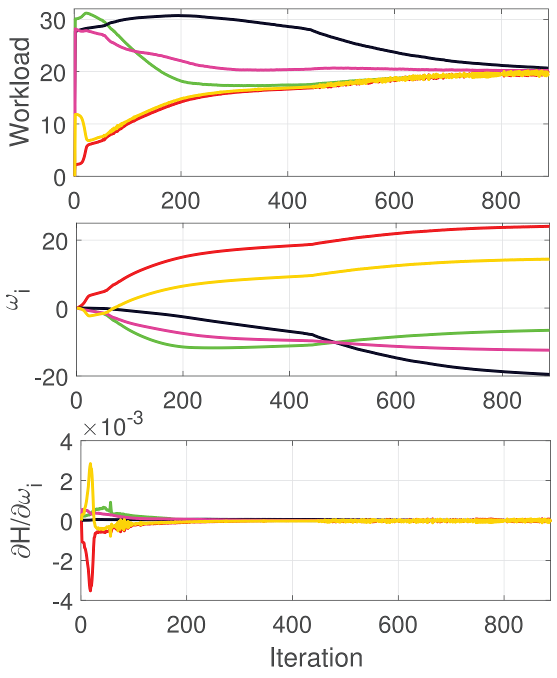


Fig. 5. Evolution of the workload, weight and gradient of the cost function with respect to the weight for the *Spiral* environment under control law (12). Different colors represent the variables for the different robots.

The 10 initial positions of the generators were selected randomly and the variation of the work function was done through the decay function. In (59) the decay is set to be a 2D Gaussian centered at $|\mathcal{Q}|_x/2$ and $|\mathcal{Q}|_y/2$, i.e., in the middle of the environment. Instead of this center, we chose a random point for each trial. We execute this analysis for (12) and (33) and refer to them as W and WG respectively, representing that only the power weights or the power weights and the generator positions are modified.

First we show in Figure 7 two boxplots of the number of iterations required for the algorithm to converge to the equitable partition. Figure 7a shows the results for different initial positions of the generators and Figure 7b, for different decay functions. The most important feature of these results is that, for the same environment, WG converges in general faster than W. Among the different environments, the speed of convergence depends on the complexity of the environment and on the tuning of the gains.

In the second place we pay attention to the connectivity of the partitions. Table 3 gathers the number of partitions that became disconnected out of the 50 regions corresponding to each cell. As expected, this number is drastically reduced with the WG control law. In fact, for *Open Rooms* and *Rooms*, it converges to connected partitions in all cases.

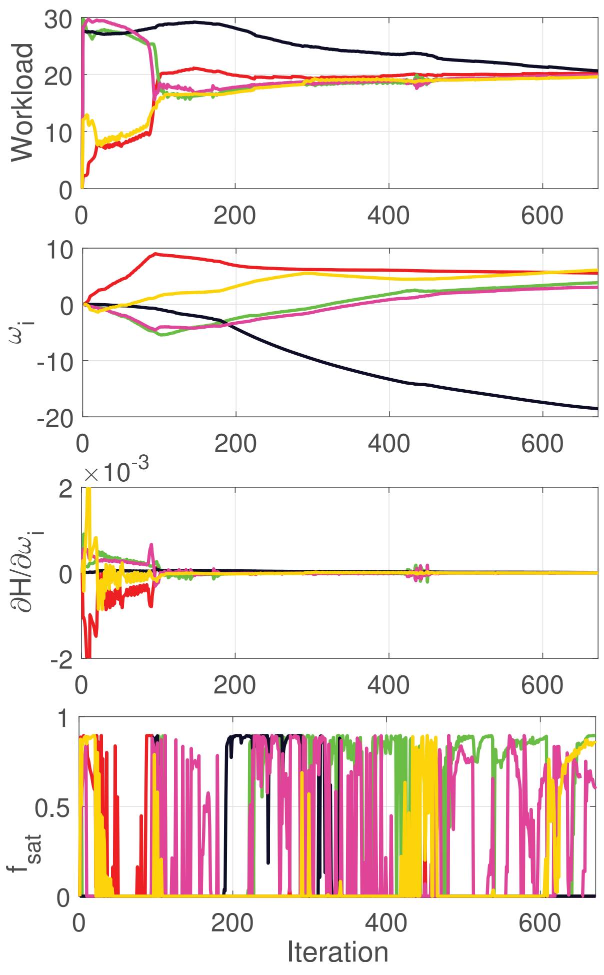


Fig. 6. Evolution of the workload, weight, gradient of the cost function with respect to the weight, and saturation function for the *Spiral* environment under control law (33). Different colors represent the variables for the different robots.

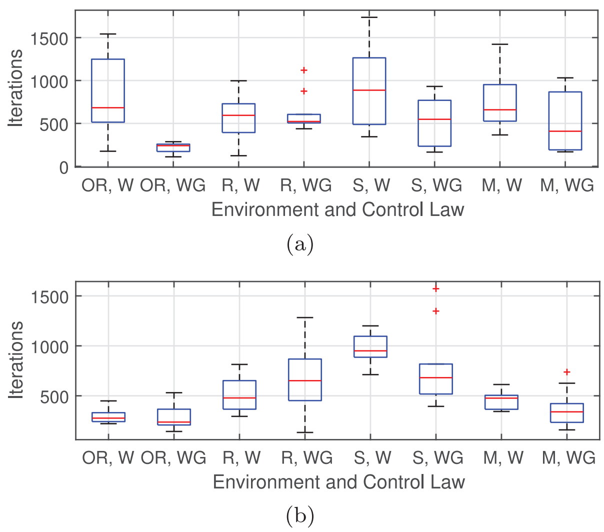


Fig. 7. Boxplot of the number of iterations required to converge to the equitable partition: (a) different initial positions; (b) different decay functions.

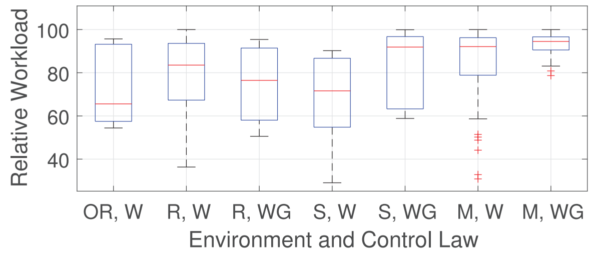


Fig. 8. Boxplot of the relative workload of the biggest connected component with respect to the workload of the entire partition.

However, for *Spiral* and *Maze* the partitions became disconnected in some cases.

To evaluate the nature and the quality of the still non-connected partitions in Figure 8 we represent in percentage the relative workload of the biggest connected component with respect to the workload of the entire partition. One can see that under WG, this value is on average above 80%. This means that almost all the work of the partition is on the biggest component, whose centroid was followed by the generator. Therefore, a simple reassignment of the smaller disconnected regions could be done to reach total connectivity with only a small deviation from the equitable partitioning.

6.3. Coverage planning results

In this part of the simulations, we analyze the coverage planning solution and the performance of the entire approach with an example. Figure 9 shows the partition of the *Rooms* environment and the path graphs of the robots.

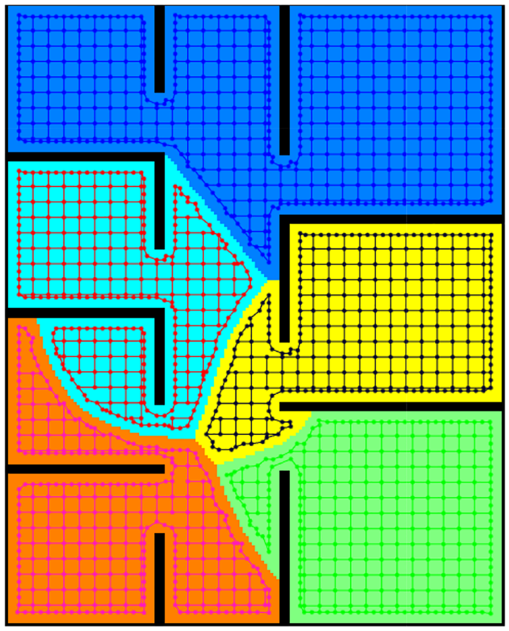


Fig. 9. Example of partitions and path graphs for the *Rooms* environment.

Table 3. Number of disconnected partitions.

	Different positions		Different decays	
	W	WG	W	WG
Open Rooms	1	0	4	0
Rooms	30	0	29	4
Spiral	12	4	26	9
Maze	30	14	23	11

According to the work map shown in Figure 3, the robot assigned to the blue partition has to cover a bigger area than the other four. These other four share the peak of the work map and each of them is additionally in charge of covering a big room on the right-hand side or one and a half small rooms on the left-hand side.

In order to evaluate the quality of the paths and the provided coverage we normalize the coverage error at a single point at time k,

$$\varepsilon(k) \equiv \varepsilon(\mathbf{q}, k) = \frac{e(k)}{Z^{*2}} \qquad (62)$$

and calculate the mean normalized error over the environment as

$$\bar{\varepsilon}(k) = \frac{\int_{Q_f} \varepsilon(k) d\mathbf{q}}{|Q_f|} \qquad (63)$$

In Figure 10 we show the coverage error $e(\mathbf{q}, k)$ of the environment for eight different time instants. In Figure 10a the robots are starting to cover the environment and, therefore, the coverage error is initially proportional to the importance of the points and their objective. The paths of the top row, Figure 10a–10d, represented in magenta, cover the partitions from bottom to top in a sweep-like manner. This demonstrates that the paths go through the points with the highest coverage error. In Figure 10e, almost all the robots have already covered their partitions for the first time. The blue robot needs a little more time since it has the biggest partition. The paths of the robots in Figure 10e and 10f, fill the gaps that they have previously left uncovered. At this point, the coverage error is already small in the majority of the points. Eventually, in the steady state, Figure 10g and 10h, the robots keep moving to maintain the coverage as close as possible to the objective, paying special attention to the zones where more work is required.

The performance of our coverage strategy is assessed in Figure 11a. We depict the mean coverage error and its standard deviation for the simulation. One can see that, at the beginning, when the environment is totally uncovered, the error is around 0.7 due to the importance function. After that, it decreases rapidly and converge to a small steady-state value. In fact, on average, the points have under 7% of error. The standard deviation increases at the beginning because the undercovered points keep deteriorating and some others become slightly overcovered when the robots

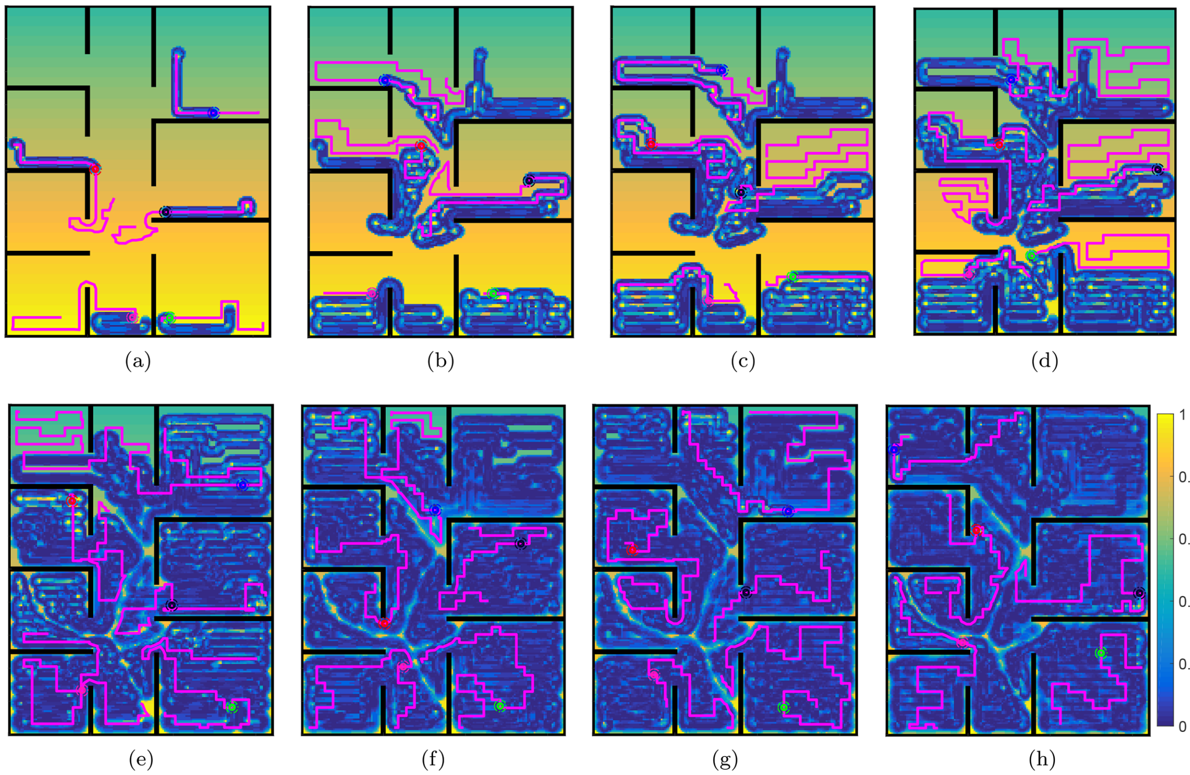
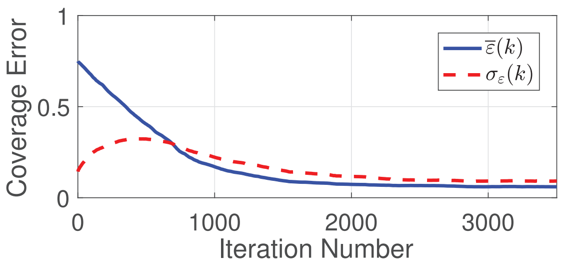
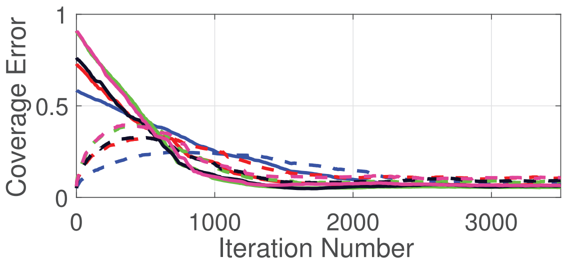


Fig. 10. Example of coverage evolution: (a) $k = 60$; (b) $k = 275$; (c) $k = 400$; (d) $k = 500$; (e) $k = 1,150$; (f) $k = 1,700$; (g) $k = 2,000$; (h) $k = 2,500$. The background color map represents the value of $\varepsilon(k)$. The robots are depicted in different colors with their coverage areas as dash-dotted circumferences. Magenta lines show the current paths of the robots.



(a)



(b)

Fig. 11. Evolution of the mean coverage error (solid lines) and its standard deviation (dashed lines): (a) complete environment; (b) separated by partitions.

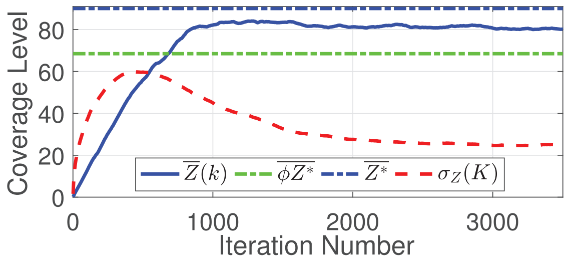


Fig. 12. Evolution of the mean coverage level of the environment (blue line) with its standard deviation (red dashed line). The blue dash-dotted line represents the mean coverage objective and the green dash-dotted one represents the mean coverage objective weighted by the importance of each point.

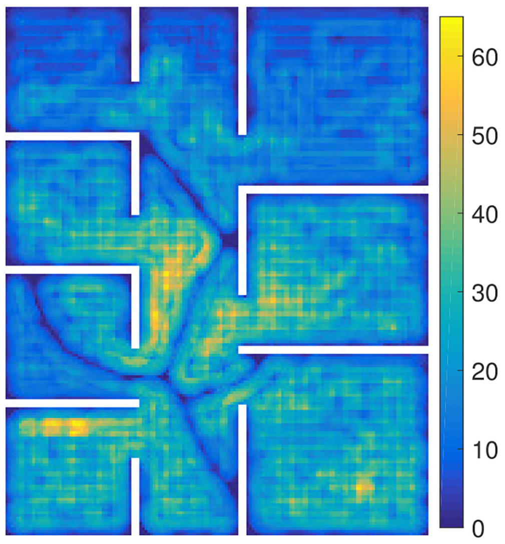


Fig. 13. Number of times that the robots have covered each point of the environment.

go over them. However, it also decreases to a small steady-state value around 0.09.

A similar tendency persists inside the partitions of the robots as shown in Figure 11b, where $\overline{\varepsilon}(k)$ is calculated only within $\mathcal{P}_i$. Nevertheless, it is interesting to see the differences between the biggest partition, the blue one, and the smaller partitions, the magenta and green ones. The error at the beginning in the magenta and green ones is higher because the importance of the points inside them is higher, as opposed to the blue one. On the other hand, the reduction of the error in those partitions is faster because they are smaller and the robots need less time to cover them completely. In contrast, the blue robot needs more time to cover its region and, therefore, its error decreases slower.

Although the coverage error is the metric that defines the optimality of the coverage, it is difficult to visualize. A more intuitive representation is shown in Figure 12. The mean coverage level of the environment increases rapidly towards the mean coverage objective. However, its steady-state value is smaller due to the different importance of the points. This happens because the robots pay more attention to the most important points at the expense of leaving the least important worse covered. For this reason, the mean value lays between the mean coverage objective and the mean coverage objective weighted by the importance.

Finally, we show in Figure 13 the number of times that the robots covered each point of the environment. The resemblance with the work map makes it clear that the robots visit more frequently the most important points and those with a lower decay, because the coverage deteriorates faster on them. The undercovered zones near the boundaries between partitions occur due to discretization in the simulation and due to the separation from the boundary to the boundary nodes of the graph.

6.4. Comparison with centroidal Voronoi partitions

To conclude the simulations, we compare the coverage results with those obtained using the Voronoi partition algorithm described in Cortes et al. (2004) to generate the partitions of the environment. Both methods have been tested in the *Rooms* environment, with the work function λ represented in Figure 3. In the Voronoi partition we have used the workload as the density function to compute the centroid of the partition.

In Figure 14(a), we show the partitions obtained with the two algorithms for one example of initial conditions of

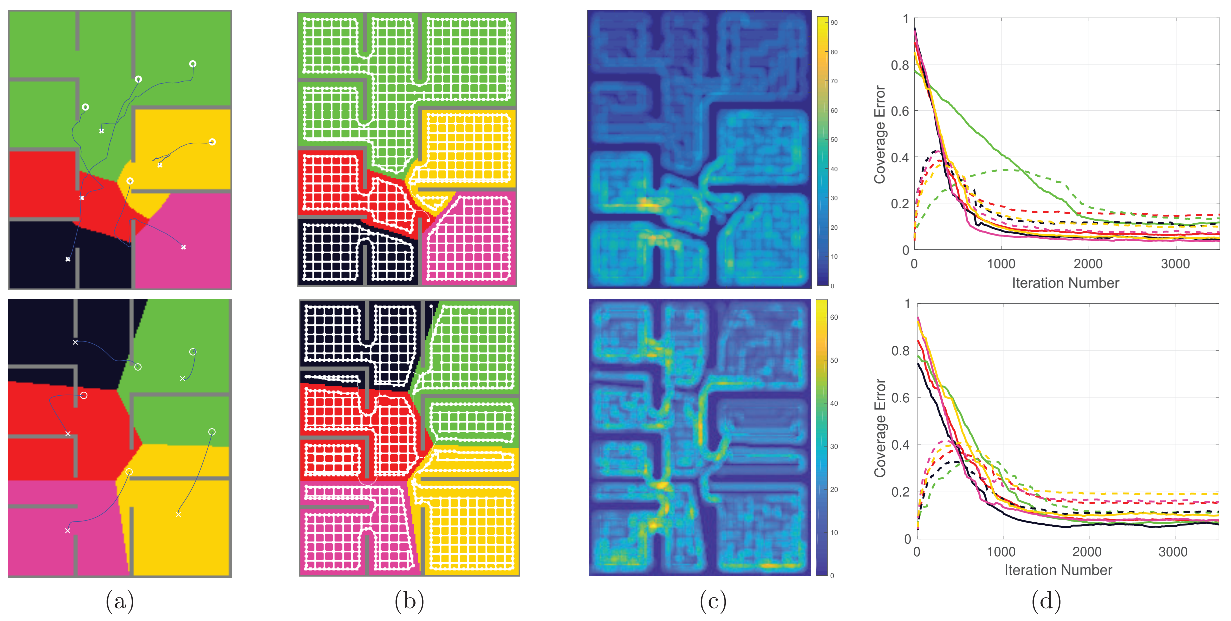


Fig. 14. Comparison of the solution with a standard Voronoi partition: (a) partitions; (b) coverage graphs; (c) number of visits; (d) coverage error. The top row shows our approach and the bottom row shows the results with the Voronoi partition.

the generators. In this particular example the yellow partition accounts for 25.64% of the total workload, whereas the black partition only accounts for 15.83%, implying that first robot will need to do considerably more work than the second. In addition, two out of the five partitions are not connected (black and green). We have repeated this analysis for the 10 runs described in Table 3 for this environment to measure the number of disconnected partitions and the difference between the maximum and minimum workload. The Voronoi tessellation method has yielded 16 disconnected regions, as opposed to the zero regions obtained with our method. The mean difference of the 10 trials was 10.3% with a standard deviation of 1.36.

In the following, we focus on the coverage quality when both sets of partitions are used together with the path-planning algorithm described in Section 5, just for the partitions given in Figure 14(a). In Figure 14(b), we show the coverage graphs and in Figure 14(c) we show the number of times that each node of the graphs was visited during the experiment. In the Voronoi partitions, some robots invade other robots' partitions in order to move between the disconnected components, e.g., the green partition . More importantly, with the equitable partition, points at the bottom of the map are visited more often, accounting for their decay and importance in a better way.

Finally, in Figure 14(d) we show the normalized error (solid lines) (63) for each partition and the standard deviation (dashed lines) computed over all the points in the integral. The transient behavior in this case is similar for both partitions, with a slightly better performance, i.e., less settling time, when using the equitable partitions for those regions with high importance (every region but the green one). In the steady state, it can be seen that the equitable partition presents a smaller standard deviation, implying a more homogeneous coverage.

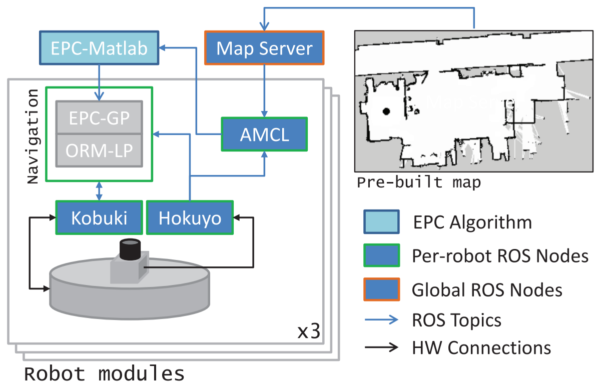


Fig. 15. Multi-robot system architecture.

7. Experimental Results

The final validation of our proposal was performed carrying out real-world experiments in a complex environment shown in Figure 16a. It is composed by a small room in the left side and a main room in the center that connects to a corridor through two different doors. The goal was to persistently cover the environment shown in the same figure in yellow, with $N = 3$ TurtleBot II robots. We refer the

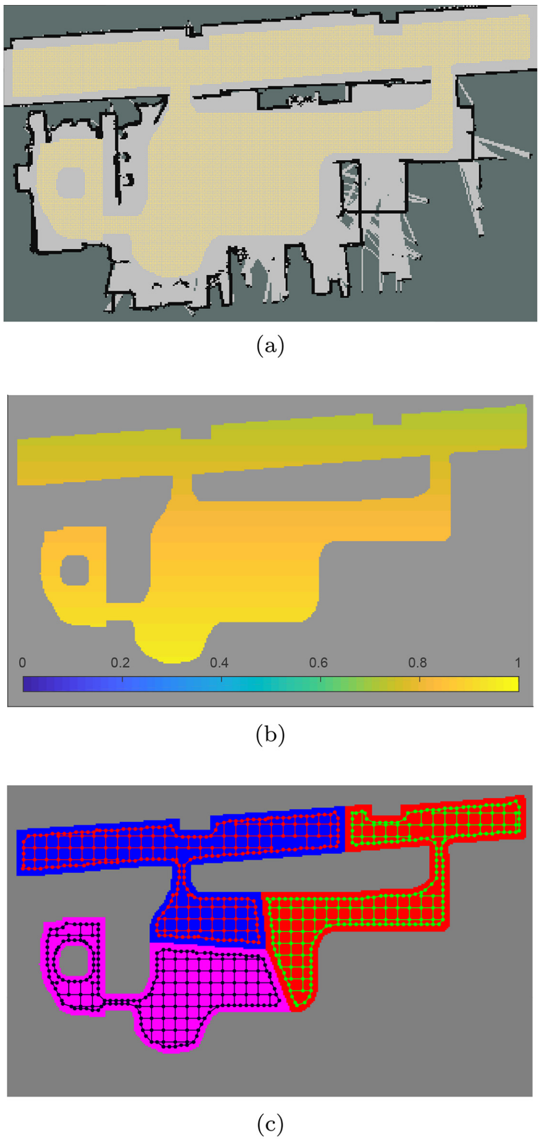


Fig. 16. Maps of the experimental environment and results from the partitioning and graph construction algorithms. (a) Environment map. The area to be covered is represented in yellow. (b) Work map. (c) Partitioned map with path graphs.

reader to the Extension 1 for the accompanying video of this experiment for a complete visualization of the coverage solution.

7.1. Setup

The multi-robot system was set up relying on the Robot Operating System (ROS) framework Quigley et al. (2009). The robotics system was organized as shown in Figure 15. From the bottom up, two ROS nodes, *Kobuki* and *Hokuyo*, manage the TurtleBot II platform equipped with an on-board Intel NUC core i7 computer and a Hokuyo URG-04LX LiDAR sensor. These nodes provide odometry and laser readings to the *AMCL* node and to the navigation stack. The *AMCL* node is in charge of localizing the robots on a pre-built map (obtained previously to the experiments) provided by the *Map Server* node. The navigation stack is, as usual, divided in two layers. At the bottom level, the *ORM*Minguez (2005), chosen because its efficacy in dense environments, implements the basic obstacle avoidance capability. On top of the *ORM*, a dedicated global planner indicated as *EPC-GP* was used. It wraps the Equitable Partition algorithm described in this article, which is presently implemented in Matlab (*EPC-Matlab*). The latter takes advantage of the Matlab robotics toolbox, which allows ROS communication between standalone ROS nodes and Matlab scripts. Specifically the *EPC-Matlab* publishes the paths computed by the algorithm in a dedicated topic which is read by the *EPC-GP*, that assumes them as global paths. In addition, the *EPC-Matlab* reads *AMCL*-computed poses to update the coverage map. From the architectural point of view, the heavier computation was centralized in a single machine: only the *Kobuki* and *Hokuyo* nodes ran on the mobile platform computers while the other nodes ($3 \times$ AMCL, $3 \times$ Navigation and $1 \times$ Map Server) and the algorithm itself (*EPC-Matlab*) reside in the base station connected with the robots through a WiFi router.

The coverage actions and values remained simulated in the experiment due to the complexity associated with measuring these values in a real setup. We defined a circular coverage area Ω_i of radius $r_i^{cov} = 0.186$ m, while the radius of the robots is equal to $r_i = 0.177$ m. The speed of the robots were fixed at 0.35 m/s. Their production and the objective functions were the same as in the simulations and the importance of the coverage was

$$\Phi = 0.7 + 0.3\,\frac{\mathbf{q}_y}{|\mathcal{Q}|_y} \qquad (64)$$

7.2 Results

In this section, we present two runs for different decays. The decay was set to a uniform value of 0.9995 in experiment A and 0.995 in experiment B. The resulting work for experiment A can be seen in Figure 16b and is proportional for experiment B. The lower part required more work whereas the corridor required less, due to its lower importance. With these settings, the resulting partitions and path graphs computed by the robots were those depicted in Figure 16c. In both experiments, they were the same since the only difference is a proportional decay. The three partitions were equitable and self-connected with the three robots sharing the main room and two of them sharing the corridor through different doors. The path graph of the each robot covers its entire partition and allows the robot to follow sweep-like paths.

In Figure 17 we show some snapshots of experiment B at three different iterations, $k = 0$, $k = 100$, and $k = 500$. In particular, Figure 17a shows the initial position of the robots while Figure 17b shows the three robots in the main

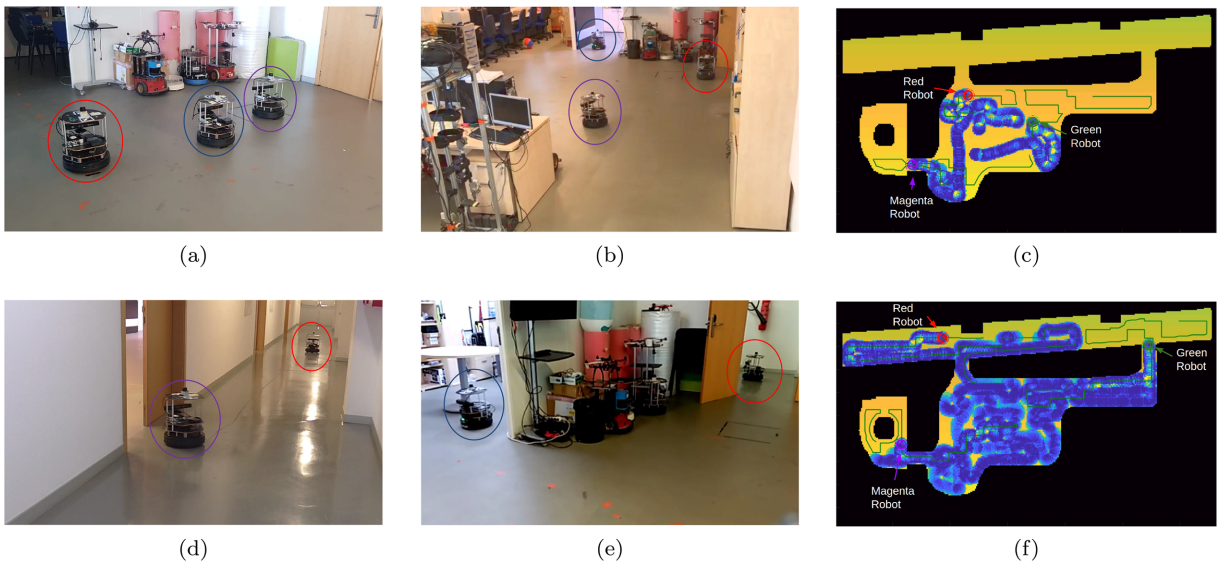


Fig. 17. Snapshots of one of the experiments: (a) main room at $k = 0$; (b) main room at $k = 100$; (c) coverage map and paths at $k = 100$; (d) corridor at $k = 500$; (e) small room and part of the main room at $k = 500$; (f) coverage map and paths at $k = 500$. The colored circles show the identity of each robot.

room covering their respective partitions after start-up. Even when the robot associated with the red partition starts outside of its partition, the planner drives it towards it improving the coverage of the other partition while going there. This coverage is taken into account by the planner in the magenta partition. They start sweeping the environment from bottom to top, according to the importance of the points, as shown in Figure 17c. After 500 iterations, the robots have covered almost the entire environment for the first time. This can be seen in Figure 17f. At that point, the blue robot is inside the small room covering the surroundings of the table (Figure 17d) and the other two are sweeping the corridor (Figure 17e). It is worth highlighting how the coverage of the central and lower part of the main room had already decayed at $k = 500$, because they were the first covered areas.

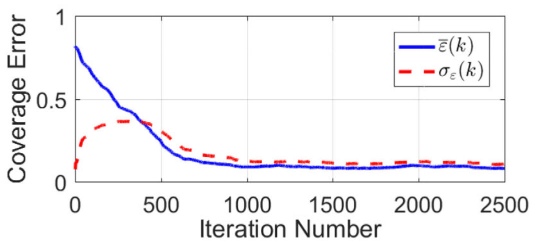
Eventually, we analyze the results from experiments A and B. It can be seen that in both cases they are very similar to the simulation results, which validates our approach for a real system in a complex environment.

In experiment A, the coverage error in the entire environment (Figure 18a) and in each partition (Figure 18b) decreased rapidly and was maintained at a low value. The mean coverage level was maintained very close to the desired level, with almost the same value as the mean of the objective weighted by the work required at each point (Figure 18c). This demonstrated that, although the robots could not maintain the desired level all the time because of the nature of the problem, they were able to maintain the most important points closer to their desired value. Figure 18d depicts the real paths that the robots followed to cover the entire environment the first time.

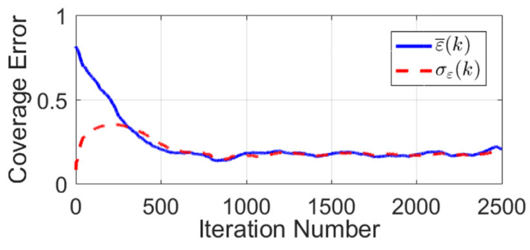
In experiment B, we decreased the value of the decay to simulate a faster deterioration of the coverage, i.e., to entail the robots a bigger workload and, therefore, forcing them to visit the important points with more frequency. This is equivalent to reducing the team size and deploying fewer robots than required by the environment. For these reasons, the results show in this case a smaller reduction of the coverage error (Figure 19a and (b)), a lower value of the coverage level and a slightly bigger standard deviation (Figure 19c). The paths followed by the robots demonstrate that they covered several times the lower parts of their partitions, i.e., the most important ones, before covering their upper parts completely. These experiments support the adaptability of our approach to environments with non-uniform workloads and, in particular in experiment B, that the robots are able to prioritize the most important areas even if their capabilities are lower than the work required by the environment.

8. Conclusions

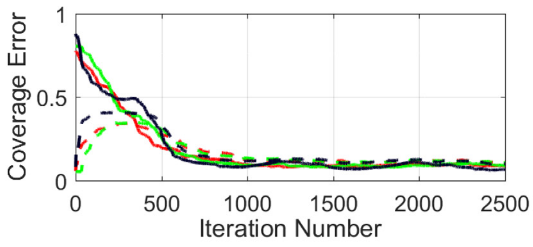
In this article, we have proposed a partitioning approach based on power diagrams along with a graph-based coverage planning strategy to solve the persistent coverage problem. We have introduced in the first place an algorithm to find the power weights that correspond to an equitable partition of the environment. This strategy can be applied to any kind of non-convex environment thanks to the use of the geodesic distance in the partitioning. We have also presented two extensions for this strategy. The first extension drives the generator of each partition to the centroid of the connected component with the highest workload. The



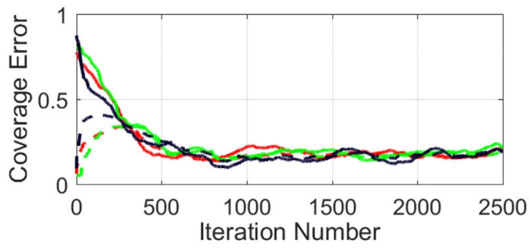
(a)



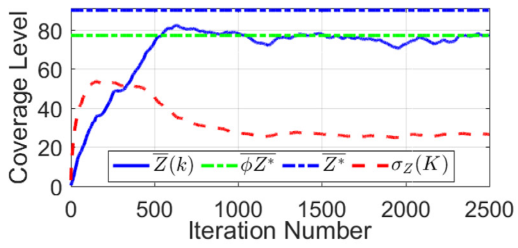
(a)



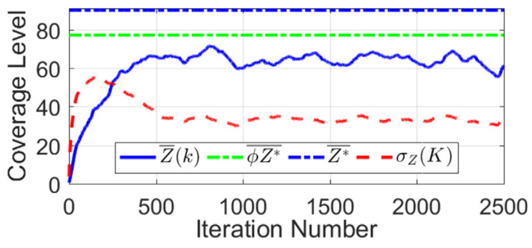
(b)



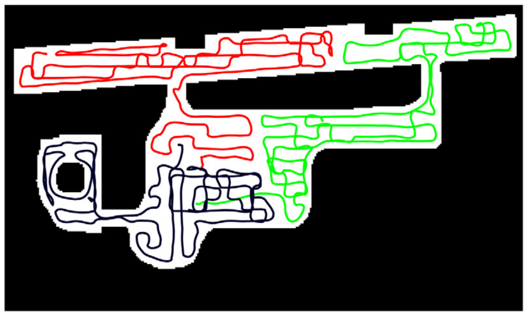
(b)



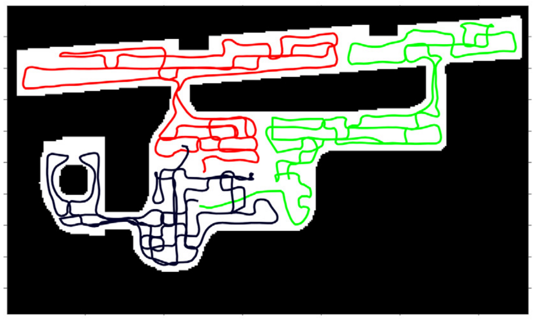
(c)



(c)



(d)



(d)

Fig. 18. Results from experiment A. (a) Mean coverage error of the environment (solid line) and its standard deviation (dashed line). (b) Mean coverage error of each partition (solid lines) and its standard deviation (dashed lines). (c) Mean coverage level of the environment (solid blue line), its standard deviation (dashed red line), mean coverage objective (dash-dotted blue line), and mean objective weighted by the workload (dash-dotted green line). (d) Paths of the robots until $k = 1,500$.

Fig. 19. Results from experiment B. (a) Mean coverage error of the environment (solid line) and its standard deviation (dashed line). (b) Mean coverage error of each partition (solid lines) and its standard deviation (dashed lines). (c) Mean coverage level of the environment (solid blue line), its standard deviation (dashed red line), mean coverage objective (dash-dotted blue line), and mean objective weighted by the workload (dash-dotted green line). (d) Paths of the robots until $k = 1,500$.

objective of this strategy is to reduce the disconnections of the partitions to the greatest extent possible. The second extension takes into account the different coverage capabilities of each robot. In the second place, we have presented an online planning algorithm that allows each robot to locally find the best path in terms of the current coverage error. This algorithm is based on the construction of a path graph that covers the entire partition of the robot with sweep-like paths. We have assigned the coverage errors of the head nodes to the edge weights and, therefore, the problem has been reduced to finding the optimal path through a graph that can be achieved efficiently with state-of-the-art methods. Finally, we have included simulation and experimental results to support our contributions and have shown how our solution can be applied to a real-world scenario.

Funding

This work was partially supported by projects RTC-2014-1847-6 and RTC-2017-5965-6 of Retos-Colaboración, DPI2015-69376-R and PGC2018-098719-B-I00 (MCIU/AEI/FEDER, UE), DGA-T45_17R/FSA, CUD2017-18 and DGA Scholarship C076/2014, partially funded by the European Social Fund.

ORCID iD

Eduardo Montijano https://orcid.org/0000-0002-5176-3767

References

Ahmadzadeh A, Jadbabaie A, Kumar V and Pappas G (2007) Cooperative coverage using receding horizon control. In: *European Control Conference*, pp. 2466–2470.

Alamdari S, Fata E and Smith SL (2014) Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations. *The International Journal of Robotics Research* 33(1): 138–154.

Araujo JF, Sujit PB and Sousa JB (2013) Multiple UAV area decomposition and coverage. In: *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pp. 30–37.

Arkin EM, Fekete SP and Mitchell JS (2000) Approximation algorithms for lawn mowing and milling. *Computational Geometry* 17(1): 25–50.

Atinç GM, Stipanović DM and Voulgaris PG (2014) Supervised coverage control of multi-agent systems. *Automatica* 50(11): 2936–2942.

Aurenhammer F (1987) Power diagrams: Properties, algorithms and applications. *SIAM Journal on Computing* 16(1): 78–96.

Barrientos A, Colorado J, del Cerro J, et al. (2011) Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics* 28(5): 667–689.

Bhattacharya S, Ghrist R and Kumar V (2014) Multi-robot coverage and exploration on riemannian manifolds with boundaries. *The International Journal of Robotics Research* 33(1): 113–137.

Boardman B, Harden T and Martinez S (2016) Spatial load balancing in non-convex environments using sampling-based motion planners. In: *American Control Conference*, pp. 5703–5708.

Breitenmoser A, Metzger JC, Siegwart R and Rus D (2010a) Distributed coverage control on surfaces in 3D space. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5569–5576.

Breitenmoser A, Schwager M, Metzger JC, Siegwart R and Rus D (2010b) Voronoi coverage of non-convex environments with a group of networked robots. In: *IEEE International Conference on Robotics and Automation*, pp. 4982–4989.

Breitenmoser A, Sommer H and Siegwart R (2014) Adaptive multi-robot coverage of curved surfaces. In: *Distributed Autonomous Robotic Systems*. Berlin: Springer, pp. 3–16.

Cormen TH (2009) *Introduction to Algorithms*. Cambridge, MA: MIT Press.

Cortes J, Martinez S, Karatas T and Bullo F (2004) Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation* 20(2): 243–255.

Durham JW, Carli R, Frasca P and Bullo F (2012) Discrete partitioning and coverage control for gossiping robots. *IEEE Transactions on Robotics* 28(2): 364–378.

Franco C, Lopez-Nicolas G, Sagues C and Llorente S (2013) Persistent coverage control with variable coverage action in multi-robot environment. In: *IEEE Conference on Decision and Control*, pp. 6055–6060.

Galceran E and Carreras M (2013) A survey on coverage path planning for robotics. *Robotics and Autonomous Systems* 61(12): 1258–1276.

Graham R and Cortés J (2012) Adaptive information collection by robotic sensor networks for spatial estimation. *IEEE Transactions on Automatic Control* 57(6): 1404–1419.

Haumann D, Breitenmoser A, Willert V, Listmann K and Siegwart R (2011) Discoverage for non-convex environments with arbitrary obstacles. In: *IEEE International Conference on Robotics and Automation*, pp. 4486–4491.

Jager M and Nebel B (2002) Dynamic decentralized area partitioning for cooperating cleaning robots. In: *IEEE International Conference on Robotics and Automation*, Vol. 4, pp. 3577–3582.

Kakalis NM and Ventikos Y (2008) Robotic swarm concept for efficient oil spill confrontation. *Journal of Hazardous Materials* 154(1): 880–887.

Kapoutsis AC, Chatzichristofis SA and Kosmatopoulos EB (2017) DARP: Divide areas algorithm for optimal multi-robot coverage path planning. *Journal of Intelligent and Robotic Systems* 86(3–4): 663–680.

Lan X and Schwager M (2013) Planning periodic persistent monitoring trajectories for sensing robots in Gaussian random fields. In: *IEEE International Conference on Robotics and Automation*, pp. 2407–2412.

Ma X, Jiao Z, Wang Z and Panagou D (2018) 3-d decentralized prioritized motion planning and coordination for high-density operations of micro aerial vehicles. *IEEE Transactions on Control Systems Technology* 26(3): 939–953.

Mackenzie D and Balch T (1993) Making a clean sweep: Behavior based vacuuming. In: *Proceedings of the AAAI Fall Symposium: Instantiating Real-world Agents*, pp. 93–98.

Maza I and Ollero A (2007) Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. *Distributed Autonomous Robotic Systems* 6: 221–230.

Minguez J (2005) The obstacle-restriction method (ORM) for robot obstacle avoidance in difficult environments. In: *IEEE International Conference on Intelligent Robots and Systems*, pp. 3706–3712.

Moon S and Frew EW (2017) Distributed cooperative control for joint optimization of sensor coverage and target tracking. In: *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 759–766.

Nigam N (2014) The multiple unmanned air vehicle persistent surveillance problem: A review. *Machines* 2(1): 13–72.

Nigam N, Bieniawski S, Kroo I and Vian J (2012) Control of multiple UAVs for persistent surveillance: algorithm and flight test results. *IEEE Transactions on Control Systems Technology* 20(5): 1236–1251.

Palacios-Gasós JM, Montijano E, Sagüés C and Llorente S (2016a) Distributed coverage estimation and control for multi-robot persistent tasks. *IEEE Transactions on Robotics* 32(6): 1444–1460.

Palacios-Gasós JM, Montijano E, Sagüés C and Llorente S (2016b) Multi-robot persistent coverage using branch and bound. In: *American Control Conference*, pp. 5697–5702.

Palacios-Gasós JM, Talebpour Z, Montijano E, Sagüés C and Martinoli A (2017) Optimal path planning and coverage control for multi-robot persistent coverage in environments with obstacles. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1321–1327.

Paley DA, Zhang F and Leonard NE (2008) Cooperative control for ocean sampling: The glider coordinated control system. *IEEE Transactions on Control Systems Technology* 16(4): 735–744.

Panagou D, Stipanović DM and Voulgaris PG (2017) Distributed dynamic coverage and avoidance control under anisotropic sensing. *IEEE Transactions on Control of Network Systems* 4(4): 850–862.

Pavone M, Arsie A, Frazzoli E and Bullo F (2011) Distributed algorithms for environment partitioning in mobile robotic networks. *IEEE Transactions on Automatic Control* 56(8): 1834–1848.

Peters JR, Wang SJ and Bullo F (2017) Coverage control with anytime updates for persistent surveillance missions. In: *American Control Conference (ACC)*, pp. 265–270.

Pierson A, Figueiredo LC, Pimenta LC and Schwager M (2017) Adapting to sensing and actuation variations in multi-robot coverage. *The International Journal of Robotics Research* 36(3): 337–354.

Pimenta L, Kumar V, Mesquita R and Pereira G (2008) Sensing and coverage for a network of heterogeneous robots. In: *IEEE Conference on Decision and Control*, pp. 3947–3952.

Quigley M, Conley K, Gerkey BP, et al. (2009) ROS: An open-source robot operating system. In: *ICRA Workshop on Open Source Software*.

Renzaglia A, Doitsidis L, Martinelli A and Kosmatopoulos EB (2012) Multi-robot three-dimensional coverage of unknown areas. *The International Journal of Robotics Research* 31(6): 738–752.

Sahin H and Guvenc L (2007) Household robotics: Autonomous devices for vacuuming and lawn mowing. *IEEE Control Systems Magazine* 27(2): 20–96.

Sea V, Kato C and Sugawara T (2017) Coordinated area partitioning method by autonomous agents for continuous cooperative tasks. *Journal of Information Processing* 25(0): 75–87.

Smith RN, Schwager M, Smith SL, Jones BH, Rus D and Sukhatme GS (2011) Persistent ocean monitoring with underwater gliders: Adapting sampling resolution. *Journal of Field Robotics* 28(5): 714–741.

Smith SL and Rus D (2010) Multi-robot monitoring in dynamic environments with guaranteed currency of observations. In: *49th IEEE Conference on Decision and Control*. IEEE, pp. 514–521.

Soltero DE, Schwager M and Rus D (2014) Decentralized path planning for coverage tasks using gradient descent adaptive control. *The International Journal of Robotics Research* 33(3): 401–425.

Thanou M, Stergiopoulos Y and Tzes A (2013) Distributed coverage of mobile heterogeneous networks in non-convex environments. In: *21st Mediterranean Conference on Control and Automation (MED)*, pp. 956–962.

Xu A, Viriyasuthee C and Rekleitis I (2014) Efficient complete coverage of a known arbitrary environment with applications to aerial operations. *Autonomous Robots* 36(4): 365–381.

Yun Sk and Rus D (2014) Distributed coverage with mobile robots on a graph: Locational optimization and equal-mass partitioning. *Robotica* 32(2): 257–277.

Appendix A. Index to multimedia extensions

Archives of IJRR multimedia extensions published prior to 2014 can be found at http://www.ijrr.org, after 2014 all videos are available on the IJRR YouTube channel at http://www.youtube.com/user/ijrrmultimedia

Table of Multimedia Extensions

Extension	Media type	Description
1	Video	Accompanying video of the experiment for a complete visualization of the coverage solution