Article

The International Journal of Robotics Research
2019, Vol. 38(7) 813–832
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0278364919845054
journals.sagepub.com/home/ijr
SAGE

A distributed, plug-n-play algorithm for multi-robot applications with a priori non-computable objective functions

Athanasios Ch Kapoutsis[1,2], Savvas A Chatzichristofis[3] and Elias B Kosmatopoulos[1,2]

Abstract

This paper presents a distributed algorithm applicable to a wide range of practical multi-robot applications. In such multi-robot applications, the user-defined objectives of the mission can be cast as a general optimization problem, without explicit guidelines of the subtasks per different robot. Owing to the unknown environment, unknown robot dynamics, sensor nonlinearities, etc., the analytic form of the optimization cost function is not available a priori. Therefore, standard gradient-descent-like algorithms are not applicable to these problems. To tackle this, we introduce a new algorithm that carefully designs each robot's subcost function, the optimization of which can accomplish the overall team objective. Upon this transformation, we propose a distributed methodology based on the cognitive-based adaptive optimization (CAO) algorithm, that is able to approximate the evolution of each robot's cost function and to adequately optimize its decision variables (robot actions). The latter can be achieved by online learning only the problem-specific characteristics that affect the accomplishment of mission objectives. The overall, low-complexity algorithm can straightforwardly incorporate any kind of operational constraint, is fault tolerant, and can appropriately tackle time-varying cost functions. A cornerstone of this approach is that it shares the same convergence characteristics as those of block coordinate descent algorithms. The proposed algorithm is evaluated in three heterogeneous simulation set-ups under multiple scenarios, against both general-purpose and problem-specific algorithms.

Keywords
Distributed robot systems, learning and adaptive systems, cognitive robotics, surveillance systems, autonomous agents

1. Introduction

The new era of artificial intelligence and robotics has an ever-increasing interest in multi-robot systems. The causality of this trend is outlined in the following three points. First, the recent *advances in hardware and communications* allow the cooperative deployment of many affordable robots. Second, the use of multiple robots introduces *redundancy*, which can be translated into mission speed-up and/or fault-tolerant characteristics (e.g., in cases when one or more robots faces a malfunction). Third, the utilization of multi-robot teams may tackle *problems that cannot be solved with a single robot* (e.g., continuous monitoring/guarding a large area). Robot missions in which the multi-robot configuration can be more appealing include surveillance in hostile environments (e.g., areas contaminated with biological, chemical, or even nuclear wastes), law enforcement missions (e.g., border patrol), agriculture activities (e.g., soil sampling), and cleaning missions (e.g., cleaning up an oil spill).

1.1. Related work

Unfortunately, many of the multi-robot tasks have been proven to be extremely difficult. For example, the online generation of robot trajectories so as to maximize

[1]Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece
[2]Information Technologies Institute, The Centre for Research and Technology, Hellas, Thessaloniki, Greece
[3]Department of Computer Science, Neapolis University, Paphos, Cyprus

Corresponding author:
Athanasios Ch Kapoutsis, Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, 67100, Greece.
Email: akapouts@ee.duth.gr

simultaneous localization and mapping (SLAM) accuracy and efficiency is NP-hard (Kollar and Roy, 2008a; Singh et al., 2009). Moreover, the offline design of multi-robot trajectories in order to cover a known area of interest in minimum time/energy has been proven NP-complete (Zheng et al., 2005), etc.

To alleviate the above problem, many multi-robot approaches attempt to solve a simplified version of the original problem. In such a way, it is possible to construct a computationally feasible solution, utilizing *optimal control or dynamic programming* techniques, at the expense, of course, of sacrificing global optimality. For instance, to render the decision-making scheme computationally feasible, many methodologies (De La Cruz and Carelli, 2008; Le Ny and Pappas, 2009; Seyboth et al., 2015) assumed relaxed or linearized versions of the multi-robot problem. A usual assumption is that the robots operate in a discrete space where their actions and measurements can also take values from a finite discrete set of values (Matignon et al., 2012; Spaan and Vlassis, 2005). The exploitation of the above assumption can lead to remarkable results in the context of multi-robot tasks, presenting many real-life applications (e.g., Capitan et al., 2013). Unfortunately, these strategies cannot be fully informed by the (usually occurring) continuous field measurements, whereas they can be computationally intractable for large state systems, e.g., a single mobile robot operating in the real world often has millions of possible states (Roy and Thrun, 1999). Other multi-robot approaches that fall into this class adapt the assumption of perfect or sufficient knowledge of the dynamics of the overall multi-robot system, i.e., the dynamics of each and every robot along with their interactions with the other robots and the external environment (Wang and Schwager, 2016; Zhou and Roumeliotis, 2011). In such cases, the multi-robot problem can be seen to be equivalent to a standard optimization problem, where the robots' decision values are generated according to, e.g., a gradient-descent or gradient-descent-like algorithm (Nesterov, 2007). However, the requirement for perfect or sufficient knowledge of the overall dynamics renders the overall control design practically infeasible in many multi-robot applications, as they typically involve a large number of controllable variables with highly complex and uncertain dynamics (Chen et al., 2015; Gomes et al., 2013; Morgan et al., 2016).

Another well-investigated class of multi-robot approaches is the *optimal one-step-ahead* methodologies. In this family of approaches, the next robots' decision variables are chosen greedily, so as to optimize an appropriately defined cost function that is related to the problem in hand. For instance, in the domain of multi-robot exploration, a common practice is to choose the next robots' positions that maximize the expected information gain (Burgard et al., 2005; Rooker and Birk, 2007; Stachniss and Burgard, 2003) or minimize the trace of the extended Kalman filter (EKF) error covariance matrix (Bourgault et al., 2002; Cui et al., 2016). Although, many of these approaches have been successfully evaluated in real-life multi-robot platforms, the majority of them suffer from the following drawbacks. First and foremost, the nonlinearities may give rise to undesirable divergence (such as in cases where the noise does not follow the additive white Gaussian noise (AWGN) model). For example, it is usually considered that a robot can accurately estimate the position of an object or a point in the environment (landmark/cell) as soon as it perceives it. In most of the existing *optimal one-step-ahead* approaches, this assumption allows in each timestamp the a priori calculation of the cost function, as well as the robots' decision variables that greedily optimize such a cost function. Moreover, such an assumption is crucial for overcoming deadlocks (local minima), which are frequently encountered when greedy approaches are employed (Palacios-Gasos et al., 2016; Rathnam and Birk, 2013). Finally, the selection of an adequate cost function that provides an efficient solution to the multi-robot problem is not always trivial.

On the other side of the spectrum are the *simulation-based* multi-robot methodologies (Kapoutsis et al., 2015b; Kohl and Stone, 2004; Kollar and Roy, 2008b). The idea behind these approaches is as follows. First, a parametrized decision-making mechanism is devised for generating the robot decisions online, with different choices for its parameters, leading to different decision-making mechanisms. Then, realistic simulations or similar tools are used in order to optimize the parameters of the decision-making mechanism. Thus, conceptually, many of the optimization computations that otherwise would take place on the real devices are "moved" offline. The drawbacks of such approaches are as follows: first, the simulations need to cover a wide range of different realistic scenarios (and, thus, they may become "expensive"); and second, because the dimensionality of the optimization problem is quite high, large numbers of parameters are needed in order to come up with an efficient decision-making mechanism.

We close this subsection by mentioning that for most of the centralized approaches, in all three classes, it is not clear how they can be extended to have a distributed nature. Furthermore, the majority of the distributed multi-robot algorithms (e.g., Morgan et al., 2016; Palacios-Gasos et al., 2016; Rathnam and Birk, 2013) exploit application-specific dynamics, therefore their solutions cannot be generalized to a broader context. In other words, if the problem objectives or the dynamics are changed, most of the existing approaches must be redesigned from scratch to adequately tackle the altered problem.

1.2. Contributions

To overcome the aforementioned problems, we propose a new resource optimization algorithm, specifically tailored to the context of multi-robot applications, that extends the cognitive-based adaptive optimization (CAO) algorithm (Kosmatopoulos, 2009). CAO was originally developed and analyzed for the optimization of functions for which an explicit form is unknown but their measurements are available, as well as for the adaptive fine-tuning of large-scale nonlinear control systems (Kosmatopoulos and Kouvelas, 2009; Kouvelas et al., 2011).

In a nutshell, an update cycle on decision variables of the proposed algorithm consists of the following steps. Initially, the robots' measurements are gathered in a central node (robot or base station) where the calculation of the global objective function takes place. In the following, each robot's contribution to the cost function is approximated and forwarded to the corresponding robot. In a fully distributed fashion, each robot constructs a linear-in-the-parameters (LIP) estimator to approximate the (unknown, problem-dependent) evolution of its subcost function. Then, each robot generates random (or pseudo-random) perturbations around its current state and neglects those that violate the operational constraints (if any). Finally, the next robot's action is the one valid perturbation that achieves the best score on the previously constructed estimator.

The proposed algorithm deviates from the original version of CAO in its distributed nature. More precisely, although each robot does not know explicitly either the decision variables of the other robots nor of their measurements, it is able to update its own decision variables effectively in a way to cooperatively achieve the team objectives. The latter can be achieved through a cost function that is exclusive to each robot, designed so as to encapsulate not only the mission objectives but also the other robots' dynamics ("data-driven gradient descent" approach: for more details see Section 3). Rigorous arguments establish that despite the fact that the dynamics that govern the multi-robot system are unknown, the proposed methodology shares the same convergence characteristics as those of block coordinate descent algorithms (Wright, 2015). As exhibited in the presented applications, the distributed nature of the proposed algorithm also allows rapid convergence, especially in cases with many robots.

The contributions with respect to the multi-robot approaches as presented in the previous subsection are as follows.

(i) The problem is formulated in a continuous domain without the need to either know all the states and measurements beforehand, or to perform a relaxation on the original multi-robot problem (*optimal control* and *dynamic programming* approaches). The ability to cope with unknown dynamics (robots–environment) and unknown cost functions imparts a generality to the proposed algorithm, regarding the spectrum of applications that can be utilized.

(ii) However, the main advantage of the proposed algorithm is that it does not require either a priori calculation of the cost function (*optimal one-step-ahead* approaches) or the analytical form of the system to be optimized to be explicitly known (*optimal control* and *dynamic programming* approaches). Instead, the proposed algorithm can cope with cost functions whose calculation can only be achieved by actually performing the corresponding course of actions. Along the same lines, the proposed algorithm does not require evaluation of the decision variables in the vicinity of their current values for calculating their corresponding updates. Instead, the proposed algorithm is able to find the (locally) optimal configuration for the decision variables by using only noise-corrupted measurements collected from the robots' sensors.

(iii) Furthermore, instead of relying on exhaustive, computationally intensive simulations (*simulation-based* approaches), the proposed scheme is able to online learn the problem-specific characteristics that affect the user-defined objectives. By doing so, the proposed algorithm does not need any elaborate model in order to learn its decision-making mechanism.

It must be emphasized that apart from rendering the optimization problem practically solvable, the proposed approach preserves additional features that make it particularly tractable:

(i) its complexity is low, allowing *real-time implementations*;

(ii) it can handle a variety of *physical constraints*;

(iii) it has *fault-tolerant characteristics*, i.e., online redesign in case one or more robots being added or removed, an extra task being added to the set of objectives, etc.;

(iv) it is able to adapt its behavior even in cases where a *time-varying objective function* is employed.[1]

1.3. Simulation testbeds

The proposed control strategy is evaluated on *three different simulation set-ups* under multiple scenarios, against both general-purpose and problem-specific algorithms. All the simulation set-ups have been chosen so that: (i) the objective of the multi-robot mission can be expressed as a cost function, and (ii) the evaluation of which cannot be performed beforehand.

In the first simulation set-up, the objective is to spread out the robots over a 2D environment while aggregating in areas of high sensory interest. An important aspect of the set-up is that the robots are not aware beforehand of the sensory areas of interest - instead, they learn this information online via sensor measurements from their current positions. The proposed algorithm is evaluated together with the approach proposed by Schwager et al. (2009) for the problem in hand.

In the second simulation set-up, the trajectories of the robots should be designed in real-time having a twofold objective (which forms a trade-off). On the one hand, the part of the 3D terrain that is monitored (i.e., visible) by the robots has to be maximized and, on the other hand, for each one of these visible points in the terrain, the closest robot has to be as close as possible to that point. This problem along with a centralized CAO-based methodology has been proposed by Renzaglia et al. (2012), therefore a detailed analysis regarding the performance of both algorithms, in different scenarios, is presented.

Last but not least, the proposed methodology is evaluated in the task of persistent coverage. The objective of this application is to maintain a user-defined level of coverage in an unknown environment (Palacios-Gasós et al., 2016). This is a quite challenging task as the mission objectives constantly change, whereas the unknown morphology of the environment does not allow the prior calculation of the improvement in the coverage task.

Conclusively, if it is possible to define a cost function which encapsulates the mission objectives and can be calculated through the robots' measurements for every decision variables configuration, the proposed methodology will be directly applicable to the corresponding problem.

1.4. Paper structure

The remainder of the paper is structured as follows. Section 2 presents the translation of a general-purpose multi-robot framework to a constrained optimization problem, highlighting the difficulties and the obstacles of the general problem. The description of the proposed algorithm, which tackles such a problem, is presented in Section 3. Sections 4, 5, and 6 present three indicative multi-robot applications: *adaptive coverage of unknown 2D environment, 3D surveillance of unmapped terrains*, and *persistent coverage of unknown 2D environments*, respectively. In all these sections, we perform a series of simulations in different scenarios to adequately analyze the performance of the proposed algorithm. The overall conclusions of the paper are drawn in Section 7.

2. Problem formulation

Consider a team (swarm) that consists of N robots interacting with each other, towards achieving a global set of objectives. Let us assume the following augmented decision vector

$$\mathbf{x}(k) \equiv \left[x_1^\tau(k), x_2^\tau(k), \ldots, x_N^\tau(k)\right]^\tau \tag{1}$$

where $x_i(k) \in \mathbb{R}^n$ denotes the decision variables of the i th robot at the k th iteration. These decision variables represent the controllable parameters of the available robots (e.g., position, motors, propellers, thrusters, rotation of the cameras, etc.). Furthermore, the augmented vector which contains the available exteroceptive measurements takes the form

$$\mathbf{y}(k) \equiv \left[y_1^\tau(k), y_2^\tau(k), \ldots, y_N^\tau(k)\right]^\tau \tag{2}$$

where $y_i(k) \in \mathbb{R}^m$ denotes the measurement vector of the i th robot at the k th iteration and its evolution can be represented as

$$y_i(k) \equiv h_i(k, x_i(k)) \tag{3}$$

where $h_i(\cdot)$ denotes an unknown, nonlinear function that depends on both $x_i(k)$ and the specific problem characteristics.

The accomplishment of the multi-robot system's objectives (e.g., mapping, surveillance, etc.) can be translated into the minimization (or maximization)[2] of a specifically defined global cost function j_k, i.e.,

$$j_k \equiv \mathcal{J}(x_1(k), x_2(k), \ldots, x_N(k)) \tag{4}$$

where $\mathcal{J}(\cdot)$ is a non-negative, nonlinear, scalar function that depends, apart from the robots decision variables, on the particular dynamics of the problem (e.g., the environment where the robots operate). Owing to the dependence of the function $\mathcal{J}$ on the particular problem characteristics, the *explicit form of the function $\mathcal{J}$ is not known* in practical scenarios; as a result, standard optimization algorithms (e.g., gradient descent with an apriori model) are not applicable. However, in most practical cases, the current value of the objective function can be approximated from the robots' measurements,

$$\mathcal{J}(x_1(k), \ldots, x_N(k)) = \mathrm{J}(y_1(k), \ldots, y_N(k)) + \xi_k \tag{5}$$

where ξ_k denotes the noise introduced in the estimation of j_k, owing to the presence of noise in the robots' sensors.[3] It must be emphasized that, in contrast to $\mathcal{J}$, J can be evaluated "offline," if the measurement vector $\mathbf{y}(k)$ is available. However, the *acquisition of a new measurement vector* requires an *actual evaluation* of the decision variables on the robotic system (2) and (3).

Apart from the problem of dealing with a criterion for which an explicit form is not known, but only its noisy measurements are available at each time, the decision vector $\mathbf{x}(k)$ should satisfy a set of constraints that, in general, can be represented as follows:

$$\mathcal{C}(\mathbf{x}(k)) \leq 0 \tag{6}$$

where $\mathcal{C}$ is a set of nonlinear functions of the decision variables $\mathbf{x}(k)$. As in the case of $\mathcal{J}$, the constraints function $\mathcal{C}$ depends on the particular problem characteristics and an explicit form of this function may be not known in many practical set-ups; however, it is natural to assume that the low-level algorithm is provided with information whether a particular selection of decision variables $\mathbf{x}(k)$ satisfies or violates the set of constraints (6).

Given the mathematical description presented above, the problem of choosing the decision variables online for a multi-robot system, so as to accomplish a set of objectives, can be mathematically described as the following constrained optimization problem:

$$\begin{aligned} &\text{minimize } j_k \\ &\text{subject to} \quad \mathcal{C}(\mathbf{x}(k)) \leq 0 \end{aligned} \tag{7}$$

As already noted, the difficulty in solving the constrained optimization problem (7) in real time lies in the

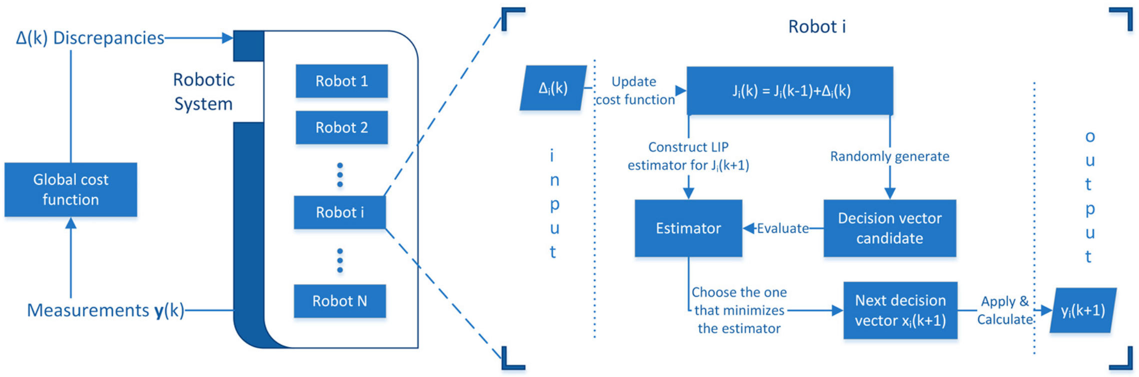

Fig. 1. High-level diagram of the proposed algorithm. At each timestamp, all the operational robots first apply their decision commands and acquire the corresponding measurements $\mathbf{y}(k)$, in order to be able to calculate the *global cost function* index. Then, the contribution $\Delta_i(k)$ of the each robot to overall accomplishment of mission objectives is calculated and sent to the ith robot. In a fully distributed fashion, each robot constructs a linear-in-the-parameters estimator to approximate the (unknown - problem dependent) evolution of its sub-cost function $J_i(k)$, which encapsulates both the mission objectives and the operational capabilities of the multi-robot team. Finally, each robot's next decision vector $x_i(k+1)$ is the one valid perturbation that achieves the best score on the previously constructed estimator.

fact that explicit forms for the function $\mathcal{J}$ and $\mathcal{C}$ are not available. Although this is not the only problem, jointly optimizing a function over multiple robots (N), each of which with multiple decision variables (n), can incur excessively high computational cost.

3. Proposed algorithm

Having defined the fundamental aspects that govern a multi-robot application, we proceed to present the proposed algorithm for updating the decision variables $\mathbf{x}(k)$ so as to minimize the cost function (4) subject to (6). A high-level diagram of the proposed algorithm is sketched in Figure 1.

3.1. Global coordination

Step 1. As a first step, and for each iteration k, the robots transmit the acquired measurements, after the execution of $\mathbf{x}(k)$ decision variables.

It must be emphasized that this step can be performed even in cases where global communication between all robots is not feasible. In such a case, each robot can send and receive measurements to and from peer (adjacent) robots, until all the measurements aggregate to the corresponding processor unit (robot or ground station). The latter can be guaranteed by introducing an extra condition on the constraints set (6), ensuring the connectivity, if applicable to the problem in hand, among the different robots.

Step 2. Thus, the global cost function can be straightforwardly derived from (see (4) and (5))

$$j_k = J(y_1(k), \ldots, y_N(k))$$

In addition, for each i th robot, calculate the following discrepancy:

$$\Delta_i(k) \equiv j_k - \mathrm{J}(y_1(k), \ldots, y_{i-1}(k), y_i(k-1), y_{i+1}(k), \ldots, y_N(k)) \tag{8}$$

In other words, $\Delta_i(k)$ encapsulates the effect of the $x_i(k)$ on the current problem for the kth timestamp.

Note that, because the last term of (8) is analytically available, we can calculate this term, although the resulting value does not necessarily correspond to the actual value when the robots have the following decision variables:

$$\{x_1(k), \ldots, x_{i-1}(k), x_i(k-1), x_{i+1}(k), \ldots, x_N(k)\}$$

Although there may be a discrepancy between the way we calculate $J(\cdot)$ and its actual value, that does not affect the convergence properties of the proposed algorithm. This discrepancy is application oriented and depicts the effect of other robots' decisions on each robot's measurements. If the measurements acquired from a robot only affects its own decision variables and the problem itself (3), then there is no discrepancy at all.

Step 3. Next, the calculated discrepancy $\Delta_i(k)$ is sent to the ith robot.

After this step all the calculations are performed locally, building a system that (i) is resilient to robot failures, (ii) does not require any global coordination, and (iii) all the decision variables' updates are made in a (parallel) distributed fashion.

3.2. Distributed decision

Each ith robot, at the same kth iteration, performs the following.

(a) Update $J_i(k)$ that corresponds to the last executed decision variables $x_i(k)$ as

$$J_i(k) = J_i(k-1) + \Delta_i(k), \quad \forall k \geqslant 1,\ J_i(0) = j_0 \tag{9}$$

Therefore, each robot is responsible for choosing the next values for its decision variables $x_i(k+1)$, having as sole objective the minimization of its corresponding cost function $J_i(\cdot)$.[4] Each such subproblem is a lower-dimensional minimization problem, and thus can typically be solved more easily than the full problem.

(b) Construct a *LIP* estimator of $J_i(k+1)$ as follows:

$$J_i(k+1) \approx \hat{J}_i(k+1) = \theta_i^\tau(k)\phi_i(x_i(k)) \tag{10}$$

where ϕ_i denotes the nonlinear vector of L *regressor terms*, θ_i denotes the vector of the *parameter estimates*, and L is a positive user-defined integer which denotes the size of the function approximator (10). Defining the vector of regressor terms ϕ_i as in Section 3.3.1, the estimator vector θ_i can be calculated using standard least-squares estimator principles, i.e., θ_i is obtained by solving the following optimization problem:

$$\theta_i(k) = \underset{\vartheta}{\operatorname{argmin}} \sum_{\ell = k-T(k)}^{k-1} (\vartheta^\tau \phi_i(x_i(\ell)) - J_i(\ell+1))^2 \tag{11}$$

where $T(k)$ denotes the time window over which the least-squares estimation is taking place.

(c) Generate (randomly or pseudo-randomly) a set of M valid candidate perturbations:

$$\delta x_i^{(1)}(k), \delta x_i^{(2)}(k), \ldots, \delta x_i^{(M)}(k)$$

where $\delta x_i^{(j)}(k)$ are vectors of the same dimension as $x_i(k)$ and M is a positive integer that is larger[5] than $2n$. A candidate perturbation j is considered valid if[6]

$$\mathcal{C}\left(\left[x_1^\tau, \ldots, x_{i-1}^\tau, x_i^\tau + \delta x_i^{(j)}, x_{i+1}^\tau, \ldots, x_N^\tau\right]^\tau\right) \leqslant 0 \tag{12}$$

The random choice for the candidates is essential and crucial for the efficiency of the algorithm, as such a choice guarantees that $\hat{J}_i(k+1)$ is a reliable and accurate estimate for $J_i(k+1)$; see Kosmatopoulos (2009) and Kosmatopoulos and Kouvelas (2009) for more details.

(d) Estimate the effect of each of the candidate perturbations on the current vector $x_i(k)$ by employing the previously constructed estimator (10) and pick the candidate perturbation with the "best" effect, i.e., choose the vector $\delta x_i^{(j^*)}(k)$ that satisfies

$$\delta x_i^{(j^*)}(k) = \underset{j=1,\ldots,M}{\operatorname{argmin}}\ \theta_i^\tau(k)\phi_i(x_i(k) + \alpha(k)\delta x_i^{(j)}(k))$$

(e) Update the i th robot decision variables as

$$x_i(k+1) = x_i(k) + \alpha(k)\delta x_i^{(j^*)}(k) \tag{13}$$

where $\alpha(k)$ is a positive function chosen to be either a constant positive function or a time-descending function satisfying $\alpha(k) > 0$, $\sum_{k=0}^{\infty}\alpha(k) = \infty$, $\sum_{k=0}^{\infty}\alpha(k)^2 < \infty$. Furthermore, $\alpha(k) \leqslant \bar{\alpha}\ \forall k$, where $\bar{\alpha}$ is a problem-specific constant, correlated with the robot's dynamics (e.g., maximum achievable movement in one timestamp) and the objectives of the multi-robot application.

(f) Finally, by applying the $x_i(k+1)$ decision vector, the corresponding $y_i(k+1)$ measurements vector will be acquired. This vector, along with all the measurements from the remaining robots, are utilized in order to evaluate the $k+1$ team configuration (see *Step 1* from the previous subsection).

Remark 1. The above distributed update of the decision variables (Section 3.2) does not need information about what is happening to the other robots. All the necessary information has been "packed" to the scalar value $\Delta_i(k)$. At each iteration, each robot attempts to minimize the objective function $J_i(k)$ by assuming that the other robots' decision variables are part of the problem to be solved.

Remark 2. The utilization of random perturbations provides the proposed algorithm with the potential to escape from local minima. In essence, the random perturbations inside the distributed decision mechanism (step (c)), could have a behavior similar to *simulated annealing*, which has been proved that under specific conditions can overcome local minima (Granville et al., 1994) that may arise from the distributed nature of the algorithm.

3.3. Estimator's implementation details

This subsection encloses the implementation details of the i th robot estimator (10), as outlined in step (b) of the distributed decision-making scheme.

3.3.1. ϕ monomial construction. The vector ϕ_i of regressor terms must be chosen so that it satisfies the so-called *Universal Approximation Property* (Polycarpou and Ioannou, 1991), i.e., it must be chosen so that the approximation accuracy of the constructed approximator (10) is an increasing function of the approximator's size L. Polynomial approximators, radial basis functions, kernel-based approximators, etc. are known to satisfy such a property (Polycarpou and Ioannou, 1991). Experimenting with different types of ϕ_i, in different multi-robot set-ups (Sections 4–6, Kapoutsis et al. (2013), and Kapoutsis et al. (2015a)), it was found that it is sufficient to construct a polynomial estimator as in Algorithm 1.

The tunable parameters of this procedure are the maximum order of monomials (maxorder) and the corresponding number of monomials per order ($L_1, L_2, \ldots, L_{\text{maxorder}}$,

Algorithm 1. ϕ_i construction

```
Input: maxorder, $L_1, L_2, \ldots, L_{\text{maxorder}}$, $x_i$, $n$
Output: $\phi_i$
1: $\phi_i = 1$
2: for $j \in \{1, \ldots, \text{maxorder}\}$ do
3:     for $v \in \{1, \ldots, L_j\}$ do
4:     $g = 1$
5:     for $l \in \{1, \ldots, j\}$ do
6:     Generate $r := $ random integer $\in \{1, \ldots, n\}$
7:     $g = g \cdot x_i^{(r)}$
8:     end for
9:     $\phi_i = [\phi_i^\tau, g]^\tau$
10:    end for
11: end for
```

where $L_1 + L_2 + \cdots + L_{\text{maxorder}} = L - 1$ should be hold). Mathematically speaking, the number of different monomials per order is given by the number of possible combinations with repetitions (multiset coefficient):

$$\left(\!\!\binom{n}{i}\!\!\right) = \binom{n+i-1}{i} = \frac{n(n+1)(n+2)\cdots(n+i-1)}{i!}$$

where $\binom{a}{b}$ denotes the binomial coefficient. However, the summation $L_1 + L_2 + \cdots + L_{\text{maxorder}}$ may exceed the number of available monomials $L - 1$. A usual practice is to downscale the number of monomials as follows:

$$L_i = \left[\binom{n+i-1}{i} s\right] \tag{14}$$

where $[\cdot]$ denotes the nearest integer and s denotes the following scaling factor

$$s = \frac{L-1}{\sum_{i=1}^{\text{maxorder}} \binom{n+i-1}{i}}$$

3.3.2. Solving the least-squares problem. It is worth pointing out that although the $\hat{J}_i(k+1)$ is evolving in a nonlinear fashion with respect to x_i, standard linear regression techniques can be utilized to find θ_i, as (10) is still linear in the parameters' vector. Therefore, the least-square problem as defined in (11) can be solved by several algorithms (normal equation, QR decomposition, SVD, etc.). Although, singular value decomposition (SVD) is more computational intensive in comparison to other alternatives, we utilize this approach due to the fact that it is more numerical stable (e.g., when the problem is ill-conditioned) (Demmel, 1997).

3.4. Convergence analysis

Remark 3. As shown in Kosmatopoulos (2009); Kosmatopoulos and Kouvelas (2009), the distributed algorithm implemented in each robot (Section 3.2) guarantees that if $M \geqslant 2 \times \dim(x_i)$, the vector ϕ satisfies the universal approximation property and the functions J_i and C are either continuous or discontinuous with a finite number of discontinuities, then the update rule of x_i (13) is equivalent to

$$x_i(k+1) = x_i(k) - A(k)\nabla_{x_i} J_i + \epsilon(k)$$

where $A(k)$ is a positive-definite matrix that depends on the choice of α (see step (e) of the distributed decision-making scheme) and $\nabla_{x_i} J_i$ denotes the gradient of J_i with respect to the x_i decision variables. In addition, $\epsilon(k)$ is a term that converges exponentially fast to zero with probability one. In simple words, the analysis of Kosmatopoulos (2009); Kosmatopoulos and Kouvelas (2009) establishes that the algorithm will converge to a local minimum of J_i.

The following theorem describes the properties of the proposed methodology; as the proof of this theorem is along the same lines as in Bertsekas (1999: Proposition 2.7.1), only a sketch of proof is provided.

Theorem 1. *The local convergence of the proposed algorithm can be guaranteed in the general case where the global cost function $\mathcal{J}$ and each robot's contribution J_i are non-convex, non-smooth functions.*

Sketch of the proof: By using Remark 3 (*projected gradient-descent* on the minimization of J_i) and Equations (8)–(9), we can establish that the distributed update on each robot is equivalent to

$$x_i = \underset{w}{\operatorname{argmin}}\, \mathcal{J}(x_1, \ldots, x_{i-1}, w, x_{i+1}, \ldots, x_N)$$

subject to (12) and, therefore, the proposed algorithm approximates the behavior of the block coordinate descent (BCD) (Wright, 2015: Algorithm 1) family of approaches. Following the proof described in Bertsekas (1999: Proposition 2.7.1), it is straightforward to see that if the minimum with respect to each block of variables is unique, then any accumulation point of the sequence $\{x(k)\}$ generated by the BCD methodology is also a stationary point.[7]

3.5. Complexity

The computational burden regarding the global coordination (section 3.1) is accumulated in the calculation of $\Delta_i(k)$ (8) for each robot i. However, the calculation of $J(\cdot)$ is problem-dependent, thus, it is not possible to analytically derive bounds regarding its complexity. In the reported cases (cost functions (19), (21), (23), and (27)), as well as in most real-world applications, the computational needs of $J(\cdot)$ grow, at most, quadratic with the number of robots $\times$ the number of measurements per robot, i.e., $\mathcal{O}(N^2m^2)$. Technically, the above threshold expresses the case where an operation is needed per different pair of measurements $\{y_a^{(i)}, y_b^{(j)}\}$, with $a, b \in \{1, \ldots, m\}$ and $i, j \in \{1, \ldots, N\}$. Overall, $J(\cdot)$ is evaluated $N+1$ times, one for each robot and one for the global cost function term (8); therefore, *Steps 1–3* are expected to have $\mathcal{O}(N^3m^2)$.

The computational requirements for the distributed decision (Section 3.2), which is computed on each robot, are dominated by the requirement of solving the least-squares problem (11). According to Golub and Van Loan (2012: Section 5.5.6, Figure 5.5.1), the best algorithms for least-squares problem using SVD procedure, take time that is proportional to $\mathcal{O}(T^2L + L^3)$. In the interest of simplicity, and owing to the fact that $T \simeq L$, we can assume that the complexity for the distributed decision scales as $\mathcal{O}(L^3)$. Although, there exist no theoretical results for providing the lower bound $\bar{L}$ for the size of the regressor vector, practical investigations on many different applications (e.g., Amanatiadis et al., 2013; Kapoutsis et al., 2015a; Korkas et al., 2016) indicate that it is sufficient enough to choose $L \geqslant \bar{L} = 2 \times n$, to adequately tackle the local approximation of J_i. Therefore, it is expected that the computational requirements will grow with $\mathcal{O}(n^3)$. Although this step is executed on each robot (N times), the distributed nature of the algorithm guarantees that no extra computational needs will be required.

Overall, it is expected that the complexity of computing N times the cost function $J(\cdot)$ dominates the requirement of solving the least-squares problem for one robot. Table 1 summarizes the complexity bounds discussed in this section.

Remark 4. We close this section by accumulating the free parameters of the proposed algorithm. The set is composed of the number of perturbations M, the total number of utilized monomials L, and the time window T over which the least-squares estimation is taking place. According to Remark 3, the number of perturbations M should be greater than $2 \times n$. Furthermore, the complexity analysis of Section 3.2 indicates that the estimator (10) should have at least $\bar{L} = 2 \times n$ number of monomials. Finally, T is a non-negative integer that expresses the desired "forgetting factor" for the constructed estimator. In the following experimental set-ups, we set the algorithm's parameters within these bounds. Alternatively, and if required, all parameters mentioned could be manually tuned in order to achieve better, application-dependent, performance.

4. Adaptive coverage control utilizing Voronoi partitioning

The first simulation set-up is the well-investigated optimal robots' placement problem (Cortes et al., 2002; Schwager et al., 2006, 2009). The objective for the network of robots is to spread out over an environment, while aggregating in areas of high sensory interest. Furthermore, the robots do not know beforehand where the areas of sensory interest are, but they learn this information online from sensor measurements. The aforementioned task can be found in applications such as environmental monitoring and clean-up, automatic surveillance of rooms/buildings/towns, or search-and-rescue missions.

Table 1. Complexity analysis.

Step	Complexity	Practical	Comments
1–3	$\mathcal{O}(J(\mathbf{y}))$	$\mathcal{O}(N^3m^2)$	Application dependent
4	$\mathcal{O}(L^3)$	$\mathcal{O}(n^3)$	Least-squares

4.1. Problem definition

It is assumed that the operational area is a bounded $Q \subset \mathbb{R}^n$. A point inside this environment is denoted by q and the decision vector x_i for the i th robot contains its position in Q. In addition, let $\{V_1, \ldots, V_N\}$ be the Voronoi partition of Q, for which the robot positions are the generator points:

$$V_i = \{q \in Q | \|q - x_i\| \leqslant \|q - x_j\|, \forall j \neq i\}$$

(Henceforth, we use $\|\cdot\|$ to denote the Euclidean norm $\|\cdot\|_2$). Let $\zeta(\cdot)$ be the unknown sensory function such that $\zeta : Q \to \mathbb{R}_{>0}$ (where $\mathbb{R}_{>0}$ is the set of strictly positive real numbers). In other words, this function $\zeta(\cdot)$ assigns in each location of the available space Q a weight of importance related to the necessity of being covered.

The global cost function for the problem in hand, admits the following form:

$$\mathcal{J}(\mathbf{x}(k)) = \sum_{i=1}^{N} \int_{V_i} \frac{1}{2} \|q - x_i\|^2 \zeta(q) dq \tag{15}$$

Apparently, the above function cannot be calculated in advance owing to the dependence of the unknown sensory function ζ. Without loss of generality, we assume that the sensory function is given by

$$\zeta(q) = \mathcal{K}(q)^\tau \upsilon + \mathcal{O}(1/W), \quad \forall q \in Q \tag{16}$$

where $\mathcal{K} : Q \to \mathbb{R}^W_{>0}$ denotes a vector of bounded, continuous basis functions (e.g., Gaussians, wavelets, sigmoids, etc.) and $\upsilon \in \mathbb{R}^W$ is the parameter vector. The deviation from the actual value of ζ is of the order of the number of basis functions $\mathcal{O}(1/W)$. Although $\mathcal{K}$ is defined a priori, the mixing parameters vector υ is environment-dependent and generally unknown. However, the value of the sensory function can be measured from the robots' sensors (e.g., temperature/chemical sensor) at their current position's configuration $\mathbf{x}(k)$:

$$y(x_i) = \zeta(x_i) \tag{17}$$

The value of the parameter estimation vector $\hat{\upsilon}$ can be approximated through these measurements, utilizing standard parameter estimation techniques (e.g., least-squares approach (11)). Therefore, after the update on the parameter vector $\hat{\upsilon}$, a new update on the belief regarding the sensory function is also available through the equation

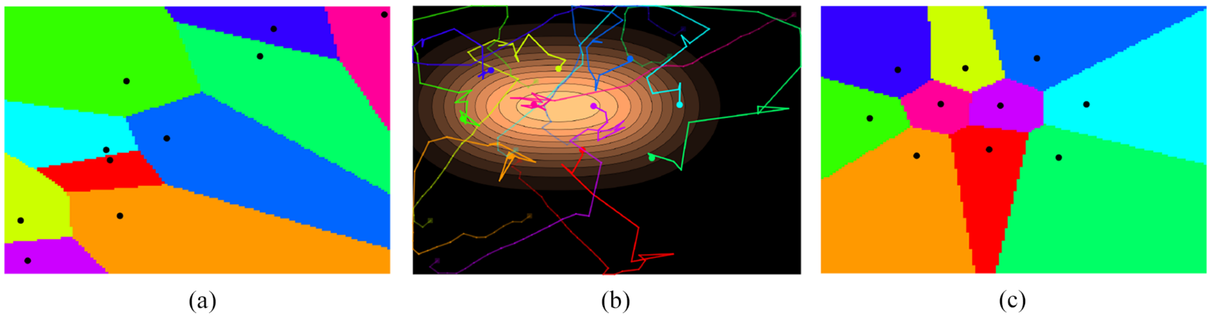

Fig. 2. Illustrative example with random initial positions for the robots. (a) Initial Voronoi partitioning. (b) Robots' trajectories on top of the heatmap of the sensory function. The squares and the circles denote the initial and the final positions of the robots, respectively. (c) Voronoi partitioning of the final configuration. In these figures, we sketch how the proposed algorithm drives the available robots so as to completely cover the space and to aggregate around areas with high sensory interest.

$$\hat{\zeta} = \mathcal{K}^{\tau}\hat{v} \tag{18}$$

Hence, the value of the unknown cost function can be approximated through the following equation:

$$J(\mathbf{y}(k)) = \sum_{i=1}^{N} \int_{V_i} \frac{1}{2}\|q - x_i\|^2 \mathcal{K}^{\tau}(q)\hat{v}\, dq \tag{19}$$

4.2. Simulation results

For implementation reasons, we assume that the operation area consists of 225 discrete points, uniformly distributed across the plane of $[0, 1]^2$. The sensory function, $\zeta(q)$, was parameterized as a linear combination of 49 Gaussians, i.e., $\mathcal{K}(j) = \frac{1}{2\pi\sigma_j^2}\exp - \frac{(q-\mu_j)^2}{2\sigma_j^2}$, $\forall j \in \{1, \ldots, 49\}$. Each standard deviation is set to be $\sigma_j = 0.02$ and the Gaussians centers μ_j are chosen so as to be uniformly distributed in the operational area (seven Gaussians in each row and column). The parameter $v = [v_1, v_2, \ldots, v_{49}]^{\tau}$ was chosen so that $v_i = 0.1, \forall i \in \{1, \ldots, 49\}$, apart from two random integers $a, b \in \{1, \ldots, 49\}$ whereas $v_a = v_b = 100$. In other words, for each simulation instance, the sensory function $\zeta(q)$ was dominated by two, randomly selected, Gaussians. Finally, the equations are integrated using a fixed step of $\alpha = dt = 0.01$ and the initial values for the estimation of parameter vector (robots' knowledge) was chosen to be $\hat{v} = [0.1, 0.1, \ldots, 0.1]^{\tau}$.

In addition, with the proposed approach, we present simulation results from the algorithm as proposed, for the problem in hand, by Schwager et al. (2009). The weights' selection was undertaken following the authors' instructions in Schwager et al. (2009: Section 7.2). To construct comparable simulations instances, we utilize the same learning rule for the parameter vector $\hat{v}$ (Schwager et al., 2009: equation (13)). In both the evaluated algorithms, the update of parameter vector was performed, by aggregating all the robots' measurements. To evaluate the performance of each approach in each timestamp, we also calculate the real value of the cost function (15), but none of the evaluated algorithms utilizes this information.

The proposed approach was employed with a constant time window for the least-squares estimation of $T = 30$ and the number random perturbations was set to $M = 100$. To approximate each robot's cost function evolution, we utilize a third-order monomial estimator with $L = 10$ and using (14) we calculate the number of monomials per order to be $L_1 = 2$, $L_2 = 3$, and $L_3 = 4$.

4.2.1. Random initial positions scenario. In the first simulation scenario, the robots were placed randomly along the x and y axes of the operation area. An example of this simulation set-up is illustrated Figure 2, where Figure 2(a) sketches the Voronoi partitioning for the initial robot configuration, Figure 2(b) illustrates the robots' trajectories from their initial positions (squares) to the final configuration (circles), and, finally, Figure 2(c) illustrates the Voronoi partitioning for the final robots' positions. As one can see, the robots gathered around the areas with the highest values of the unknown sensory function $\zeta(\cdot)$.

Figure 3 presents a comparison study between the evaluated algorithms, over different sizes of robot teams. The number of robots was chosen to be 10, 15, 20, and 25 robots, and for each configuration 60 experiments with randomly selected initial robots' placement and sensory function were performed. The average, final achieved cost function (15) values, along with the corresponding confidence intervals are illustrated in Figure 3(a). In addition, we present the summation of the cost function over the course of each simulation pair (Figure 3(b)). It must be emphasized that, although the summation of the cost function may be strongly dependent on the initial robots' positions the final achieved value has a small variance around the average value. This feature highlights the ability of the proposed approach to converge to an optimal configuration, independently of the initial conditions.

4.2.2. Right half-plane scenario. In the second simulation scenario, the robots' initial positions were constrained inside the right half-plane of the operation area. In general, this

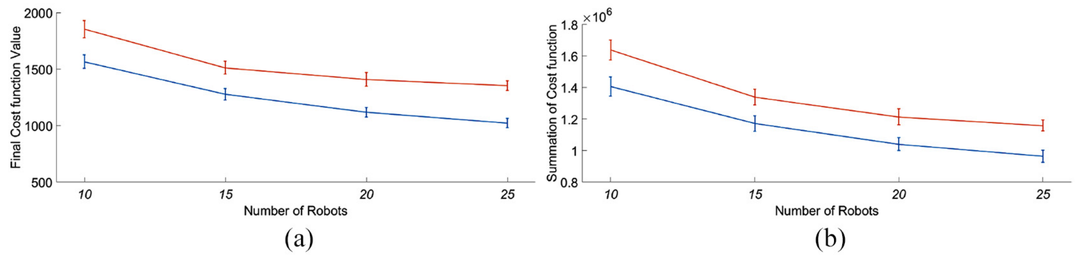

Fig. 3. Comparison study for the *random initial positions scenario*: proposed algorithm (blue) and approach presented by Schwager et al. (2009) (red). (a) Final achieved value of the cost function. (b) Summation of the cost function over the experiment's horizon.

scenario has a greater level of difficulty, compared with random initialization, as the robots can easily get stuck in highly suboptimal situations. Figure 4 illustrates an instance of such a scenario where the proposed approach was utilized.

As in the previous scenario, we present a comparison between the evaluated algorithms for different sizes of robot teams. The results are illustrated in Figure 5. Again, the proposed approach utilizes all the available team resources in order to achieve optimal robot configurations with small variance around the average values.

5. Three-dimensional surveillance of unknown areas

A more elaborate variation of the previously described set-up has been proposed by Renzaglia et al. (2012) and applied in several domains (e.g., Kapoutsis et al., 2015a; Scaramuzza et al., 2014). Although the problem is again the optimal placement of robots in real time, the details of the simulation set-up are important. First and foremost, the robots are moving inside a 3D space (e.g., unmanned aerial vehicles). The terrain to be covered is considered an unknown, non-convex, 3D surface the formation of which may form an arbitrary number and shape of obstacles. Furthermore, a realistic model for the robots' sensors is employed and utilized in all the simulation scenarios.

5.1. Problem definition

In this simulation testbed, the decision variables (1) represent the positions of the robots in 3D space, i.e., $\mathbf{x} = \left[x_1^\tau, \ldots, x_N^\tau\right]^\tau$, where $x_i \in \mathbb{R}^3$. It is assumed that the area to be monitored is constrained within a rectangle in the (x, y) coordinates as

$$\mathcal{U} = \{x, y | x \in [x_{\min}, x_{\max}], y \in [y_{\min}, y_{\max}]\}$$

where $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ are real numbers that define the "borders" of the area of interest. Using the definition of $\mathcal{U}$, the area can be defined as a function that maps each point $(x, y) \in \mathcal{U}$ to a point $z = z(x, y)$ (height of unknown terrain at (x, y)). A point $q = (x, y, z)$ of the terrain is *visible* if there exists at least one robot so that:

- the robot x_i and the point q are connected by line of sight;
- $\|x_i - q\| \leq thres$, where *thres* defines the maximum distance the i th robot can "see."

Given the robot configuration $\mathbf{x}(k)$ at timestamp k, we let $\mathcal{V}$ denote the *visible* area of the terrain, i.e., $\mathcal{V}$ consists of all points $q \in \mathcal{U}$ that are *visible* from the robots.

Furthermore, the measurements' model for all the robots admits the following form:

$$y_{x_i - q} = \begin{cases} \|x_i - q\| + h_\xi(x_i, q)\xi & \text{if } q \in \mathcal{V} \\ \text{undefined} & \text{otherwise} \end{cases} \quad \forall q \qquad (20)$$

where $h_\xi(x_i, q)$ is the multiplicative sensor noise term ($e.g.$, $\propto \|x_i - q\|^2$) and ξ is a standard Gaussian noise. The above nonlinear noise model is a realistic representation of the noise effect in many real robotic systems; see Salavasidis et al. (2016) and Teixeira (2007: Chapters 3 and 4). For instance, in the case of sonar or cameras, the noise affecting such sensors is proportional to the sensor-to-sensing-point distance, i.e., the larger the robot-to-sensing point distance, the larger the sensor noise (Scaramuzza et al., 2014).

Having the above formulation in mind, we define the following combined cost function that the team of robots has to minimize

$$J(\mathbf{y}(k)) = \int_{q \in \mathcal{V}} \min_{i = 1, \ldots, N} y_{x_i - q} \, dq + K \int_{q \in \mathcal{U} \setminus \mathcal{V}} dq \qquad (21)$$

The fist term is equivalent to the cost function considered in many coverage problems for known 2D environments (Choset, 2001; Cortes et al., 2002). The second term of (21) is related to the invisible area in the terrain. The positive constant K serves as a weight for giving less or more priority to one of the objectives.

Moreover, the set of nonlinear constraints (6), which must be held for each new robots' configuration $\mathbf{x}(k)$, include the following:

- the robots remain within the terrain's limits, i.e., within $[x_{\min}, x_{\max}]$ and $[y_{\min}, y_{\max}]$ in the x and y axes, respectively;

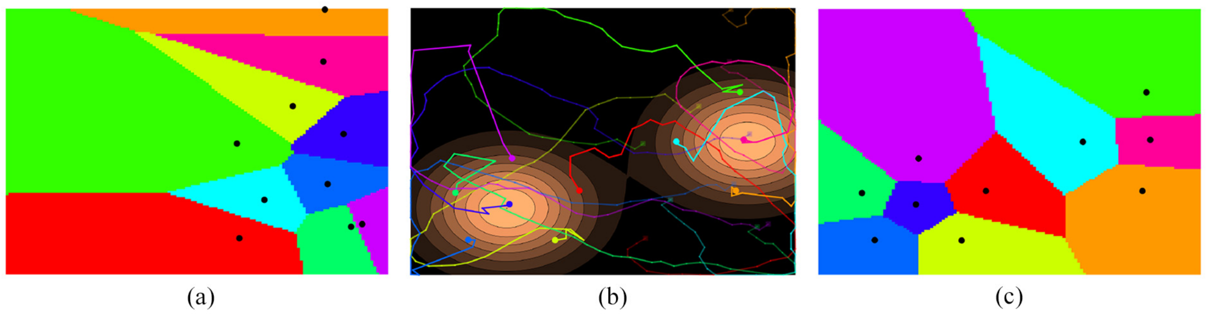

Fig. 4. Illustrative example where the robots initial positions are constrained inside the right half-plane of the operational environment. The proposed algorithm navigates the robots around the space, utilizing only their measurements on their current positions, to achieve the mission objective. (a) Initial Voronoi partitioning. (b) Robots' trajectories on top of the heatmap of the sensory function. The squares and the circles denote the initial and the final positions of the robots, respectively. (c) Voronoi partitioning of the final configuration.

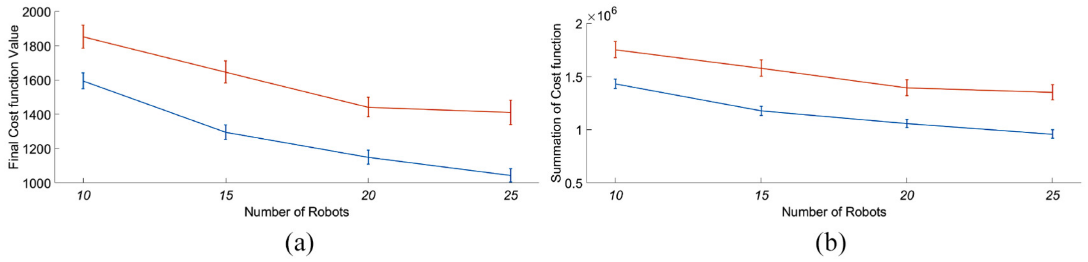

Fig. 5. Comparison study for the *right half-plane scenario*: proposed algorithm (blue) and approach presented by Schwager et al. (2009) (red). (a) Final achieved value of the cost function. (b) Summation of the cost function over the experiment's horizon.

- the robots satisfy a maximum height requirement, while they do not hit the terrain, i.e., they remain within $[z+d_h, z_{\max}]$ along the z axis, where d_h denotes the minimum safety distance the robots should always have from the terrain and $z_{\max}$ denotes the maximum allowable operational height for the robots;
- $\|x_i - x_j\| \geqslant d_r$, $\forall i,j \in \{1, \ldots, N\}$ and $i \neq j$, i.e., the safety distance between two robots is d_r.

5.2. Simulation results

The centralized CAO-based approach that has been proposed for the problem in hand (Renzaglia et al., 2012) is utilized for comparison purposes. The proposed approach was parametrized with a time window $T=40$ for the least-squares estimation, with $M=100$ random perturbations, the corresponding approximator was a third-order monomial estimator with $L=18$, and the number of monomials per order (14) were $L_1=2$, $L_2=5$, and $L_3=10$. Acknowledging the fact that the CAO algorithm performs optimization in a higher-dimensional space (centralized optimization scheme), a different set of parameters was chosen. Evaluating the CAO version for different numbers of random perturbations, we found that after $M=900$ the number of random perturbations does not affect its performance. Furthermore, to cope with the higher-dimensional state space, the time window was set to $T=60$ and the approximator was chosen to be a third-order monomial estimator with $L_1=3$, $L_2=12$, and $L_3=40$ (with overall size of $L=56$). In both algorithms, we utilize $\alpha=0.1$ to update the robot's positions. For the rest of this section, we use these values in all the presented experiments.

To perform simulations in a realistic environment, we utilized the morphology of an area located in Zürich, Switzerland (Figure 6(a)). This map was generated using a state-of-the-art visual-SLAM algorithm (Doitsidis et al., 2012), which tracks the pose of the camera while, simultaneously and autonomously, building an incremental map of the surrounding environment. The terrain's dimension is [0,162] m and [0, 84] m for x and y axes, respectively, while the height of the terrain is between $[0, 7.2]$ m and the maximum operational height was set to 25 m. Following the authors instructions (Renzaglia et al., 2012), K weight (21) was chosen to be 30, whereas both the safety distance from the terrain and the minimum allowable distance between two robots were set to be $d_h = d_r = 0.5$ m. Finally, the duration of each experiment was set to $k_{\max}=600$ timestamps.

Figure 6 depicts such a simulation instance with six robots. The initial positions of the robots, as sketched in Figure 6(b), were selected to be "crowded" inside a subarea of the terrain. Figures 6(c) and 6(d) illustrate the final robots' configuration, as calculated by the CAO-based algorithm in 3D and 2D representation, respectively. The corresponding final robots' assignment as calculated by the proposed approach is presented in Figures 6(e) and 6(f). In both cases, the 3D representation reports which subarea of the terrain is covered by each robot, whereas the 2D representation reveals the exact positions of the robots in the x–y plane and the distance between them. Figure 6(g) depicts the evolution of the cost function (21) for both the evaluated algorithms. Apart from the difference in the convergent state, the proposed approach is able to find this solution from its early steps (<50). The centralized CAO needs more iterations to learn the dynamics of the robots and the unknown terrain, because it performs its optimization scheme in the higher-dimensional space of $\mathbb{R}^{3N}$ ($\mathbb{R}^{18}$ for 6 robots). In contrast, the proposed algorithm separately, although cooperatively, solves N (=6 robots for this instance) optimization problems of the size of $\mathbb{R}^3$.

In the specific problem set-up, the speed of convergence requires extra attention, as a slow convergence rate may lead to instability or loss of convergence at all. More specifically, if a navigation algorithm does not converge fast enough to the optimal configuration, one or more robots may have reached high-altitude positions, from which they cannot acquire useful measurements (out of their sensor capabilities (20)). This is a non-recoverable situation, as the robots do not have any "feedback" from the terrain to properly evaluate their actions.

5.2.1. Scalability analysis. To validate both the efficiency and the effectiveness of the proposed algorithm in the case of larger robot teams, we performed experiments with 5, 10, 15, and 20 robots.[8] For each different size of robotic team, we created 20 experiment instances with randomly chosen initial robots' positions. The aforementioned simulation instances are evaluated on both the proposed approach and the centralized CAO-based approach.

The results of these simulations are summarized in Figure 7. Figure 7(a) displays the average value of the resulting cost function $J(k_{\max})$, along with the corresponding confidence interval, over the different number of robots. In addition, Figure 7(b) displays a statistical analysis on the summation of cost function $\sum_0^{k_{\max}} J(k)$, to investigate the convergence rate of each pair (scenario–algorithm).

Overall, the proposed approach achieves an average improvement of 23% on the *final achieved cost function value*, with 55.33% improvement on the deviation around that average value. Moreover, the *summation of cost function* has been improved by 23.84% with a corresponding improvement on the deviation of 65.06%, against the centralized CAO-based approach. The proposed approach achieves these performance enhancements mainly due to the two following reasons.

(i) The proposed algorithm has a better perspective on the change of the overall cost function by evaluating the appropriate combinations of historical measurements on that cost function (*Steps 2–3* of the proposed approach).
(ii) The fast convergence of the proposed approach eliminates the chances for a robot to be found out of its sensors capabilities. Therefore, the proposed approach is able to converge on approximately the same robots' configuration (per different team size), independently on the robots' initial positions. The latter is depicted in the substantial improvements on the corresponding confidence intervals.

5.2.2. Fault-tolerant characteristics. In this scenario, we investigate the performance of the proposed algorithm in the case of catastrophic events or hardware failures. More precisely, five robots were initially deployed to perform the aforementioned coverage task, whereas the duration of the experiment was increased to $k_{\max} = 1,000$ timestamps. It is assumed that, at timestamp 330, one robot did not correspond to our control commands and the measurements' flow had been interrupted. Under these new circumstances, the surveillance task has to be undertaken by remaining, properly working robots. After the completion of two-thirds of the available timestamps, we assume that another robot had an equipment malfunction and cannot continue its covering task. Thus, the number of available robots, which are called to cover the area of interest for the ~ 300 remaining timestamps, has dropped to three.

Figures 8(a)–8(f) illustrate the evolution of the robots positions during the course of the previously described scenario, utilizing the proposed approach. After both the robots' malfunctions, the algorithm redesigns the remaining robot positions to achieve the best possible coverage. Overall, Figure 8(g) demonstrates the evolution of the objective function for the proposed approach in comparison with the centralized CAO-based approach (Renzaglia et al., 2012).

It must be emphasized that the proposed algorithm does not need any separately designed, fault-detection mechanism (e.g., failure in establishing communication, operator to detect the malfunction, etc.), as it is able to implicitly derive this kind of information from the changes in the cost function J with respect to the commanded positions. The above feature is of paramount importance in real-life multi-robot applications, because it removes the tedious, and in many applications impossible, task to predict (or identify online) all the possible malfunctions, as well as to design the appropriate course of actions.

5.2.3 Target monitoring. We close this section by investigating the algorithm's capability to process objectives that can be alternated/activated on the fly, without stopping and restarting the mission. To achieve this, simultaneously with the coverage task, we introduce the task of monitoring a target. For the sake of this simulation set-up, it is assumed that, in addition to the sensors which are responsible for

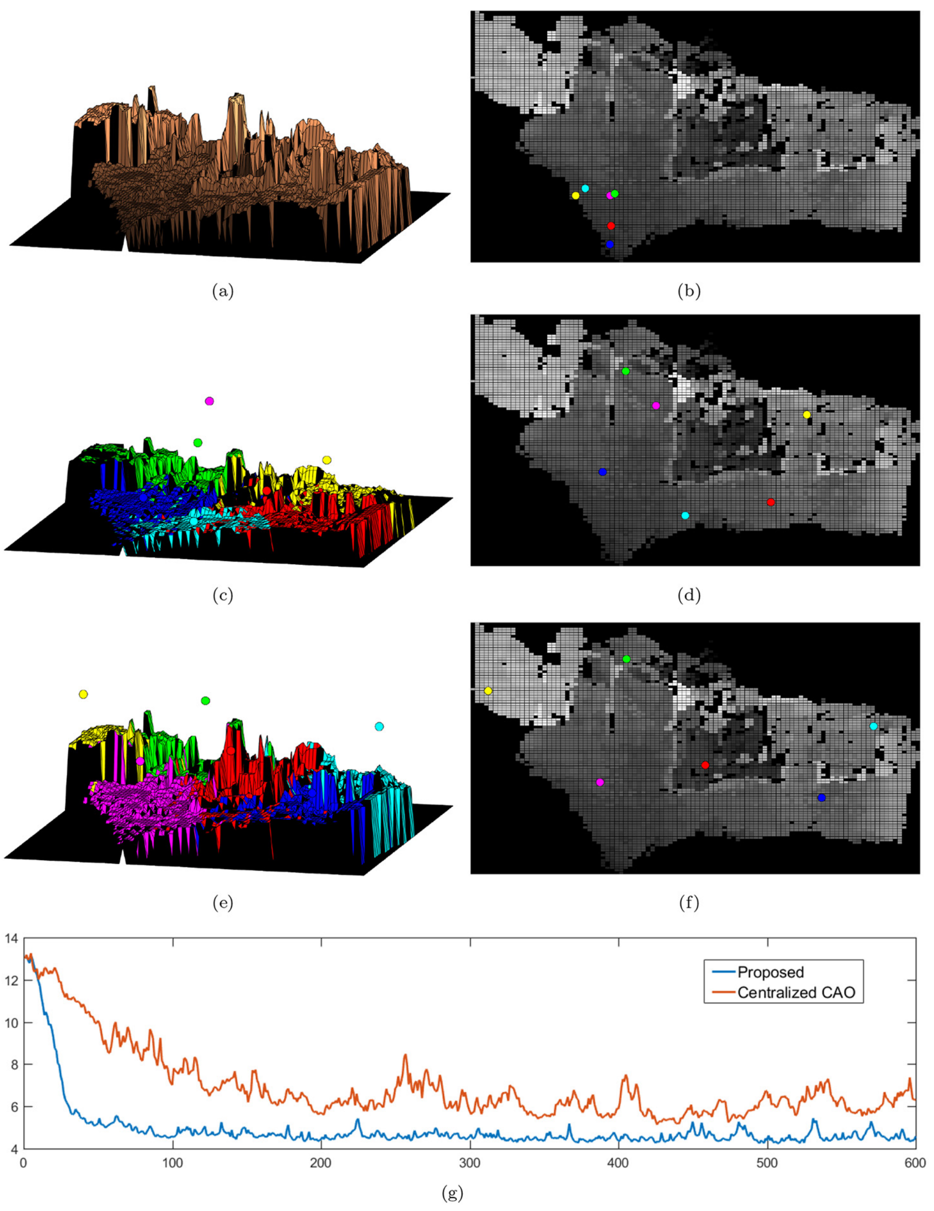

Fig. 6. Indicative example: surveillance of unknown terrain by a team of robots: (a) 3D representation of the surface to be covered; (b) initial positions of the available robots; (c) 3D view, CAO-based approach; (d) top view, CAO-based approach; (e) 3D view, proposed approach; (f) top view, proposed approach; (g) cost function evolution. The proposed algorithm and the CAO-based approach (Renzaglia et al., 2012) are evaluated on the same set-up (environment, robots initial positions, robots sensor capabilities).

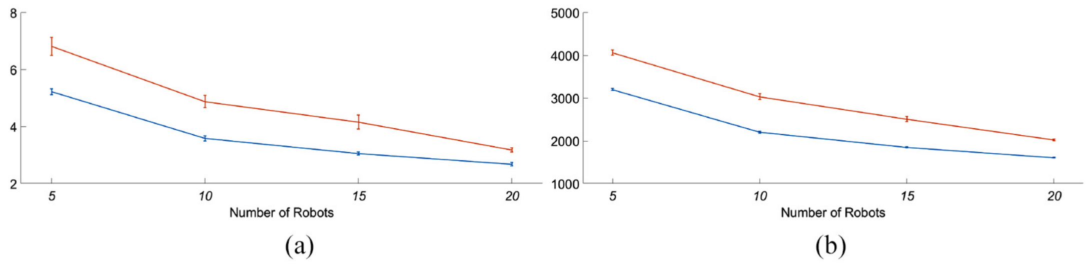

Fig. 7. Comparison study over different numbers of robots: proposed algorithm (blue) and CAO-based approach (Renzaglia et al., 2012) (red). (a) Final achieved value of the cost function. (b) Summation of the cost function over the experiment's horizon.

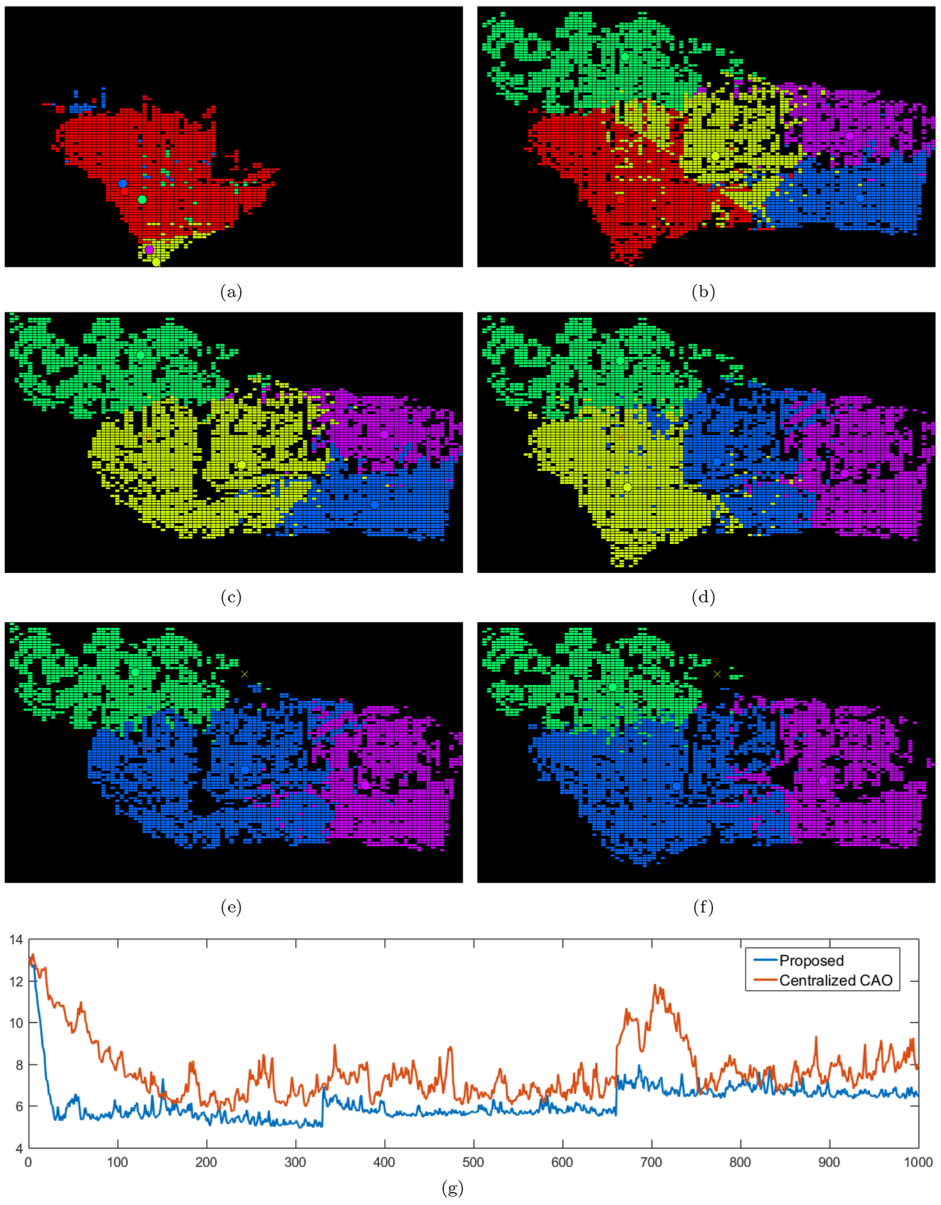

Fig. 8. Malfunction scenario: five robots were initially deployed for the surveillance task. At two distinct timestamps, the swarm of robots loses one of its member due to a simulated malfunction. The surveillance task have to be continued with the remaining team resources. (a) Initial positions of the five available robots. (b) Coverage task with all five available robots. (c) One timestamp after the malfunction on the red robot. (d) Coverage task with four robots. (e) One timestamp after the malfunction on the yellow robot. (f) Again, the algorithm redesigns the robots positions to cover the area in the best possible way utilizing the available resources. (g) Cost function evolution.

the coverage task (20), the robots are equipped with exteroceptive sensors (e.g., cameras, sonars, etc.) which are able to estimate the targets' positions, according to the following measurement model:

$$y_{x_i - \chi_j} = \begin{cases} \|x_i - \chi_j\| + h_\xi(x_i, \chi_j)\xi & \text{if } \chi_j \textit{ has been detected} \\ \text{undefined} & \text{otherwise} \end{cases} \tag{22}$$

where χ_j denotes the jth target's position in 3D space, $h_\xi(x_i, x_t)$ and ξ, similar to Equation (20), denote the multiplicative sensor noise term and the standard Gaussian noise, respectively. Therefore, an extra term has to be added to the cost function (21) to appropriately evaluate the progress of targets' monitoring, as follows:

$$J(\mathbf{y}(k)) = \int_{q \in \mathcal{V}} \min_{i=1,\dots,N} y_{x_i - q}\, dq + K \int_{q \in \mathcal{U} \setminus \mathcal{V}} dq + K_t \sum_{j=1}^{n_t} \min_{i=1,\dots,N} y_{x_i - \chi_j} \tag{23}$$

where K_t serves as a weight to give more or less priority to the monitoring task in comparison with the coverage. In addition, n_t denotes the number of targets to be monitored.

The experiments were performed in the same terrain, under the previously defined set-up parameters. Figure 9 illustrates four key snapshots, which demonstrate the functionality of the proposed algorithm. Figure 9(a) depicts the robots' initial positions along with the corresponding coverage on the terrain. After 367 timestamps (Figure 9(b)), the algorithm has converged to the (locally) optimal robots' configuration for the coverage-only problem. At $k = 370$ timestamp, it is assumed that a target, which requires closer examination, appears inside the operation area. The proposed algorithm, after the time needed to learn the changed problem dynamics (activation of the third term in (23)), starts to adapt the robots' positions to minimize the updated cost function (23). More precisely, as illustrated in Figure 9(c), the purple robot (which was, at the time, closer to the target) starts to gain height to minimize its distance from the detected target. However, such an action leads to poor coverage on the subarea underneath that robot. To alleviate the above undesirable situation, the proposed algorithm redesigns the remaining robots' positions so as to achieve the best coverage of the terrain with the available resources. The final robots' positions with the corresponding coverage of the terrain is sketched in Figure 9(d). The evolution of the objective function for the proposed approach in comparison with the centralized CAO-based approach is demonstrated in Figure 10. Conclusively, for this simulation scenario, the proposed algorithm:

- chooses to assign a robot to be as close as possible to the target without any explicit command;
- adapts the other robots positions so as to "fill the hole" in the coverage task; and
- achieves almost the same level of terrain coverage with the centralized CAO-based approach for five robots (Figure (10), dashed line), whereas one (out of five) robots is occupied with another task.

6. Persistent coverage inside unknown environment

In the final application, we focus on the problem of persistent coverage in an area of interest with a team of robots. In this application, it is assumed that the operational robots are equipped with the appropriate sensors that are able to cover a portion of the environment. The objective in a persistent coverage application is to continuously cover an area of interest, assuming that the coverage level follows a time-decaying function. The problem along with a specifically designed algorithm has been proposed in Palacios-Gasós et al. (2016). The authors also established a well-defined, heuristic mechanism to online share the coverage evolution between the robots in a distributed way.

Although the results are remarkable, the proposed decision-making mechanism in Palacios-Gasós et al. (2016) utilizes a model that accurately predicts the improvement in the coverage level with respect to the robots movement (Palacios-Gasós et al., 2016: Equations (10), (18)–(21)). In real-world applications, the above assumption does not always hold, as the increase in coverage level (i) is usually corrupted by nonlinear noise, (ii) can be affected by environmental specific characteristics, such as local morphology, obstacles, other robots' positions, etc., (iii) may follow a time-varying model (e.g., coverage level deteriorates over time). To circumvent these difficulties, we propose a variation of the above problem, where the changes in the level of coverage cannot be accurately predicted before the action. The actual information about the exact covered area is only available after the execution of each corresponding action through the robot's measurements. The above formulation is not only more *realistic*, as it does not require an exact model of the environment or robot's coverage capabilities, but also more *generic*, as it does not need to redesign the approach when robots with different or unknown coverage models are deployed.

6.1. Problem definition

It is assumed that the operational area is a bounded $Q \subset \mathbb{R}^2$, which a team of robots has to persistently cover. The decision variables (1) represent the collective vector of all the robots' positions, i.e., $\mathbf{x} = \left[x_1^T, \ldots, x_N^T\right]^T$, where $x_i \in Q$.

Inside the environment there are several positions $q \in O \subset Q$ that cannot be traversed by the robots and additionally the presence of these obstacles affects each robot's coverage distribution. Although the exact positions of the obstacles are generally unknown, we assume that the robots are able to sense their presence when they are in close proximity. The above assumption is in line with the most commercial robots which are also equipped with proximity sensors to avoid collisions (e.g., Nieuwenhuisen et al., 2014). Thus, each robot's new candidate position x_i^{cand} should verify the following constraint (see Equation (6) of the general problem formulation):

$$\min_{q \in O}\left(\left\|x_i^{\text{cand}} - q\right\|\right) \geqslant b \tag{24}$$

where b denotes the safety distance. At each timestamp k, the overall coverage increase is given by $y(q,k) = \sum_{i \in \{1,\ldots,N\}} y_i(q,k),\ \forall q \in Q$, where

$$y_i(q,k) = \begin{cases} \gamma_i(q, x_i) & \text{if } \|x_i - q\| \leqslant r_i^{\text{cov}} \text{ and there is line of sight between } x_i \text{ and } q \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

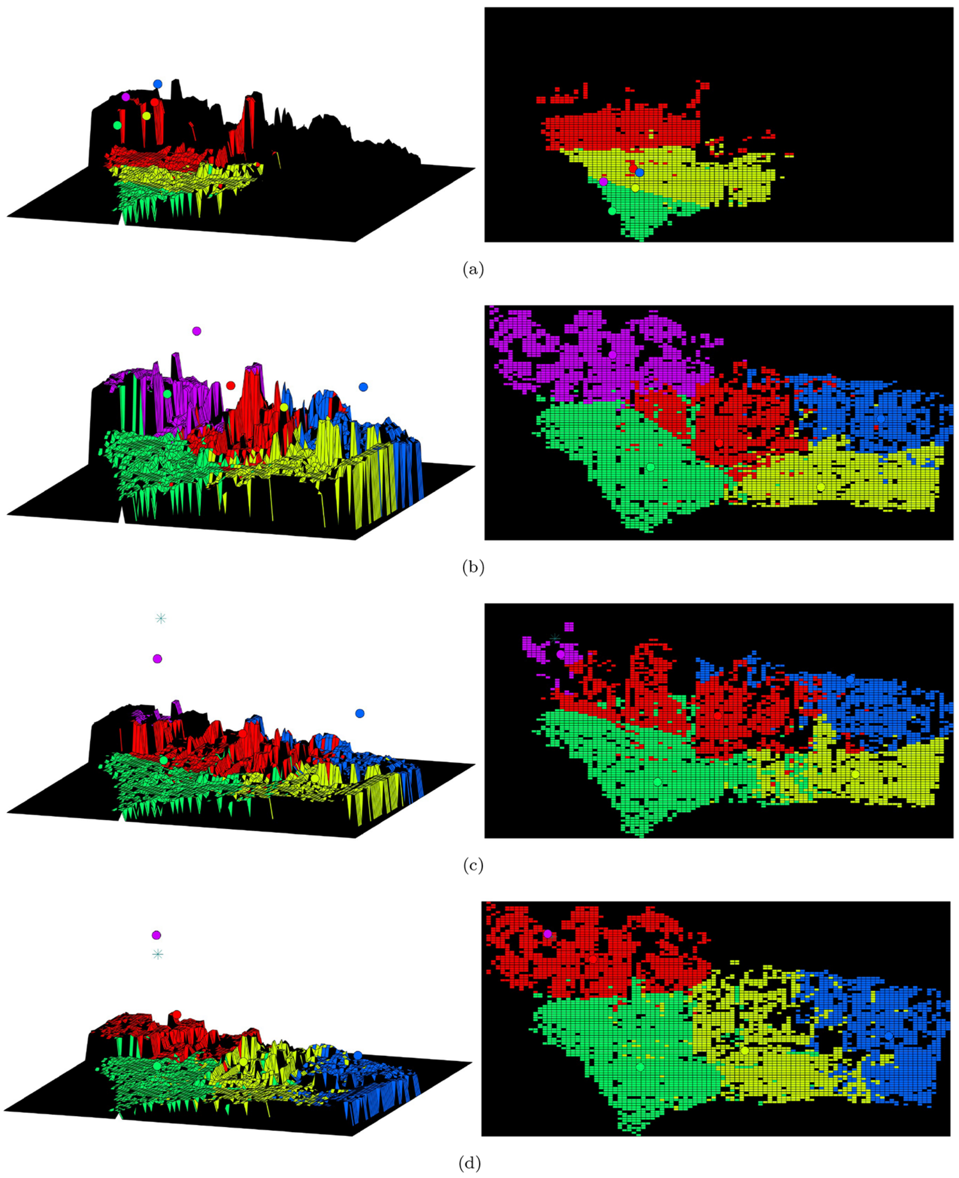

Fig. 9. Target monitoring scenario. The robots have been deployed with an extra objective (apart from the surveillance task) to get as close as possible to a target. The target appears inside the operation area of the robots in the middle of the mission. (a) Timestamp 1: initial positions of the five available robots. (b) Timestamp 367: coverage task with all five available robots. (c) Timestamp 427: the purple robot starts to gain height to minimize the distance from the target. As a consequence, it cannot cover adequately its underneath surface. (d) Timestamp 1,000: finally, the algorithm redesigns the robots positions so as to cover the area in the best possible way utilizing the available resources.

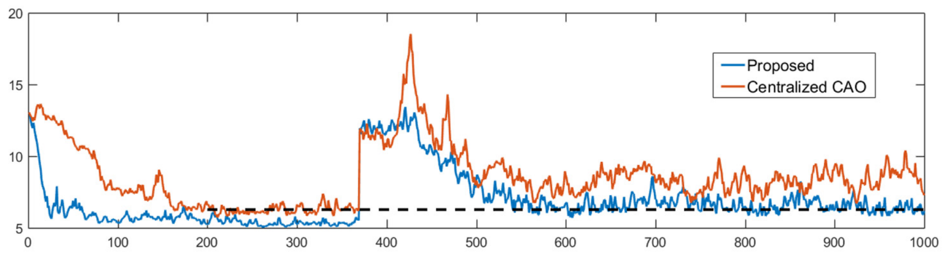

Fig. 10. Cost function evolution in target monitoring scenario.

and $\gamma_i(q, x_i)$ denotes a nonlinear function that models how the coverage level evolves in the area around the i th robot's position. Note, that coverage distribution model $\gamma_i(q, x_i)$ may be different for each robot as it expresses the functionality of its on-board sensors.

The coverage of the operational area can be modeled by a time-varying field and, in general, admits the following form:

$$Z(q,k) = d(q)Z(q,k-1) + y(q,k), \quad \forall q \in Q \tag{26}$$

In other words, the coverage level decreases to a constant decay gain $d(q)$, with $0 < d(q) < 1$, and increases according to the $y(q,k)$. The objective of the multi-robot team is to maintain a desired coverage level, $Z^*(q) > 0$, $\forall q \in Q$.

Having the above formulation in mind, we define the *quadratic coverage error* the robot team has to minimize

$$J(k) = \int_Q (Z^*(q) - Z(q,k))^2 \, dq \tag{27}$$

6.2. Simulation results

All simulations were performed in a rectangle environment consisting of 100×150 units, with uniformly distributed decay rate $d(q) = 0.995$, $\forall q \in Q$. The desired coverage level is $Z^*(q) = 100$, $\forall q \in Q$. The number of robots was $N = 6$, whereas their maximum motion is $u^{\max} = 5$. The coverage increase in *open space* (obstacle-free), caused by the robots' movements, can be simulated by

$$\gamma_i(q,k) = \frac{P}{r_i^{\text{cov}2}} \left(\|x_i - q\| - r_i^{\text{cov}} \right)^2 \tag{28}$$

The maximum value is set to $P = 17$ and the coverage radius is set to $r_i^{\text{cov}} = 10$ units. Please note that this equation is not utilized during the decision-making process, but it is only employed to simulate the increase in the area coverage, with respect to the robot's movement. Finally, the experiments' duration is set to $k_{\max} = 900$ timestamps.

To adapt the parameters of the proposed algorithm to the current application, we have to take into consideration that the navigation algorithm has to rapidly change its behavior owing to the time-varying nature of the cost function. Therefore, the time window for the least-squares estimation was only $T = 5$ timestamps and the number of perturbations was $M = 100$ candidates. To solve the underlying least-squares optimization problem (11) with such a reduced historical values, we utilize only a second-order monomial estimator with $L_1 = 2$ and $L_2 = 2$ (with overall size of $L = 5$). Finally, following also the problem definition in Palacios-Gasós et al. (2016: Section II.), we utilize $\alpha = 1$ to update the robot's positions.

6.2.1. Obstacle-free environment. In the first simulation scenario, we deploy the team of robots in an obstacle-free environment. An indicative simulation run of this scenario is summarized Figure 11. Figures 11(a)–11(c) present the evolution of the coverage across the environment Q, for three different timestamps. In addition, Figures 11(d), 11(e), and 11(f) depict the evolution of the *average coverage level*, the corresponding *standard deviation*, and the *quadratic coverage error* for the course of the experiment, respectively. After the experiment execution, the average coverage level in all the operational environment Q was 97 with a standard deviation of 21.2 and the corresponding quadratic coverage error was 6.9×10^6.

It should be highlighted that, the objective (27) is a time-varying function with high rate of change, i.e., the evaluation of (27) may result in significantly different scores for the same robots positions, even for very close timestamps. However, the proposed scheme is able to appropriately tackle the above problem, by constantly learning these cost function variations with respect to the robots' positions.

Although the proposed algorithm presents an equivalent performance compared with the dedicated one (Palacios-Gasós et al., 2016: Section VI), if the problem is defined as in this scenario and the coverage evolution with respect to the robots movement being accurately predicted, a dedicated approach should be preferred to avoid the extra time due to learning (Equations (10) and (11) of the proposed algorithm). However, the proposed approach has several advantages when it is deployed in a real-world environment, where the evaluation of the coverage increase cannot be performed beforehand. Such a scenario is presented in the following paragraph.

6.2.2. Unknown cluttered environment. In the final simulation scenario, we investigate the performance of the proposed approach for the persistent coverage task, when it is evaluated on an unknown environment with non-convex obstacles. The obstacles have been created randomly and do not hold any kind of pattern. The minimum distance between the obstacles and any robot (24) has been set to $b = 2.5$.

Again, an illustrative example is presented in Figure 12. Following the same presentation policy, Figures 12(a), 12(b), and 12(c) illustrate the evolution of the of the coverage across the environment Q, for three different timestamps. Figures 12(d), 12(e), and 12(f) depict the evolution of the *average coverage level*, the corresponding *standard deviation*, and the *quadratic coverage error* (27), respectively.

The cost function (27) does not need any adaptation to this scenario as the coverage values $Z(q)$ that correspond to obstructed locations $q \in O$ will remain zero, independently of their distance from any robot. In other words, the calculation of (27) does not need the information of the unknown obstacles, as the robots would never send coverage updates (25) about the obstacles' positions. However, to construct comparable metrics with the previous scenario, we exclude the values that correspond to obstacles' locations from the calculation of the average coverage level (Figure 12(d)).

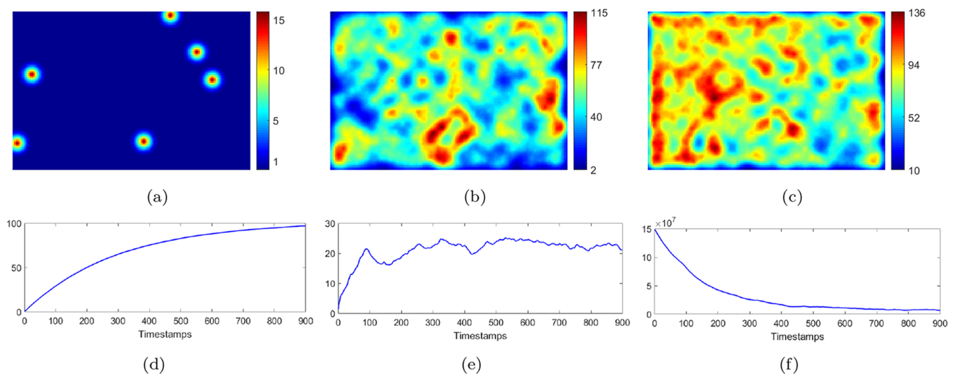

Fig. 11. Obstacle-free scenario: the coverage level for three different timestamps ((a) timestamp 1, (b) timestamp 200, and (c) timestamp 400) and the corresponding performance indices ((d) average coverage level, (e) standard deviation of coverage level, and (f) cost function, quadratic coverage error).

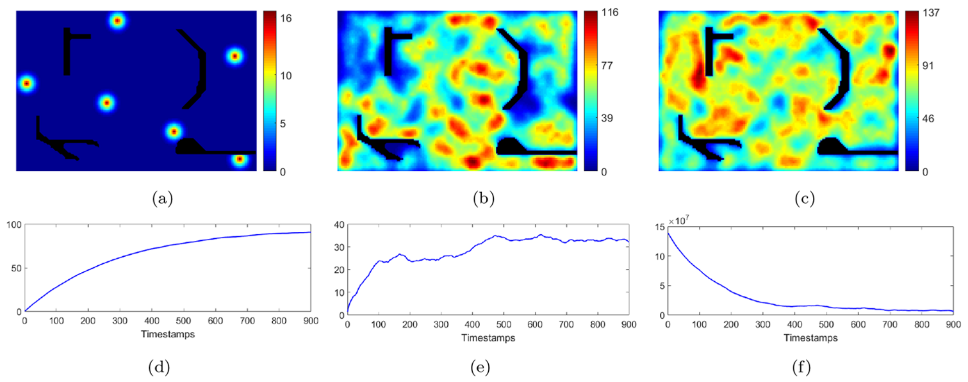

Fig. 12. Scenario in an unknown environment with non-convex obstacles: the coverage level for three different timestamps ((a) timestamp 1, (b) timestamp 200, and (c) timestamp 400) and the corresponding performance indices ((d) average coverage level, (e) standard deviation of coverage level, and (f) cost function, quadratic coverage error).

After the experiment execution, the average coverage level inside Q was 90.7 with a standard deviation of 32.1 and the corresponding quadratic coverage error was 6.6×10^6.

Comparing the outcomes of two scenarios side by side, we can draw the following observations.

- In the cluttered environment scenario, the robots can more easily get "trapped" in overcovered areas, resulting in a higher standard deviation. In other words, when a robot detects (implicitly from the changes in its corresponding cost function) that its position deteriorates the coverage level, may have only a small subset of possible new positions.
- During the course of the experiment in the cluttered environment, the obstacles "blocked" a portion of the robots' coverage capabilities. Therefore, for the cluttered environment scenario, the robots achieved a smaller average coverage level (excluding the obstacles positions).

7. Conclusions

A distributed methodology for dealing with multi-robot problems, where the mission objectives can be translated into an optimization of a cost function, has been proposed. In contrast to the majority of the multi-robot approaches, where the objectives are accomplished in a cost function optimization scheme, the proposed approach has been designed for multi-robot problems where the a priori calculation of the cost function is not feasible. In a nutshell, the proposed approach has the following key advantages:

- it does not require any knowledge of the dynamics of the overall system;
- it can incorporate any kind of operational constraint or physical limitation;
- it shares the same convergence characteristics as those of BCD algorithms;
- it has fault-tolerant characteristics;
- it can appropriately tackle time-varying cost functions;
- and it can be realized in embedded systems with limited power resources.

Conclusively, we expect that many interesting tasks in mobile robotics can be approached by the proposed scheme. This is basically due to the fact that the proposed approach, instead of explicitly solving a particular problem, which requires prior knowledge of the system dynamics, learns, from the real-time measurements, exactly the features of the system which affects the user-defined objectives. Furthermore, the proposed approach can be appealing in many real-life application owing to its fault-tolerant characteristics, without an explicitly designed fault-detection mechanism. All the above issues are considered of paramount importance in the emerging field of multi-robot applications.

As future directions, we are interested in performing an extensive set of experiments, ideally with a large number of robots (e.g., a large swarm of femtosatellites (100 g class spacecraft) (Hadaegh et al., 2016)). In particular, in such a set-up, it is impossible to explicitly program each and every robot to perform a subtask, therefore the goal will be to achieve an abstract set of objectives, which are defined in the form of cost function optimization. The idea behind the above formulation is, by excluding the intermediate steps from the design process, we enrich the multi-robot decision-making scheme with autonomy, regarding the “type” of converged solutions.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement number 740593 (ROBORDER)).

Notes

1. The rate of change in the objective function should be smaller than the learning capabilities of the algorithm (see Section 2).
2. Without loss of generality, in the rest of the paper we assume a minimization problem.
3. Note that although it is natural to assume that the noise sequence ξ_k is a stochastic zero-mean signal, it is not realistic to assume that it satisfies the typical AWGN property, even if the robots sensors do; as $\mathcal{J}$ is a nonlinear function of the robots decision variables and, thus, of the robots sensor measurements (3), the AWGN property is typically lost.
4. In the general case, $\sum_{i=1}^{N} J_i(k) \neq j_k$ and $\prod_{i=1}^{N} J_i(k) \neq j_k$, $\forall k$.
5. See Kosmatopoulos (2009) for more details about the sufficiency of this condition.
6. The distributed nature of the algorithm may impose a stricter set of constraints, in comparison with cases where a centralized control is applied.
7. Moreover, recent studies imply that BCD methodologies can achieve global convergence even in cases where the global cost function (4) is non-convex but holds some properties. For example, Xu and Yin (2013) established global convergence of the BCD algorithm in the general case where the global cost function $\mathcal{J}$ and each robot’s contribution J_i are non-convex functions, but the so-called Kurdyka-Łojasiewicz (KL) property is satisfied.
8. Note that for the current experiment set-up with the previously defined sensor’s capabilities, the utilization of more than 15 robots cannot significantly affect the coverage task.

ORCID iD

Athanasios Ch Kapoutsis https://orcid.org/0000-0002-1688-036X

References

Amanatiadis A, Chatzichristofis S, Charalampous K, et al. (2013) A multi-objective exploration strategy for mobile robots under operational constraints. *IEEE Access* 99: 691–702. 20 September 2013. IEEE.

Bertsekas DP (1999) *Nonlinear Programming*. Belmont, MA: Athena Scientific.

Bourgault F, Makarenko AA, Williams SB, Grocholsky B and Durrant-Whyte HF (2002) Information based adaptive robotic exploration. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002, vol. 1*. IEEE, pp. 540–545.

Burgard W, Moors M, Stachniss C and Schneider FE (2005) Coordinated multi-robot exploration. *IEEE Transactions on robotics* 21(3): 376–386.

Capitan J, Spaan MT, Merino L and Ollero A (2013) Decentralized multi-robot cooperation with auctioned POMDPs. *The International Journal of Robotics Research* 32(6): 650–671.

Chen J, Gauci M, Li W, Kolling A and Groß R (2015) Occlusion-based cooperative transport with a swarm of miniature mobile robots. *IEEE Transactions on Robotics* 31(2): 307–321.

Choset H (2001) Coverage for robotics–a survey of recent results. *Annals of Mathematics and Artificial Intelligence* 31(1–4): 113–126.

Cortes J, Martinez S, Karatas T and Bullo F (2002) Coverage control for mobile sensing networks. In: *Proceedings IEEE International Conference on Robotics and Automation, 2002 (ICRA’02)*, Vol. 2. IEEE, pp. 1327–1332.

Cui R, Li Y and Yan W (2016) Mutual information-based multi-AUV path planning for scalar field sampling using multidimensional RRT. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46(7): 993–1004.

De La Cruz C and Carelli R (2008) Dynamic model based formation control and obstacle avoidance of multi-robot systems. *Robotica* 26(03): 345–356.

Demmel JW (1997) *Applied Numerical Linear Algebra*. Philadelphia, PA: SIAM.

Doitsidis L, Weiss S, Renzaglia A, et al. (2012) Optimal surveillance coverage for teams of micro aerial vehicles in GPS-denied environments using onboard vision. *Autonomous Robots* 33(1–2): 173–188.

Golub GH and Van Loan CF (2012) *Matrix computations*, Vol. 3. Baltimore, MD: JHU Press.

Gomes J, Urbano P and Christensen AL (2013) Evolution of swarm robotics systems with novelty search. *Swarm Intelligence* 7(2–3): 115–144.

Granville V, Křivánek M and Rasson JP (1994) Simulated annealing: A proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(6): 652–656.

Hadaegh FY, Chung SJ and Manohara HM (2016) On development of 100-gram-class spacecraft for swarm applications. *IEEE Systems Journal* 10(2): 673–684.

Kapoutsis A, Chatzichristofis S, Doitsidis L, Borges de Sousa J and Kosmatopoulos EB (2013) Autonomous navigation of teams of unmanned aerial or underwater vehicles for exploration of unknown static and dynamic environments. In: *2013*

21st Mediterranean Conference on Control and Automation (MED). IEEE, pp. 1181–1188.

Kapoutsis AC, Chatzichristofis SA, Doitsidis L, et al. (2015a) Real-time adaptive multi-robot exploration with application to underwater map construction. *Autonomous Robots* 40: 987–1015.

Kapoutsis AC, Salavasidis G, Chatzichristofis S, et al. (2015b) The Noptilus Project overview: A fully-autonomous navigation system of teams of AUVs for static/dynamic underwater map construction. *IFAC-PapersOnLine* 48(2): 231–237.

Kohl N and Stone P (2004) Policy gradient reinforcement learning for fast quadrupedal locomotion. In: *Proceedings 2004 IEEE International Conference on Robotics and Automation, 2004 (ICRA'04)*, Vol. 3. IEEE, pp. 2619–2624.

Kollar T and Roy N (2008a) Efficient optimization of information-theoretic exploration in SLAM. In: *AAAI*, Vol. 8, pp. 1369–1375.

Kollar T and Roy N (2008b) Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research* 27(2): 175–196.

Korkas CD, Baldi S, Michailidis I and Kosmatopoulos EB (2016) Occupancy-based demand response and thermal comfort optimization in microgrids with renewable energy sources and energy storage. *Applied Energy* 163: 93–104.

Kosmatopoulos EB (2009) An adaptive optimization scheme with satisfactory transient performance. *Automatica* 45(3): 716–723.

Kosmatopoulos EB and Kouvelas A (2009) Large scale nonlinear control system fine-tuning through learning. *IEEE Transactions on Neural Networks* 20(6): 1009–1023.

Kouvelas A, Aboudolas K, Kosmatopoulos EB and Papageorgiou M (2011) Adaptive performance optimization for large-scale traffic control systems. *IEEE Transactions on Intelligent Transportation Systems* 12(4): 1434–1445.

Le Ny J and Pappas GJ (2009) On trajectory optimization for active sensing in gaussian process models. In: *Proceedings of the 48th IEEE Conference on Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference (CDC/CCC 2009)*. IEEE, pp. 6286–6292.

Matignon L, Jeanpierre L and Mouaddib AI (2012) Coordinated multi-robot exploration under communication constraints using decentralized Markov decision processes. In: *AAAI 2012*, pp. 2017–2023.

Morgan D, Subramanian GP, Chung SJ and Hadaegh FY (2016) Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *The International Journal of Robotics Research* 35(10): 1261–1285.

Nesterov Y (2007) Gradient methods for minimizing composite objective function. Technical Report, UCL.

Nieuwenhuisen M, Droeschel D, Beul M and Behnke S (2014) Obstacle detection and navigation planning for autonomous micro aerial vehicles. In: *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, pp. 1040–1047.

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

Polycarpou MM and Ioannou PA (1991) *Identification and control of nonlinear systems using neural network models: Design and stability analysis*. University of Southern Calif.

Rathnam RK and Birk A (2013) A distributed algorithm for cooperative 3d exploration under communication constraints. *Paladyn, Journal of Behavioral Robotics* 4(4): 223–232.

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.

Rooker MN and Birk A (2007) Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice* 15(4): 435–445.

Roy N and Thrun S (1999) Coastal navigation with mobile robots. In: *NIPS*, Vol. 13, pp. 1043–1049.

Salavasidis G, Harris C, McPhail S, Phillips AB and Rogers E (2016) Terrain aided navigation for long range AUV operations at arctic latitudes. In: *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*. IEEE, pp. 115–123.

Scaramuzza D, Achtelik MC, Doitsidis L, et al. (2014) Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in GPS-denied environments. *IEEE Robotics and Automation Magazine* 21(3): 26–40.

Schwager M, McLurkin J and Rus D (2006) Distributed coverage control with sensory feedback for networked robots. In: *Proceedings of Robotics: Science and Systems*, Philadelphia, PA. DOI:10.15607/RSS.2006.II.007.

Schwager M, Rus D and Slotine JJ (2009) Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research* 28(3): 357–375.

Seyboth GS, Dimarogonas DV, Johansson KH, Frasca P and Allgöwer F (2015) On robust synchronization of heterogeneous linear multi-agent systems with static couplings. *Automatica* 53: 392–399.

Singh A, Krause A, Guestrin C and Kaiser WJ (2009) Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research* 34: 707–755.

Spaan MT and Vlassis N (2005) Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research* 24: 195–220.

Stachniss C and Burgard W (2003) Exploring unknown environments with mobile robots using coverage maps. In: *IJCAI*, pp. 1127–1134.

Teixeira F (2007) Terrain-aided navigation and geophysical navigation of autonomous underwater vehicles. *Instituto Superior Técnico, Lisboa*.

Wang Z and Schwager M (2016) Multi-robot manipulation without communication. In: *Distributed Autonomous Robotic Systems*. New York: Springer, pp. 135–149.

Wright SJ (2015) Coordinate descent algorithms. *Mathematical Programming* 151(1): 3–34.

Xu Y and Yin W (2013) A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences* 6(3): 1758–1789.

Zheng X, Jain S, Koenig S and Kempe D (2005) Multi-robot forest coverage. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005 (IROS 2005)*. IEEE, pp. 3852–3857.

Zhou K and Roumeliotis SI (2011) Multirobot active target tracking with combinations of relative observations. *IEEE Transactions on Robotics* 27(4): 678–695.