Auton Robot
DOI 10.1007/s10514-017-9652-y

Strategies for coordinated multirobot exploration with recurrent connectivity constraints

Jacopo Banfi[1] · **Alberto Quattrini Li**[2] · **Ioannis Rekleitis**[2] · **Francesco Amigoni**[1] · **Nicola Basilico**[3]

Received: 11 December 2016 / Accepted: 23 June 2017

Abstract During several applications, such as search and rescue, robots must discover new information about the environment and, at the same time, share operational knowledge with a base station through an *ad hoc* network. In this paper, we design exploration strategies that allow robots to coordinate with teammates to form such a network in order to satisfy *recurrent connectivity* constraints—that is, data must be shared with the base station when making new observations at the assigned locations. Current approaches lack in flexibility due to the assumptions made about the communication model. Furthermore, they are sometimes inefficient because of the synchronous way they work: new plans are issued only once all robots have reached their goals. This paper introduces two novel *asynchronous strategies* that work with arbitrary communication models. In this paper, 'asynchronous' means that it is possible to issue new plans to subgroups of robots, when they are ready to receive them. First, we propose a single-stage strategy based on Integer Linear Programming for selecting and assigning robots to locations. Second, we design a two-stage strategy to improve computational efficiency, by separating the problem of locations' selection from that of robot-location assignments. Extensive testing both in simulation and with real robots show that the proposed strategies provide good situation awareness at the base station while efficiently exploring the environment.

Keywords Multirobot systems · Exploration · Communication constraints · Recurrent connectivity

This is one of several papers published in Autonomous Robots comprising the Special Issue on Online Decision Making in Multi-Robot Coordination.

✉ Nicola Basilico
nicola.basilico@unimi.it

Jacopo Banfi
jacopo.banfi@polimi.it

Alberto Quattrini Li
albertoq@cse.sc.edu

Ioannis Rekleitis
yiannisr@cse.sc.edu

Francesco Amigoni
francesco.amigoni@polimi.it

[1] Artificial Intelligence and Robotics Laboratory, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy

[2] Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA

[3] Department of Computer Science, University of Milan, Milano, Italy

1 Introduction

Exploration of unknown environments through the deployment of multirobot systems is a task required in many applications, including map building (Thrun 2002) and search and rescue (Tadokoro 2010). In such scenarios, the process of discovering unknown features of the environment can be generally modeled with the following operations iteratively undertaken by each robot:

(a) perceive the surrounding environment,
(b) integrate perceived data in a map representing the environment known so far,
(c) decide the next locations to reach, and
(d) move to the selected locations.

In the basic and most common formulation, the choice of where to sense next (Step (c)) is guided by the selected *exploration strategy*. Moreover, it is often assumed that robots can

Published online: 25 July 2017

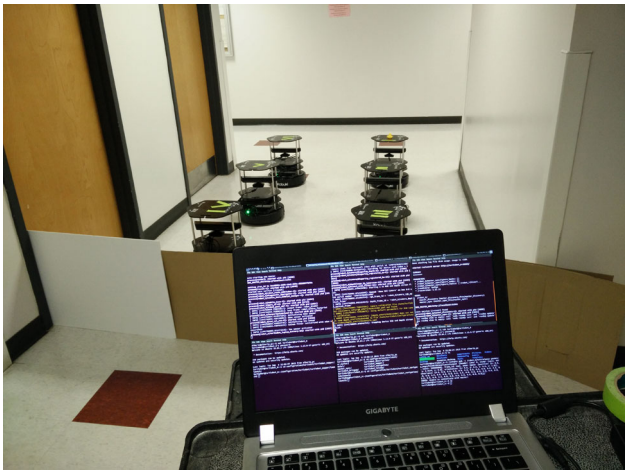


Fig. 1 Six robots with a supervising base station, with the task to explore an environment

always communicate with each other with high-bandwidth, e.g., (Yamauchi 1998; Quattrini Li et al. 2016).

However, such a strong assumption is not necessarily satisfied in real-life applications and has an impact on the performance of the system, as it has been shown by Tuna et al. (2013). This holds true especially when exploration is performed in disaster mitigation domains (Ochoa and Santos 2015), where centralized situational awareness at a *base station* (BS) is often required for the effective supervision of the mission; see Fig. 1 for an experimental setup where the BS overlooks six exploring robots. One important consequence is that robots not only have to efficiently explore, but also need to report and share the data they gather by communicating with each other and with the BS. Different types of connectivity constraints are assumed in literature, as reported in the short taxonomy provided in (Banfi et al. 2015). Loose connectivity constraints allow robots to explore the environment more efficiently, but reduce the situational awareness at the BS. On the other hand, strict connectivity constraints — e.g., requiring the whole team to be always globally connected — restrict the explored area but increase mission awareness at the BS. The literature proposes multi-robot exploration strategies that take into account different types of connectivity constraints (Rooker and Birk 2007; Stump et al. 2011; Spirin et al. 2013; Pei et al. 2013). Among them, *recurrent connectivity* constraints can provide a good trade-off between situation awareness and exploration efficiency. With recurrent connectivity, robots have to connect with each other and with the BS each time they gather new information. This entails an *online* constraint scheme, where robots can disconnect for arbitrarily long periods, but they must be able to coordinate in order to report to the base station as soon as new information is acquired.

This paper addresses the problem of multirobot exploration under recurrent connectivity in a novel *asynchronous* way: new plans can be computed for arbitrary groups of θ robots (where θ is a parameter) as soon as they become *ready*. Readiness can be defined as the state when robots have reached their goal locations and transmitted all the information acquired therein. The exploration strategy we propose is fundamentally centralized: new plans are computed at the BS, which can monitor the readiness status of each robot. Once a sufficient number of robots becomes ready, the BS issues a new plan.

Within this framework, we formulate the problem of computing the optimal set of locations robots should reach during the exploration of an environment. The solution should achieve efficient exploration, while accounting for recurrent connectivity constraints. We propose two strategies for solving this problem. In the first strategy, robots' locations are selected with an exact method based on integer linear programming (ILP). In the second strategy, planning is decomposed in two stages. First, an optimal set of connected locations is computed abstracting away from the robot-location assignments. Then, the most efficient assignment of robots to the locations found in the first stage is computed. For this second approach, we propose an exact (ILP-based) method and an approximation algorithm.

Experimental results, obtained both in simulation and with a real team of TurtleBot 2 robots, show that the first approach can provide better deployments, but with limited scalability in highly asynchronous settings (with low θ). As the number of ready robots increases, the two-stage approach turns out to be the preferred choice. In general, results show that this latter approach is able to achieve a good trade-off in terms of explored area, traveled distance, and situation awareness at the BS. The proposed approach is also competitive compared to a state-of-the-art exploration strategy (Spirin et al. 2013) not imposing any strict connectivity constraint, with which robots can acquire an arbitrary amount of information before sharing it with the BS.

The contributions of this paper significantly extend the preliminary results from (Banfi et al. 2016), adding the design of the single-stage approach and a richer experimental analysis both in simulation and with real robots.

This paper is structured as follows. The next section reviews communication-constrained multirobot exploration. Section 3 formalizes the multirobot exploration framework considered in this paper. Section 4 introduces the proposed one-stage solution for finding optimal connected robots' locations, while Sect. 5 describes the two-stages approach. Section 6 discusses robot readiness in the proposed context of asynchronous planning. Sections 7 and 8 present the experimental results in simulation and with real robots, respectively. Section 9 discusses the experimental results more in depth, trying to draw some conclusions of general validity. Finally, Sect. 10 concludes the paper.

2 Related work

The problem of online multirobot exploration in presence of a fixed BS to which the gathered information is transmitted has been investigated in different variants. Most of the works are built upon the seminal paper of Yamauchi (1998) on multirobot frontier-based exploration, where the idea is to have robots moving towards the boundaries (*frontiers*) of the known free space without any communication constraint. A multitude of approaches have been proposed during the years, not explicitly considering communication constraints, e.g., (Wurm et al. 2008; Quattrini Li et al. 2016; Simmons et al. 2000; Basilico and Amigoni 2011; Rekleitis 2013). The survey by Julia et al. (2012) provides an overview of exploration strategies.

More recently, research has also been focusing on how robots should coordinate to satisfy some communication constraints. An experimental analysis on the effects of communication constraints on the exploration has been presented in (Banfi et al. 2015). In the following, we group relevant works according to the enforced connectivity requirements.

2.1 Continous connectivity

A first way to address communication constraints is to maintain continuous connectivity between all the robots and the BS, either directly or in a multi-hop fashion. This could be useful, for instance, in situations where real-time image streaming is important (e.g., in search and rescue). The algorithm proposed by Mukhija et al. (2010) constructs a connected exploration tree in which the robots are organized as *explorers* and *link stations*: explorers are placed at the leaves of the tree, while the link stations are at the inner nodes and ensure the connectivity of the BS (the root) with the explorers. Rooker and Birk (2007) devise a local search method where the utility of a team configuration is computed in terms of distances from the closest frontiers: a configuration that does not satisfy full connectivity is highly penalized and is never chosen by the algorithm. Reich et al. (2012) propose a distributed protocol so that the physical layer connectivity of a mobile wireless network is maintained. Clearly, guaranteeing continuous connectivity can introduce non-negligible costs for the exploration performance. Therefore, if such a requirement is not strictly needed, other approaches are usually preferred.

2.2 Periodic connectivity

Several exploration strategies allow robots to explore regions in autonomy, but force them to communicate their discoveries to the BS under a more or less strict periodic regime. Therefore, we can say that such approaches enforce a *periodic reconnection* scheme. Hollinger and Singh (2012) consider a general mission scenario in which robots must *synchronously* regain global connectivity with the BS after a fixed time interval. The authors prove the inapproximability of the problem and propose a heuristic algorithm based on planning robots' paths in turns. The best path is chosen from a pool of samples according to a utility function which, in an exploration context, is typically related to the information gain of the path.

Some works also consider periodic connectivity as an *asynchronous* condition that, although desired, is not enforced as a hard constraint. In some cases, connections are established as the result of an emergent behavior of the algorithms. For example, Visser and Slamet (2008) include a criterion in the exploration strategy that takes the communication probability into account in order to favor locations in which it is high. de Hoog et al. (2009) propose the so-called Role-Based exploration, a distributed strategy in which robots are allowed to explore without considering communication limits. Rendezvous points, namely selected locations where exploring robots can communicate to the BS (possibly through relays), allow asynchronous updates of the environment map at the BS. In (Spirin et al. 2013), the robots' behavior is regulated by a utility function, which considers the amount of information not delivered yet by a robot to the BS and the predicted amount of information known by the BS. Tuning a parameter, the mission planner is able to specify strategies ranging from a completely greedy exploration, with no returns to the BS, to an exploration ensuring the maximum update frequency at the BS. Stronger forms of asynchronous connectivity are investigated by the works of Arkin and Diaz (2002) and Jensen et al. (2016). The former focuses on line-of-sight connectivity. A behavior-based architecture is proposed and tested in exploration scenarios with increasing prior information about the environment. The latter, although not explicitly considering a fixed BS, is able to fully explore an unknown environment in a distributed way. The proposed architecture relies on a small set of behaviors and messages exchanged between robots and dropped beacons. In both these last works, lost connectivity triggers an appropriate recovery behavior.

2.3 Recurrent connectivity

A third way in which the communication constraints can be defined is to ensure global connectivity only at the *deployment locations* of the robots, thus enforcing *recurrent connectivity* each time a robot collects new data. Such communication constraints constitute the main focus of our work. This is motivated by the fact that, typically, new information is gathered at the robots' goal locations, and robots can get disconnected for arbitrarily long periods of time while traveling to them. Howard et al. (2002) study the problem of mobile sensors placement for maximizing the coverage of an

unknown area while keeping each node connected to a BS via a multi-hop mutual-visibility constraint. The algorithm proceeds by sequentially deploying nodes after selecting the best goal locations. More recent work related to communication nodes deployment has been presented by Stump et al. (2011). Here, a set of agents is assumed to be already present in an environment (e.g., exploring), and two problems are tackled: (i) finding a deployment of relay nodes, which ensures global connectivity between each agent and the BS (again stated in terms of mutual visibility), and (ii) given the current deployment and new locations agents should reach, finding the redeployment that minimizes robots' traveling time. The former problem is reduced to the computation of a minimum Steiner tree with the agents' locations as terminal set, while the latter is solved by using a (generally sub-optimal) dynamic programming algorithm. Finally, the problem setting addressed in (Pei et al. 2013) shares some basic features with the one considered in this paper. That work proposes an approach that takes into account bandwidth constraints over the robots relay chain under the "disk" communication model—i.e., two robots can communicate if they are within a given maximum distance. New plans are computed once the *whole* network has been formed. The general optimization problem is split into sub-problems: explorers placement, relays placement, and robot path generation. In particular, given a set of candidate locations to be connected, relays placement is achieved by solving variations of the Steiner minimum tree problem with minimum number of Steiner points and bounded edge length (Cheng et al. 2008). However, this choice is intimately related to the adoption of the disk communication model under a synchronous planning setting. The contributions of our paper lie on a complementary direction: our proposed methods do not depend on a specific communication model and do not require synchronous coordination among robots.

3 Problem definition

In this section, we formalize the problem considered in this paper, which has the same setup as the one considered in (Banfi et al. 2016). Table 1 summarizes the notation used in the paper.

3.1 Assumptions

An initially-unknown, two-dimensional, continuous, and bounded environment $\text{Env} \subset \mathbb{R}^2$ is considered. The interior points of the environment can belong to obstacles of arbitrary shape, whose set is denoted by Env_o, or can belong to the free space, denoted by $\text{Env}_f = \text{Env} \setminus \text{Env}_o$. A supervising control center called base station (BS) is deployed in Env at a fixed location, along with a team of m mobile robots $R = \{r_1, r_2, \ldots, r_m\}$. Each mobile robot is equipped with a finite-range sensors able to perceive the surrounding free space and outer boundaries of obstacles (e.g., laser range scanners or RGB-D sensors) and is capable of exchanging data with other robots or with the BS over an *ad hoc* network. The BS and the robots maintain and update a *map* representing the portion of environment discovered so far, represented as an occupancy grid. Note that the BS can also receive other types of data from the robots, such as a video feed in a search and rescue scenario. In particular, each robot is endowed with its own Simultaneous Localization and Mapping (SLAM) system (Thrun et al. 2005), so that temporary disconnections from the rest of the team do not result in loss of autonomy. The BS updates the map of the explored area each time new information is sent by the robots. Note that the choice of adopting a particular SLAM algorithm is orthogonal to the exploration strategy we present in this paper.

For simplicity, we assume that time evolves in discrete steps $t \in \{1, 2, \ldots, T\}$, where T denotes the last step of the exploration mission. Upon the grid-based map known at a generic time step t, the BS is able to construct a graph-based representation of the environment $G^t = (V^t, C^t)$, where vertices in V^t encode some discretization of the portion of Env_f known so far. Such a discretization should represent a reasonable trade-off between the ability to represent the most salient communication features of the environment and the size of the graph. Each vertex $v \in V^t$ is associated with a candidate robots' goal location, except for the vertex b which denotes the fixed position of BS. A set $F^t \subseteq V^t \setminus \{b\}$ denotes the *exploration frontiers*, that is, vertices corresponding to locations of Env_f lying on the boundary between explored and unexplored portions of Env. As customarily done in robotic exploration, each vertex $v \in V^t$ is associated with a numerical value $g(v)$ representing the (expected) *information gain* obtainable by taking a perception from v. Typically, the information gain is 0 when $v \notin F^t$, and proportional to the new area expected to be seen from v, otherwise. Each pair of vertices $i, j \in V^t$ is associated with a value $d(i, j)$, representing the distance between them as known by the BS and the robots. The edge set C^t encodes the communication features of the environment. In particular, this set is determined by link-detection mechanisms in charge of recognizing the availability of a communication link between any two known vertices. Link-detection methods range from simple visibility-based criteria — that is, two vertices can communicate if in direct line of sight within a maximum range — to more sophisticated approaches considering models of signal decay through distance and obstacles. As in (Stump et al. 2011; Pei et al. 2013; Hollinger and Singh 2012), we make three assumptions. First, edges in C^t are static. Second, the link-detection mechanism is not affected by false positives. Third, if the BS and one or more robots form a connected component in C^t, then any of such robots

Table 1 Summary of the notation used in the paper

$t \in \{1, 2, \ldots, T\}$	$\triangleq$	Generic mission time step
$R = \{r_1, r_2, \ldots, r_m\}$	$\triangleq$	Team of robots
$G^t = (V^t, C^t)$	$\triangleq$	Planning graph at time t (V^t = locations, C^t = links)
$b \in V^t$	$\triangleq$	BS vertex (fixed)
$F^t \subseteq V^t \setminus \{b\}$	$\triangleq$	Frontier vertices at time t
$g(v)$	$\triangleq$	Information gain of vertex $v \in V^t$
$d(i, j)$	$\triangleq$	Distance between vertices i and j ($i, j \in V^t$)
R_e^t	$\triangleq$	Ready robots at time t
R_c^t	$\triangleq$	Robots connected with the BS at time t
$\pi^t = \langle p_1^t, p_2^t, \ldots, p_m^t \rangle$	$\triangleq$	Robots deployment computed at time t ($p_r^t \in V^t \setminus \{b\}$ is the goal vertex of r)
$Q^t = \{q_1^t, \ldots, q_m^t\}$	$\triangleq$	Configuration (occupied vertices) computed at time t, $Q^t \subseteq V^t \setminus \{b\}$
w_d, w_g	$\triangleq$	Weights of the objective function (one-stage approach)
α	$\triangleq$	Parameter used to express w_d and w_g
$\mathcal{C}^+, \mathcal{C}^-(v)$	$\triangleq$	Arcs leaving (+) or entering (-) vertex v in the directed version of G^t
$\delta^+(S), \delta^-(S)$	$\triangleq$	Directed cuts induced by $S \subseteq V^t \setminus \{b\}$ in the directed version of G^t
$U(f), U(S)$	$\triangleq$	Utility of frontier f and vertices $S \subseteq V^t \setminus \{b\}$ (two-stage approach)
δ	$\triangleq$	Parameter of the approximation algorithm (two-stage approach)
P_f	$\triangleq$	The first forming path connecting frontier f and the BS in a new deployment
θ	$\triangleq$	Parameter representing the number of ready robots required to compute a new plan

can exchange data with the BS using some protocol. The bandwidth is assumed to be large enough to allow data transfer between robots. Such assumptions are validated in Sect. 8 through experiments with real robots.

3.2 Exploration process

Within the framework described above, the exploration mission under recurrent connectivity constraints evolves as follows. New robots' deployments in the known portion of space are dictated by the BS and issued at selected time steps. In particular, at any $t \in \{1, 2, \ldots, T\}$, a new *deployment*

$$\pi^t = \langle p_1^t, p_2^t, \ldots, p_m^t \rangle,$$

where $p_r^t \in V^t \setminus \{b\}$, can be computed. This means that planning is performed at selected time steps in $\{1, 2, \ldots, T\}$. The new deployment specifies the goal vertices each robot r is headed to. Each deployment π^t implicitly defines also a *configuration* $Q^t = \{q_1^t, \ldots, q_m^t\}$, $Q^t \subseteq V^t \setminus \{b\}$, containing the m vertices that will be occupied by the robots and abstracting away from the robot-location assignment. Thus, a deployment π^t can be thought as an assignment of robots to goal locations taken from a configuration Q^t, that is $\pi^t(r) = p_r^t \in Q^t$. Note that multiple deployments π^t can be obtained from the same configuration Q^t.

Each robot follows the directives coming from the BS. In particular, once a deployment π^t is computed, the robot r travels to the corresponding goal vertex p_r^t. Once there, if $p_r^t \in F^t$, then r must take a new range scan — e.g., execute a complete rotation to maximize the explored area, and transmit the newly gathered data (and/or forward data received by others) towards the BS. Instead, if $p_r^t \notin F^t$, then r just acts as a relay to convey information back to the BS. Notice that all the robots transmit to the BS also the sensing data they have acquired while going to the assigned locations.

A robot r is marked *ready* if (a) it has reached its previous goal vertex and completed its sensor measurements in case of frontier vertex and (b) it has transmitted to the BS its perceived data and no other robot still requires it as a relay. R_e^t denotes the set of ready robots at step t. Note that, to declare a robot to have completed its service as relay, several specific conditions could be defined, either offline or online i.e., jointly specified with new deployments and fixed, or changeable according to suitable policies. Section 6 discusses a possible implementation of the readiness condition. Finally, let $R_c^t \supseteq R_e^t$ be the set of robots *connected* with the BS at time t.

The following constraints are posed for a deployment π^t:

(I) $Q^t \cup \{b\}$ must form a single connected component in G^t;
(II) if $F^t \neq \emptyset$, then $Q^t \cap F^t \neq \emptyset$, that is at least one frontier must be reached by a robot;
(III) for each robot $r \notin R_e^t$ it must hold that $p_r^{\bar{t}} \in Q^t$, where $\bar{t} < t$ denotes the time step of the last issued deployment;

(IV) π^t must not change the robot-goal assignments for each robot $r \notin R_c^t$ (i.e., non-connected robots).

Constraint I imposes recurrent connectivity, by forcing robots to be able to exchange data with the BS when they occupy their goal vertices. Constraint II requires a minimum exploration progress rate: if no frontier can be further visited, then no additional space can be explored and the mission ends. (Note that, in this case, exploration can progress only by relaxing the connectivity constraint.) Constraints III and IV define feasible *asynchronous re-deployments*. Specifically, Constraint III allows the BS to accommodate a number of new goals equal to that of ready robots. That is, preemption at configuration level for unreached goals is forbidden. Instead, robots' preemption is allowed in favor of a new deployment. This might be preferred in the presence of new goals. Finally, by Constraint IV, a robot's deployment can be changed only if it is able to receive data from the BS (so, a non-ready robot's assignment could be modified if more convenient). To guarantee the correct reception of a new plan, a temporary halt of all the non-ready, connected robots can be enforced.

The above constraints do not model the feasibility of the transition between one configuration to another. For instance, it might be possible that the path planning module of a robot fails to produce a plan allowing to reach the assigned goal due to, e.g., interference with paths of other robots, or noise in the map. If this happens, a suitable procedure must be in charge of detecting the problem and computing a new plan. We think that such an approach is much simpler than trying to incorporate into the above constraints a multirobot path planning subproblem, which is very difficult *per se*; see, e.g., (Yu 2016).

In principle, the computation of new deployments can be triggered by different conditions. This generalizes the approach adopted in (Pei et al. 2013), where new plans are issued only when all robots have reached their goal locations. While performing a deployment, some robots could become ready before others, hence becoming able to receive a new location to reach. As it will be shown in the next sections, new deployments involving only part of the robot team can exploit communication links that will be made available by robots currently not ready to establish communication links with the BS. Figure 2 shows a snapshot of an exploration process with ready and not ready robots.

3.3 Method overview

Given the setting described above, the objective is to compute robots' deployments $\pi^t(\cdot)$ allowing the multirobot system to incrementally achieve efficient exploration of the environment in terms of explored area and traveled distance. At the same time, such a deployment should comply to recurrent connectivity constraints. In the following, we present two approaches for computing a new deployment.

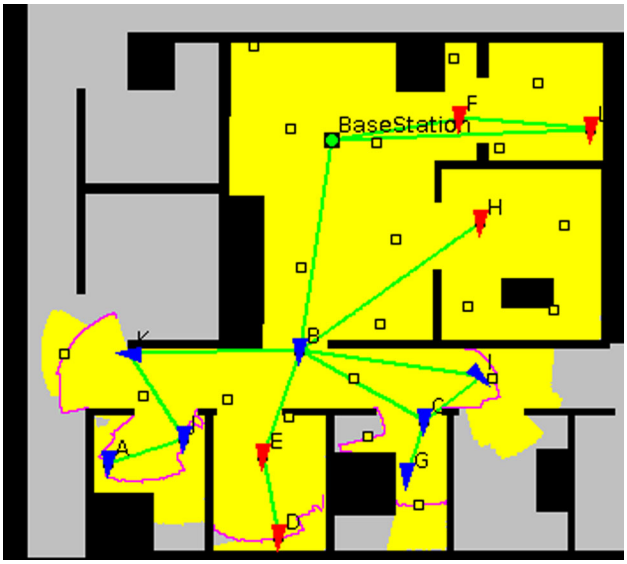


Fig. 2 Exploration snapshot. *Blue* and *red* not ready and ready robots, respectively. *Black-edged squares* vertices of G^t. *Green* current communication links. *Purple* frontiers of the last issued plan. Image from (Banfi et al. 2016) (Color figure online)

The first is described in Sect. 4. The *optimal deployment* under Constraints I-IV is computed by solving a single bi-objective Integer Linear Program (ILP). The proposed ILP maximizes the information gain $g(v)$ collected from the vertices to be visited while, at the same time, minimizes the distance traveled by the robots.

In Sect. 5, we present our second approach. In general, it is less computationally demanding and, as a consequence, could be more suitable for real online settings. The main idea is to solve the planning problem in two stages. The first stage computes an *optimal configuration* Q^t under a given utility function defined on the information gain of the frontier vertices. The second stage finds an *optimal deployment* $\pi_t(\cdot)$ that minimizes the traveled distance, given the optimal configuration Q^t.

4 Optimal one-stage approach

The first approach we propose finds the optimal robots' deployment $\pi^t(\cdot)$ for a set of ready robots R_e^t by solving a single Integer Linear Program (ILP). For this reason, we refer to it as an optimal *one-stage* approach. Our ILP is such that its optimal solution encodes a deployment that maximizes a weighted combination of the information gains and traveling costs.

Our ILP is inspired by (Álvarez-Miranda et al. 2013), where a related problem, called *Rooted Maximum Node-*

Weight Connected Subgraph with Budget Constraint (B-RMWCS), is solved. The formulation works on a *directed* version of G^t obtained as follows: each undirected communication edge is replaced by two symmetric arcs except for the edges incident to the BS vertex b, which are replaced by outgoing arcs. With a slight overload of notation, we call C^t the set of arcs obtained after such replacements.

Each ILP solution is a robot deployment $\pi^t(\cdot)$ that is encoded by the following decision variables:

- z_{rv}, taking value 1 if and only if robot r is associated with vertex $v \in V^t \setminus \{b\}$ (number of variables: $|R|(|V^t|-1)$);
- y_v, taking value 1 if and only if vertex $v \in V^t \setminus \{b\}$ is selected in the solution (number of variables: $|V^t|$);
- x_{ij}, taking value 1 if and only the directed arc (i, j) is selected in the solution (number of variables: $|C^t|$).

In any feasible solution, variables z_{rv} and y_v encode a well-formed deployment π^t and the associated configuration Q^t, respectively. Variables x_{ij}, instead, encode an arborescence rooted at the BS and spanning the vertices of Q^t (those selected with variables y_v). By definition, an arborescence guarantees the existence of a unique directed path (a connected sequence of selected arcs) from the BS to each selected vertex. The existence of such paths implies that, in the original undirected graph, the deployment encoded by variables z_{rv} is connected.

Let us call, with a slight overload of notation, $d(r, v)$ the distance between the current pose of robot $r \in R_c^t$ and vertex $v \in V^t$ estimated at the time the ILP is instantiated to be solved. (Notice that $d(r, v)$ can be accurately computed at the BS, since $r \in R_c^t$ is in communication by definition.) The objective function we optimize reads as follows:

$$\text{maximize} \sum_{r \in R_c^t} \sum_{v \in V^t} [w_g g(v) - w_d d(r, v)] z_{rv} \tag{1}$$

This function balances the cumulative information gain $g(\cdot)$ robots can get from a joint perception and the cost as the total traveled distance $d(\cdot)$. The trade-off between the two is given by the parameters w_d, w_g. In particular,

$$w_d = \frac{\alpha}{\max_{r \in R_c^t v \in V^t} d(r, v)} \text{ and } w_g = \frac{1-\alpha}{\max_{v \in V^t} g(v)},$$

with $\alpha \in [0, 1]$, so that the contribution given by each robot in the deployment is a number varying between 1 and -1. Note that the contributions given by the robots currently not in communication with the BS are excluded from the objective function, as they will be equal in all the candidate solutions.

Now, to introduce the set of constraints, we denote with $\mathcal{C}^+(v)$ and $\mathcal{C}^-(v)$ the arcs leaving (+) or entering (-) vertex $v \in V^t$, respectively. The directed cuts induced by the set of vertices $S \subseteq V^t$ are defined as $\delta^+(S) = \{(i, j) \in C^t \mid i \in S, j \notin S\}$ and $\delta^-(S) = \{(i, j) \in C^t \mid i \notin S, j \in S\}$. In Fig. 3 we report a simple example of a graph G^t and the associated directed version where an example of directed cuts is depicted. Also, let $\pi^{\bar{t}} = \langle p_1^{\bar{t}}, p_2^{\bar{t}}, \ldots, p_m^{\bar{t}} \rangle$ be the deployment associated to the last issued plan (or to the starting position of the robots if exploration is just starting). The maximization of Eq. 1 is subject to the following set of constraints:

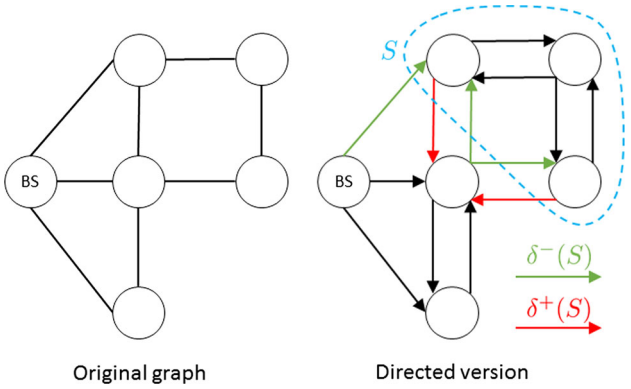


Fig. 3 An example of G^t and its directed version. Directed cuts induced by a set of vertices S are shown too

$$\sum_{(i,j) \in \mathcal{C}^-(v)} x_{ij} = y_v \qquad \forall v \in V^t \setminus \{b\} \tag{2}$$

$$\sum_{(i,j) \in \delta^-(S)} x_{ij} \geq y_v \qquad \forall S \subseteq V^t \setminus \{b\}, \forall v \in S \tag{3}$$

$$\sum_{r \in R} z_{rv} = y_v \qquad \forall v \in V^t \setminus \{b\} \tag{4}$$

$$\sum_{v \in V^t \setminus \{b\}} z_{rv} = 1 \qquad \forall r \in R \tag{5}$$

$$\sum_{f \in F^t} y_f \geq 1 \tag{6}$$

$$y_{p_r^{\bar{t}}} = 1 \qquad \forall r \in R \setminus R_e^t \tag{7}$$

$$z_{rp_r^{\bar{t}}} = 1 \qquad \forall r \in R \setminus R_c^t \tag{8}$$

Constraints 2 and 3, whose number is $|V^t| - 1$ and $(|V^t| - 1)2^{|V^t|-2}$, respectively, force the new deployment to be connected in the form of an arborescence rooted at the BS, formalizing Constraint I of Sect. 3. Constraints 4, whose number is $|V^t| - 1$, impose that each vertex belonging to Q^t must be occupied by exactly one robot. Constraints 5, whose number is $|R|$, imposes the allocation of each robot to exactly one vertex in the new deployment. Constraint 6 imposes the inclusion in the new deployment of at least one frontier vertex (Constraint II of Sect. 3). Constraints 7, which are at most $|R|$, force to include in the new configuration those vertices representing goal locations of non-ready robots (Constraint III

of Sect. 3). Finally, Constraints 8, again at most $|R|$, enforce not to change vertex allocation for robots currently not in communication with the BS (Constraint IV of Sect. 3).

We complete the description of the ILP we solve to find an optimal deployment with the following observations:

- Our problem formulation is such that, in presence of non-ready and/or non-connected robots, the new deployment of ready robots can exploit communication links that are made available by the former, due to the constraints imposed on the y_v variables.
- ILPs are NP-hard, thus modern solvers have exponential worst-case running times; however, on the average case, good efficiency is often achieved (as we show in our experiments).
- The number of Constraints 3 is exponential in the size of the input. Therefore, as detailed in Sect. 7, to optimally solve the model, a Branch&Cut approach similar to that used in (Álvarez-Miranda et al. 2013) is employed. The idea is to gradually introduce violated inequalities (3) as soon as the solution of a new LP relaxation is available. The problem of recognizing such violated constraints can be solved in polynomial time.

5 Optimal and approximate two-stage approaches

The second approach we present was originally introduced in (Banfi et al. 2016) and, differently from the previous one, exploits a decomposition into two sub-problems. Each sub-problem is then solved in a separate stage. The first stage (Sect. 5.1) is the *optimal configuration* problem, in which we seek for the configuration Q^t that maximizes a utility function defined on frontiers and that satisfies the recurrent connectivity constraints. The second stage (Sect. 5.2) is the *optimal deployment* problem, in which, given the optimal configuration calculated in the previous stage, we compute the robot-location assignment π^t that minimizes the traveling costs.

5.1 First stage: optimal configurations

Configurations are evaluated by using the cumulative information gain achievable by acquiring sensor data from the selected locations. Instead of explicitly accounting for the traveled distance in this first stage of planning, the information gain of each frontier is heuristically weighted by a measure of distance (a lower bound) that any ready robot will need to travel to reach it. In particular, the utility $U(\cdot)$ of a frontier vertex f combines the estimated information gain $g(f)$ and its minimum traveling cost (Spirin et al. 2013; Pei et al. 2013) as:

$$U(f) = \frac{g(f)}{\min\limits_{r \in R_e^t} d^2(r, f)},$$

where $d(r, f)$ denotes the current estimated distance of robot r to frontier f, and gains and distances can be thought as normalized with respect to their maximum value. Non-frontier vertices have null utility and, with a slight overload of notation, $U(S) = \sum_{v \in S} U(v)$ denotes the utility of any subset of vertices $S \subseteq V^t$.

The goal is to find a configuration Q^t of size at most $|R_e^t|$ (the number of ready robots) on G^t that is optimal, i.e., that maximizes $U(Q)$ while maintaining connectivity with the BS and accounting for the presence of non-ready and non-connected robots. This problem, in case all robots are ready, can be seen as particular case of known NP-hard problems such as the *Constrained Maximum-Weight Connected Graph problem* (CMCG) (Lee and Dooly 1996), the *Rooted Budget Prize-Collecting Steiner Tree problem* (B-RPCST) (Johnson et al. 2000), and the *Rooted Maximum Node-Weight Connected Subgraph with Budget Constraint* (B-RMWCS) (Álvarez-Miranda et al. 2013). In particular, the problem of computing an optimal configuration can be shown to be NP-hard with a simple adaptation of the reduction outlined in (Hochbaum and Pathria 1994) for the unconstrained version of CMCG. Moreover, notice that the computation of the optimal configuration can be thought as a particular case of computing an optimal deployment as formalized in Sect. 4 (it suffices to assume that the constraint on the number of robots belonging to the configuration is tight), where all the distances robots have to travel are equal to a fixed constant. In the following, two alternative approaches to find the optimal configuration are presented.

5.1.1 Finding the optimal configuration

Similarly to the formulation presented in Sect. 4, an ILP formulation can be derived by simplifying that discussed in (Álvarez-Miranda et al. 2013) for solving B-RMWCS. The idea is to find the best connected configuration in the form of an arborescence computed on the *directed* version of G^t (generated as in Sect. 4). First, G^t is further modified by removing the vertices assigned to non-ready robots and introducing fictitious communication edges directly connecting the neighbors of the removed vertices with the vertex b of the BS. This modification clearly simplifies the problem while it encodes the fact that non-ready robots already have a goal assigned. Such robots could be relays for the current replanning robots. The following binary variables are defined:

- y_f, taking value 1 if and only if frontier $f \in F^t$ is selected in the configuration (number of variables: $|F^t|$);
- x_{ij}, taking value 1 if and only if the directed arc $(i, j) \in C^t$ is selected in the configuration (number of variables: $|C^t|$).

Notice that, contrary to the ILP model in Sect. 4, vertex variables y_f are defined only for frontier vertices. Again, $\delta^+(S)$ and $\delta^-(S)$ denote the directed cuts induced by the set of vertices $S \subseteq V^t$. The ILP model reads as follows:

$$\text{maximize} \sum_{f \in F^t} U(f) y_f \tag{9}$$

subject to

$$\sum_{(i,j) \in C^t} x_{ij} \leq |R_e^t| \tag{10}$$

$$\sum_{(i,j) \in \delta^-(S)} x_{ij} \geq y_f \quad \forall f \in F^t, \quad \forall S \subseteq V^t \setminus \{b\} \text{ s.t. } f \in S \tag{11}$$

The objective function 9 maximizes the configuration utility. Constraint 10 limits the size of the new configuration to the available number of robots. If frontier f is occupied, Constraints 11 require a connected sequence of links from the BS to f. The number of this latter set of constraints is upper-bounded by $(|V^t| - 1)2^{|V^t|-2}$. Note that, compared to the ILP of the one-stage approach, this model contains significantly less variables and constraints.

Given an optimal solution of the above ILP, the (weakly) connected component containing b, pruned of some possible useless arcs, represents an optimal configuration from which it is easy to retrieve the final Q^t. Despite the exponential multiplicity of the last set of constraints, the model can still be solved optimally for instances of moderate size by the same Branch&Cut algorithm introduced with the formulation of Sect. 4; see Sect. 7 and (Álvarez-Miranda et al. 2013) for details.

5.1.2 Approximating the optimal configuration

Real operational conditions require computational efficiency and to keep the problem tractable when the number of robots and/or the size of G^t increase. We now discuss an approximation method, able to introduce efficiency in the computation of the new Q^t with a bounded loss of optimality. The literature presents different studies on the approximation of CMCG (Lee and Dooly 1996), B-RMWCS (Moss and Rabani 2007), and B-RPCST (Chekuri et al. 2012). The best result applicable to the problem of finding an optimal configuration is the $4+\epsilon$ approximation algorithm of Chekuri et al. (2012). However, differently from the standard approach followed in optimization literature, we do not seek for an effective solution for *any* instance. Instead, we leverage our domain knowledge where vertices are physical locations and ready robots are not so many. In this section it is shown how, by exploiting such features, a different approximation algorithm can be developed. Our approach, despite not being competitive with the best known one for general instances (as it achieves only a non-constant approximation factor) is able to provide a *better approximation guarantee* in *typical exploration settings*.

The main idea is to iteratively compute the cheapest tree (in terms of number of required robots) allowing to connect the BS with fixed subsets of frontiers of at most a given size. To compute such a tree, we need to solve a Steiner tree problem, formally defined below.

Steiner tree problem given an undirected graph $G = (V, E)$, a subset of terminal vertices $S \subseteq V$, and a non-negative edge cost $c_e \geq 0$ for each edge $e \in E$, find a minimum-cost subset of edges $E' \subseteq E$ such that $G = (V, E')$ contains a path between each pair of vertices in S.

Our problem maps on the Steiner tree problem as follows: terminal vertices are associated to a particular subset of frontiers and the BS vertex b, while edge costs are unitary. The number of vertices in the minimum-cost tree is then the minimum number of robots required to connect the subset of frontiers we chose. Our algorithm is also based on two further considerations. For any $\delta \in \mathbb{N}$:

- if the number of vertices to connect is upper bounded by $\delta + 1$, then the optimal Steiner tree can be computed in polynomial time (Kimelfeld and Sagiv 2006); let us call $(\delta + 1)$-*Steiner* an algorithm performing such a task; for any instance with up to $\delta + 1$ terminals the $(\delta + 1)$-Steiner algorithm computes efficiently the optimal solution; clearly, since the Steiner tree problem is NP-hard (even when edge costs are equal) (Garey and Johnson 1979), this does not imply an efficient algorithm for the general case (any number of terminals), for which $(\delta + 1)$-Steiner cannot be applied;
- the enumeration of the subsets of at most δ frontiers can be done in polynomial time.

The proposed approximation algorithm works on a modified version of G^t, where vertices assigned to non-ready robots are removed as previously explained. Also, it operates by fixing to δ the maximum number of frontiers that can be occupied by the configuration (thus allowing suboptimalities). Then, for each subset F_δ of at most δ frontiers, the algorithm searches for the optimal Steiner tree T^* connecting $F_\delta \cup \{b\}$ by employing the $(\delta + 1)$-Steiner algorithm. If the number of vertices in T^* does not exceed the budget of robots $|R_e^t|$, then T^* is considered a feasible configuration and checked against the best configuration found so far. Otherwise, if T^* exceeds the budget, then, being T^* the cheapest tree, no feasible configuration exists for F_δ.

Algorithm 1 formally presents the steps of the proposed method. The `genNewSubset()` function is used to retrieve the next subset of frontiers to examine (Step 2). It returns a subset F_δ of at most δ frontiers such that no other $F'_\delta \subseteq F_\delta$ has already been determined as non-feasible; this simple prun-

ing rule is applied by maintaining a closed list of discarded solutions in Steps 7 and 13. Subsets are heuristically ranked by decreasing utility and increasing size. Notice that, in an efficient implementation, new subsets are generated lazily, and they need to be checked only against the newly added discarded solution; alternatively, the algorithm can be easily adapted for a recursive implementation not needing any closed list.

For each returned subset F_δ, two simple tests are performed to determine if it can be discarded without running $(\delta+1)$-Steiner on it. First, the longest frontier-frontier or frontier-BS shortest path in G^t (Step 6) is checked: subtracting 1 to the number of spanned vertices, we have a simple lower bound on the minimum dimension of the Steiner tree connecting F_δ and b. If the value computed exceeds $|R_e^t|$, F_δ is discarded. Then, $U(F_\delta)$ is evaluated (Step 10). This is a simple lower bound of the total utility possible: if it does not exceed the current best solution, then F_δ is discarded. Note that this jump in the search is safe. The reason is that those frontiers not explicitly considered in F_δ, but potentially part of the "phantom" solution that could have been obtained by solving the Steiner tree on F and b (i.e., frontier nodes not considered as terminals when building the Steiner tree), will necessarily be taken into account later in the search in a subsequent set F'_δ s.t. $F_\delta \subset F'_\delta$. The Steiner tree problem on $F_\delta \cup \{b\}$ is solved in Step 11, and in Step 12 budget feasibility is checked. At the end of the while loop, it may be possible that some unoccupied frontiers are still reachable from the newly created connected configuration T^* with the remaining budget of robots. In this case, the `completeConfiguration()` function (Step 21) greedily adds branches from T^* to the unoccupied frontier with the highest utility, until no other frontier can be reached.

Algorithm 1 Compute approximate optimal configurations.

1: **while** True **do**
2: F_δ = `genNewSubset`(F^t, u, δ, closedList)
3: **if** $F_\delta = \emptyset$ **then**
4: break
5: **end if**
6: **if** `lowerbound`(G^t, F_δ, b) > $|R_e^t|$ **then**
7: closedList.add(F_δ)
8: continue
9: **end if**
10: **if** $U(F_\delta) > z^*$ **then**
11: $T = (\delta+1)$-Steiner(G^t, $F_\delta \cup \{b\}$)
12: **if** $|T| > |R_e^t|$ **then**
13: closedList.add(F)
14: continue
15: **else if** $U(T) > z^*$ **then**
16: $z^* = U(T)$
17: $T^* = T$
18: **end if**
19: **end if**
20: **end while**
21: `completeConfiguration`(G^t, b, F^t, U, $|R^t|$, T^*)

Theorem 1 *Let* $k = \min(|R_e^t|, |F^t|)$. *For* $\delta \in \{1, \ldots, k\}$, *Algorithm 1 is a* k/δ*-approximation algorithm for the problem of finding the new connected configuration. Its running time is bounded by* $O(|F^t|^\delta[3^{\delta+1}|V^t| + 2^{\delta+1}|C^t| \log |V^t|])$.

Proof For the definition of the `genNewSubset()` function, it is ensured that all the possible subsets of frontiers containing at most δ elements and able to improve the current solution can be generated. Therefore, at the end of the while loop, it is guaranteed that the computed tree T^* is connecting, within the budget limit, the BS vertex b with $F_{\delta\text{-BEST}}$ — i.e., the set of frontiers, possibly bigger than δ, when the computed Steiner tree makes use of other frontiers as non-terminal vertices, collecting the maximum utility. Let $F_{\text{OPT}} = \{f_1, f_2, \ldots, f_{|F_{\text{OPT}}|}\}$ be the set of frontiers included in the optimal solution where $\text{OPT} = U(F_{\text{OPT}})$. Clearly, if $\delta \geq |F_{\text{OPT}}|$, then the algorithm finds the optimal solution.

If instead $\delta < |F_{\text{OPT}}|$, let us assume, without loss of generality, that frontiers in F_{OPT} are labeled such as $U(f_i) \geq U(f_{i+1})$ and let $\beta = \lfloor |F_{\text{OPT}}|/\delta \rfloor$ and $\gamma = |F_{\text{OPT}}| \bmod \delta$. Moreover, to ease the notation, rename frontiers $f_{\beta\delta+1}, \ldots, f_{|F_{\text{OPT}}|}$ as $\bar{f}_1, \ldots, \bar{f}_\gamma$, respectively (these are the γ least-utility frontiers in F_{OPT}). Then we have:

$$\text{OPT} = \sum_{i=0}^{\beta-1} [U(f_{i\delta+1}) + \cdots + U(f_{(i+1)\delta})] + \sum_{i=1}^{\gamma} U(\bar{f}_i),$$

where the first summation iterates over subsets of frontiers of size δ and the second summation covers the possibly remaining frontiers. Since terms are ordered according to non-increasing utilities, we have that:

$$\text{OPT} \leq \beta U(F_{\delta\text{-BEST}}) + \gamma U(f_\delta)$$

by definition of $F_{\delta\text{-BEST}}$ and since $U(f_\delta) \geq U(\bar{f}_1)$. If $\gamma = 0$, then the first part of the claim directly follows since $|F_{\text{OPT}}| \leq \min(|R_e^t|, |F^t|)$. Otherwise, notice that $[U(f_1) + \cdots + U(f_\delta)]/\delta \geq U(f_\delta)$. This implies that

$$\text{OPT} \leq \beta U(F_{\delta\text{-BEST}}) + (\gamma/\delta) U(F_{\delta\text{-BEST}}) = \\ = \frac{|F_{\text{OPT}}|}{\delta} U(F_{\delta\text{-BEST}})$$

and the first part of the claim again follows.

The asymptotic running time is obtained by noticing that the while loop performs at most $\sum_{i=1}^{\delta} \binom{|F^t|}{i} = O(|F^t|^\delta)$ calls to the $(\delta+1)$-Steiner algorithm of (Kimelfeld and Sagiv 2006), whose running time is bounded by $O(3^{\delta+1}|V^t| + 2^{\delta+1}|C^t| \log |V^t|)$. The greedy completion of T^* does not influence neither the approximation nor the running time bound. □

It is easy to show that such an approximation bound, for suitable choices of δ, can yield a better approximation than the one provided by the best general algorithm presented in

literature (Chekuri et al. 2012), which has an approximation factor of $4+\epsilon$. For example, with a team of 10 robots and $\delta = 5$ an approximation factor of 2 can be obtained. Also, our algorithm is much more simple than that of (Chekuri et al. 2012) (a whose detailed complexity analysis of this last one is not straightforward).

Finally, notice that, when the algorithm is run with $\delta > \min(|R_e^t|, |F^t|)$, an optimal solution is always obtained.

5.2 Second stage: optimal deployments

Given a new connected configuration Q^t calculated as described in the previous section, it must be decided which robot goes to which vertex. Since we are not only interested in completing the *whole* deployment in the shortest time, instead of minimizing the maximum distance a robot has to travel, as in (Pei et al. 2013), in this paper the cumulative travel distance is minimized, as in (Stump et al. 2011). In particular, the Hungarian algorithm (Burkard et al. 2009) is used. Actually, since some non-ready robots may be in communication with the BS when replanning takes place, a new allocation of robots to vertices is computed taking into account also these robots and their previous target locations. This reduces possible path overlaps between ready and non-ready robots.

6 Readiness and asynchronicity

In this work, a communication protocol able to dynamically discover multi-hop paths between frontiers and BS is assumed to be in place, such as the Optimized Link State Routing Protocol (Clausen and Jacquet 2003). Accordingly, we now provide a specific definition of readiness that enables the BS to receive the data from the robots at the frontiers in minimum expected time, under the assumptions of sufficient bandwidth along each communication link and of negligible transmission times. First, notice that, although Sects. 4 and 5 focus on finding a connected tree on G^t, other links not explicitly considered in the solution could be available to transmit data between the vertices Q^t. Intuitively, the robots placed on the vertices of the *first* forming path on G^t from a selected frontier to the BS should remain available for serving as relays until all the frontier data have been transmitted to the BS (possibly, by taking other routes). Formally, given a deployment π^t, for each frontier $f \in F^t \cap Q^t$, call P_f the path connecting f and the BS vertex b in G^t that is firstly built according to the time that each robot r needs to travel from its current position to p_r^t. A robot r is defined to be *ready* if (a) it has reached its goal vertex p_r^t, and (b) for each $f \in F^t \cap Q^t$ whose P_f contains p_r^t, the data of f have been received at the BS, and every other robot r' for which $p_{r'}^t \in P_f$ is at its goal position. For each frontier f, the corresponding P_f can be conservatively computed right after the computation of $\pi^t(\cdot)$ (thus not considering possible other links discovered throughout the plan execution) by means of the following algorithm: order the robots in increasing order of traveling times; for each robot r, build a restricted version of G^t containing only the vertices of Q^t associated with robots expected to reach their goal before r; if a path exists, take the shortest, otherwise examine the next robot.

The reader might have noticed that, in practice, new deployments could prescribe to simply "append" some robots to a previously formed connected configuration. This happens, for instance, when there are less frontiers than robots, whose corresponding vertices in F^t are directly connected to the BS vertex b. Of course, such robots can become immediately ready once they arrive at their goal locations.

As soon as a robot becomes ready, it is available to receive a new plan from the BS (our asynchronicity notion is built upon this feature). Several options may be taken in consideration for triggering a new plan computation, e.g., replan when a fixed threshold θ of ready robots is reached, or as soon as a sufficiently interesting region is discovered. In the experiments described in the next sections, the effect of choosing different θ – i.e., of replanning as soon as at least θ robots become ready – is studied.

7 Simulation experiments

We choose the MRESim simulator (Spirin et al. 2013) since it focuses on communication and we select three environments of size 80 m × 60 m (represented by occupancy grids, whose cell edge length is 10 cm), shown in Fig. 4. Office and Open are from the Radish repository (Howard and Roy 2003) ("sdr_site_b" and "acapulco_convention_center", respectively), while Cluttered is from the MRESim repository ("grass"). We run simulations with teams of 6 and 12 TurtleBot-like robots moving at a constant speed of 0.4 m/s, and equipped with a depth camera with a maximum range of 5 m, a 60° FOV, and an angular resolution of 1°. For each experimental set, 5 runs of 900 s (steps) are executed for each environment, randomly varying the starting positions of the BS and robots. For the simulation experiments, we assume that the communication model robots are endowed with coincides with the actual possibility of communicating in the simulated world while ensuring enough bandwidth. Clearly, this assumption will not hold in the real-robots experiments, which will thus validate our method in realistic scenarios. In particular, the disk communication model is used with maximum distance 15 m for 6 robots (i.e., robots can communicate if within a given distance), and the limited-distance line-of-sight communication model with the same maximum distance for 12 robots. (A more relaxed communication model for teams of 6 robots is adopted to avoid to

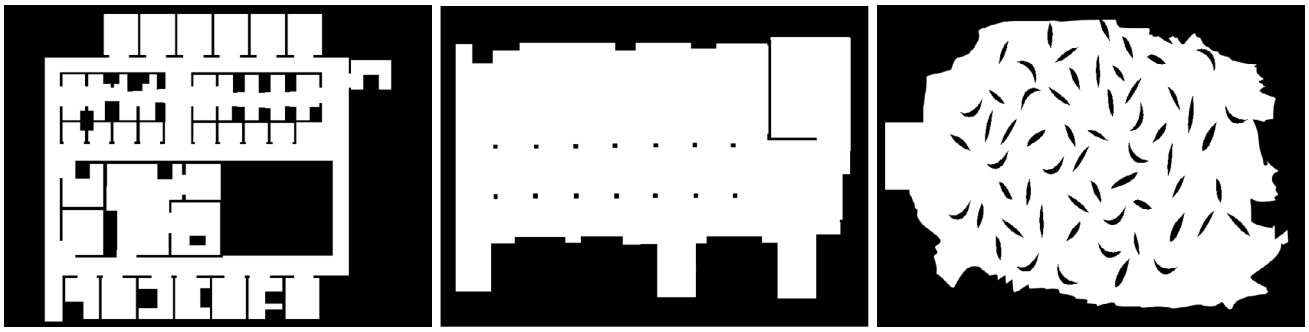


Fig. 4 Simulation environments, approximate size 80×60 m

excessively constrain the exploration process, especially in the Office and Cluttered environments.) On top of the BS global grid map, the exploration graph G^t is iteratively built by adding as vertices the locations representing a sufficiently big cluster of frontier cells, so that already visited frontiers become candidate relay locations for future plans. To evaluate the proposed strategies, three performance metrics are logged over time: the percentage of explored area, the average robots' traveled distance (that can be related to energy consumption), and the average time a robot is not in communication with the BS (an indicator of the situation awareness achieved at the BS). Figure 2 shows a snapshot of an experiment running in MRESim on the Office environment under the limited-distance line-of-sight communication model with 12 robots.

In a first set of experiments, we start by considering a synchronous setting ($\theta = 6$ and $\theta = 12$ for 6 and 12 robots, respectively) and study the performance obtained by the three proposed algorithms: the optimal one-stage ILP-based algorithm presented in Sect. 4 (1S-ILP) and the two-stage algorithms described in Sect. 5, where configurations are computed optimally (2S-ILP) or approximately (2S-APX).

The GUROBI solver (Gurobi 2015) is used for solving the ILPs on a laptop equipped with an i5-4310M processor and 8 GB of RAM as follows. Initially, the two ILP models contain all their constraints except for (3) and (11). Violations of such constraints are checked at each node of the Branch&Bound tree by means of the standard separation procedure involving the resolution of a max-flow problem in the underlying graph, combined with the usage of *nested cuts* to find more violated cuts at each node; see (Álvarez-Miranda et al. 2013) for details. The α parameter of 1S-ILP is set to 0.5 after some preliminary experiments, while 2S-APX is run with $\delta = 2$ for 6 robots and $\delta = 4$ for 12 robots to obtain a 3-approximation for both team sizes.

First, we show two different views for our results that we call *real time* and *instant replanning*. The real time view entails an exploration mission that unfolds in time just like it would be in reality with our implementation: both traveling and planning take some time. Instead, the instant replanning view filters out the time spent in computing plans. It shows how the explored area would grow under an ideal implementation achieving instantaneous plan computations. This second view evaluates the goodness of our plans independently of the efficiency of our implementation.

Figure 5 shows the results obtained in the Office and Cluttered environments with the team of 6 robots in terms of percentage of explored area as function of mission time. Bars correspond to the standard deviation. Results for Open environment are not reported here as they show a similar trend to those in Office and Cluttered. With real-time planning, Fig. 5a, c, the one-stage approach shows better performance than the two-stage ones in both environments, although not statistically significant. For instance, at the end of the mission in the Office environment the gap between 1S-ILP and 2S-APX has p-value $= 0.0575$ according to one-way ANOVA. The disadvantage of higher computational time required to solve a more complicated ILP is compensated by the much higher quality of the solution. Moreover, notice that the performance of 2S-ILP and 2S-APX are very similar. This can be explained by the fact that the ILP models obtained in such instances require just up to a few seconds to be optimally solved. Further, this shows that the approximation algorithm performs comparatively well against its optimal counterpart. Of course, the advantage in terms of explored area between the one-stage and the two-stage planning approaches is amplified when considering planning instantaneous (Fig. 5b, d), especially in the Cluttered environment.

Figure 6 shows the results obtained for 12 robots in the Office and Cluttered environments (again, Open environment presents a similar trend). When considering real-time planning (Fig. 6a, c), 1S-ILP seems to be only slightly better than the two-stage approaches; as such it is not clear whether 1S-ILP provides a significant advantage. Considering planning instantaneous, it is also less evident the gain in solution qual-

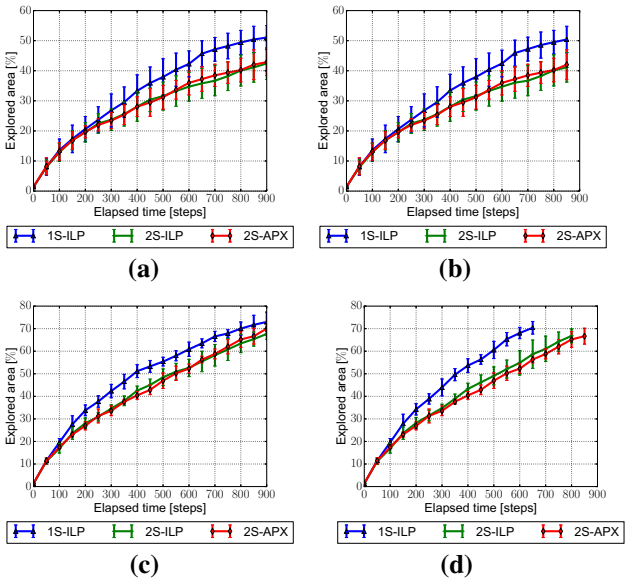


Fig. 5 Comparison of the planning approaches for 6 robots ($\theta = 6$) in the Office and Cluttered environments. **a** Office, real-time. **b** Office, instant replanning. **c** Cluttered, real-time. **d** Cluttered, instant replanning

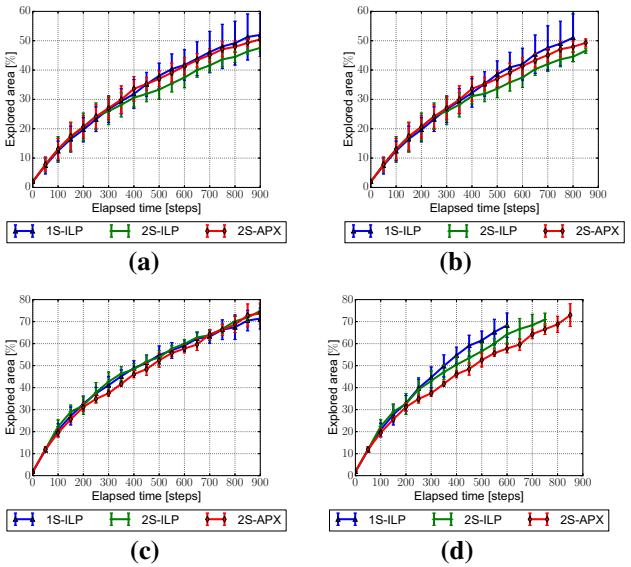


Fig. 6 Comparison of the planning approaches for 12 robots ($\theta = 12$) in the Office and Cluttered environments. **a** Office, real-time. **b** Office, instant replanning. **c** Cluttered, real-time. **d** Cluttered, instant replanning

ity when adopting 1S-ILP. A higher density of robots seems to be able to alleviate the penalty due to a lower solution quality of the deployment.

In a second set of experiments, we study the effect of choosing different θ when considering real-time planning, since the delays due to a high planning time might turn out to be the bottleneck of the system when introducing asynchronicity. In particular, our aim is to investigate (a) if 1S-ILP remains a practical approach to adopt, given that new plans need to be computed much more frequently when $\theta < m$, and (b) which replanning threshold θ should be adopted in the different environments. Since 2S-ILP and 2S-APX have been shown to offer comparable performance, only 2S-APX is tested as the two-stage approach, in order to have the guarantee of polynomial worst-case running time. In particular, we again set $\delta = 2$ for 6 robots and $\delta = 4$ for 12 robots to obtain a 3-approximation for both the team sizes, regardless of the particular replanning threshold chosen. Note that this means that there are cases in which the algorithm returns an optimal solution.

To provide a reference comparison, we evaluate the proposed approach against the one presented by Spirin et al. (2013). This method, that we label as *Utility*, represents a communication-aware exploration strategy where exploration of new areas can be efficiently traded-off with communication requirements. We compare against it since it has been shown to complete exploration faster than the role-based strategy of de Hoog et al. (2009) (besides, the strategy is already available in MRESim). In short (see also Sect. 2), Utility defines a distributed strategy where each robot chooses autonomously which frontier to explore in a greedy fashion and returns to the BS as soon as the ratio between the area supposed to be known at the BS and that known by the robot goes below a predefined value $0 \leq r \leq 1$. In our experiments, we set $r = 0.5$ to obtain a balanced, yet more exploration-prone, behavior. Note that this strategy does not embed the recurrent connectivity constraint, so robots can remain disconnected from the BS for an unpredictably long amount of time and are thus expected to explore more freely.

Figure 7 shows the percentage of explored area known at the BS for 6 robots in the three environments as the mission unfolds for 1S-ILP with different θ and for Utility. A low replanning threshold seems to provide a very limited advantage in the first phase of exploration. However, towards the end, replanning once for half (in Office) or whole (in Open and Cluttered environments) team offers better performance. Nevertheless, the difference is not statistically significant. For example in the Open environment, p-value $= 0.1072$. This is due to the increasingly high planning time required to solve the model to optimality, which leads to a congestion in the computation of new plans. Comparing the performance of 1S-ILP with 2S-APX (Fig. 8), a statistically significant improvement can be observed at the end of the mission only in the Cluttered environment when selecting $\theta = 1$ (p-value $= 0.0019$). However, notice that the lower planning times required by 2S-APX are directly reflected in a trend of statistically significant better performance for decreasing values of θ in all the environments (for instance, for $\theta = 1$ and $\theta = 6$ in Cluttered, p-value $= 0.003$). Comparing our recurrent connectivity approach with the Utility method, it

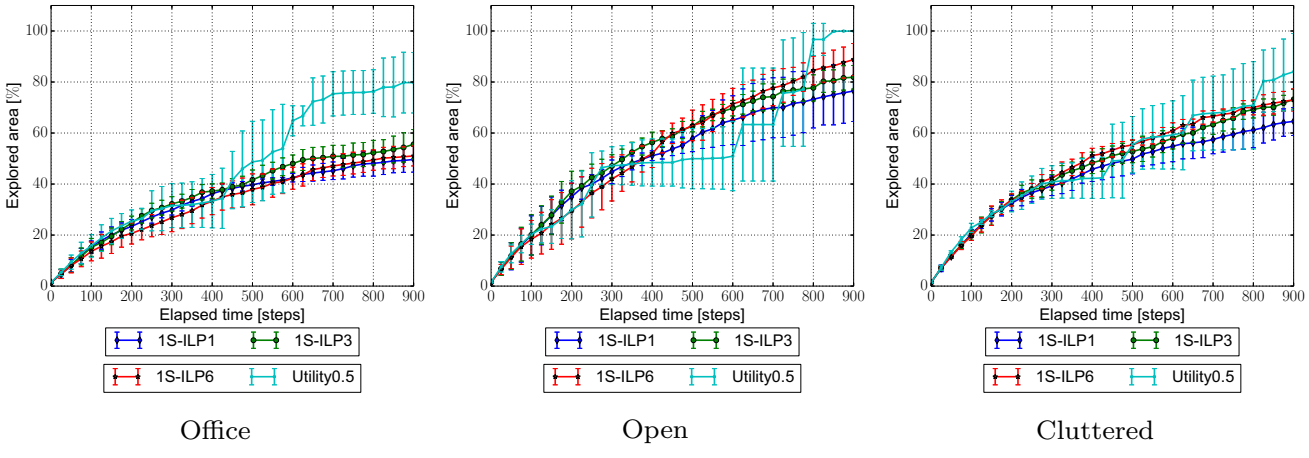


Fig. 7 Explored area for 6 robots: 1S-ILP with different replanning thresholds ($\theta = 1, 3, 6$) and Utility method with $r = 0.5$ (Spirin et al. 2013)

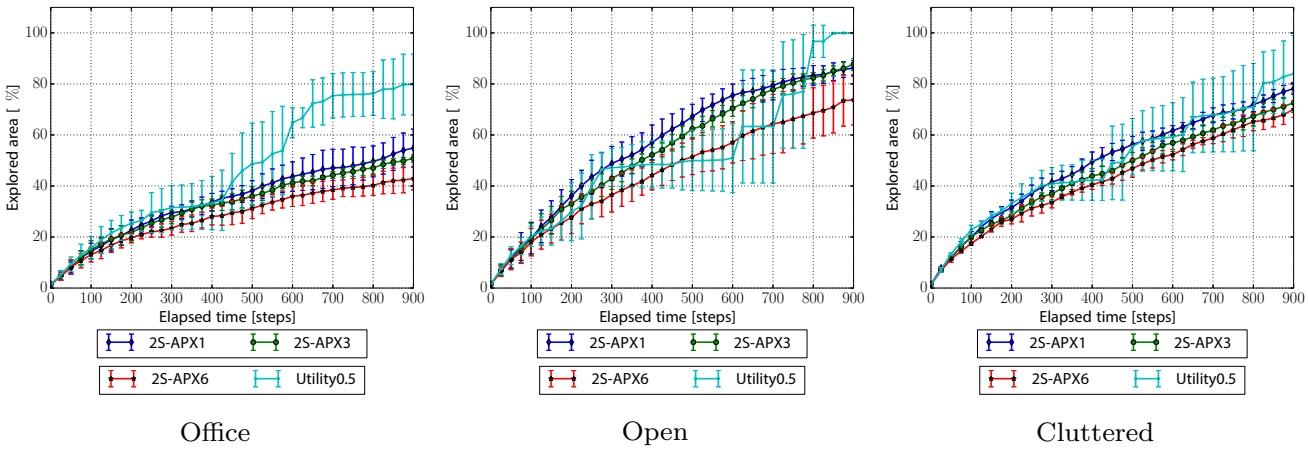


Fig. 8 Explored area for 6 robots: 2S-APX with different replanning thresholds ($\theta = 1, 3, 6$) and Utility method with $r = 0.5$ (Spirin et al. 2013)

can be observed that, in general, the latter offers better performance in terms of explored area, but with a much lower situational awareness at the BS and a much higher energy consumption; see the results of the Cluttered environment in Fig. 9; the results for the other two environments exhibit similar trends. Figure 10 shows some snapshots of a simulation in the Office environment with 2S-APX and $\theta = 1$.

We now examine the performance with a team of 12 robots. Figure 11 shows the percentage of explored area known at the BS in the three environments for 2S-APX and Utility. As in the previous case, a low replanning threshold provides a statistically significant advantage in terms of explored area in all the environments; for instance, in Office, 2S-APX1 versus 2S-APX12 has a p-value $= 0.0035$.

Again, compared to Utility, our approach allows a better situational awareness at the BS, while sacrificing some performance in terms of explored area (actually, less than in the previous case). The results for 1S-ILP, partially reported in Fig. 12, confirm the previous insights. In particular, notice how $\theta = 1$, that yields the best results in terms of explored area with 2S-APX, now results in a heavy congestion in the computation of new plans. The situational awareness at the BS and the distance traveled, whose graphs are not reported here, show the same trends as in Fig. 9 for 6 robots.

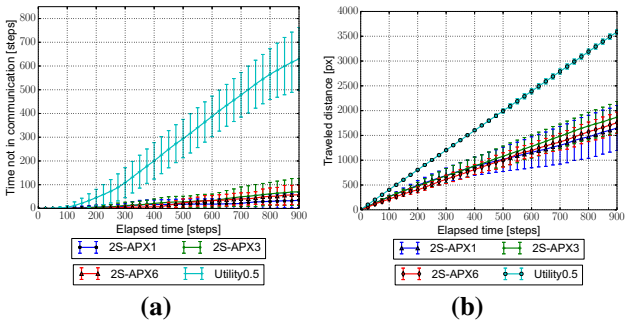


Fig. 9 Time not in communication with the BS (**a**) and traveled distance (**b**) for 6 robots in Cluttered: 2S-APX with different replanning thresholds ($\theta = 1, 3, 6$) and Utility method (Spirin et al. 2013)

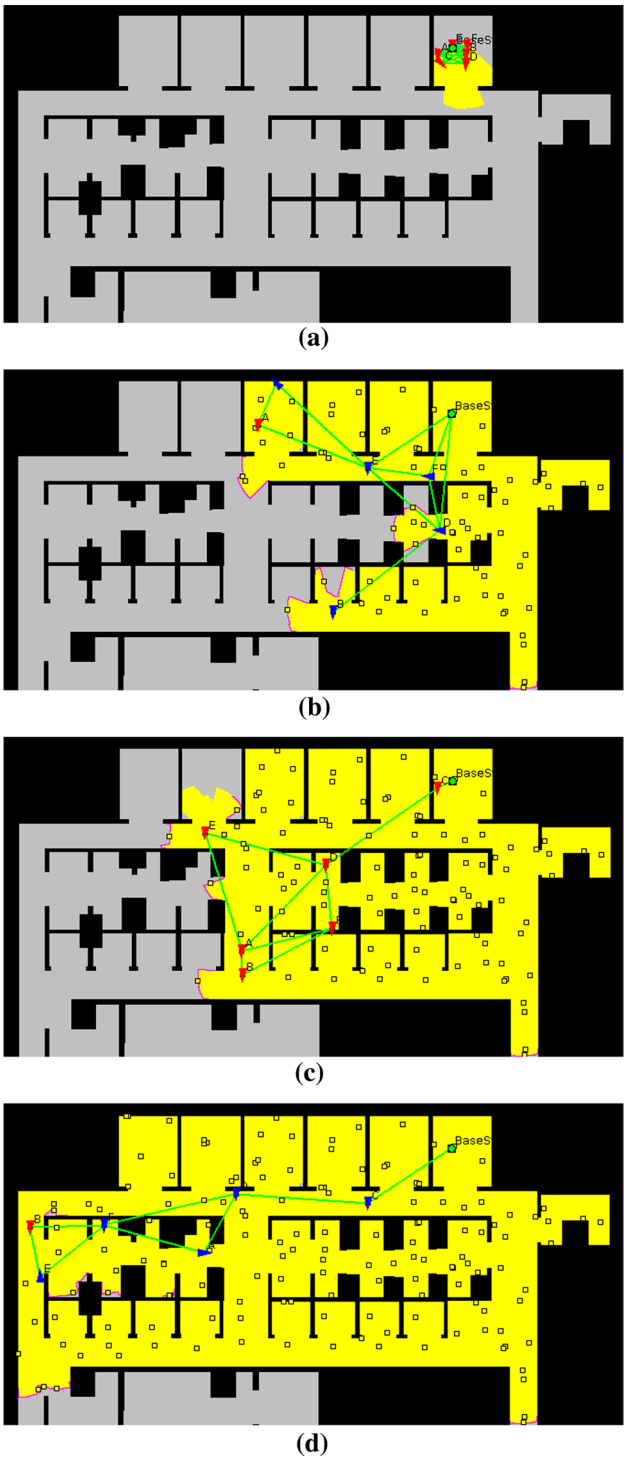


Fig. 10 Exploration example (6 robots, 2S-APX, $\theta = 1$, Office environment). **a** All the robots are initially close to the BS and in the ready status (*red*). **b** The exploration proceeds along several directions in parallel. (c) After some time, the robots focus on mapping the *left* part of the environment. All the robots become ready at once due to the fact that the robot closest to the BS, required to connect to it with all the other robots, is the last to reach its goal. **d** Once large part of the environment has been mapped, and the mission approaches its deadline, the graph link topology starts to resemble a tree (Color figure online)

8 Real-world experiments

The proposed methods are also validated in a real scenario, with a laptop (the BS) and 6 TurtleBot 2 mobile robots,[1] each one equipped with a netbook and a Microsoft Kinect (see 1). An *ad hoc* network is set through the WiFi interfaces of the computers and the Optimized Link State Routing Protocol is run to manage a multihop network (Clausen and Jacquet 2003). ROS (Quigley et al. 2009; O'Kane 2013) is used to control the multirobot system. In such a setup, robots are able to communicate only if the signal quality is good enough. It might be possible that a communication edge in the communication graph does not necessarily translate into an actual communication link. However, in the experiments, given the conservative way in which we build the communication graph (explained later), robots are able to communicate when expected. We use the "nav2d" stack[2] for multirobot (graph-based) SLAM and path planning. The exploration strategy is implemented as a plugin of the "nav2d_exploration" package.

The Swearingen Engineering Center at the University of South Carolina is used as testing ground for the proposed approach; Fig. 13 shows one of the floors where robots have been deployed. The size of the portion of the environment is approximately 90 m $\times$ 65 m. There are two long corridors, with some other intersecting corridors/halls, and a looping corridor, thus requiring the robots to form a chain to guarantee communication with the BS. After some preliminary tests, we assumed that two locations can communicate if they are within 15 m range. Experiments involved running 1S-ILP and 2S-APX with $\theta = 1$ and $\theta = 6$. For $\theta = 6$, initially two coalitions of robots are formed, each one going towards the frontiers of the corridors. Figure 14 shows the partial map known by the BS (and the robots) at some time step using 1S-ILP, with $\theta = 6$. It is possible to observe that some noise is introduced, due to perception and motion errors. One immediate consequence is that, in order to let the exploration proceed, a relaxed version of the communication model has to be adopted: given the current known map, two locations can communicate if they are in line of sight, with some tolerance. Without this relaxation, many goal locations for the robots could not be connected, resulting in a premature end of the exploration task. Such a relaxation is realistic, as WiFi signal can be received even in presence of some obstacles that obstruct the view.

When the robots in one branch of the corridor cannot proceed further without violating the communication constraint, they become ready and are allocated to the other branch of the

[1] http://www.turtlebot.com

[2] http://wiki.ros.org/nav2d

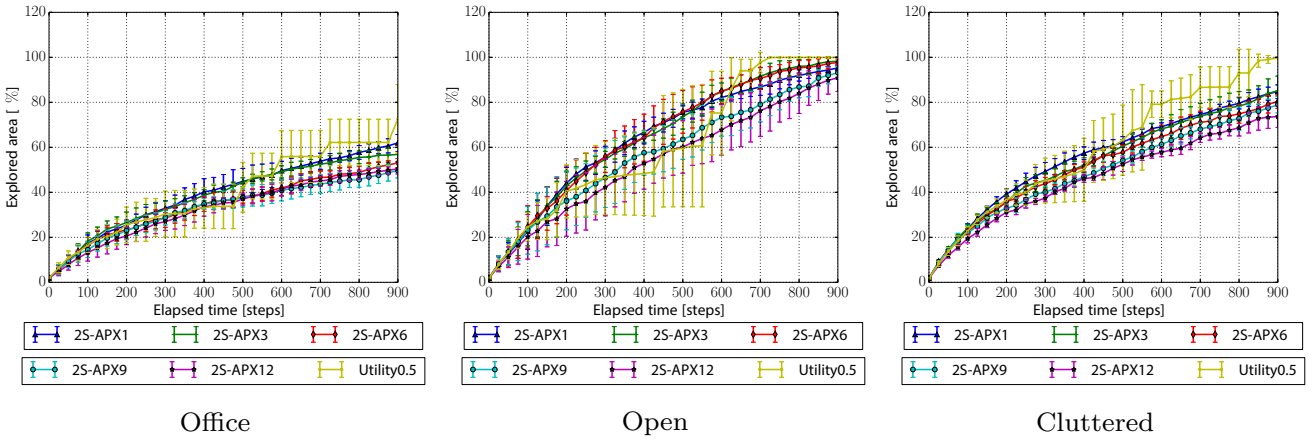


Fig. 11 Explored area for 12 robots: 2S-APX with different replanning thresholds ($\theta = 1, 3, 6, 9, 12$) and Utility method with $r = 0.5$ (Spirin et al. 2013)

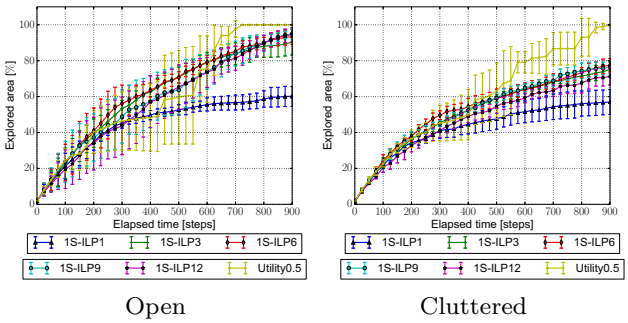


Fig. 12 Explored area for 12 robots: 1S-ILP with different replanning thresholds ($\theta = 1, 3, 6, 9, 12$) and Utility method (Spirin et al. 2013)

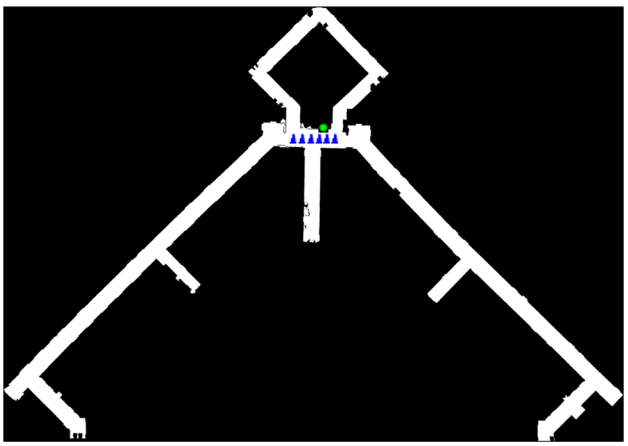


Fig. 13 Part of the third floor at the Swearingen Engineering Center for some of the experiments with real robots (*blue*) and base station (*green*), whose initial poses are indicated on the map (Color figure online)

corridor to form a longer chain. Note that, with real robots, recovery mechanisms are necessary. For instance, some of the robots could not actually move to the assigned goal because of the noise in the map. In such cases, the robot aborts the motion to the location, notifying the BS. To guarantee that

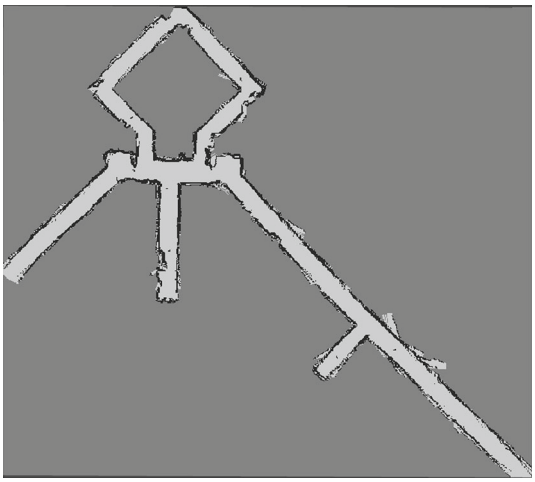


Fig. 14 Partial map built by 6 TurtleBots and known by the BS using 1S-ILP, $\theta = 6$

the communication topology always contains a tree, all the robots belonging to the branches to which the aborted robot belongs are preempted and become ready for a new assignment. Figure 15 shows the trend of the explored area during the mission. After some time, we observe that the robots basically do not move. This can be explained by two facts. One is that the BS is computing a new plan for the robots. The second cause is that, with a noisy map, several frontiers could be generated in non-reachable areas, leading to a sequence of plan re-computations. In this case, a proper filtering of the frontiers should be adopted in such a way that planning time, that is typically the bottleneck, is not wasted. It is possible to observe that 1S-ILP6 is the strategy that outperforms the rest. Compared to 2S-APX, the quality of the solution found by 1S-ILP is higher, confirming the insights from the simulation results, presented in the previous section. In contrast to the simulation results, setting $\theta = 1$, while initially provides some benefits, eventually worsens

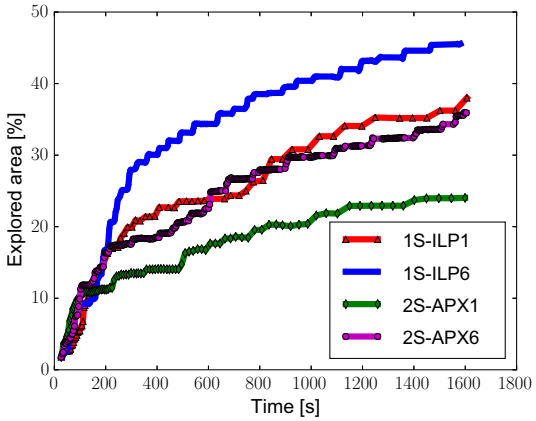


Fig. 15 Area explored by the real TurtleBot 2 robots

the performance. With frequent replanning, robots tend to interfere in the motion of each other: this possibly results in collisions that can degrade the quality of the map, leading to situations in which many locations are not valid. Notice that the bandwidth is not an issue during the experiments, as robots and base station are not sharing very intensive data.

9 Discussion of the experimental results

In this section, we make some general observations from the results obtained in simulations and in real-robots experiments. In general, we cannot say that one approach performs better than another. However, we think that the following key points will be of help in guiding a practitioner in deploying a multirobot system for exploring an initially unknown environment under recurrent connectivity constraints.

1. ILP-based algorithms are (relatively) fast, but sometimes not enough for real-world settings In all the experiments, we are always able to solve the ILP models of the one-stage and two-stage approaches to optimality in a few seconds on average, rarely reaching 1 minute or more of computation (this usually happens towards the end of the mission, when the underlying graph is large and the number of variables and constraints is large, too). For 6 robots, this computational disadvantage is compensated by the higher quality of the computed plans. However, for larger team sizes, an ILP-based algorithm could turn out to be the bottleneck of the system when replanning happens frequently (see, e.g., the performance of 1S-ILP for small values of θ in Fig. 12). This could also happen when exploring very large environments.

2. Optimality of configurations does not necessarily imply best performance Due to the online nature of the exploration problem, it is not surprising that locally optimal decisions that return the best possible configurations (as defined in the one-stage approach) are not necessarily translated into a globally optimal performance. For example, in Fig. 6c, the curves of 1S-ILP, 2S-ILP, and 2S-APX are almost overlapped. Moreover, from what observed before, the usage of the one-stage approach does not remain a viable option for very large teams when a small replanning threshold is adopted.

3. The choice of the replanning threshold should depend on the team size and on the chosen approach From our experiments both in simulation and on real robots, it seems that a high replanning threshold θ should be preferred for small team sizes using the one-stage approach. This might be due to the fact that, with few robots, the system has less chances to "recover" from an erroneous myopic decision based on partial information (that might have taken some time to be produced).

4. Real-world implementations must be accompanied by backup procedures, whose activation negatively impacts on the performance In general, exploration with real robots proceeds at a much slower pace compared to the one in simulation (which is error-free). In our current implementation, when one (or more) robot(s) cannot reach the assigned goal location(s), all the robots depending on it (them) are preempted, and a new configuration is computed. This introduces a non-negligible delay with respect to an error-free setting, which seems to be minimized when replanning for the whole team, i.e., when $\theta = 6$. The reason is that, with $\theta = 1$, robots tend to interfere more frequently in the motion of each other.

10 Conclusions

Recurrent connectivity is a way for introducing communication constraints in multirobot exploration missions by requiring that robots must communicate with a base station whenever they reach goal locations from which new knowledge can be acquired. In this work, we defined and extensively evaluated two planning techniques for efficient exploration under recurrent connectivity constraints. First, we proposed a single-stage strategy based on an ILP whose objective function accounts for both exploration costs and information gains. Then, we defined a two-stage strategy, separating the problem of locations selection from that of the robot-location assignments. The experimental analysis showed that the proposed methods can be effectively applied to multirobot exploration missions, because they provide performance comparable with that of a state-of-the-art method that leaves more freedom to explore.

Future work will improve the selection of the locations for the graph (for example, in an indoor environment, it would be possible to use the Generalized Voronoi Graph (Zhang et al. 2014)), and fast detection of frontiers, building on the work of (Keidar and Kaminka 2014). Furthermore, our method could be extended to allocations that consider shared resources such as space (Nam and Shell 2015). Currently,

we are also investigating the possibility of adopting a more speculative link-detection mechanism to build the set of communication edges, like resorting to a dedicated subteam of robots to build a "communication map" by means of an online learning method such as Gaussian Processes (Banfi et al. 2017), which are particularly suitable for learning spatial phenomena. Finally, this development would also entail the need for backup plans to be used in the presence of false-positives in predicted links.

References

Álvarez-Miranda, E., Ljubić, I., & Mutzel, P. (2013). The rooted maximum node-weight connected subgraph problem. In *Proceedings of CPAIOR* (pp. 300–315). Springer.

Arkin, R., Diaz, J. (2002). Line-of-sight constrained exploration for reactive multiagent robotic teams. In *Proceedings of AMC*, (pp. 455–461).

Banfi, J., Quattrini, Li, A., Basilico, N., Amigoni, F. (2015). Communication-constrained multirobot exploration: Short taxonomy and comparative results. In *IROS Workshop on On-line decision-making in multi-robot coordination*.

Banfi, J., Quattrini Li, A., Basilico, N., Amigoni, F., Rekleitis, I. (2016). Asynchronous multirobot exploration under recurrent connectivity constraints. In *Proceedings of ICRA*, pp. 5491–5498.

Banfi, J., Quattrini Li, A., Basilico, N., Amigoni, F., Rekleitis, I. (2017). Multirobot online construction of communication maps. In *Proceedings of ICRA*, pp. 2577–2583.

Basilico, N., & Amigoni, F. (2011). Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. *Autonomous Robots*, *31*(4), 401–417.

Burkard, R., Dell'Amico, M., & Martello, S. (2009). *Assignment problems* (pp. 79–87). Society for Industrial and Applied Mathematics.

Chekuri, C., Korula, N., & Pál, M. (2012). Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms*, *8*(3), 23.

Cheng, X., Du, D. Z., Wang, L., & Xu, B. (2008). Relay sensor placement in wireless sensor networks. *Wireless Networks*, *14*(3), 347–355.

Clausen, T., & Jacquet, P. (2003). *Optimized link state routing protocol (OLSR)* (p. 3626). RFC: Technical Report.

Garey, M., & Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman.

Gurobi (2015). Gurobi optimizer reference manual. http://www.gurobi.com.

Hochbaum, D., Pathria, A. (1994). Node-optimal connected k-subgraphs. Manuscript, UC Berkeley, 1994. http://goo.gl/QoB8IB.

Hollinger, G., & Singh, S. (2012). Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE Transactions on Robotics*, *28*(4), 967–973.

de Hoog, J., Cameron, S., Visser, A. (2009). Role-based autonomous multi-robot exploration. In *Proceedings of COGNITIVE*, pp. 482–487.

Howard, A., Roy, N. (2003). The robotics data set repository (Radish). http://radish.sourceforge.net/.

Howard, A., Matarić, M., & Sukhatme, G. (2002). An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robotots*, *13*(2), 113–126.

Jensen, EA., Lowmanstone, L., Gini, M. (2016). Communication-restricted exploration for search teams. In *Proceedings of DARS*, (to appear).

Johnson, D. S., Minkoff, M., & Phillips, S. (2000). The prize collecting seiner tree problem: Theory and practice. *Proceedings of SODA*, *1*, 760–769.

Julia, M., Gil, A., & Reinoso, Ó. (2012). A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, *33*(4), 427–444.

Keidar, M., & Kaminka, G. A. (2014). Efficient frontier detection for robot exploration. *The International Journal of Robotics Research*, *33*(2), 215–236.

Kimelfeld, B., Sagiv, Y. (2006). New algorithms for computing Steiner trees for a fixed number of terminals. http://researcher.ibm.com/researcher/files/us-kimelfeld/papers-steiner06.pdf

Lee, H. F., & Dooly, D. R. (1996). Algorithms for the constrained maximum-weight connected graph problem. *Naval Research Logistics*, *43*(7), 985–1008.

Moss, A., & Rabani, Y. (2007). Approximation algorithms for constrained node weighted Steiner tree problems. *SIAM Journal on Computing*, *37*(2), 460–481.

Mukhija, P., Krishna, K., & Krishna, V. (2010). A two phase recursive tree propagation based multi-robotic exploration framework with fixed base station constraint. In *Proceedings of IROS*, pp. 4806–4811.

Nam, C., & Shell, D. A. (2015). Assignment algorithms for modeling resource contention in multirobot task allocation. *IEEE Transactions on Automation Science and Engineering*, *12*(3), 889–900.

Ochoa, S., & Santos, R. (2015). Human-centric wireless sensor networks to improve information availability during urban search and rescue activities. *Information Fusion*, *22*, 71–84.

O'Kane, J.M. (2013). A Gentle Introduction to ROS. Independently published, available at http://www.cse.sc.edu/~jokane/agitr/.

Pei, Y., Mutka, M., & Xi, N. (2013). Connectivity and bandwidth-aware real-time exploration in mobile robot networks. *Wireless Communications and Mobile Computing*, *13*(9), 847–863.

Quattrini Li, A., Cipolleschi, R., Giusto, M., & Amigoni, F. (2016). A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings. *Autonomous Robots*, *40*(4), 581–597.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. (2009). ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Reich, J., Misra, V., Rubenstein, D., & Zussman, G. (2012). Connectivity maintenance in mobile wireless networks via constrained mobility. *IEEE Journal of Selected Area in Communications*, *30*(5), 935–950.

Rekleitis, I .(2013). Multi-robot simultaneous localization and uncertainty reduction on maps (MR-SLURM). In *Proceedings of ROBIO*, pp .1216–1221.

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

Simmons, R. G., Apfelbaum, D., Burgard, W., Fox ,D., Moors, M., Thrun, S., & Younes, H. L. S .(2000). Coordination for multi-robot exploration and mapping. In *Proceedings of AAAI*, pp. 852–858.

Spirin, V., Cameron. S., & de Hoog, J. (2013). Time preference for information in multiagent exploration with limited communication. In *Proceedings of TAROS*, pp. 34–45.

Stump, E., Michal, N., Kumar, V., & Isler, V. (2011). Visibility-based deployment of robot formations for communication maintenance. In *Proceedings of ICRA*, pp. 4498–4505.

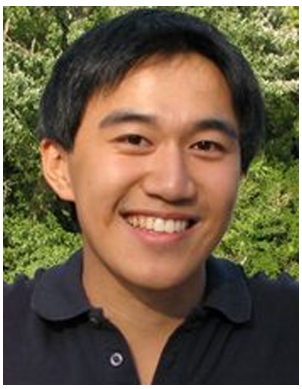
Tadokoro, S. (2010). *Rescue Robotics*. Springer.

Thrun, S. (2002). Robotic mapping: A survey. In: Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, pp. 1–35.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. The MIT Press.

Tuna, G., Gulez, K., & Gungor, V. (2013). The effects of exploration strategies and communication models on the performance of cooperation exploration. *Ad Hoc Networks*, *11*(7), 1931–1941.
Visser, A., & Slamet, B. (2008). Including communication success in the estimation of information gain for multi-robot exploration. In *Proceedings of WiOpt*, pp. 680–687.
Wurm, K. M., Stachniss, C., & Burgard, W. (2008). Coordinated multi-robot exploration using a segmentation of the environment. In *Proceedings of IROS*, pp. 1160–1165.
Yamauchi, B. (1998) Frontier-based exploration using multiple robots. In *Proceedings of AGENTS*, pp. 47–53.
Yu, J. (2016). Intractability of optimal multirobot path planning on planar graphs. *IEEE RA-L*, *1*(1), 33–40.
Zhang, Q., Whitney, D., Shkurti, F., & Rekleitis, I. (2014). Ear-based exploration on hybrid metric/topological maps. In Proceedings of IROS, pp 3081–3088.

Jacopo Banfi received a B.Sc. and a M.Sc. degree in Computer Science in 2011 and 2014, respectively, from the Politecnico di Milano (Italy). From February 2013 to April 2014, he worked at his M.Sc. thesis at the Dalle Molle Institute for Artificial Intelligence Studies (Manno, Switzerland). Since 2014, he is a Ph.D. student in Information Technology at the Politecnico di Milano. His research interests lie in the field of cooperative multi-robot systems. In particular, his thesis is devoted to the study of navigation strategies for cooperative tasks (like exploration and patrolling) in communication-restricted environments.



Alberto Quattrini Li After being a postdoctoral fellow in the Autonomous Field Robotics Laboratory (AFRL) at the Computer Science & Engineering Department of the University of South Carolina, Alberto is currently working as research assistant professor in the same department. He received a M.Sc. in Computer Science Engineering (2011) and a Ph.D. in Information Technology (2015) from Politecnico di Milano. From February to July 2014, he was a visiting Ph.D. student at Research on Sensor Networks Lab at the Computer Science Department of the University of Minnesota. His main research interests include autonomous mobile robotics and underwater robotics, dealing with problems that span from multirobot exploration and coverage to visual-based state estimation.

Ioannis Rekleitis Currently an Assistant Professor at the Computer Science and Engineering Department, University of South Carolina, and an Adjunct Professor at the School of Computer Science, McGill University. In 2004–2007 he was a visiting fellow at the Canadian Space Agency working on Planetary exploration and On-Orbit-Servicing of Satellites. During 2004 he was at McGill University as a Research Associate in the Centre for Intelligent Machines with Professor Gregory Dudek in the Mobile Robotics Lab (MRL). Between 2002 and 2003, he was a Postdoctoral Fellow at the Carnegie Mellon University in the Sensor Based Planning Lab with Professor Howie Choset. He was granted his Ph.D. from the School of Computer Science, McGill University, Montreal, Quebec, Canada in 2003. His Research has focused on mobile robotics and in particular in the area of cooperating intelligent agents with application to multi-robot cooperative localization, mapping, exploration and coverage. He has done extensive work on space, aerial, ground, surface, and underwater robots, with more than 60 journal and conference articles. His interests extend to computer vision and sensor networks.

Francesco Amigoni was born in Soresina (CR), Italy, on December 2, 1971. He got the Laurea degree in Computer Engineering from the Politecnico di Milano in 1996 and the Ph.D. degree in Computer Engineering and Automatica from the Politecnico di Milano in 2000. From December 1999 to September 2000 he has been a visiting scholar at the Computer Science Department of the Stanford University (USA). From February 2002 to April 2007 he has been an assistant professor and from May 2007 he is an associate professor at the Dipartimento di Elettronica, Informazione e Bioingegneria of the Politecnico di Milano. His main research interests include: agents and multiagent systems, autonomous mobile robotics, and the philosophical aspects of artificial intelligence. He is author of about 110 papers published on international journals, books, and conference proceedings.

Nicola Basilico received a M.Sc. degree in Computer Science and Engineering in 2007 and a Ph.D. in Information Technology in 2011 from Politecnico di Milano (Italy). In 2011 and 2012 he has been a postdoctoral scholar at the Robotics Laboratory at the University of California, Merced. In 2013 he worked as a research assistant at the Swiss AI lab IDSIA. Since 2014, he is an Assistant Professor at the deparment of Computer Science of the University of Milan (Italy). His main research interests develop in the Artificial Intelligence area with particular focus on Multi-Agent Systems and Autonomous Robotics.