2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Asynchronous Multirobot Exploration under Recurrent Connectivity Constraints

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

Abstract— **In multirobot exploration under centralized control, communication plays an important role in constraining the team exploration strategy. Recurrent connectivity is a way to define communication constraints for which robots must connect to a base station only when making new observations. This paper studies effective multirobot exploration strategies under recurrent connectivity by considering a centralized and asynchronous planning framework. We formalize the problem of selecting the optimal set of locations robots should reach, provide an exact formulation to solve it, and devise an approximation algorithm to obtain efficient solutions with a bounded loss of optimality. Experiments in simulation and on real robots evaluate our approach in a number of settings.**

I. Introduction

Research on multirobot exploration showed how communication constraints represent a critical issue in a number of real-life applications. This holds true especially when exploration is performed in disaster mitigation domains [1], where centralized situational awareness is often required for the effective supervision of the mission. 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 a base station. The literature proposes multirobot exploration strategies that take into account different types of communication constraints [2]–[5]. Among them, *recurrent connectivity* is a method that can be applied to ensure situation awareness without excessively constraining the exploration task. In this method, robots are required to be connected each time they gather new information. Under such a scheme, the constraints can be thought in an *online* fashion, allowing robots to be disconnected for arbitrarily long periods, provided that they are able to report to a base station as soon as new information is collected.

This work addresses the problem of multirobot exploration under recurrent connectivity in an asynchronous fashion, meaning that new plans can be submitted to arbitrary groups of robots as soon as they become *ready*. Our approach leverages a variant of the Steiner tree problem (i.e., given a subset of vertices, find a minimum-cost subtree that connects them [6]) that appears as a particular case of different known graph optimization problems, and that, to the best of our knowledge, has never been adopted for this kind of settings. In particular, we formulate the problem of selecting the optimal set of locations robots should explore, propose an exact Integer Linear Programming (ILP) formulation, and discuss its applicability scope. To overcome the limits of such an exact formulation, we define a novel approximation algorithm that introduces significant quality guarantees in the robotic scenario considered. Then, given the best set of connected locations, the most efficient robot assignment is easily computed by minimizing the cumulative travel distance. Simulation results validate our method against a state-of-the-art technique. Finally, we validate the proposed approach using a team of Turtlebot2 robots; see Fig. 1.

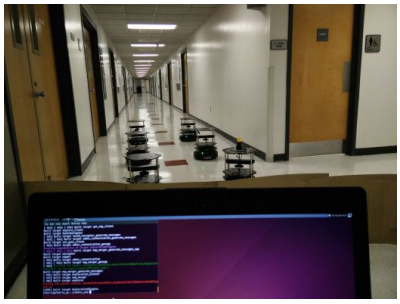


Fig. 1: An experimental setup used in this paper.

The paper is structured as follows. Section II provides a short review of communication-constrained multirobot exploration. Section III formalizes our multirobot exploration framework. Section IV describes our solution methods for finding optimal connected configurations while Section V addresses the optimal robot deployment. Section VI discusses the concept of robot readiness. Section VII presents our experimental results and Section VIII concludes the paper.

II. Related Work

Exploration is the task of building a complete *map* representing the structure of an environment. A number of works in the literature addressed the problem of multirobot exploration under communication constraints by focusing on *continuous* connectivity (direct or multi-hop) to a Base Station (BS). This conveniently applies to situations where two-way communication with each robot must be continuously maintained, e.g., in applications like teleoperated search with continuous real-time image streaming. To this aim, [2] proposes a local search algorithm, while [7] relies on a depth-first procedure to build the skeleton of the connected network. Clearly, guaranteeing continuous connection can introduce non-negligible costs for the exploration performance.

J. Banfi and F. Amigoni are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy {jacopo.banfi,francesco.amigoni}@polimi.it.

A. Quattrini Li and I. Rekleitis are with the Deparment of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA {albertoq,yiannisr}@cse.sc.edu.

N. Basilico is with the Department of Computer Science, University of Milan, Milano, Italy nicola.basilico@unimi.it.

Therefore, if such requirement is not strictly needed, a more flexible approach should be selected.

Another class of approaches addresses central situation awareness by considering *periodic* connectivity. Robots are allowed to temporarily disconnect from the BS to explore autonomously, but are required to regularly regain connection. In particular, some works cast periodic connectivity as a "soft" requirement, trying to comply with it under a best-effort perspective [4], [8]. Here, the periodic reconnection is distributedly obtained, with no strict guarantees, as a collective emerging behavior. Other solutions embrace a "hard" constraining approach. One significant work falling in this category is [9], where regaining connectivity under a strict frequency constraint is addressed by resorting to a scalable local search algorithm.

The above works model the communication constraint in an offline fashion, meaning that the communication requirements are fixed and mostly do not depend on how exploration unfolds. A different method taking an online view is that of *recurrent* connectivity, where robots need to be connected only at their deployment positions, namely at the locations where they gather new information. This method can provide up-to-date situational awareness to the BS without excessively constraining robots in their movements. Indeed, robots are allowed to disconnect for an arbitrarily long time, while reaching a desired deployment. Solutions based on this method are proposed in [3], [5], [10]. In particular, the problem setting addressed in [5] 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 unit disk communication model, and 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, relay placement is achieved by solving variations of the *Steiner minimum tree problem with minimum number of Steiner points and bounded edge length* [11]. However, this choice is intimately related to the adoption of the unit disk communication model under a synchronous planning setting. The contributions of our paper lie on a complementary direction: bandwidth constraints are not considered, but a method is devised not depending on a specific communication model and not requiring synchronous coordination among robots.

III. Exploration Model

A two-dimensional, continuous, and bounded environment $Env \subset \mathbb{R}^2$ is considered, whose points belong either to the free space, Env_f, or to obstacles of any shape, Env_o. A supervising facility termed Base Station (BS) is present in Env at a known location. BS will act as a central planning entity supervising the mission and computing new plans for a team of m robots $R = \{r_1, r_2, \ldots, r_m\}$. Each mobile robot is equipped with finite-range sensors to perceive the surrounding free space and outer boundaries of obstacles (e.g., a laser range scanner or a RGB-D sensor) and is capable of exchanging data with other robots or with the BS over an *ad hoc* network. Following a standard approach, we work on a graph-based representation of the environment $G = (V, C)$ where vertices in V encode some discretization of Env. Each vertex $v \in V$ is associated with a location, and BS vertex is marked as b. To each pair (i, j) of vertices, a value d_{ij} is associated, representing (an estimate of) the geodesic distance between them. Finally, the edge set C encodes the availability of communication links between pairs of vertices. In particular, we assume that, if the BS and one or more robots form a connected component in C, then any of such robots can exchange data with the BS under some protocol. (Experimental verification using real robots proves the validity of such an assumption.)

The exploration mission develops as follows. Robots explore and discover the environment in an incremental way, by taking local perceptions and repeatedly transmitting the collected data to the BS where a global map of the currently explored space is maintained. Plans are dictated by the BS and happen at discrete stages $t = 1, \ldots, T$, where T denotes the time of the last plan of the mission. We denote with $G^t = (V^t, C^t)$ the portion of G known by the BS at stage t. This graph can be defined as a subgraph of G whose structure depends on the exploration mission up to the present time. In particular, set V^t is obtained from the perceptions robots have taken and reported to the BS. A set $F^t \subseteq V^t$ denotes the *frontiers*, that is, vertices corresponding to locations of Env_f lying on the boundary between explored and unexplored portions of Env. Finally, set $C^t \in C$ is determined by some link-detection mechanism in charge of recognizing the availability of a communication link between any two known vertices. Link-detection methods range from simple visibility-based criteria to more sophisticated approaches where the signal decay through obstacles is modeled. The standard assumption [3], [5], [9] that network C is static and that the link-detection mechanism is not affected by false positives is used; see Section VII for more details.

At any planning stage t a new vector $\langle p_1^t, p_2^t, \ldots, p_m^t \rangle$ (where $p_r^t \in V^t$) is computed, denoting the goal vertices each robot r is headed to. More specifically, a plan is formally represented by a *configuration* Q^t and a *deployment* $\pi^t(\cdot)$. A configuration is the subset of vertices $Q^t \subseteq V^t$ containing m vertices to be occupied: $Q^t = \{q_1^t, \ldots, q_m^t\}$. A deployment specifies an assignment of each robot to one of the vertices, where $\pi^t(r) \in Q^t$. When a plan is computed the assignment $p_r^t = \pi^t(r)$ is performed and the robot r is instructed to reach its goal vertex p_r^t. Once reached, 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. Note that the robots flush to the BS also the sensing data acquired while going to the assigned locations. Moreover, we characterize a robot r as *ready* if (a) it has reached its goal vertex p_r^t and, in case such a vertex is a frontier, it has completed sensor measurements, and (b) it has transmitted to the BS its new perception data and no other robot still requires it as a relay. We denote with R^t the set of ready

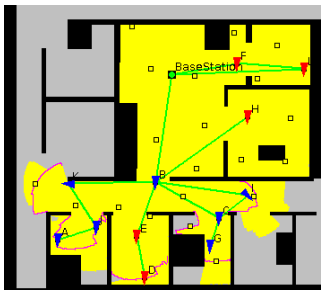


Fig. 2: Exploration snapshot. Blue and red: not ready and ready robots. Black-edge squares: vertices of G^t. Green: current communication links. Purple: frontiers of the last issued plan.

robots at time t. The following constraints are posed in the framework just described:

(I) $Q^t \cap F^t \neq \emptyset$, that is at least one frontier must be reached by a robot;
(II) $Q^t \cup \{b\}$ must form a single connected component in G^t;
(III) for each robot $r \notin R^t$ it must hold that $p_r^t \in Q^t$;
(IV) π^t must not change the robot-goal assignments for non-connected robots (robots momentarily not able to exchange data with the BS).

Constraint (I) requires a minimum exploration progress rate: if no frontier can be further visited, then no additional space can be explored and the mission ends. Recurrent connectivity is enforced by Constraint (II) forcing robots to be able to exchange data with the BS when they occupy their goal vertices. Constraints (III) and (IV) define feasible asynchronous replanning. Specifically, Constraint (III) allows to overwrite goals only for ready robots. That is, preemption at configuration level for unreached goals is forbidden. On the other side, we allow robots preemption in favor of a new deployment which, in presence of the new goals, might be preferred. By Constraint (IV), to change a robot's deployment this must be able to receive data from the BS.

Notice that, by definition, ready robots are connected to the BS, which can keep track of them. So, in principle, the BS can decide to replan over them as soon as some *replanning condition* is verified. We allow for arbitrary replanning conditions generalizing the approach adopted in [5] where new plans are issued for each robot when each of them reached the respective goal location. During the execution of a plan, some robots can become ready before others, hence becoming able to receive and execute a new plan. As we will discuss later, plans involving only part of the robot team are allowed to exploit communication links that will be made available by other robots (those currently not ready) to establish communication links with the BS. Fig. 2 shows a representative snapshot of the exploration process.

Given the setting described above, our objective is to compute plans for a given set of ready robots R^t to iteratively achieve efficient exploration of the environment. We decompose this into two sub-problems. The first one is the *optimal configuration problem*, where the set of robot locations Q^t that maximizes a utility function defined on frontier vertices under recurrent connectivity constraints is computed (Section IV). The second one is the *optimal deployment problem*, where a robot-location assignment $\pi^t(\cdot)$ minimizing traveling costs is computed (Section V).

IV. Optimal Configurations

A configuration is evaluated using the cumulative information gain achievable by acquiring sensor data from the reached frontiers. The utility $U(\cdot)$ of a frontier node f combines the estimated information gain $g(f)$ (usually proportional to the area associated to f when the environment is discretized) and its minimum traveling cost (see [4], [5]):

$$U(f) = \frac{g(f)}{\min\limits_{j \in R^t} d^2_{p_j^t f}}. \tag{1}$$

Non-frontier vertices have null utility and, with a slight overload of notation, $U(Q) = \sum_{v \in Q} U(v)$ denotes the utility of any configuration Q as defined in the previous section.

The goal is to find a configuration of size at most $|R^t|$ on G^t that maximizes the above utility function while maintaining connectivity with the BS. This problem can be seen as a particular case of known problems such as:

- *Constrained Maximum-Weight Connected Graph* (CMCG) [12], i.e., given a weight function $V \to \mathbb{R}$, a root vertex, and a positive integer k, find a connected subgraph containing the given root of exactly k vertices maximizing the cumulative weight. In our case all weights are non-negative, and the constraint on the number of vertices is not tight.
- *Rooted Maximum Node-Weight Connected Subgraph Problem with Budget Constraint* (B-RMWCS) [13], i.e., given a weight function $V \to \mathbb{R}$, a cost function $V \to \mathbb{R}_0^+$, a cost budget, a root vertex, and a (possibly empty) subset of terminal vertices, find a connected subgraph connecting the given root to the terminals maximizing the cumulative weight, without exceeding the budget. In our case all the weights are non-negative, there are no terminals, and each vertex consumes one unit of budget.
- *Rooted Budget Prize-Collecting Steiner Tree* (B-RPCST) [14], i.e., given a reward function $V \to \mathbb{R}_0^+$, an edge cost function $C \to \mathbb{R}_0^+$, a cost budget, and a root vertex, find a tree containing the given root maximizing the cumulative reward, without exceeding the budget. In our case, edge costs are unitary, and the budget limit is $|R^t|$.

Our problem can be shown to be $\mathcal{NP}$-*Hard* with a simple adaptation of the reduction outlined in [15] for the unconstrained version of CMCG.

A. Exact formulation

A simple ILP formulation can be derived from one discussed in [13] 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. In fact, a second formulation is presented in [13], including only vertex binary variables: in presence of dense graphs, it could provide lower planning times. We leave its study to future works.

First, in order to encode the fact that some robots may already have a goal assigned, and hence could be able to behave as relays for the currently replanning robots, we modify G^t by substituting the vertices assigned to non-ready robots with ficticious communication edges directly connecting the neighbors of the removed vertices with the BS. Moreover, we double each undirected communication edge, except for the set of edges incident to the BS, for which we keep only the outgoing arcs. With a slight abuse of notation, let us denote again as $G^t = (V^t, C^t)$ the modified graph derived as described above. We define the following binary variables:

- y_f, taking value 1 if and only if frontier $f \in F^t$ is chosen in the solution;
- x_{ij}, taking value 1 if and only if arc $(v_i, v_j) \in C^t$ is included in the solution.

We denote with $\delta^{\pm}(S)$ the directed cuts induced by the set of vertices $S \subseteq V^t$. The exact ILP model reads as follows:

$$\text{maximize} \sum_{f \in F^t} U(f) y_f \quad \text{s.t.} \quad (2)$$

$$\sum_{(i,j) \in C^t} x_{ij} \leq |R^t| \quad (3)$$

$$\sum_{(i,j) \in \delta^-(S)} x_{ij} \geq y_f \quad \forall f \in F^t, \forall S \subseteq V^t \setminus \{b\}, f \in S \quad (4)$$

The objective function (2) maximizes the configuration utility. Constraint (3) limits the size of the configuration to the available number of robots. If frontier f is occupied, Constraints (4) require a connected sequence of links from f to the BS. Given the model optimal solution, the connected component containing b represents the optimal solution of our problem. Despite the exponential multiplicity of the last constraints, the model can still be solved to optimality in instances of moderate size by a Branch&Cut algorithm which iteratively adds the violated inequalities found in the current solution (see Section VII and [13] for details).

B. Approximation algorithm

Real operational conditions require computational efficiency and to keep the problem tractable at higher dimensions. We now discuss an approximation method, able to introduce efficiency with a bounded loss of optimality. The literature presents different studies on the approximation of CMCG [12], B-RMWCS [16], and B-RPCST [17]: the best result applicable to our problem that we were able to find is the $4 + \epsilon$ approximation algorithm of [17]. However, differently from the standard approach followed in optimization literature, we do not seek an effective solution for *any* instance. Instead, we try to leverage our domain knowledge for which vertices are physical locations and budget is the number of robots, which do not typically come in very large quantities. In this section we show how, by exploiting such features, a different approximation algorithm can be developed. Our approach, despite not being competitive with the best known general one, achieves a non-constant approximation factor and is able to outperform it in our typical settings. The rationale is twofold. 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; 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; obviously, this does not imply an efficient algorithm for the general case (any number of terminals) for which $(\delta + 1)$-Steiner cannot be applied [6];
- the enumeration of the subsets of at most δ frontiers can be done in polynomial time.

Our method works on the original version of G^t after vertices assigned to non-ready robots have been removed as previously done. 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 it searches for the optimal Steiner tree T^* connecting $F^\delta \cup \{b\}$ by employing the $(\delta+1)$-Steiner algorithm. If the number of edges in T^* does not exceed $|R^t|$ then T^* is considered a feasible configuration and checked against the best configuration found so far. If, otherwise, 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 method. The *genNewSubset()* function is used to retrieve the next subset of frontiers to examine (Step 2). It returns a subset F of at most δ frontiers such that no other $F' \subseteq F$ has already been determined as non-feasible; this simple pruning rule is applied by maintaining a list of discarded solutions in Steps 7 and 13. We heuristically rank subsets by decreasing utility and increasing size.

For each returned subset F, we perform two simple tests to determine if it can be discarded without running $(\delta + 1)$-Steiner on it. First, we check the largest minimum frontier-frontier or frontier-BS distance in G^t (Step 6): subtracting 1, this is a simple lower bound on the minimum dimension of the Steiner tree connecting F and b. If the value computed exceeds $|R^t|$, we discard F. Then, $U(F)$ is evaluated (Step 10). This is a simple lower bound of the total price possible, if it does not exceed the current best solution F is discarded. The Steiner tree problem on $F \cup \{b\}$ 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 20) greedily adds branches from T^* to the unoccupied frontier with the highest utility, until no other frontier can be reached.

Theorem 4.1: For $\delta \in \mathbb{N}$, Algorithm 1 is a k/δ-approximation algorithm for the problem of finding the new connected configuration, where $k = \min(|R^t|, |F^t|)$. Its running time is bounded by $O(|F^t|^\delta [3^{\delta+1}|V^t| + 2^{\delta+1} e \log |V^t|])$.

Proof: For our definition of the *genNewSubset()* function, it is ensured that all the candidate feasible solutions can

Algorithm 1 Compute approximate optimal configuration.

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

be generated. Therefore, at the end of the while loop, we are guaranteed to have found a tree T^* connecting, within the budget limit, the BS vertex b with $F^{\delta\text{-BEST}}$, i.e., the set of frontiers with size $\leq \delta$ 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 $\gamma = |F^{\text{OPT}}| \mod \delta$. Moreover, to ease 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

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

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 $|F^{\text{OPT}}| \leq \min(|R^t|, |F^t|)$. Otherwise, notice that $[U(f_1) + \ldots + U(f_\delta)]/\delta \geq U(f_\delta)$. 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 algorithm of [6], whose running time is bounded by $O(3^{\delta+1}|V^t| + 2^{\delta+1} e \log |V^t|)$. The greedy completion of T^* performed by *completeConfiguration()* does not influence neither the approximation nor the running time bound. ■

Clearly, we have a meaningful bound only for $\delta \leq \min(|R^t|, |F^t|)$. Notice that this is always verified since we interpret δ as the maximum number of frontiers we can occupy by robots. Moreover, it is easy to see how such an approximation bound can yield an approximation better than the one provided by the best general algorithm presented in literature [17], with 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.

V. Optimal Deployments

Given a new connected configuration, it must be decided which robot goes to which new vertex. Instead of minimizing the maximum distance a robot has to travel, as in [5], since we are not only interested in completing the *whole* deployment in the shortest time, in this paper we minimize the cumulative travel distance, as in [3], by means of the Hungarian algorithm. Actually, since some non-ready robots may be in communication with the BS when replanning takes place, we compute a new allocation taking into account also these robots and their previous destinations, in order to reduce possible path overlaps between ready and non-ready robots. Notice that the newly computed connected configuration might reserve less than $|R^t|$ vertices for a new deployment. In this case, we use again the Hungarian algorithm to allocate the remaining robots to the vertices adjacent to the connected configuration w.r.t. C^t: these vertices will be treated as "fake" frontiers in the forthcoming definition of robot readiness.

VI. Readiness and Asynchronicity

As introduced in Section III, robots can take part in new plans as soon as they become ready, i.e., when they have reached their goal positions and possibly served as relay to other robots. In order to declare a robot to have completed its service as relay, several formal conditions could be defined, either offline or online (i.e., specified at the moment of issuing a plan or during its execution), and/or by taking into account the routing rules of a specific communication protocol. 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 [18]. Accordingly, we now provide a more formal definition of readiness that enables the BS to receive the data from the chosen frontiers in minimum expected time, under the assumptions of sufficient bandwidth along each communication link and negligible transmission times. First, notice that, although in Section IV we focused on finding a connected tree on G^t, many other links not explicitly considered in the solution could be available to transmit data between the vertices of Q^t. Intuitively, the definition will bind the robots placed on the *first* forming link path between a frontier and the BS to 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 first path connecting f and the BS in

G^t that will be built according to the time that each robot j has to travel from its current position to reach p_j^t. We define a robot j to be *ready* if (a) it has reached its destination vertex p_j^t, and (b) for each $f \in F^t \cap Q^t$ whose P_f contains q_j^t, the data of f has been received at the BS, and every other robot j' for which $p_{j'}^t \in P_f$ is at its goal position. For each frontier f, the corresponding P_f can be conservatively computed offline (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 j, build a restricted version of G^t containing only the vertices of Q^t associated with robots expected to reach their destination not after j; if a path exists, take the shortest, otherwise examine the next robot.

As soon as a robot becomes ready, it is available to receive a new plan from the BS. 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 our experiments, we study the effect of choosing different θ, i.e., replanning as soon as at least θ robots become ready.

VII. Experimental Evaluations

In our implementation robots maintain local grid maps representing the portion of environment they explored. On top of the BS global grid map, the exploration graph G^t is iteratively built by adding as vertices locations representing a sufficiently big cluster of frontier cells. Robots receive from the BS vertices of G^t (goals) to reach. Visited frontiers become candidate relay locations for future plans. We build C^t with the conservative limited line-of-sight model of [3].

A. Simulation activity

We choose the MRESim [4] simulator for its focus on the communication aspects and select three environments as shown in Fig. 3. Office and Open are from the Radish repository [19] ("sdr_site_b" and "acapulco_convention_center"), while Cluttered is from the MRESim repository ("grass"). We simulate 12 Turtlebot-like robots equipped with a depth camera with a maximum range of 50 pixels (cells), a $60°$ FOV, and an angular resolution of $1°$. The limit distance for the line-of-sight model is set to 150 pixels. Note that the sensor and communication ranges could realistically correspond to 5 m and 15 m, respectively, considering 10 cm a pixel. Robot speeds are set to 4 pixels/timestep (40 cm/s, assuming 1 second per timestep). For each experimental set, we execute 5 runs of 900 steps for each environment, varying the starting positions of BS and robots. Fig. 2 shows an experiment running on MRESim on the Office environment and the video accompanying the paper shows example runs of the experiments with simulated and real robots.

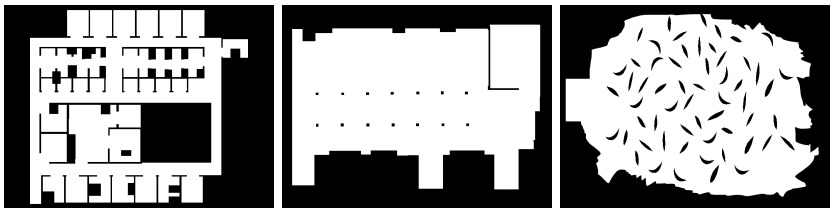
A first set of experiments gives some insights on the performance obtainable without considering asynchronous plans (i.e., $\theta = 12$) and evaluates the approximation algorithm (APX) against its optimal counterpart. The GUROBI solver [20] is used for solving the ILP on a laptop equipped with a i5-4310M processor and 8 GB of RAM as follows: initially, the model contains only Constraints (3); violations of Constraints (4) are checked, for 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 [13]). APX is run with $\delta = 4$ to obtain an approximation of 3.

For space reasons, Fig. 4 shows only the results obtained for the Cluttered environment (the other two present a similar trend). Fig. 4a shows the evolution of the average explored area when considering replanning time in counting the simulation steps. The results obtained are very similar: the better configurations obtained by means of the ILP are compensated by the faster replanning time of APX. All the instances, containing up to ≈ 200 vertices and ≈ 2000 edges, are always solved within 40 and 6 seconds, respectively. (Most of the times, the ILP is able to produce the optimal solution within few seconds.) Fig. 4b reports the evolution of the explored area when considering planning instantaneous: APX empirically shows basically no quality losses w.r.t. the exact method, given the online nature of the problem.

We now report on the effect of choosing different θ. Preliminary experiments showed that solving the ILP model in an asynchronous setting becomes not practical (at least in our current implementation) given the frequency with which robots may become ready, especially in early phases of the exploration, and given the increased size of the exploration graph. Therefore, we test only APX, again with $\delta = 4$, and compare our approach against the Utility method of [4], a distributed strategy in which 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 $0 \leq r \leq 1$ (here set to 0.5 to obtain a balanced 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. Fig. 5 shows the percentage of explored area known at the BS in the three environments. A low replanning threshold provides a statistically significant increment in the explored area across the mission (e.g., for Cluttered, in one-way ANOVA *p-value* $< 10^{-9}$ considering $\theta = 1$ and $\theta = 12$ at the end of the simulation). In particular, replanning as soon as at least one robot becomes ready offers competitive performance w.r.t. the Utility method, even if the latter seems to strongly depend from the starting positions of robots and BS. Recall that running APX with $\theta \leq \delta$ is guaranteed to return the optimal solution. Fig. 6 shows the average time in which robots and BS are not in communication. Our approach, although allowing temporary disconnections, allows a greater situational awareness at the BS than Utility.

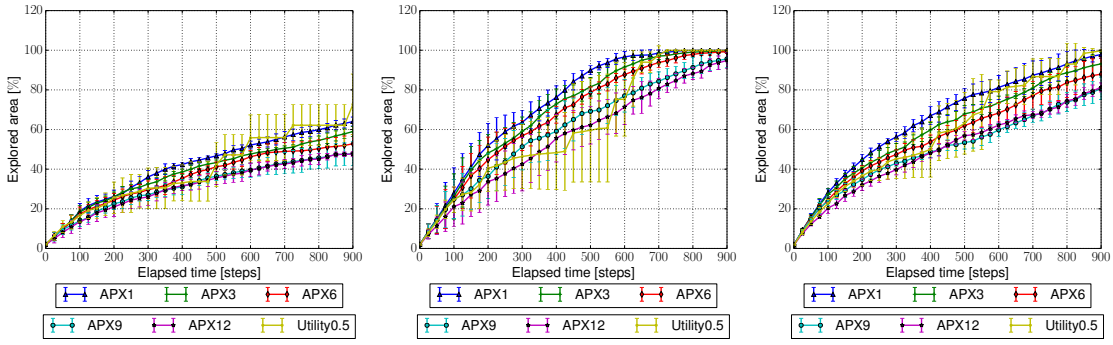
B. Implementation on real robots

Our method is also validated in a real scenario, with a laptop (the BS) and 6 TurtleBot 2 mobile robots equipped with a netbook and a Microsoft Kinect. An ad hoc network is set through the WiFi interfaces of the computers and



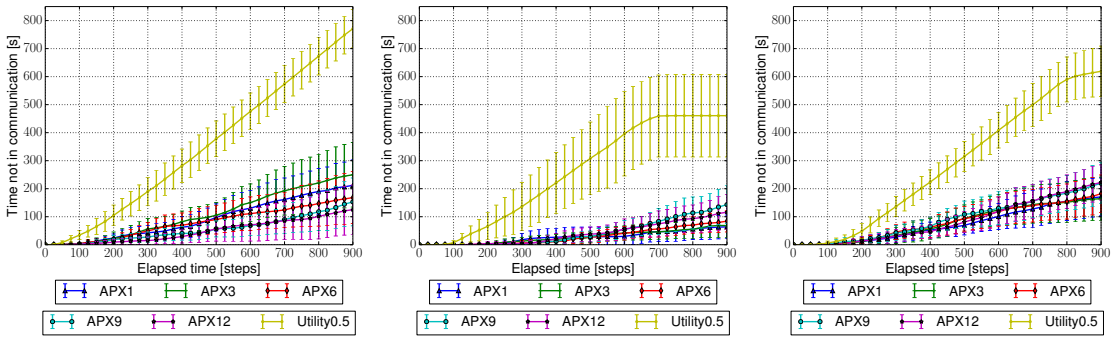
Office Open Cluttered

Fig. 3: Simulation environments, approximate size 80 x60 m.



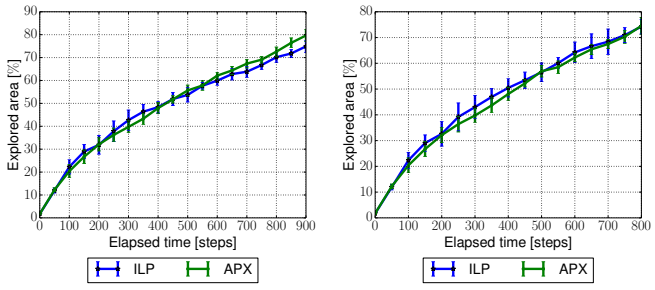
Office Open Cluttered

Fig. 5: Explored area for different replanning thresholds ($\theta = 1, 3, 6, 9, 12$) and Utility method [4].



Office Open Cluttered

Fig. 6: Time not in communication for different replanning thresholds ($\theta = 1, 3, 6, 9, 12$) and Utility method [4].



(a) (b)

Fig. 4: Comparison between ILP and approximation algorithm in the Cluttered environment, with (a) and without (b) replanning time.

the Optimized Link State Routing Protocol is run to allow multihop topology of the network [18]. ROS [21] is used to control the multirobot system. One part of the floor of the Swearingen Engineering Center at the University of South Carolina is used as testing ground for our approach; see Fig. 7. The size of the portion of the environment is approximately 75 by 35 m. There is a long corridor, with some other intersecting corridors/halls, thus requiring the robots to form a chain to guarantee communication with the BS. After some preliminary tests, we assume that two locations can communicate if they are within 15 m range in line of sight. We run APX with $\delta = 4$ and $\theta = 1$ and $\theta = 6$. For $\theta = 6$, initially two coalitions of robots are formed, each one going towards the frontiers of the corridors. Fig. 7 shows the partial map known by the BS and the robots. It is possible to observe that some noise is introduced, due to, e.g., perception and motion errors. One immediate consequence is

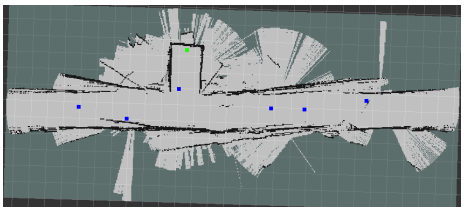


Fig. 7: Partial map built by the TurtleBots (blue squares) and known by the BS (green square) for $\theta = 6$.

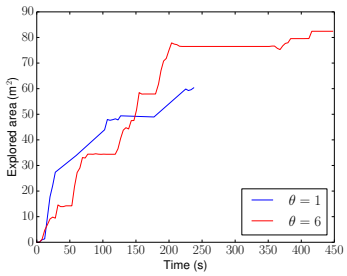


Fig. 8: Area explored by the real TurtleBots.

that, in order to let the exploration proceed, a relaxed version of the communication model should 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 destination locations could not be connected, resulting in a premature end of the exploration task. As 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 corridor to form a longer chain. Note that, with real robots, recovery mechanisms are necessary. For instance, some of the robots could not find a path to the assigned destination locations or some of the frontiers are found behind a wall because of the noisy map. In such cases, the robot aborts the motion to the location, notifying the BS. To guarantee that 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. Fig. 8 shows the trend of the explored area during the mission. At some time interval, we observed 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. Differently of the simulation results, setting $\theta = 1$, while initially providing some benefits, leads to a worsening in the performance and no candidate location available. With frequent replanning, robots tend to interfere in the motion of each other: this possibly results in collisions, and, given the low quality of the map, eventually all the locations become non-valid.

VIII. Concluding Remarks

In this work, we proposed a multirobot exploration strategy operating under recurrent connectivity constraints in a centralized and asynchronous framework. Our simulation results suggest that frequent replanning can offer a good situation awareness at the base station without significantly limiting the explored area. However, experiments on real robots revealed a tradeoff between frequent replanning and the need for more robust path planning and map building methods to cope with emerging robots interferences.

Currently, we are studying extensions of our exploration strategy to account for robot heterogeneity, that could result in preferential placement of robots as relays. Moreover, we are investigating the use of topological representations such as skeletons to guide the robots through the free space during the exploration process.

References

[1] S. Ochoa and R. Santos, "Human-centric wireless sensor networks to improve information availability during urban search and rescue activities," *Inform Fusion*, vol. 22, pp. 71–84, 2015.

[2] M. Rooker and A. Birk, "Multi-robot exploration under the constraints of wireless networking," *Control Eng Pract*, vol. 15, no. 4, pp. 435–445, 2007.

[3] E. Stump, N. Michal, V. Kumar, and V. Isler, "Visibility-based deployment of robot formations for communication maintenance," in *Proc. ICRA*, pp. 4498–4505, 2011.

[4] V. Spirin, S. Cameron, and J. de Hoog, "Time preference for information in multiagent exploration with limited communication," in *Proc. TAROS*, pp. 34–45, 2013.

[5] Y. Pei, M. Mutka, and N. Xi, "Connectivity and bandwidth-aware real-time exploration in mobile robot networks," *Wirel Commun Mob Comput*, vol. 13, no. 9, pp. 847–863, 2013.

[6] B. Kimelfeld and Y. Sagiv, "New algorithms for computing Steiner trees for a fixed number of terminals." `http://researcher.ibm.com/researcher/files/us-kimelfeld/papers-steiner06.pdf`.

[7] P. Mukhija, K. Krishna, and V. Krishna, "A two phase recursive tree propagation based multi-robotic exploration framework with fixed base station constraint," in *Proc. IROS*, pp. 4806–4811, 2010.

[8] J. de Hoog, S. Cameron, and A. Visser, "Autonomous multi-robot exploration in communication-limited environments," in *Proc. TAROS*, pp. 68–75, 2010.

[9] G. Hollinger and S. Singh, "Multirobot coordination with periodic connectivity: Theory and experiments," *IEEE T Robot*, vol. 28, no. 4, pp. 967–973, 2012.

[10] A. Howard, M. Matarić, and G. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Auton Robot*, vol. 13, no. 2, pp. 113–126, 2002.

[11] X. Cheng, D.-Z. Du, L. Wang, and B. Xu, "Relay sensor placement in wireless sensor networks," *Wirel Netw*, vol. 14, no. 3, pp. 347–355, 2008.

[12] H. F. Lee and D. R. Dooly, "Algorithms for the constrained maximum-weight connected graph problem," *Nav Res Log*, vol. 43, no. 7, pp. 985–1008, 1996.

[13] E. Álvarez-Miranda, I. Ljubić, and P. Mutzel, "The rooted maximum node-weight connected subgraph problem," in *Proc. CPAIOR*, pp. 300–315, Springer, 2013.

[14] D. S. Johnson, M. Minkoff, and S. Phillips, "The prize collecting Steiner tree problem: theory and practice," in *Proc. SODA*, vol. 1, pp. 760–769, 2000.

[15] D. Hochbaum and A. Pathria, "Node-optimal connected k-subgraphs." Manuscript, UC Berkeley, 1994. `http://goo.gl/QoB8IB`.

[16] A. Moss and Y. Rabani, "Approximation algorithms for constrained node weighted steiner tree problems," *SICOMP*, vol. 37, no. 2, pp. 460–481, 2007.

[17] C. Chekuri, N. Korula, and M. Pál, "Improved algorithms for orienteering and related problems," *ACM TALG*, vol. 8, no. 3, p. 23, 2012.

[18] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," Tech. Rep. RFC 3626, 2003.

[19] A. Howard and N. Roy, "The robotics data set repository (Radish)." `http://radish.sourceforge.net/`, 2003.

[20] "Gurobi optimizer reference manual." `http://www.gurobi.com`, 2015.

[21] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.