Optimal Redeployment of Multirobot Teams for Communication Maintenance

Jacopo Banfi Nicola Basilico Stefano Carpin

Abstract— **In this paper, we consider the problem of maintaining and restoring connectivity among a set of agents (humans or robots) by incrementally redeploying a team of mobile robots acting as communication relays. This problem is relevant in numerous scenarios where humans and robots are jointly deployed for tasks like urban search and rescue, surveillance, and the like. In this case, as the humans move in the environment, connectivity may be broken, and consequently, robots need to reposition themselves to restore it. We study the computational complexity of the problem, also in terms of approximation hardness, and present an Integer Linear Programming formulation to compute optimal solutions. We then analyze the performance of the proposed resolution approach against a heuristic algorithm taken from the literature, and we demonstrate how our method favorably compares in terms of solution quality and scalability.**

I. Introduction

The ability to install and maintain a resilient communication infrastructure among a set of cooperating mobile agents is one of the fundamental building blocks in many situations requiring to maintain situational awareness. It is known that mobile multirobot teams can be employed to solve this task. For example, in search and rescue scenarios, first responders search the environment for victims, while robots are dynamically redeployed to provide them with a communication infrastructure to support their operations [18]. Similar settings can be found in exploration of dangerous environments and surveillance, just to name a few.

Despite significant efforts in this area, effective redeployment of robots for communication maintenance still represents and open problem characterized by significant theoretical and practical challenges. Usually, it is tackled in heuristic ways and/or by over-provisioning the communication infrastructure. Among its leading complications is the fact that estimating the presence of communication links between locations of an environment is a complex task. Indeed, signal propagation in the physical world is subject to countless sources of uncertainty (scattering, multipath, fading, etc.), and analytic models fall short of being accurate or useful in most practical scenarios [9]. Consequently, incrementally deployed communication infrastructures are often subject to loss of connectivity, and the problem of recovering connectivity once it is lost is therefore of great practical importance.

In this paper, motivated by the above scenarios, we tackle the problem of maintaining and restoring connectivity among a set of agents (humans or robots) by incrementally redeploying a team of mobile robots acting as communication relays. This problem, introduced by Stump et al. in [26], was originally investigated in the context of *line-of-sight* communication maintenance and only addressed by means of a heuristic method. Starting from this prior work, three novel contributions advancing the state of the art are provided in this paper:

- we propose a more general problem formulation that efficiently leverages recently proposed methods for communication links estimation [7], [21], [23];
- we perform a rigorous characterization of the computational complexity of the problem;
- we provide an exact resolution method, showing how Integer Linear Programming (ILP) and state-of-the-art separation techniques [11] can be leveraged to compute optimal solutions in realistic instances with moderate computational efforts.

To corroborate the practical impact of our findings, an experimental comparison between our exact approach and an heuristic method adapted from [26] is presented as well, and we demonstrate how our method favorably compares in terms of solution quality and scalability.

The remainder of the paper is organized as follows. In Section II we discuss selected related literature in the area of robot-networks and algorithmic matters related to the combinatorial structure of the problem at hand. The problem is formally introduced in Section III together with its complexity characterization, with solving techniques given in Section IV. The formerly proposed heuristic solution is shortly presented in Section V along with a simple enhancement, and then compared against our method in Section VI. Finally, in Section VII we discuss the lessons learned, and outline venues for future research.

II. Related Work

The problem we consider in this work was originally introduced in [26] as a possible heuristic for maintaining communication among a set of mobile agents operating in an environment. The proposed solution exploits a team of robots acting as communication relays. The underlying motivation is intuitive: instead of maintaining the agents connected at all times, the continuous connectivity constraint is relaxed in favour of a sequence of connected agents' deployments. In such deployments, the agents can exploit an additional set of robots to convey information in a multi-hop fashion. The problem objective is therefore to form connected relay chains that minimize the robots' total travel

J. Banfi and N. Basilico are with the department of Computer Science, University of Milan, Milan, Italy.

S. Carpin is with the Department of Computer Science and Engineering, University of California, Merced, CA, USA.

cost. The authors' intuition related to the need of simplifying a continuous communication requirement was appropriate because the discrete multiagent connected path planning problem (MCPP) [16] was recently shown to be PSPACE-complete under a generic communication model [27].

From a more general perspective, we point out that the study of the impact of different types of communication constraints in multirobot information-gathering missions has become a trending research topic in the last few years. In a very recent survey [2], the connectivity requirement of [26] (and hence the one investigated in this paper) is classified as *event-based*. In the same class, [2] includes also *periodic* connectivity ("be connected each T time steps") [16] and *recurrent* connectivity ("be connected when new information is acquired") [3], [8], [20]. Noticeably, the already mentioned *continuous connectivity* has also been investigated in information gathering settings [19], [22]. To ensure these kinds of communication requirements without renouncing robust long-term planning, it is common to resort to *conservative* link-detection mechanisms capable of safely predicting the availability of a communication link between pairs of locations. For example, this can be done offline either by exploiting some *a priori* communication model (like limited-distance line-of-sight [4], [26]), or by devoting an additional team of robots to a systematic mapping of the communication characteristics of the environment [17], [23]. The latter task can also be carried out in an online fashion using a regression framework such as Gaussian Processes [7], [21], but in this case additional effort must be devoted to keeping the prediction confidence into account. To cope with possible false positives in the link-detection mechanism, [6] proposes optimal and heuristic algorithms for computing backup plans.

The problem we consider in this work is related to two well-known combinatorial optimization problems, i.e., the Steiner Tree Problem (STP) and the Linear (Sum) Assignment Problem (LAP). The former is an NP-Hard optimization problem [13] that can be informally stated as follows: given a subset of "terminal" vertices in an undirected, edge-weighted graph, find the sub-tree connecting them at minimum cost. In spite of the general intractability of STP, very good results can be achieved by leveraging suitable ILP formulations, like [11]. In LAP, instead, one is given n "persons", n "jobs', and a $n \times n$ matrix of costs for each person/job assignment. The objective is to assign each person to exactly one job so as to minimize the total assignment cost. This problem can be instead solved in polynomial time. The Hungarian method is one of the most famous algorithms for LAP, and has complexity $\mathcal{O}(n^3)$ [10].

The connection between the above combinatorial problems and the communication maintenance problem, which was already noted in [25], is as follows. In a graph whose edges represent the availability of communication links between any two locations, we need to connect a set of agents, which can be thought as the STP terminals, by occupying a number of locations equal to the number of robots at minimum cost. However, contrarily to the STP, in the connectivity maintenance problem such minimum cost is not given by some weights on the edges. Instead, this is equal to the value that is obtained by solving a LAP on an $n \times n$ cost matrix where the cost c_{ij} represents the distance that robot i needs to travel to reach location j.

III. Problem Formulation

We model the environment with a multigraph $G = (V, E, C)$ with a set of vertices V and two set of edges E and C describing both its physical and communication features. The set of vertices V represents the set of possible locations and can be obtained either by manually discretizing the environment, or with automated techniques (see, e.g., the experimental section of [26]).

The first edge set E models the physical topology of the environment. Each physical edge $(u, v) \in E$ with $u \neq v$ is associated with a positive integer number $d(u, v) \in \mathbb{Z}^+$ denoting the cost for traveling from u to v[1]. With a slight abuse of notation, given any two vertices $u, v \in V$, we denote with $d(u, v) \in \mathbb{Z}^+ \cup \{0\}$ the minimum cost for traveling between them. Note that in this case we add 0 to the set of possible costs, because we define $d(v, v) = 0$. To ease notation, we call $G_E = (V, E)$ the physical graph and we assume it is connected[2].

The second edge set C encodes the communication topology. If $(u, v) \in C$, a communication link is available between the locations represented by u and v, and any two robots occupying those locations are assumed to be capable of exchanging data by means of some protocol. As commonly assumed in literature, we assume that the set C is not affected by false positives and that it is time invariant. Similarly to what we did for the set E, we call $G_C = (V, C)$ the communication graph.

A team of agents $A = \{a_i\}$ and a team of robots $R = \{r_i\}$ are located in G, with starting locations denoted by the function $s : A \cup R \to V$. We assume that the function $s(\cdot)$ induces a *connected subgraph* on G_C, meaning that agents and robots are all connected at their starting locations, either directly or indirectly. The team of agents is then relocated to a new and possibly disconnected set of *target* vertices $t : A \to V$ such that the subgraph induced by $\{t(a) : a \in A\} \cup \{s(r_i) : r \in R\}$ on G_C is not connected. With a slight abuse of notation, we denote by $t(A)$ the set of all the target vertices occupied by the agents. The connectivity maintenance problem can then be stated as follows:

Problem 1: Given $\langle G, A, R, s, t \rangle$, compute the robots' redeployment goal locations $g : R \to V$ such that:

1) the subgraph of G_C induced by $t(A) \cup \{g(r) | r \in R\}$ is connected;
2) the total distance traveled by the robots $\sum_{r \in R} d(s(r), g(r))$ is minimized.

[1]Integer (or rational) numbers are usually preferred to reals in the definition of combinatorial optimization problems to avoid representation issues.

[2]Since G_E represents a physical environment, it can also be assumed that the cost $d(u, v)$ associated with a physical edge $(u, v) \in E$ is not greater than that of any other (u, v)-path computed on G_E.

A. Some Observations

Some remarks about Problem 1 are in order before continuing the discussion of its properties. First, we note that the problem formulation of [26] considers an additional cost term, scaled by a factor μ, in the objective function. Such cost favors the mutual proximity of the goal locations of robots in direct communication. In this work, we instead choose to adopt an objective function based only on the distance. We deem that the distance cost should have a prominent role while other minor cost terms can be mitigated during the problem modeling phase. For example, with respect to the additional cost term of [26], the maximum robot-robot and robot-agent communication distance can be enforced by a suitable construction of the communication graph G_C.

Second, the reader might have noticed that the agents' starting positions, as well as the connectedness of the starting deployment, do not play any role in the definition of the objective function: therefore, it could be possible to omit them from the definition of Problem 1. However, the heuristic method of [26] (primarily designed for settings where the agents' locations do not change much between $s(\cdot)$ and $t(\cdot)$) requires them to prune the search space. We will show that such a pruning rule can indeed work well when the agents' target positions are close to the corresponding start positions, but it is not suitable when the agents' redeployment is substantially different from the previous one.

Finally, we note that Problem 1 does not contain any constraints about the possible concurrent occupancy of the same vertex by more than one agent/robot. In general, enforcing each vertex to be occupied by at most one entity is a requirement that should depend on the chosen environment discretization. Regardless of such considerations, the following proposition shows that, under two very mild assumptions, it is always convenient to force each robot to occupy a vertex on its own in the final connected deployment.

Proposition 1: Consider a generic instance of Problem 1 where:

(A1) all the robots' starting positions are different and such that $\{s(r)|r \in R\} \cap t(A) = \varnothing$;

(A2) for any $u, v \in V$, there exists one shortest path between them on G_E s.t. if the physical edge $(i, j) \in E$ is traversed along the path, then $(i, j) \in C$ too.

Then, in any optimal solution $g(\cdot)$ no robot occupies a vertex already occupied by another agent/robot.

Proof: Any solution where at least one robot shares its goal vertex with another agent/robot can be improved by means of the following iterative procedure. At any iteration, consider an arbitrary robot r still sharing its goal vertex $g(r)$, and relocate it along the shortest path to its starting position $s(r)$ corresponding to assumption (A2) until either (a) $s(r)$ is reached, or (b) an empty vertex v is encountered. Each time a robot is relocated, the objective function improves and the subgraph induced by the solution remains connected. Moreover, each robot can be relocated at most $|V|$ times, since it will eventually reach its starting position. This implies that the procedure will always terminate in a finite number of steps. At the end of the procedure, each robot will either be the unique robot on a vertex, or at its starting position. By assumption (A1), this will not be shared with any other agent/robot. ■

Since assumption (A1) often holds in practice (at worst, it can be easily enforced by slightly modifying G) and assumption (A2) should hold in any reasonable environment discretization, throughout the rest of the paper we assume that these two hypotheses hold.

B. Complexity

The next theorem shows that even finding a feasible solution of a generic instance of Problem 1 is computationally hard in the general case.

Theorem 1: Deciding whether a generic instance of Problem 1 admits a feasible solution is NP-complete, even when G_E is a tree with unitary edge costs and $E \subset C$.

Proof: The above decision problem is clearly in NP, since the robots' goal function $g(\cdot)$ (serving as certificate) can be stored in polynomial space w.r.t. the input size and the connectedness of the subgraph of G_C induced by $t(A) \cup \{g(r)|r \in R\}$ can be checked in polynomial time. For what concerns NP-hardness, it is possible to build (in polynomial time) from a generic decision instance of STP (see Section II) with equal edge weights, which remains NP-complete [13], a particular instance of Problem 1 such that the former has "yes" answer if and only if the latter has a feasible solution. Let S and B be the STP terminal vertices and the upper bound on the STP edges available to connect the terminals, respectively. One can map the STP graph to G_C, the STP terminal vertices S to $t(A)$, and any spanning tree of the STP graph to G_E. Robots' starting positions can be added as dummy vertices suitably connected on G_E and G_C to any STP vertex. Finally, the cardinality of R is set equal to $B + 1 - |S|$. We can assume that B is less than the number of STP vertices and s.t. $B + 1 > |S|$ since otherwise the original STP instance is trivial. ■

Note, however, that the feasibility of instances where G_E is a tree and such that $C = E$ (or even $C \subset E$), which are usually treated with particular attention from a theoretical point of view [24], can be decided in polynomial time. Since in this paper we focus on a resolution approach for generic problem instances, we leave for future work the study of optimal algorithms for these special classes of graphs, as well as a more precise characterization of the complexity profile of the problem. For now, we just observe that Problem 1 is an optimization problem belonging to the NPO class[3], which can be classified according to the existence of *polynomial-time* algorithms with specific *approximation guarantees* [5]. The reader may be familiar with the APX complexity class, namely, the class including problems for which there exist constant-factor approximation algorithms. Since any feasible solution of an NPO problem has a performance ratio bounded

[3]Informally, this is the class of optimization problems such that (a) the set of instances is recognizable in polynomial time, (b) feasible solutions have polynomial size and can be recognized in polynomial time, and (c) the objective function can be computed in polynomial time [5].

by $h2^{n^k}$ for some h and k (n denotes the input size), as a corollary to Theorem 1 we have the following result.

Corollary 1: Unless P=NP, Problem 1 does not belong to the exp-APX complexity class, namely, the class of NPO problems for which there exists a $\mathcal{O}(2^{n^k})$-approximation algorithm for some $k \geq 0$.

IV. Optimal Resolution

Despite the problem's hardness, in this section we show how to tackle its optimal resolution using an Integer Linear Programming (ILP) formulation combined with state-of-the-art separation techniques that leverage our problem's combinatorial structure.

We build upon the parallelism that, as outlined in Section II, relates our setting to the Steiner Tree Problem. The optimal resolution of STPs was recently addressed in [11]. Precisely, the work presents an ILP-based resolution method for STPs with uniform edge costs and, more in general, problems involving the optimization of a linear objective function defined on the vertices of a connected subgraph. We take inspiration from this recent result and build an ILP model for Problem 1 exploiting the concept of *node separators*, formally defined as follows:

Definition 1: For two distinct nodes $u, v \in V$, a subset of nodes $N \subseteq V \setminus \{u, v\}$ is called (u, v) node separator if and only if after eliminating N from V there is no (u, v) path in G_C. A separator N is minimal if $N \setminus \{w\}$ is not a (u, v) separator, for any $w \in N$. Let $\mathcal{N}(u, v)$ denote the set of all (u, v) separators.

We also define $\delta_v = \{u \in V | \exists (v, u) \in C\}$ as the set of all the neighboring vertices of v in G_C. Our ILP model encodes the optimal robots' goal locations $g(\cdot)$ with a set of binary variables x_{rv} assuming value 1 iff robot $r \in R$ is placed in vertex $v \in V \setminus t(A)$. Moreover, an additional set of binary y_v variables is used to indicate whether vertex $v \in V$ is occupied by an agent or a robot in the solution. To ease the notation, we use V' to denote the set $V \setminus t(A)$. The ILP reads as follows:

$$\text{minimize} \sum_{r \in R} \sum_{v \in V'} d(s(r), v) \cdot x_{rv} \tag{1}$$

subject to

$$\sum_{v \in V'} x_{rv} = 1 \qquad \forall r \in R \tag{2}$$

$$\sum_{r \in R} x_{rv} = y_v \qquad \forall v \in V' \tag{3}$$

$$y_{t(a)} = 1 \qquad \forall a \in A \tag{4}$$

$$\sum_{n \in N} y_n \geq y_i + y_j - 1 \qquad \forall i, j \in V, i \neq j, \ \forall N \in \mathcal{N}(i, j) \tag{5}$$

$$y_v \in \{0, 1\} \qquad \forall v \in V \tag{6}$$

$$x_{rv} \in \{0, 1\} \qquad \forall r \in R, v \in V' \tag{7}$$

The objective function (1) minimizes the sum of the robots' redeployment costs[4]. Constraints (2) ensure that each robot is placed in a single vertex not already occupied by an agent, while Constraints (3) make sure that each vertex is assigned to at most a single robot. Constraints (4) force agents' target vertices to be part of the final connected subgraph. Finally, Constraints (5) make sure that the vertices occupied by agents and robots induce a connected subgraph on G_C. This happens for the following reason: if vertex i and j are occupied, then at least one vertex from any node separator $N \in \mathcal{N}(i, j)$ must also be part of the solution in order to ensure the existence of a path between i and j.

In order to solve ILP models with such an exponential number of constraints, the customary approach prescribes to gradually add them to the model as soon as the solver provides a fractional or integer solution violating them. In this work, we follow the suggestion of [11] and focus on cutting off only infeasible integer points enumerated by the solver. In particular, let $\hat{y}$ be the integer solution vector corresponding to y variables, and let $G_{\hat{y}} = (V, C_{\hat{y}})$ be a graph where $C_{\hat{y}} = \{(i, j) \in C | \hat{y}_i = \hat{y}_j = 1\}$. If $\hat{y}$ is infeasible, there exist at least two connected components in $G_{\hat{y}}$, V_i and V_j, such that $i \in V_i$, $j \in V_j$, and $\hat{y}_i = \hat{y}_j = 1$. Let $\delta(V_i)$ be the set of the neighbor vertices of V_i, i.e. $\delta(V_i) = \{v \in V \setminus V_i | \exists (u, v) \in C, u \in V_i\}$. The algorithm below shows how to obtain a *minimal* node separator $N \in \mathcal{N}(i, j)$, and can be implemented to run in $\mathcal{O}(|C|)$ [11]. Note that we are interested in finding minimal node separator inequalities (5) since they dominate the other ones. In our current implementation, given an infeasible solution, we add a violated Constraint (5) for each pair of distinct, disconnected (i, j) vertices.

Algorithm 1 Detecting a minimal node separator between two components V_i and V_j in $G_{\hat{y}}$

```
Delete all edges in $C[V_i \cup \delta(V_i)]$ from $G_C$
Find the set $R_j$ of nodes reachable from a vertex in $V_j$
Return $N = \delta(V_i) \cap R_j$
```

To speed-up the model resolution, we also initialize the model with the following additional valid inequalities:

$$\sum_{u \in \delta_v} y_u \geq y_v \quad \forall v \in V \tag{8}$$

Constraints (8) express the following simple fact: if a vertex is occupied, then so must be one of its neighbors.

Now, consider Constraints (2)-(3). For any integer feasible y vector, the corresponding constraint matrix is *totally unimodular*: therefore, it is possible to relax the integrality requirement on the x variables (7) while being ensured that, among the set of all the optimal solutions, there will always exist one where they all take integer values [10]. In fact, we are precisely solving a LAP between the robots R and the vertices in $v \in V'$ such that $y_v = 1$, while the x_{rv} variables associated with vertices such that $y_v = 0$ all take value 0.

Consider also the following fact: if a feasible solution with total cost U is available, then it is possible to reduce the number of x_{rv} variables by creating them only if $d(s(r), v) \leq U$ and feed the model with such a feasible

[4]The model can easily be adapted for the minimization of the maximum traveled distance; see [6].

solution. Constraints (2)-(3) have to be slightly modified to keep this variable pruning rule into account, but the total unimodularity property still holds.

To conclude, we mention that different ways of enforcing connectivity through a set of ILP constraints exist, resulting either in compact formulations, such as those based on single- and multi-commodity flows [14], or in non-compact ones, such as those enriching the above model with an additional set of binary variables modeling a "root vertex", or those explicitly modeling connections among vertices by means of additional binary edge variables [1]. Evaluating such formulations in our context is an interesting venue for future research.

V. Heuristic Resolution

As formerly discussed, the method proposed in [26] by Stump et al. can be used to heuristically solve Problem 1. We shortly recall some its salient aspects to contextualize the experimental comparisons with our exact method that we present in the next section. The formalization we report here is the result of the adaptation of the algorithm to our problem setting, and we refer the reader to [26] for a detailed description.

In [26], the authors leverage the fact that any deployment satisfying connectivity induces a connected topology among agents and robots. This is represented by a tree where vertices are given by $A \cup R$ and edges correspond to direct connections between them. The method, based on a dynamic programming approach, deploys robots at minimum cost with respect to a given candidate topology T. The cost function $Q_{b,v}$ quantifies the cost of deploying a robot or an agent b to vertex v, under the requirement of forming the topology described by T. If b is an agent, then $Q_{b,v}$ is initialized to 0 if $v = t(b)$ and to ∞ otherwise. Conversely, if b is a robot $Q_{b,v}$ is initially set to $d(s(b), v)$ for any $v \in V$. The algorithm selects a root vertex b in T and, for any vertex k, calls $\mathcal{N}_T(k)$ the set of k's children. Then, it computes Q with the following recursive formula, where the superscript $^{(0)}$ denotes the initial values:

$$\forall v \in V,\ Q_{b,v} = Q^{(0)}_{b,v} + \sum_{k \in \mathcal{N}_T(r)} \min_{q \in V} \{Q_{k,q} + K_{v,q}\}$$

In the above formula, $K_{v,q}$ is an additional cost term that, as we anticipated in Section III-A, is a trade off with the distance cost, In [26], this cost was set to ∞ for non connected pairs of locations. Since in this work we only consider the distance cost, in our implementation we set $K_{v,q} = 0$ if $(v, q) \in C$ and ∞ otherwise.

To avoid an exhaustive enumeration of all the $|V|^{|V|-2}$ tree topologies, the authors start from an initial topology associated with $s(\cdot)$, and proceed by single-edge edits relying on the underlying assumption of limited movement by the agents, thus preferring topologies that are "close" to the previous one.

Consider now the simple example instance show in Fig. 1, where the agents "swap" their locations in $t(\cdot)$. In this case, proceeding as suggested in [26] by single edge edits results in a suboptimal robots' redeployment (having cost 4) where the final topology is the same as the (unique) one associated with $s(\cdot)$. Clearly, an optimal solution would have been obtained by simply "swapping" the agents' connections, i.e., having a_2 connected with r_1 and a_1 with r_2. However, this would have required to edit two edges. This example shows that there are simple cases where the algorithm of [26] does not return the optimal robots' allocation on the selected vertices of $g(\cdot)$. To ensure this useful property, we can enhance the dynamic programming algorithm by simply solving a LAP between the robots' and the selected locations in $g(\cdot)$. In the example, this allows to obtain the optimal solution whose associated cost is 2.

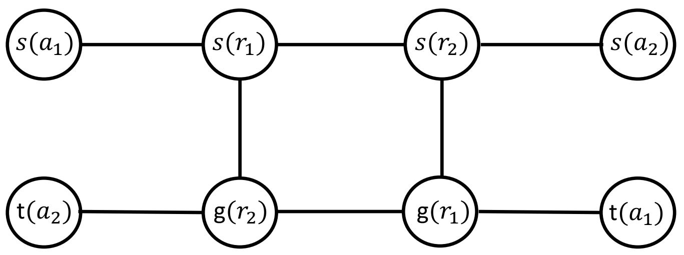

Fig. 1: A simple example instance with $G_E = G_C$ and unitary edge costs. The robots' goal function was obtained with the algorithm of [26].

VI. Experimental evaluation

In this section, we compare our proposed ILP based solution with an implementation of the dynamic programming algorithm of [26] in its "enhanced" version, as discussed above (called DP from now on). Our tests were run on a desktop computer equipped with an i7-7700K 4.2GHz processor and 64 GB RAM. The code of DP was implemented in MATLAB, while the ILP was solved with GUROBI [15] (vers. 7.5.1) with default settings[5]. Preliminary experiments on a restricted set of instances showed that, in general, it is convenient to relax the integrality requirement on the x variables. We remain consistent with this choice in all the experiments. The separation procedure was implemented through the GUROBI-Python interface, and all the graph-related computations were performed using the freely available igraph C library [12]. In all cases, if the algorithm did not terminate within one hour of computation, we stopped it and returned the best solution found (if present).

A. Square grid graphs

To assess the scalability of our approach against DP in a significant number of instances, we start by considering randomly generated robots' and agents' deployments in multigraphs G where G_E is a square (4-connected) grid graph, and G_C approximates a limited-range communication model. In particular, we considered square grid graphs of size $N \times N$, with $N \in \{10, 20, 30, 40\}$, with the distance between two vertices being the Manhattan distance. For what concerns

[5]The code used in our experiments is open-source and available at `https://github.com/jacoban/rdcm`.

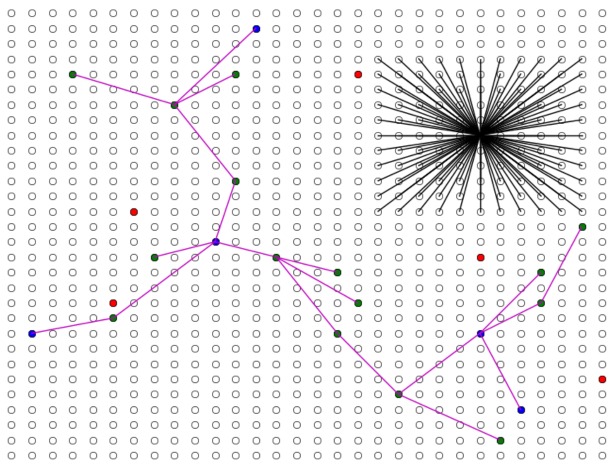

Fig. 2: A randomly-generated C-type instance (grid size 30×30, $|A| = 5$, $|R| = 15$). Blue and red: start and target agents' locations. Green: start robots' locations. Magenta: initial topology provided to DP. The communication edge set of a single vertex is also shown in black ($D = 5$).

the communication model, each vertex $v \in V$ is assumed to be in communication with all the vertices lying within a square of side length $2D$ centered in v, where the value of D depends on the particular set of experiments (see the top-right corner of Fig. 2 for an example).

In the first set of experiments, we compare our approach (ILP) with the heuristic method DP. To do this, we run both algorithms on instances where the agents' target locations are *close* to their starting ones. This choice provides a fair common ground for the two methods, by generating instances satisfying the agents-proximity assumption required by the DP heuristic.

Specifically, we build a random dataset of instances as follows. Starting locations, that is $s(\cdot)$, are obtained by randomly sampling a subset of vertices of G and keeping only those inducing a connected subgraph on G_C. Agents' goal locations, instead, are chosen by uniformly drawing from vertices that are connected to their starting locations (that is, within the boundaries of the square of edge 2D centered at the starting location) and such that (i) the subgraphs induced on G_C by $t(A)$ and (ii) $t(A) \cup s(R)$ are not connected. We set D to $\{2, 3, 5, 7\}$ for $N = \{10, 20, 30, 40\}$, respectively, and we consider two robots/agents configurations: $|A| = 2, |R| = 7$, as done in [26], and $|A| = 5, |R| = 15$. The initial connected topology for DP is obtained by computing a spanning tree on the subgraph induced by the vertices in $s(\cdot)$. For each configuration we generate 20 random instances. We dub these instances C-type (Close-type), and show an example in Fig. 2.

The results of the comparison are shown in Fig. 3, where we plot the heuristic gap value computed as $(DP - ILP)/ILP$ (algorithms names here denote the solution cost they provide). For this set of instances ILP was able to terminate with the optimal solution in all cases within 1 minute. The gaps show an increasing trend when dealing with more agents and robots and demonstrate how DP is able to actually find the optimal solution, although in a limited number of cases. The overall assessment, however, proves that the suboptimalities introduced by the heuristic can be at times very large, with instances where the heuristic solution is more than 20 times worse than the optimal one. No significant trend can be observed as the grid edge size becomes larger.

We do not provide a quantitative comparison between the time spent by the two algorithms because they are implemented using two languages with intrinsic performance differences. Just to give a qualitative idea, ILP always terminated with the optimal solution within 1 minute while DP reached the 1 hour deadline approximately 60% of the times, although this could be improved with a more efficient implementation.

In the second set of experiments, we assess the scalability of our approach in terms of the size of V and of the number of agents/robots. We consider the same multigraphs G of the previous set, but generate more challenging agents' target functions, which will correspond to vertices lying *far* from the starting locations. To build such instances, we start from coordinate $(0, 0)$ in the grid and place all the agents' starting locations at $(i, 0)$ for $a_i \in A$. As soon as the first row is completely filled, we change the y coordinate and iterate this procedure until all the agents are deployed. Robots' starting positions are then placed according to the same pattern, starting from the last agent. The agents' target function is selected by randomly drawing $|A|$ vertices of the grid while respecting the same constraints on $t(\cdot)$ of the previous set of experiments (we make sure that such instances admit at least a feasible solution). We dub these instances F-type (Far-type).

First, we consider a fixed $N = 20$ (with $D = 3$) and study how the computation times of ILP vary as a function of the number of agents/robots. In particular, we consider combinations ranging from $2/7$ to $25/75$. The box plots are shown in Fig. 4(a) and highlight two interesting facts. First, ILP is always able to obtain an optimal solution in less than 1 minute, except for an instance with 2 agents and 7 robots. Second, the median trend assumes a bell shape. This is easily explained by noticing that higher densities of agents/robots result in instances where it is very easy to find connected subgraphs inducing feasible solutions (recall that we are not varying the communication range).

Then, we consider the same two combinations of agents/robots of the first set of experiments, and study how the compute times vary as a function of N. The results are shown in Figs. 4(b)-(c). As expected, increasing the size of the multigraph results in larger computation times. The trend, however, shows a moderate growth and ILP is able to compute an optimal solution within the deadline most of the times. Precisely, in all the experiments reported in Fig. 4 ILP was always able to find the optimal solution except from three distinct cases observed with $|A| = 2, R = 7$ where the solver returned a solution with MIP gap not larger than 0.2.

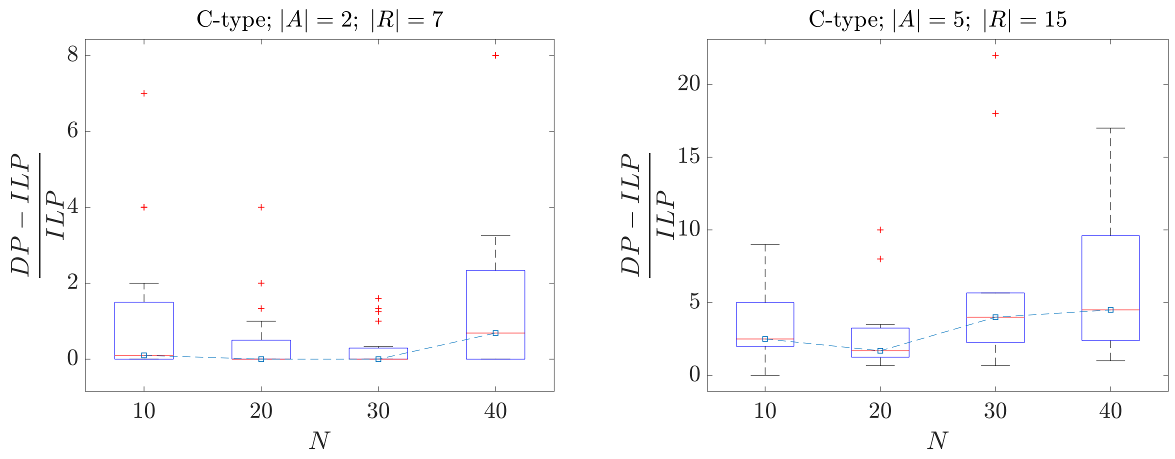

Fig. 3: Comparison between DP and ILP: heuristic gap.

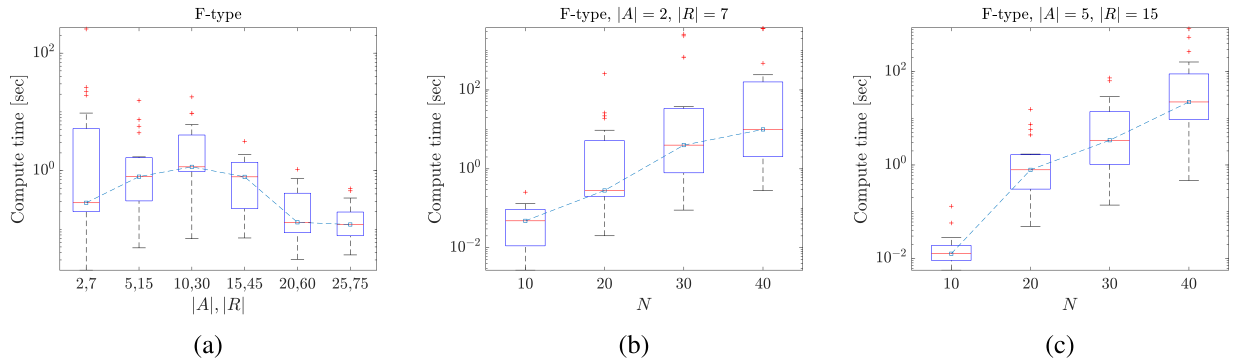

Fig. 4: Scalability analysis for the ILP. (a) Varying agents and robots (for fixed $N = 20, D = 3$). (b) Varying N (for fixed $|A| = 2, |R| = 7$). (c) Varying N (for fixed $|A| = 5, |R| = 15$).

B. *Real environment*

To conclude, we present the results obtained on the same real environment used in [26] and shown in Fig 5, whose approximate size is 100m ×80m. We consider a simple grid discretization in cells with side length $\approx$ 0.7m, resulting in $|V| = 1198$. G_C encodes a line-of-sight communication model, with $|C| = 101573$. As observed in [26], such a fine-grained grid discretization is not strictly required to obtain good results in such a structured environment. Therefore, it is reasonable to assume that, if ILP works well in this case, it will also work well with discretizations resulting in substantially less vertices, provided that they are able to capture the environment topology (for instance, those based on a polygonal decomposition of the environment). We consider a setting where $|A| = 2$ and $|R| = 7$. Agents' and robots' starting positions are chosen by randomly drawing a subset of vertices inducing a connected subgraph on G_C, while agents' target positions are chosen randomly on V (while respecting the same conditions discussed in the first set of experiments of Section VI-A). Fig. 6 shows the results obtained on 19 random instances[6] in terms of time and MIP gap after 1h of computation, confirming the viability of the proposed approach on real environments. On the same instances, DP was able to provide a solution in only 9 out of 20 instances (approx. 47%), but with a fairly satisfactory heuristic ratio whose median was approx. 0.74.

[6]We generated 20 random instances, but one of them was actually unfeasible. ILP was able to detect this fact in a few seconds.

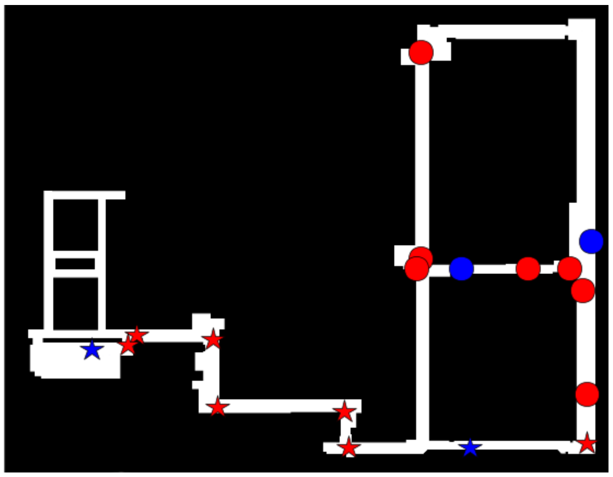

Fig. 5: A real environment. Dots and stars represent starting and goal locations, respectively, of an example optimal solution where robots are depicted in red and agents in blue.

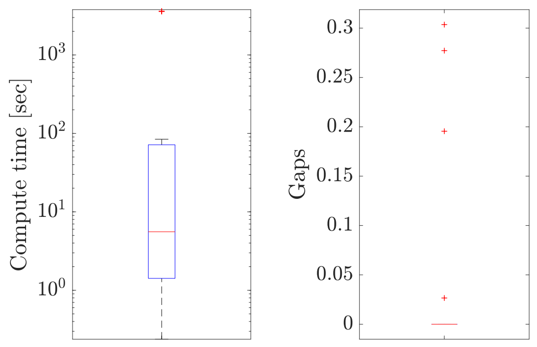

Fig. 6: Compute time and MIP gaps obtained by ILP on the real environment instances.

VII. Conclusions

In this paper we have presented a novel, exact solution for the connectivity maintenance problem formerly studied in [26]. Our method is based on an ILP formulation exploiting a connection with the STP and LAP optimization problems. By leveraging recent state-of-the-art methods to solve STP instances, we obtain a viable method capable of solving problem instances with sizes compatible with real world applications. Our experimental comparison with the heuristic method given in [26] identifies instances where the gap between the two methods can become arbitrarily large and, moreover, shows that under certain scenarios the heuristic scales very poorly with the problem size. Venues for future work include the extension to situations where the underlying graphs change over time, with the consequent necessity of repeatedly solving various instances over time.

Acknowledgment

The authors would like to thank Mattia Ornaghi, Alberto Quattrini Li, and Francesco Amigoni for the source code of the DP heuristic [26].

References

[1] E. Álvarez-Miranda, I. Ljubić, and P. Mutzel. The maximum weight connected subgraph problem. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization*, pages 245–270. Springer, 2013.

[2] F. Amigoni, J. Banfi, and N. Basilico. Multirobot exploration of communication-restricted environments: A survey. *IEEE Intel Syst*, 32(6):48–57, 2017.

[3] D. Anisi. *On cooperative surveillance, online trajectory planning and observer based control*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2009.

[4] R. Arkin and J. Diaz. Line-of-sight constrained exploration for reactive multiagent robotic teams. In *Proc. AMC*, pages 455–461, 2002.

[5] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and approximation*. Springer-Verlag, 1999.

[6] J. Banfi, N. Basilico, and F. Amigoni. Multirobot reconnection on graphs: Problem, complexity, and algorithms. *IEEE T Robot*, 2018. To appear (available online at https://goo.gl/NxvBtD).

[7] J. Banfi, A. Quattrini Li, N. Basilico, I. Rekleitis, and F. Amigoni. Multirobot online construction of communication maps. In *Proc. ICRA*, pages 2577–2583, 2017.

[8] J. Banfi, A. Quattrini Li, I. Rekleitis, F. Amigoni, and N. Basilico. Strategies for coordinated multirobot exploration with recurrent connectivity constraints. *Auton Robot*, 42(4):875–894, 2017.

[9] W. Braun and U. Dersch. A physical mobile radio channel model. *IEEE T Veh Technol*, 2(40):472–482, 1991.

[10] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. SIAM, 2009.

[11] M. Fischetti, M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl. Thinning out steiner trees: a node-based model for uniform edge costs. *Math Program Comput*, 9(2):203–229, 2017.

[12] C. Gabor and N. Tamas. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

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

[14] S. Gollowitzer and I. Ljubić. Mip models for connected facility location: A theoretical and computational study. *Comput & Oper Res*, 38(2):435–449, 2011.

[15] Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2017.

[16] G. Hollinger and S. Singh. Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE T Robot*, 28(4):967–973, 2012.

[17] M. Hsieh, V. Kumar, and C. Taylor. Constructing radio signal strength maps with multiple robots. In *Proc. ICRA*, pages 4184–4189, 2004.

[18] S. Kohlbrecher, A. Romay, A. Stumpf, A. Gupta, O. Von Stryk, F. Bacim, D. Bowman, A. Goins, R. Balasubramanian, and D. Conner. Human-robot teaming for rescue missions: Team vigir's approach to the 2013 darpa robotics challenge trials. *J Field Rob*, 32(3):352–377, 2015.

[19] T. Nestmeyer, P. Robuffo Giordano, H. Bülthoff, and A. Franchi. Decentralized multi-target exploration using a connected network of multiple robots. *Auton Robot*, 41(4):989–1001, 2017.

[20] Y. Pei, M. Mutka, and N. Xi. Connectivity and bandwidth-aware real-time exploration in mobile robot networks. *Wirel Commun Mob Comput*, 13(9):847–863, 2013.

[21] P. K. Penumarthi, A. Quattrini Li, J. Banfi, N. Basilico, F. Amigoni, I. Rekleitis, J. O'Kane, and S. Nelakuditi. Multirobot exploration for building communication maps with prior from communication models. In *Proc. MRS*, 2017.

[22] R. Rathnam and A. Birk. Distributed communicative exploration under underwater communication constraints. In *Proc. SSRR*, pages 339–344, 2011.

[23] A. Riva, J. Banfi, C. Fanton, N. Basilico, and F. Amigoni. A journey among pairs of vertices: Computing robots' paths for performing joint measurements. In *Proc. AAMAS*, pages 229–237, 2018.

[24] M. Sinay, N. Agmon, O. Maksimov, S. Kraus, and D. Peleg. Maintaining communication in multi-robot tree coverage. In *Proc. IJCAI*, pages 4515–4522, 2017.

[25] E. Stump. *On cooperative surveillance, online trajectory planning and observer based control*. PhD thesis, University of Pennsylvania, 2009.

[26] E. Stump, N. Michael, V. Kumar, and V. Isler. Visibility-based deployment of robot formations for communication maintenance. In *Proc. ICRA*, pages 4498–4505, 2011.

[27] D. Tateo, J. Banfi, A. Riva, F. Amigoni, and A. Bonarini. Multiagent connected path planning: PSPACE-completeness and how to deal with it. In *Proc. AAAI*, pages 4735–4742, 2018.