This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



# Rigidity-Preserving Team Partitions in Multiagent Networks

Daniela Carboni, Ryan K. Williams, *Student Member, IEEE*, Andrea Gasparri, *Member, IEEE*, Giovanni Ulivi, *Member, IEEE*, and Gaurav S. Sukhatme, *Fellow, IEEE*

***Abstract*—Motivated by the strong influence network rigidity has on collaborative systems, in this paper, we consider the problem of partitioning a multiagent network into two sub-teams, a bipartition, such that the resulting sub-teams are topologically rigid. In this direction, we determine the existence conditions for rigidity-preserving bipartitions, and provide an iterative algorithm that identifies such partitions in polynomial time. In particular, the relationship between rigid graph partitions and the previously identified Z-link edge structure is given, yielding a feasible direction for graph search. Adapting a supergraph search mechanism, we then detail a methodology for discerning graphs cuts that represent valid rigid bipartitions. Next, we extend our methods to a decentralized context by exploiting leader election and an improved graph search to evaluate feasible cuts using only local agent-to-agent communication. Finally, full algorithm details and pseudocode are provided, together with simulation results that verify correctness and demonstrate complexity.**

***Index Terms*—Distributed robot systems, graph rigidity, networked robots.**

## I. Introduction

COORDINATED autonomous systems continue to demand strong attention from researchers, particularly within recent years. The driving force of this surge is the increasing promise of multiagent systems in reality, a product of rapid advancements in computation and communication, and the implications that collaborative systems have for impactful applications. Examples range from base behaviors such as tracking and coverage [1]–[3], formation control [4], [5], and state consensus [6], [7], synchronization and optimization [8], [9], to higher-level objectives such as collective transport [10], wireless network optimization [11], [12], environmental monitoring [13], and data fusion [14]. Additionally, multiagent systems may yield significant advantages over single-agent solutions in terms of heterogeneity of mobility and sensing, fault tolerance and flexibility, and scalability [15]–[18].

In this paper, we are concerned with identifying two properties of the graph that describes a multiagent network. First, given a single team of connected agents, we wish to identify partitions in the network graph that yield two sub-teams, i.e., graph bipartitions or split maneuvers. Partitioning or splitting a team emerges as an important behavior primitive in navigating uncertain or cluttered environments by endowing the team with the flexibility to change in both composition and scale. Further, network partitioning is compelling in the context of task assignment (see [19]), as it would enable task-centric collaboration, allowing for example the decoupling of tasks across spatiotemporal scales. There is also promise in applying the team splitting primitive for collective transport [20], multitarget entrapment or encirclement [21], and multiteam cooperation [22].

Our second and arguably most important concern is that each partitioned team is rigid (which also implies connected sub-teams). Broadly speaking, rigidity represents an important requirement when a system demands collaboration. For example, rigidity guarantees formation stability when only relative sensing information is available [23]–[25]. Specifically, the asymptotic stability of a formation is guaranteed when the graph that defines the formation is rigid by construction. Thus, in maintaining sub-team rigidity, it becomes possible to move beyond the classical formation methodology, to that of dynamic formations, precisely as rigidity can guarantee stability as the underlying network changes. Sub-teams could then take on dynamically generated and stable formations based on the objectives of a given task assignment. Rigidity is also a necessary (and in certain settings sufficient) condition for localization tasks with distance or bearing-only measurements [24], [26], [27]. The ability of a network to self-localize is of clear importance across various application contexts, and for example in [27], it is shown that if the rigidity conditions for localizability for traditional noiseless systems are satisfied, and measurement errors are small enough, then the network will be approximately localizable. Bipartitions that are rigidity-preserving are then immediately localization-preserving as well. Finally, the flavor of rigidity studied here is also a necessary component of global rigidity [28]–[30], which

Manuscript received June 2, 2014; revised October 23, 2014; accepted December 3, 2014. This work was supported in part by the Italian Grant FIRB Futuro in Ricerca, project NECTAR, code RBFR08QWUV, the Italian Ministry of Research and Education (MIUR), and in part by the ONR MURI Program under Award N00014-08-1-0693, in part by the National Science Foundation (NSF) CPS program under Grant CNS-1035866, and in part by the NSF under Grant CNS-1213128. The work of R. K. Williams was supported by a fellowship from the USC Viterbi School of Engineering. A preliminary version of this paper was presented at the 19th World Congress of the International Federation of Automatic Control (IFAC 2014). This paper was recommended by Associate Editor H. M. Schwartz.

D. Carboni, A. Gasparri, and G. Ulivi are with the Department of Engineering, University of Roma Tre, Roma 00146, Italy (e-mail: gasparri@dia.uniroma3.it).

R. K. Williams and G. S. Sukhatme are with the Departments of Electrical Engineering and Computer Science, University of Southern California, Los Angeles, CA 90089 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2378552

2168-2267 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



can further strengthen the guarantees of formation stability and localizability, as the uniqueness of a given topological embedding is more easily characterized.

The general study of rigidity has a rich history in science, mathematics, and engineering [30]–[33]. In [32], combinatorial operations are defined which preserve rigidity, with [23] and [25] extending the ideas to multiagent splitting and formation control [34], [35]. In [36], an algorithm is proposed for generating rigid graphs in the plane based on the Henneberg construction [32]. Similarly, [37] defines decentralized rigid constructions that are edge length optimal. However, to our knowledge no previous work has considered the issue of identifying the conditions for and an algorithmic solution to rigidity-preserving graph partitioning. There exists work that considers for example the identification of $k$-connected subgraphs (see [38]), however, not from a team splitting or multiagent perspective. In [23], [25], [39], and [40], certain rigidity-preserving operations are described, but methodologies for decentralized rigidity-based partitioning are absent. Finally, insight into the splitting of rigid teams is given in [41], however, from the perspective of sub-team connectedness, which can be seen as complementary to our contributions in this paper.

Motivated by the collaborative impact of rigidity, we propose in this paper iterative algorithms that determine the existence of a rigidity-preserving bipartition of a graph in the plane, with guaranteed polynomial complexity. While previous work has provided constructive intuition for rigid splitting and rejoining (see [23]), instead in this paper, we identify how to find such partitions and the conditions under which they will be found. We first provide an analysis of the relationship between the existence of rigid graph partitions and the topological conditions that must then hold, yielding a sound direction for searching the graph for feasible partitions. In particular, we determine that for minimally rigid teams, a three-edge set always preserves sub-team rigidity when it cuts the network graph. This intuition guides us to consider the three-edge structure termed a Z-link proposed in [23], which possesses a local structure that is amenable for decentralization. Then, we reason on determining through graph search, when a candidate three-edge set represents a rigidity-preserving bipartition of the graph. Specifically, exploiting the combinatorial properties of rigidity, we demonstrate that a classical depth-first graph traversal is sufficient in making such an identification. Finally, resting on the local nature of the Z-link, and the amenability of depth-first search (DFS) to local communication, we decentralize our methods by exploiting leader election and a heuristically pruned DFS. Full algorithm details and pseudocode are provided as well as simulation results and Monte Carlo analysis that verifies our claims of complexity and correctness.[1]

The outline of this paper is as follows. In Section II, we provide preliminary materials including agent and network models, a primer on the relevant aspects of rigidity theory, and a problem formulation. Theoretical developments and a centralized algorithm for identifying rigidity-preserving bipartitions is presented in Section III, with a full decentralization then given in Section IV. Simulation results are provided in Section V, and the conclusion as well as directions for future work are stated in Section VI.

## II. Preliminaries

### A. Network and Agent Modeling

Consider a system composed of $n$ agents (e.g., robots) indexed by $\mathcal{I} = \{1, \ldots, n\}$ operating in $\mathbb{R}^2$, each possessing communication capabilities, denoting by $(i, j)$ a bi-directional communication link between agents $i$ and $j$. Note that, in a networking context every agent possesses a unique ID for the purposes of communication. Indeed, this will turn out to be useful for the leader election process in the decentralized version of the proposed algorithm. Each agent may or may not be mobile, depending on the underlying application, and we simply assume each agent possesses some position $x_i \in \mathbb{R}^2$, possibly with time-varying dynamics.

To characterize the interconnected system, we define undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, having vertices $\mathcal{V} = \{v_1, \ldots, v_n\}$ associated with each agent $i \in \mathcal{I}$, and edge set $\mathcal{E} = \{(i, j) \mid i, j \in \mathcal{V}\}$ with unordered pairs $(i, j)$, where by definition $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$, $\forall\, i \neq j \in \mathcal{I}$, excluding the possibility for self loops, $(i, i) \notin \mathcal{E}$, $\forall\, i \in \mathcal{I}$. Agents $i$ and $j$ sharing an edge $(i, j) \in \mathcal{E}$ are referred to as neighbors, where the set of neighbors for the $i$th agent is given by $\mathcal{N}_i = \{v_j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$. The undirected[2] graph topology can be seen as a communication network and a sensing graph. In our case, the concept of rigidity that we introduce in the sequel is particularly suited for sensing topologies, specifically as rigidity is necessary in sensing-based objectives such as formation control and localization. However, we require that the communication topology coincides with the sensing topology so that our decentralized algorithms (Section IV) properly respect the underlying sensing topology, which we wish to rigidly partition.

### B. Rigidity Theory

A primary concern of this paper is the rigidity property of the underlying graph $\mathcal{G}$ describing the network topology, again given the fundamental guarantees that rigid graphs provide for example in both localizability and formation stability of multiagent systems [25]. We provide a brief overview of rigidity theory here. We direct the reader to [31], [32], and [42] for a technical primer on the subject. To begin we require the notion of a graph embedding in the plane, captured by the framework $\mathbb{F}_p \triangleq (\mathcal{G}, p)$ comprising graph $\mathcal{G}$ together with a mapping $p : \mathcal{V} \to \mathbb{R}^2$, assigning to each node in $\mathcal{G}$, a location in $\mathbb{R}^2$. The natural embedding for us is to assign each node the position $x_i$ associated with each agent, defined by the mapping $p(i) = x_i$, otherwise known as a realization of $\mathcal{G}$ in $\mathbb{R}^2$.

[1] We also point out that by iteratively applying our methods, a network could further split into multiple partitions, although the general problem of multipartitioning rigid networks is the subject of future work.

[2] The concepts of rigidity that we introduce in this paper are formulated only for undirected communication and sensing. Extension to directed notions of rigidity is the focus of future work.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



The rigidity of a graph can be view in terms of infinitesimal motion of $\mathbb{F}_p$ that can be described by assigning to the vertices of $\mathcal{G}$ a velocity $\dot{p}_i \triangleq \dot{x}_i \in \mathbb{R}^2$ such that

$$\left(\dot{x}_i - \dot{x}_j\right) \cdot \left(x_i - x_j\right) = 0, \qquad \forall (i,j) \in \mathcal{E}$$

where $\cdot$ is the standard dot product over $\mathbb{R}^m$. That is, the edge lengths (interagent distance) over $\mathcal{G}$ are preserved in time, in other words, no edge is compressed or stretched over time. The framework is said to undergo a finite flexing if $p_i$ is differentiable and edge lengths are preserved, with trivial flexings defined as translations and rotations of $\mathbb{R}^2$ itself. If for $\mathbb{F}_p$ all infinitesimal motions are trivial flexings, then $\mathbb{F}_p$ is said to be infinitesimally rigid. Otherwise, the framework is called infinitesimally flexible. It is worthy to note that while this description of rigidity relies on motion of the framework, a network being evaluated for infinitesimal rigidity (or in fact, the notion of generic rigidity that follows) need not be mobile.

Intuitively, the concept of rigidity can be thought of in a physical way, that is if $\mathbb{F}_p$ were a bar and joint framework, it would be mechanically rigid against external forces. In general, the rigidity of $\mathbb{F}_p$ is tied to the specific embedding of $\mathcal{G}$ in $\mathbb{R}^2$, however, it has been shown that the notion of rigidity is a generic property of $\mathcal{G}$, specifically as almost all realizations of a graph are either rigid in infinitesimal motion or flexible (i.e., they form a dense open set in $\mathbb{R}^2$) [43]. Thus, we can consider rigidity from the perspective of $\mathcal{G}$, abstracting away the necessity to check every possible realization. The first combinatorial characterization of graph rigidity was described by Laman in [31], and is summarized as follows (also called generic rigidity).

*Theorem 1 [31]:* A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with realizations in $\mathbb{R}^2$ having $n \geq 2$ nodes is rigid if and only if there exists a subset $\bar{\mathcal{E}} \subseteq \mathcal{E}$ consisting of $|\bar{\mathcal{E}}| = 2n - 3$ edges satisfying the property that for any nonempty subset $\hat{\mathcal{E}} \subseteq \bar{\mathcal{E}}$, we have $|\hat{\mathcal{E}}| \leq 2|\hat{\mathcal{V}}| - 3$, where $|\hat{\mathcal{V}}|$ is the number of nodes in $\mathcal{V}$ that are endpoints of edges $(i,j) \in \hat{\mathcal{E}}$.

We refer to the above as the Laman conditions, where it follows that any rigid graph in the plane must then have $|\mathcal{E}| \geq 2n - 3$ edges, with equality for minimally rigid graphs. The impact of each edge on the rigidity of $\mathcal{G}$ is captured in the notion of edge independence, a direct consequence of Theorem 1.

*Theorem 2 [33]:* The edges $(i,j) \in \mathcal{E}$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ are independent in $\mathbb{R}^2$ if and only if no subgraph $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$ has $|\bar{\mathcal{E}}| > 2|\bar{\mathcal{V}}| - 3$.

The above conditions imply that a graph is rigid in $\mathbb{R}^2$ if and only if it possesses $2n - 3$ independent edges, where edges that do not meet the conditions of Theorem 2 are called redundant (see Fig. 1 for a depiction of graph rigidity). Thus, in determining the rigidity of $\mathcal{G}$, we must verify the Laman conditions to discover a suitable set of independent edges, a task that was originally solved in a centralized manner by [33], with decentralization and parallelization achieved in [42] and [44].

*Remark 1:* The extension of Laman's conditions to higher dimensions is at present an unresolved problem in rigidity theory, and thus our contributions will be limited to the case of multiagent networks in the plane. While there exist special cases under which extensions to three dimensions have been achieved, we point out that results in the plane do not necessarily limit application. For example, in aerial platforms, rigidity may only be necessary in a 2-D subspace, where team altitude is often fixed or relatively easy to measure.

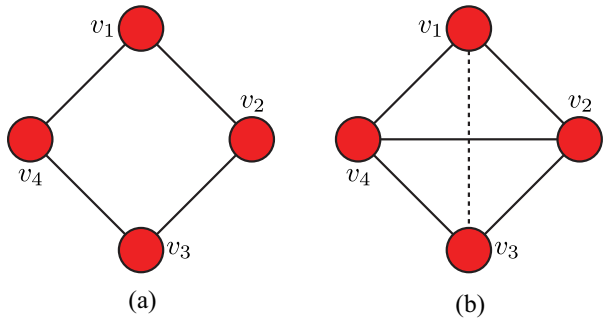


Fig. 1. Example graphs demonstrating rigidity, where dashed links indicate edges that have been added to form a new network. Notice that all solid edges in graphs (a) and (b) are independent, while edge $(v_1, v_3)$ in (b) is redundant. (a) Nonrigid. (b) Rigid.

### C. Henneberg Construction

In this section, we describe the Henneberg construction, an inductive technique to construct rigid graphs introduced by [45], which will prove useful in evaluating the complexity of our bipartition search. The reader is referred to [32], [46], and [47] for further details. The construction process starts from a graph with two vertices linked by an edge. At each step in the procedure, either a vertex and two edges are added, or a vertex and three new edges are added while one existing edge is removed. The new edges are incident to the new vertex. These operations are termed vertex addition and edge splitting, respectively. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the starting graph and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ the graph obtained after the application of one of the Henneberg operations, which are described as follows.

*Definition 1 (Vertex Addition):* A new vertex v is adjoined, then $\mathcal{V}' = \mathcal{V} \cup \{v\}$, as well as two edges so that $\mathcal{E}' = \mathcal{E} \cup \{(v,j), (v,k)\}$ for some $j, k \in \mathcal{V}$.

*Definition 2 (Edge Splitting):* A new vertex v is adjoined, then $\mathcal{V}' = \mathcal{V} \cup \{v\}$ as well as three edges, while an edge is removed, so that $\mathcal{E}' = \mathcal{E} \cup \{(v,j), (v,k), (v,m)\} \setminus \{e\}$ for some $j, k, m \in \mathcal{V}$ with at least two of the vertices $j, k, m$ adjacent in $\mathcal{G}$ and the edge $e$ either $(j,k)$, $(j,m)$, or $(k,m)$.

In Fig. 2, an example of Henneberg construction is given. Finally, we provide an important result that relates the Henneberg construction to minimally rigid graphs.

*Lemma 1 [32]:* A graph is minimally rigid if and only if it has a Henneberg construction.

### D. Definitions and Problem Formulation

Our goal in this paper will ultimately be to partition (or split) the graph $\mathcal{G}$ such that each resultant component is rigid and follows the properties outlined above. Thus, we define the following.

*Definition 3:* A bipartition of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a division of the graph into two disjoint components,

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



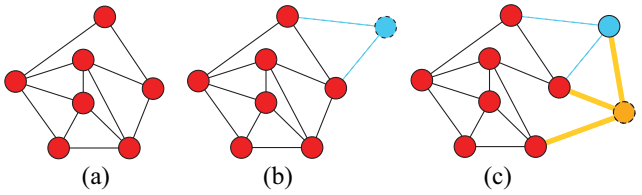


Fig. 2. Example of Henneberg construction. (a) Rigid graph during Henneberg construction. (b) Vertex addition (dashed blue). (c) Edge splitting (thick yellow).

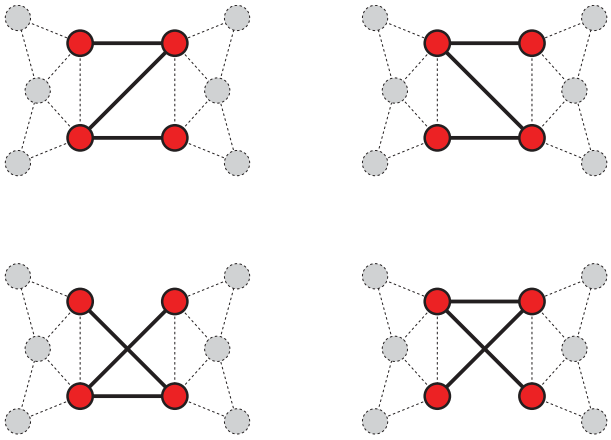

Fig. 3. [23, Fig. 7] Z-link in four possible configuration of edges.

namely $\mathcal{G}_1(\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2(\mathcal{V}_2, \mathcal{E}_2)$, such that $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$, $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$, $\mathcal{E}_1 \cup \mathcal{E}_2 = \mathcal{E}$, and $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$. We refer to a $k$-bipartition as the case in which we choose $k \geq 1$ such that $|\mathcal{V}_1| = k$ and $|\mathcal{V}_2| = n - k$, dictating the size of the resulting partitions.

Further, if a graph can be partitioned, it follows that there exists a cut.

*Definition 4:* A cut $C = \{(i, j) \in \mathcal{E} \mid i \in \mathcal{V}_1, j \in \mathcal{V}_2\}$, is a set of edges that when removed from $\mathcal{G}$ yields disjoint components $\mathcal{G}_1$ and $\mathcal{G}_2$ such that $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ and $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$.

Next, we recall two useful results and intuition from [23] which will aid us in determining proper graph partitions.

*Definition 5 [23]:* We refer to a bipartite graph $\mathcal{K}_{2,2}$ with three edges, shown in Fig. 3, as a Z-link.

*Corollary 1 [23]:* Two minimally rigid graphs that are connected using a Z-link construct a minimally rigid graph.

Notice that the Z-link structure is not only useful in characterizing edges that preserve rigidity, but also takes on a two-hop structure which is amenable to decentralization.

*Remark 2:* It is important to note that we will be restricted to two-hop decentralization precisely by the nature of the rigidity problem. As pointed out in [48, Lemma 3.3], "For an agent $i$, the local graph containing neighboring nodes and incident edges possesses only independent edges." That is, at least locally all edges appear necessary for a graph to remain rigid, and thus they will be perceived as nonremovable in a cut operation. Therefore, as expressed by the Z-link structure, we must consider at least two-hop information when determining cuts in the network.

Finally, our target problem is stated formally as follows.

*Problem 1:* Given a minimally rigid graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$, partition $\mathcal{G}$ into two disjoint subgraphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ having $|\mathcal{V}_1| = k$ and $|\mathcal{V}_2| = n - k$ such that $\mathcal{G}_1$ and $\mathcal{G}_2$ are minimally rigid. Notice that due to the Laman conditions of Theorem 1, it is implied that $k \geq 2$, yielding $|\mathcal{V}_1| \geq 2$ and $|\mathcal{V}_2| \geq 2$, with $|\mathcal{V}| = n \geq 4$.

In other words, the problem is to find a proper cut over the graph $\mathcal{G}$; we refer to this as the rigid bipartitioning problem.

## III. Identifying Rigid Bipartitions

We now turn our attention to deriving an algorithm that identifies rigid bipartitions for networks in the plane. According to the minimal rigidity assumption, we know that:

1) $\mathcal{G}$ must have $2n - 3$ independent edges;
2) $\mathcal{G}_1$ must have $2k - 3$ independent edges;
3) $\mathcal{G}_2$ must have $2(n - k) - 3$ independent edges.

Hence, we can conclude that $|\mathcal{V}_1| + |\mathcal{V}_2| = 2n - 6$, and thus there must exist three independent edges that connect $\mathcal{G}_1$ and $\mathcal{G}_2$ that are lost when $\mathcal{G}$ is partitioned, i.e., edges that are independent with respect to $(\mathcal{E}_1 \cup \mathcal{E}_2)$. Therefore, we can argue that a feasible cut must be composed of exactly three edges. Furthermore, we can prove that, after identifying a cut over $\mathcal{G}$, it is sufficient to check if the two induced components $\mathcal{G}_1$ and $\mathcal{G}_2$ have the desired number of vertices, in order to ensure they are two minimally rigid graphs. Thus, as stated in the following theorem, there is no requirement to count the number of edges in the graph, a convenient property for our purposes.

*Theorem 3:* Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a minimally rigid graph and let be $C$ a cut over $\mathcal{G}$ such that $|C| = 3$. Let $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ be the two subgraphs of $\mathcal{G}$ that are obtained after the cut, again with $|\mathcal{V}_1| \geq 2$ and $|\mathcal{V}_2| \geq 2$. Then, $\mathcal{G}_1$ (or $\mathcal{G}_2$) has $k$ vertices $\iff$ it has $2k - 3$ edges.

*Proof:* ($\Longrightarrow$) Let us assume that $\mathcal{G}_1$ has $k$ vertices. It follows that $\mathcal{G}_2$ has $n - k$ vertices. Since $\mathcal{G}_1$ and $\mathcal{G}_2$ are subgraphs of $\mathcal{G}$, condition 2 of Theorem 1 must hold for both and then $|\mathcal{E}_1| \leq 2k - 3$ and $|\mathcal{E}_2| \leq 2(n - k) - 3$. We know also that $|\mathcal{E}_1| + |\mathcal{E}_2| = |\mathcal{E}| - |C| = 2n - 6$. This equation is satisfied only when $|\mathcal{E}_1| = 2k - 3$ and $|\mathcal{E}_2| = 2(n - k) - 3$.

($\Longleftarrow$) Let us assume that $|\mathcal{E}_1| = 2k - 3$ edges, then $|\mathcal{E}_2| = |\mathcal{E}| - |C| - |\mathcal{E}_1| = 2(n - k) - 3$. Three cases are possible.

1) $|\mathcal{V}_1| < k$: This implies that condition 2 of Theorem 1 does not hold for $\mathcal{G}_1$. However, this contradicts the minimal rigidity assumption over $\mathcal{G}$.
2) $|\mathcal{V}_1| > k$: This implies that $|\mathcal{V}_2| < n - k$ and thus condition 2 of Theorem 1 does not hold for $\mathcal{G}_2$. Again, this contradicts the minimal rigidity assumption over $\mathcal{G}$.
3) $|\mathcal{V}_1| = k$: This is the only admissible case. ■

From Theorem 3, we can argue that given a cut $C$ over minimally rigid graph $\mathcal{G}$, if the cut is composed by three edges, then the two induced components, namely $\mathcal{G}_1$ and $\mathcal{G}_2$, are minimally rigid. Among all three edge cuts of $\mathcal{G}$, we concern ourselves with cuts that have the Z-link structure from Corollary 1. Although extending our methods to account for all rigidity-preserving cuts is straightforward, Z-links are the only cuts that are structurally guaranteed to be amenable for decentralization. This choice is supported by simulation results, as described in Section V, which demonstrate

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



that the number of undetected cuts is reasonably low. That is, we believe that losing a few cuts is a bearable disadvantage, when compared with the benefits of decentralized autonomy.

Now that, we understand the structural elements we seek in order to partition the graph, let us characterize our worst-case expectations in searching for such cuts (a bound which we tighten in Section III-B).

*Remark 3:* An absolute bound on the number of Z-links in a graph can be obtained by considering that, in the worst case, each group of three edges is a Z-link. So, the number of Z-links must be bounded by $\binom{n}{3} = O(|\mathcal{E}|^3)$ as the graph is minimally rigid, with $|\mathcal{E}| = 2n - 3$.

Therefore, we can expect that our bipartition search will possess polynomial complexity; a fact that we corroborate in the sequel. During our search, when a Z-link has been identified it is necessary to verify two properties: that it is a cut over $\mathcal{G}$, since there could be some Z-links in the graph that are noncuts, and if the two subgraphs obtained by the cut are minimally rigid graphs with $k$ and $n-k$ nodes, as we could discover feasible cuts that do not meet this condition. Given the result in Theorem 3, i.e., a cut yields two minimally rigid subgraphs over which counting vertices is sufficient, we can simply perform an exploration of the graph by means of a DFS. Starting from a node of the Z-link, the graph exploration is carried out avoiding all the edges that belong to the Z-link. The search ends when no more nodes can be visited. Letting $\mathcal{V}'$ be the set of visited nodes, we can obtain one of the following results.

1) $|\mathcal{V}'| = n$: The considered Z-link is not a cut because the DFS algorithm has explored the entire graph.
2) $|\mathcal{V}'| < n \ \wedge \ |\mathcal{V}'| \neq k \ \wedge \ |\mathcal{V}'| \neq n-k$: The considered Z-link is a cut but it is not our desired cut.
3) $|\mathcal{V}'| = k \vee |\mathcal{V}'| = n-k$: The considered Z-link is a proper cut.

This intuition is illustrated in Fig. 4. The graph has $n = 8$ vertices and for the bipartition we fix $k = 3$. Dashed red edges belong to the Z-link. A vertex is blue if it is visited. In Fig. 4(a), exactly $k = 3$ vertices can be visited, in fact the selected Z-link is a proper cut. In Fig. 4(b), all $n$ vertices are reachable indicating that the Z-link is not a cut.

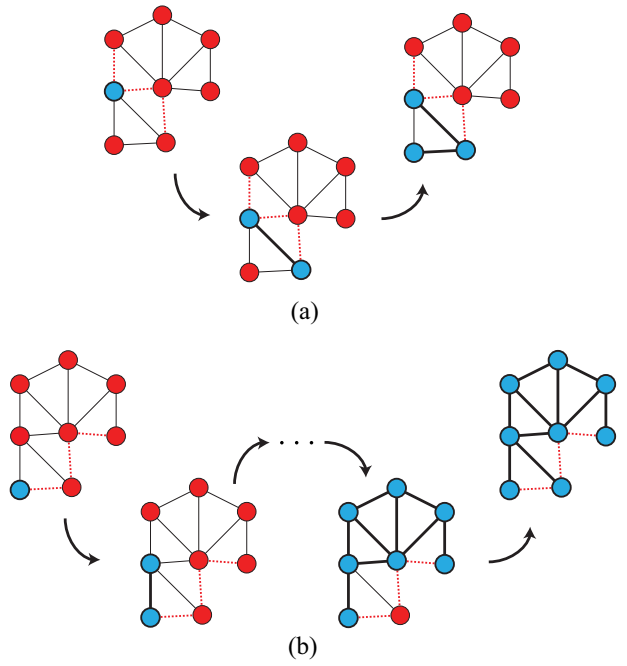


Fig. 4. For this example $n = 8$ and $k = 3$. In dashed red are the Z-link edges. In blue are the visited nodes. (a) At the end of the graph exploration exactly $k$ vertices have been visited, hence the Z-link is a proper cut. (b) All $n$ vertices have been explored hence the Z-link is not a cut.

### A. Algorithm Details and Pseudocode

The proposed algorithm, which takes a centralized form, is now described in detail. As described above, the Z-link structure is vital to identifying the graph cut that yields rigid partitions. To find Z-links in $\mathcal{G}$ we adapt the graph search mechanism applied in [38], (which originated from [49] and [50]) known simply as the $X - e + Y$ method. Briefly, the method is concerned with generating all members of the set $\mathcal{F} = \{X \,|\, X \subseteq \mathcal{E} \,|\, \pi(X) = 1\}$, where $\pi(X) : 2^{\mathcal{E}} \to \{0, 1\}$ is a monotone boolean function indicating the satisfaction of some desired property for $X$. In our case, we wish to generate elements $X \subseteq \mathcal{E}$ that are quads, that is, sets of four nodes which contain at least one Z-link. The $X - e + Y$ method applied in this context will then generate all quads that contain Z-links in $\mathcal{G}$. The search for quads occurs over a supergraph having vertexes in $\mathcal{F}$, i.e., the supergraph contains supernodes with quads that possess Z-links. The neighborhood $N(X)$ of $X \in \mathcal{F}$ and the family $\mathcal{Y}_{X,e}$ are defined by

$$N(X) = \left\{(X \setminus e) \cup Y \,|\, e \in X, Y \in \mathcal{Y}_{X,e}\right\} \tag{1}$$

and

$$\mathcal{Y}_{X,e} = \{Y \,|\, Y \subseteq \mathcal{E} \setminus X \ \wedge \ \pi((X \setminus e) \cup Y) = 1\}. \tag{2}$$

That is, for every valid quad $X \in \mathcal{F}$ and $e \in X$, we extend $X \setminus e$ to the set $(X \setminus e) \cup Y$; in our case, this constitutes a swapping operation, where we refer to $\mathcal{Y}_{X,e}$ as a swap set, with $\mathcal{Y}_{X,e}$ containing nodes $Y$ which yield Z-links when swapped into $X \setminus e$. For each extension of $X$, a new supernode in the supergraph is generated yielding a directed graph structure over $\mathcal{F}$. It follows that the supergraph is strongly connected and thus performing a breadth-first search can generate all elements of $\mathcal{F}$ [38], as opposed to a brute force search.[3] Our adaptation of this method for finding Z-links and ultimately a rigidity-preserving partition is shown in Algorithm 1, a complete explanation of which is now given.

At a high-level, the algorithm operates as follows. The set of quads that contain Z-links is generated by the swap operations of Algorithm 2, and for each Z-link we determine if it is a graph cut which yields rigid disjoint components, exploiting the conclusions of Proposition 3. Starting from a minimally rigid graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ the first step is to find an initial Z-link containing quad for the supergraph search, which is done in a

[3]In a minimally rigid graph, as Z-links scale like $O(n^3)$, a brute force search may not be intractable. However, in nonminimally graphs, the focus of our future work, the scaling becomes $O(n^6)$, demanding a pruned search method as is given here.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



**Algorithm 1** Partition a Graph Into Two Rigid Components

```
1: procedure SPLITGRAPH(G = (V, E), k)
2:     [quad, zlinks] ← INITIALSUPERNODE(G)
3:     for each zlink ∈ zlinks do
4:         if ISACUT(G, k, zlink) then
5:             return zlink
6:         end if
7:     end for
8:     P ← quad
9:     while P ≠ ∅ do
10:        [quad, P] ← REMOVEFROMQUEUE(P)
11:        Q ← INSERTINTOQUEUE(Q, quad)
12:        for v ← 1 to 4 do
13:            [S, zlinks] ← SWAPSETFROMQUAD(G, quad, v)
14:            for nbrQuad ∈ S do
15:                if nbrQuad ∩ (P ∪ Q) = ∅ then
16:                    for each zlink ∈ zlinks do
17:                        if ISACUT(G, k, zlink) then
18:                            return zlink
19:                        end if
20:                    end for
21:                    P ← INSERTINTOQUEUE(P, nbrQuad)
22:                end if
23:            end for
24:        end for
25:    end while
26: end procedure
```

brute force way. For each node $i \in \mathcal{V}$ the two-hop neighborhood is considered in order to identify the set of all quads to which $i$ contributes, i.e., the set nodes $= \{i\} \cup \mathcal{N}_i \cup \{\mathcal{N}_j, \forall j \in \mathcal{N}_i\}$ is generated. Then the set quads is obtained simply by picking from nodes all possible combinations of four distinct elements of nodes so if |nodes| $= p$ then |quads| $= \binom{p}{4}$. Next, for each quad $\in$ quads the presence of a Z-link is revealed as in Algorithm 3. Consider the basic idea that there are only three ways to partition a quad {1 2 3 4} into pairs.

1) $\{1, 2\}$ and $\{3, 4\}$.
2) $\{1, 3\}$ and $\{2, 4\}$.
3) $\{1, 4\}$ and $\{2, 3\}$.

The presence of a Z-link can then be revealed according to the following proposition.

*Proposition 1:* Let us consider a graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ with four vertices. $\hat{\mathcal{G}}$ is a Z-link if there exists a partition of $\hat{\mathcal{V}}$ into two sets $\hat{P}_1 = \{i \in \hat{\mathcal{V}} : |\hat{P}_1| = 2\} = \{v_1, v_2\}$ and $\hat{P}_2 = \hat{\mathcal{V}} \setminus \hat{P}_1$ such that given the set $\hat{\mathcal{E}}_{12} = \{(i, j) \in \hat{\mathcal{E}} : i \in \hat{P}_1, j \in \hat{P}_2\}$, all the following conditions hold.

1) $|\hat{\mathcal{E}}_{12}| = 3$.
2) $\exists (i, j) \in \hat{\mathcal{E}}_{12} : i = v_1$.
3) $\exists (i, j) \in \hat{\mathcal{E}}_{12} : i = v_2$.

In other words, if a pair has exactly three outgoing edges, where each nodes contributes at least a single edge, then there is a Z-link. The conditions introduced in Proposition 1 are explained through an example in Fig. 5. Two possible pairwise partitions of a graph with four vertices are shown. The red edges belong to the set $\mathcal{E}_{12}$ so they are candidates to be removed if the Z-link is chosen. In Fig. 5(a), a Z-link is found by partitioning the graph into $P_1 = \{1, 2\}$ and $P_2 = \{3, 4\}$. In Fig. 5(b), a Z-link cannot be found since $|\mathcal{E}_{12}| = 4$, in fact the removal of the red edges does not lead to a bipartite graph.

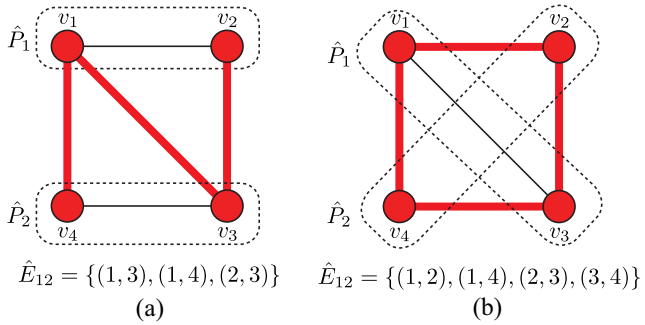


Fig. 5. Example of the conditions stated in Proposition 1. The red (thicker) edges belong to the set $\mathcal{E}_{12}$ so they are the edges that will be removed. (a) Z-link is found. (b) It is not possible since $|\mathcal{E}_{12}| = 4$.

**Algorithm 2** Returns Nodes Which Yield Z-Links When Swapped

```
1: procedure SWAPSETFROMQUAD(G, quad, v)
2:     S ← ∅
3:     s ← quad(v)
4:     for each j ∈ N_s do
5:         nodes ← nodes ∪ N_j
6:     end for
7:     nodes ← nodes \ quad
8:     for each node ∈ nodes do
9:         swapQuad ← quad
10:        swapQuad(v) ← node
11:        zls ← ZLINKSFROMQUAD(G, swapQuad)
12:        if zls ≠ ∅ then
13:            zlinks ← zlinks ∪ zls
14:            S ← S ∪ node
15:        end if
16:    end for
17:    return {S, zlinks}
18: end procedure
```

After a Z-link is detected we must check if it is a cut over the graph $\mathcal{G}$, and if the removal of the edges that belong to the Z-link yields two disjoint components, one with $k$ nodes and the other with $n - k$ nodes. This operation is described in Algorithm 4. As introduced above, a DFS is performed, taking only a single arbitrary node from the current Z-link as root of the exploration tree. At the end of this recursive search the visited nodes are counted and if this number is equal to $k$ or equal to $n - k$ it implies that a proper Z-link is found. Otherwise, the Z-link has to be discarded and the search must continue until either a valid bipartitioning Z-link is found, or the graph is deemed infeasible for the desired $k$-bipartition.

### B. Complexity Analysis

To close, we now provide a proof of the expected complexity of our proposed algorithm.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



**Algorithm 3** Returns the Z-Links in a Given Set of Four Nodes

```
procedure zLinksFromQuad(G, quad)
    zlinks ← ∅
    for each {p1, p2} ∈ quad do
        if |N_p1| + |N_p2| = 3 ∧ |N_p1| ≥ 1 ∧ |N_p2| ≥ 1 then
            for each j ∈ N_p1 do
                zlinks ← zlinks ∪ (p1, j)
            end for
            for each j ∈ N_p2 do
                zlinks ← zlinks ∪ (p2, j)
            end for
        end if
    end for
    return zlinks
end procedure
```

**Algorithm 4** Check if a Z-Link is a Proper Cut Over a Graph

```
procedure isACut(G = (V, E), k, zlink)
    Ê ← E \ zl
    Ĝ ← (V, Ê)
    visited ← DFS(Ĝ, v ∈ zlink)
    if |visited| = k then
        G1 ← visited
        G2 ← V \ visited
        return true
    else if |visited| = n − k then
        G1 ← V \ visited
        G2 ← visited
        return true
    end if
    return false
end procedure
```

*Theorem 4:* Consider a minimally rigid graph $\mathcal{G}$ in the plane. By construction, Algorithm 1 and its constituent components exhibit polynomial complexity when applied to $\mathcal{G}$, and therefore the rigid bipartitioning problem is solved in polynomial time.

*Proof:* From [38], it follows that the search mechanism illustrated in Algorithm 1 runs in polynomial time if and only if the set of reachable quads returned by swapSetFromQuad can be generated in polynomial time. In other words, the determination of swaps which yield new Z-links dominates execution. In our case, as each quad must contain exactly four elements, while swapping only a single element, each set is generated by inspecting at most $O(n)$ swap possibilities, with each inspection trivially requiring $O(1)$ operations (i.e., by applying Algorithm 3). Thus, we conclude that swapSetFromQuad runs in polynomial time, and our result follows. ■

The correctness of the algorithm follows from the preceding propositions of this section. As will be shown in Section V, our proposed algorithm executes in $O(n^3)$ time, in a Monte Carlo sense.

*Theorem 5:* In a minimally rigid graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n$ nodes, the number of Z-links is bounded above by $O(n^2)$.

*Proof:* As described in Section II-C each minimally rigid graph can be obtained through Henneberg construction. At each step of the Henneberg construction, a new vertex is added to the graph and this vertex is connected with two or three old vertices. For a graph with $n$ nodes $n-2$ steps are required, and at the end of the $k$th step we have constructed a minimally rigid partial graph $\mathcal{G}_k = (\mathcal{V}_k, \mathcal{E}_k)$ with $|\mathcal{V}_k| = k+2 = n_k$ and $|\mathcal{E}_k| = 2n_k - 3$. Now, consider the addition of Z-links at each step of the construction. The worst case is when a new vertex is added through vertex addition, i.e., it is connected with two existing nodes. There could be a new Z-link by adding to this set of three vertices each of the remaining vertices, and hence at the end of $k$th step there could be $n_k - 3 = O(n_k) = O(n)$ new Z-links. At the end of the graph construction we have a number of Z-links that is bounded by

$$\sum_{k=1}^{n-2}(n_k - 3) = \sum_{k=1}^{n-2} O(n) = (n-2)O(n) = O(n^2)$$

concluding the proof. ■

*Theorem 6:* Given a minimally rigid graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$ the split search is $O(n^3)$.

*Proof:* According to Theorem 5, we know that the number of Z-links in the graph is $O(n^2)$. In order to check if a Z-link is a proper cut a DFS-visit of the graph has to be performed. It starts from a node of the Z-link and visits each node in the graph not more than one time, so it is $O(n)$. Hence the split search is $O(n^2)O(n) = O(n^3)$. ■

### C. Alternative Problem Formulations and Applications

We close our analysis of the centralized $k$-bipartitioning algorithm by pointing out possible robotic applications and extensions.

1) In our original problem formulation, we fix a value of $k$ and then check if there exists a $k$-bipartition of the graph. This could be useful when an external supervisor provides split commands to a team of mobile agents, without specifying which robots have to belong to a partition and which to the other, but only how many agents are needed. There exist swarm density driven explorations (see [51]) where cardinality-based team partitioning could prove useful, particularly in preserving the localizability of sub-teams to guide exploration.
2) Alternatively, for each value of $k$, with $1 \leq k \leq \lceil n/2 \rceil$, we could check if there exists a $k$-bipartition of the graph. We can imagine a scenario in which there is a list of tasks that the robots must execute, and for each task there is a specification for the minimum number of needed robots. Thus, it may be important to know which tasks are feasible for a given graph topology, such that an external operator could choose the tasks with higher priority among those which are feasible, or provide for a redistribution of the robots in such a way as to make executable a task that previously was not. This check can be performed, in a way similar to that described in Algorithm 1, with the appropriate changes in order to

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



continue the check until either one cut for each value of $k$ is found or all Z-links have been checked.

3) Finally, for each cut in the graph, we could identify the size of the induced $k$-bipartition. This functionality would allow a supervisor to choose where the team has to be split, for example in navigating environments with obstacles, requiring shifts in team size and composition.

In Theorem 4, we have proven that Algorithm 1 exhibits polynomial complexity, and in particular we have shown that it is bounded above by $O(n^3)$. Thus, we can argue that this complexity is associated to the solution of the above considerations. This follows from the observation that these algorithms have the same worst case of evaluating $O(n^2)$ Z-links using an $O(n)$ search. The above formulations can be either addressed in a centralized manner (as discussed thus far), or in a decentralized manner, i.e., the mobile agents must perform the commands autonomously and guarantee rigidity in the resulting split teams. In Section IV, we will address the $k$-bipartition problem from a decentralized point of view.

## IV. Extensions for Decentralization

The decentralized version of the proposed algorithm follows directly from the centralized formulation, and therefore inherits much of its structure and primary insights. To begin, consider the following assumptions which will facilitate sound agent interaction.

*Assumption 1 (Asynchronicity):* We assume an asynchronous model of time, where each agent $i \in \mathcal{I}$ has a clock which ticks according to some distribution with bounded support (allowing for finite termination), independently of the clocks of the other agents [52] (allowing also for the possibility of delayed communication over links $(i, j) \in \mathcal{E}$). Equivalently, this corresponds to a global clock which ticks according at times $t_k,\ k \geq 1$ having time-slots $[t_k, t_{k+1})$ which discretizes system time according to clock ticks (for convenience, we will use simply $t$) [53]. Such assumptions induce asynchronicity in both the execution of local agent computation and the broadcast and reception of interagent messages.

Under the above network conditions,[4] we address the decentralization with an algorithm that operates in two phases. First, all agents search in parallel for the Z-links and build decentralized Z-link sets which guide graph exploration. Then, auctions are applied to determine leaders in order to sequentially evaluate all local Z-link sets in the graph, until a cut is found. For convenience, we will denote by $\mathbb{D}$ our decentralization of Algorithm 1, with pseudocode given in Algorithm 5.

To begin, we associate with each agent $i \in \mathcal{I}$ the following variables (with initialization indicated by $\leftarrow$).

1) $zlinks_i \leftarrow \emptyset$: Z-links set containing the Z-links to which node $i$ contributes.
2) $isLeader_i \in \{0, 1\} \leftarrow 0$: Current leadership status.
3) $beenLeader_i \in \{0, 1\} \leftarrow 0$: Prior leadership status.
4) $status_i \in \{0, 1\} \leftarrow 0$: Takes value 1 if the node has been visited in the current DFS-visit, 0 otherwise.

[4]Note that in this paper we do not fully detail interagent messaging schemes for the sake of clarity. For an illustration of messaging related to leader-based decentralization and parallelization, we refer the reader to [42].

**Algorithm 5** Decentralized Bipartition of the Graph $\mathcal{G}$

```
1: procedure SPLITGRAPHDIST(G = (V, E), k)
2:     ▷ Parallel Z-link search phase:
3:     for each i ∈ V do
4:         zlinks_i ← ZLINKSFROMNODE(i)
5:     end for
6:     ▷ Leader election phase:
7:     while ∃ i ∈ V:¬beenLeader_i and ¬cutFound do
8:         leader ← LEADERELECTION(G)
9:         LEADERPROC(leader, k)
10:    end while
11: end procedure
```

### A. Parallelized Z-Link Search

In order to determine in parallel the Z-links to which each node contributes, a two-hop neighborhood inquiry is performed (achievable with localized communication). This is the minimum number of hops that must be considered as the Z-link is by construction a two-hop structure.[5] The presence of a Z-link can be revealed according to the following proposition, which characterizes Z-links is a manner more amenable for decentralization.

*Proposition 2:* Let us consider a graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ with four vertices. $\hat{\mathcal{G}}$ is a Z-link if there exists a partition of $\hat{\mathcal{V}}$ into two sets $\hat{P}_1 = \{i \in \hat{\mathcal{V}} : |\hat{P}_1| = 2\} = \{v_1, v_2\}$ and $\hat{P}_2 = \hat{\mathcal{V}} \backslash \hat{P}_1 = \{u_1, u_2\}$ such that all the following conditions hold.

1) $u_1 \in \mathcal{N}_{v_1} \wedge u_2 \in \mathcal{N}_{v_1}$.
2) $v_2 \in \mathcal{N}_{u_1} \setminus \mathcal{N}_{u_2} \oplus v_2 \in \mathcal{N}_{u_2} \setminus \mathcal{N}_{u_1}$.

In other words, if a node $u_1$ has two neighbors $v_1$ and $v_2$ and if the node $u_2$ is in the neighborhood of either $v_1$ or $v_2$, then the graph whose nodes are $v_1$, $v_2$, $u_1$, and $u_2$ is a Z-link. The edges of this Z-links are $(v_1, u_1)$, $(u_1, v_2)$, and either $(u_2, v_1)$ or $(u_2, v_2)$. Like the centralized case, we require that the Z-link search is both exhaustive, and mutually exclusive, i.e., Z-links exist in only a single agent's Z-link set. More formally, at the end of the Z-link search the following conditions must hold:

1) $\bigcup_{i \in \mathcal{I}} z\text{links}_i = z\text{links}$;
2) $\bigcap_{i \in \mathcal{I}} z\text{links}_i = \emptyset$;

where $z$links is the set of all Z-links as determined by Algorithm 1. Satisfaction of the first condition is guaranteed by the fact that a Z-link is by definition a local structure, i.e., a structure that involves two-hop neighboring nodes that is precisely the size of the neighborhood explored by Algorithm 7. The second condition indicates that each Z-link should be added to the list of only one of the four nodes that compose it. Let us consider a Z-link whose nodes are $u_1$, $u_2$, $v_1$, and $v_2$ as described in Proposition 2. By applying such conditions, the nodes $u_2$ and $v_2$ are not able to discover the Z-link so the problem of duplication is only between $u_1$ and $v_1$. To this end it is sufficient that when $u_1$ finds a Z-link, it must query $v_1$ to know if it has been already found or not (Algorithm 7, lines 4 and 9). We reiterate that Algorithm 7 is run in parallel by all agents, and with no synchronization,

[5]As previously concluded, the Z-link structure is also the minimum-hop structure that in general admits rigidity-preserving bipartitions.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



**Algorithm 6** Leader Procedure

```
1: procedure LEADERPROC(i, k)
2:     for each zlink ∈ zlinks_i do
3:         V ← ISACUTDIST(i, zlink, k)
4:         if V = k then
5:             return
6:         else
7:             if V ≠ |V| then
8:                 ▷ Undesired cut found, mark it:
9:                 STORECUT(zlink, V)
10:            else
11:                ▷ Not a cut, allow future traversal:
12:                RESTOREEDGES(zlink)
13:            end if
14:        end if
15:    end for
16: end procedure
```

**Algorithm 7** Determine the Z-Links Incident to a Given Node

```
1: procedure ZLINKSFROMNODE(i)
2:     ▷ Check distinct neighbor pairs:
3:     for each {p1, p2} ∈ N_i × N_i | p1 ≠ p2 do
4:         if p1 has not explored i then
5:             for each j ∈ N_p1 \ N_p2 do
6:                 zlinks_i ← zlinks_i ∪ {(i, p1), (i, p2), (p1, j)}
7:             end for
8:         end if
9:         if p2 has not explored i then
10:            for each j ∈ N_p2 \ N_p1 do
11:                zlinks_i ← zlinks_i ∪ {i, p1), (i, p2), (p2, j)}
12:            end for
13:        end if
14:    end for
15: end procedure
```

yielding significant speedup to the Z-link search, a property enabled by the local Z-link structure.

### *B. Leader Election and Execution Logic*

When all Z-links have been found we make use of an auction for electing an agent in the network to become the leader, such as in [42] and [54]. Some initiating agent begins by triggering[6] an auction for electing the first lead agent and then an auction runs after each leader evaluates its Z-links. Specifically, to each agent $i \in \mathcal{I}$ we associate a bid for leadership $r_i = [i, b_i]$ with $b_i \in \mathbb{R}_{\geq 0}$ indicating the agent's fitness in becoming the new leader, with $b_i = 0$ if agent $i$ has previously been a leader, and $b_i \in \mathbb{R}_+$ otherwise. Denoting the local bid set by $\mathcal{R}_i = \{r_j \,|\, j \in \mathcal{N}_i \cup \{i\}\}$, the auction then operates according the following agreement process:

$$r_i(t^+) = \underset{r_j \in \mathcal{R}_i}{\operatorname{argmax}}(b_j) \tag{3}$$

[6]For example, in response to a supervisor command, or in response to an environmental percept such as an obstacle which must be traversed.

**Algorithm 8** DFS Visit Which Avoids Forbidden Edges

```
1: procedure SMARTDFSVISIT(i, zlink)
2:     status_i ← 1
3:     V ← 1
4:     for each j ∈ N_i do
5:         if status_j = 0 then
6:             V ← V + SMARTDFSVISIT(j, zlink)
7:         end if
8:     end for
9:     ▷ Prune based on identified cuts:
10:    for each cut ∈ cuts_i do
11:        ▷ cut reachable if transversed while avoiding zlink
12:        if ¬REACHED(cut) ∧ REACHABLE(cut) then
13:            SETREACHED(cut)
14:            V ← V + cut.size
15:            ▷ p ∈ cut and in the same partition as i
16:            if status_p = 0 ∧ REACHABLE(p) then
17:                V ← V + SMARTDFSVISIT(p, zlink)
18:            end if
19:        end if
20:    end for
21: end procedure
```

where the notation $t^+$ indicates a transition in $r_i$ after all neighboring bids have been collected through messaging. As $\mathcal{G}$ is rigid and thus connected, (3) converges uniformly to the largest leadership bid

$$r_i = \underset{r_j(0)}{\operatorname{argmax}} \big(b_j(0)\big), \qquad \forall\, i, j \in \mathcal{I} \tag{4}$$

after some finite time [55], [56]. After convergence of (3) the winning agent then takes on the leadership role, with the previous leader relinquishing its status. The bid for each agent $i$ is obtained as $b_i = (1 - \text{status}_i) * \text{fitness}_i$, where $\text{fitness}_i$ is chosen relative to the application domain (see Remark 4). The term $1 - \text{status}_i$ allows us to reduce the number of nodes that take part in the auction to those for which status $= 0$ hold. The reasons of this choice will be explained below as they are strictly related to the graph exploration algorithm.

The task of the leader is to evaluate the Z-links it has identified in order to find a cut that will induce a $k$-bipartition of the graph. This is accomplished by the execution of the Algorithm 6. With respect to the centralized version, a cut search is performed by means of a more efficient DFS-visit of the graph that is described in Algorithm 8. In the decentralized version the set of visited nodes is not available and a counter is introduced, $V \geq 0$, which stores how many nodes have been visited by the current DFS-visit. This algorithm is invoked by ISACUTDIST as depicted in Algorithm 6. The role of the function ISACUTDIST is to first make nontraversable the edges belonging to the Z-link and then to perform the DFS traversal of the graph respecting this information. As explained in detail below, we have improved the DFS visit such that we prune the search space relative to previously identified, but not desired, Z-links.

*Remark 4:* In electing leaders to inspect local Z-link sets, it is apparent that by choosing appropriate bid fitness, we can

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



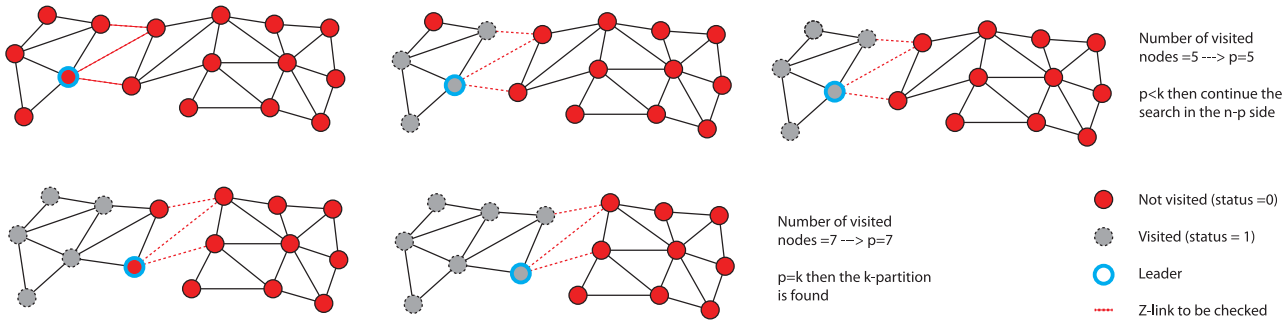


Fig. 6. Illustration of the pruning methods for the DFS of graph cuts in Algorithm 8. In this example, a previously identified, but undesirable cut yields a pruning of the search space for the desired cut.

control the order in which leaders are elected, and thus the order of Z-link evaluation. This immediately admits control of the identified partition, particularly when there are multiple possible solutions. We suggest that future work aimed at determining bid fitness for example for optimal rigid partitions, may prove useful.

### C. Identifying Rigid k-Bipartitions

When an agent becomes the leader, it must determine whether each Z-link to which it contributes is a proper cut, i.e., if it leads to a $k$-bipartition of the graph. In our decentralized approach, we have addressed the cut search problem by modifying the classical DFS to yield significantly improved complexity (as will be demonstrated in Section V). In the basic DFS algorithm, each node is marked as it is visited, in order to guarantee search correctness. However, in its standard form, DFS visitation conveys no insight into partition rigidity. We will show that with simple modifications, we are able to reduce both the size of the Z-link search space by discarding infeasible Z-links, and the time required for graph exploration using the implications of previously discovered cuts.

The general idea is to store information during DFS traversal that is supplemental to visitation, and which may be useful in future graph explorations. To begin, let us consider a cut that induces a $p$-bipartition of the graph, i.e., a split of the graph into two components with $p$ and $n-p$ nodes, respectively. We observe that all cuts whose edges belong to the $p$-side lead to $p'$-bipartitions for which $p' < p$ holds. Thus, if we desire a $k$-bipartition with $k > p$, then we can *a priori* discard all Z-links in the $p$-side of the graph. The consequence is that each undesirable cut reduces the number of Z-link candidates over which we must search. In particular, if $\hat{p}$ is the number of nodes that are reached by a graph exploration for an undesired cut, then $O(\hat{p}^2)$ Z-links can be discarded, following the conclusion of Proposition 5. Similarly, when a node of a previously discovered undesirable cut is encountered, it is unnecessary to further explore the graph beyond the edges of the cut. This fact follows from the intuition that, through previous traversal, there must be exactly $p$ (or $n-p$) nodes beyond an undesirable cut.

With the above pruning heuristics in mind, our smart graph exploration is described in Algorithm 8 (with some logic in the leader procedure, Algorithm 6), where the variable status$_i$ controls traversal and ultimately, leader election. Specifically, as introduced in Section IV-B, only nodes with status $= 0$ participate in the leader election auction, and thus status$_i$ implements the discarding of infeasible Z-links, simply by disallowing leaders who reside in infeasible $p$-partitions. The information related to undesirable cuts is stored by all involved nodes, by calling the function STORECUT (again, this action is achievable through localized communication as all cuts are Z-links). Finally, if a Z-link is not a cut, i.e., the number of visited nodes equals $n$, we cannot retrieve any useful information to prune the space search, and the leader invokes the function RESTOREEDGES such that the edges of the Z-links become traversable for future searching. For an illustration of the pruning methods discussed above, see the basic example given in Fig. 6.

*Remark 5:* Notice that the search improvements described above are easily integrated into the centralized methods of Section III, in cases where decentralization is not an application concern.

### D. Correctness and Complexity

For the sake of completeness, we conclude by analyzing the correctness, finite termination and cost properties of the $\mathbb{D}$ algorithm. First, we formally establish the stopping condition for the $\mathbb{D}$ algorithm.

*Definition 6 (Algorithm Stopping Condition):* As previously discussed, the $\mathbb{D}$ algorithm terminates upon satisfaction of the following condition:

$$f^{\mathbb{D}}_{\text{stop}} \triangleq \left\{ \left( \sum_{i=1}^{n} b_i = 0 \right) \vee \text{cutFound} \right\} \tag{5}$$

where $\sum_i b_i = 0$ indicates that all agents have been a leader, and cutFound is detected by the lead agent and conveyed to the network.

*Proposition 3 (Algorithm Termination):* By construction, executions of the $\mathbb{D}$ algorithm terminate in finite time.

*Proof:* The finiteness of execution is a direct consequence of the finiteness of the number of Z-links of each list $z\text{links}_i, \forall i \in \mathcal{I}$ and the finite convergence of auction (3), as there exists no leader reelection by construction. ■

*Proposition 4 (Algorithm Correctness):* Consider the execution of the $\mathbb{D}$ algorithm applied to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. By construction, it follows that it properly identifies (without false positives) a cut over $\mathcal{G}$ such that a $k$-bipartition is induced.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



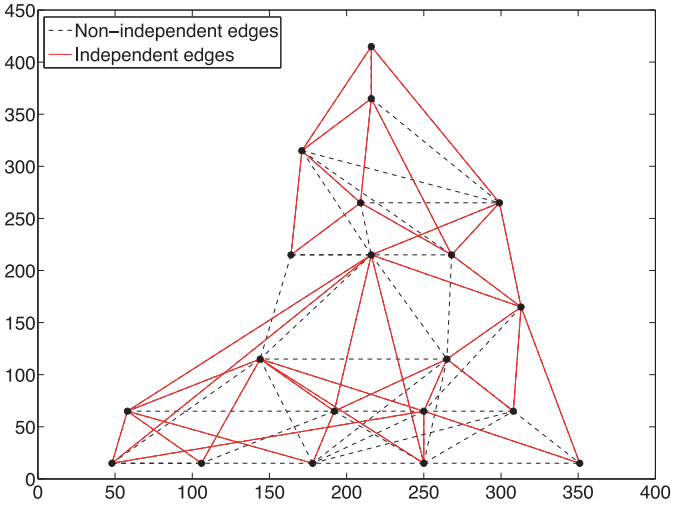


Fig. 7. Nonminimally rigid network. The solid red edges belong to the independent set.

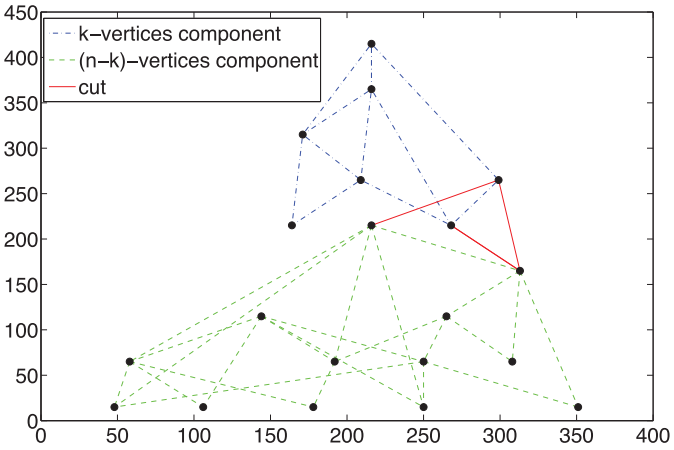


Fig. 8. The network can be partitioned into two subgraph of $k = 7$ nodes (blue dash-dotted edges) and $n - k = 13$ nodes (green dashed edges). The solid red edges represent the cut.

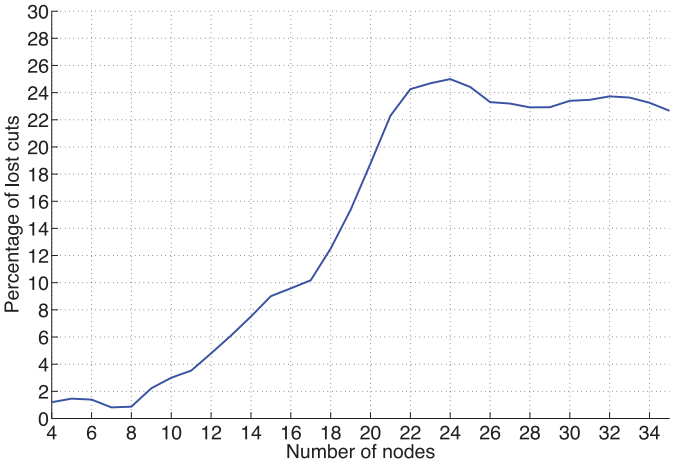


Fig. 9. Percentage of cuts that are not Z-links, for varying number of nodes.

*Proof:* By assumption we know that the graph $\mathcal{G}$ is minimally rigid, hence it is connected. The correctness of the algorithm is shown by arguing that a DFS-visit of the graph allows us to visit either the entire graph or all the nodes of a connected component (depending on which node is selected as root). For each Z-link, such a visit is performed by avoiding the edges that compose it. Thus, if less than $n$ nodes are reachable then the graph is not connected, i.e., the Z-link identifies a cut. Furthermore, from Theorem 3, we can conclude that if the number of visited nodes is exactly $k$ or $n-k$ then the cut induces a $k$-bipartition of the graph. ■

*Proposition 5 (Algorithm Complexity):* The complexity of the decentralized algorithm is bounded from above by the complexity of the centralized algorithm.

*Proof:* From Theorem 5, we know that the number of Z-links in the graph bounded by $O(n^2)$. In the decentralized algorithm each Z-link can be directly identified with cost $O(1)$. For each Z-link, a traversal of the graph is performed that is upper bounded by $O(n)$ complexity, yielding $O(n^3)$ overall, our desired result. ■

*Proposition 6 (Memory Requirements):* An upper bound for the memory requirements for each agent $i$ scales like $O(n^2)$.

*Proof:* The memory requirements for an agent $i$ is dictated by the storage of $Z$-link induced cuts known by the agent itself. Therefore, in the worst case scenario, a single leader may be able to see the entire graph in two-hops and thus such an agent would need to store all potential cuts in the graph. At this point, from Theorem 5, we know that the number of $Z$-links is $O(n^2)$, thus the thesis follows. ■

## V. Simulation Results

To demonstrate the correctness and complexity of our proposed methods, we simulated various rigid networks and identified their feasible rigid bipartitions, the results of which we now report. First, to illustrate the identification of a bipartition, consider the nonminimally rigid network in Fig. 7 with $n = 20$ nodes. Solid edges belong to the independent set, i.e., the edges that ensure the minimal rigidity of the network, where independence was determined using our decentralized methods detailed in [42]. The identified rigid bipartition is then depicted in Fig. 8: the first component has $k = 7$ nodes (blue dash-dotted edges) and the second has $n-k = 13$ nodes (green dashed edges). The solid red edges belong to the cut, i.e., they are the Z-link edges.

It may be perceived that evaluating only Z-links as candidate cuts could be very limiting in terms of identifying bipartitions in the network. For this reason, to validate our strategy, we performed a Monte Carlo simulation to determine the ratio of cuts that are overlooked in randomly generated graphs by only considering Z-links as candidate cuts. In this regard, Fig. 9 shows that the percentage of cuts that are lost is relatively low (at least statistically), that is on the order of twenty-five percent. Note that, there may be degenerate cases where Z-links fail to identify cuts, however, the Monte Carlo simulation suggests that, at least in the sense of an average case, the proposed algorithm performs well in identifying cuts in a network. In our opinion, the relatively low impact of considering only Z-links represents a reasonable tradeoff for the purposes of decentralization.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



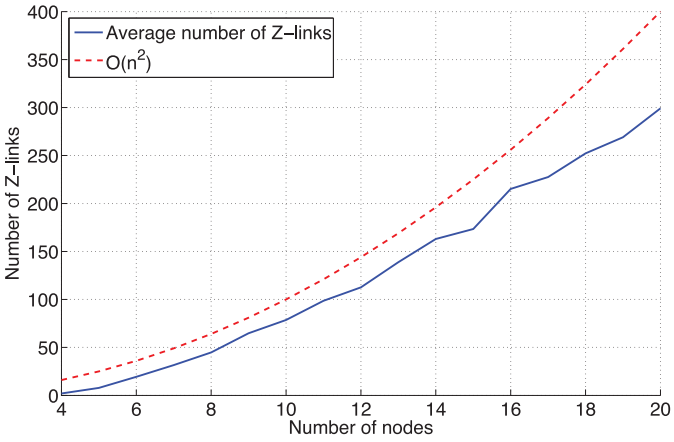


Fig. 10. Average number of Z-links found by varying the number of nodes.

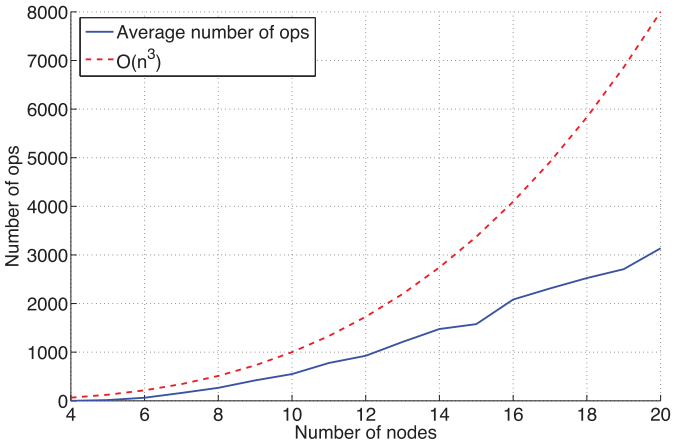


Fig. 11. Average number of operations performed by Algorithm 1, as a function of the number of nodes.

In order to analyze complexity, an extensive Monte Carlo analysis for a network with a number of nodes varying from 4 to 20 has been considered. Thousand trials were run for each configuration and average values were taken. Fig. 10 shows the average (blue solid line) of the number of Z-links found in the network by varying the number of nodes, corroborating the conclusion in Proposition 5 on the $O(n^2)$ scaling of the Z-link search problem. Our claim of overall polynomial complexity is then verified by Fig. 11 showing the average of the number of operations performed by centralized Algorithm 1, as a function of the number of nodes. It is apparent that our method exhibits $O(n^3)$ complexity, making it a feasible means of rigid splitting in multiagent teams.

Finally, our Monte Carlo analysis illustrates the performance of the decentralized algorithms proposed in Section IV. Fig. 12 depicts that the average number of operations for the parallelized Z-link search, by means of Algorithm 7, is $O(n^2)$. Fig. 13 compares the complexity of the decentralized Algorithm 5, using our proposed smart search heuristics, with a standard DFS as in the centralized Algorithm 1. It is apparent that our heuristics have asymptotically improved complexity, which is demonstrated[7] to be of order $O(n^2 \log(n))$.

[7]Our future work aims to provide a formal proof of this complexity.

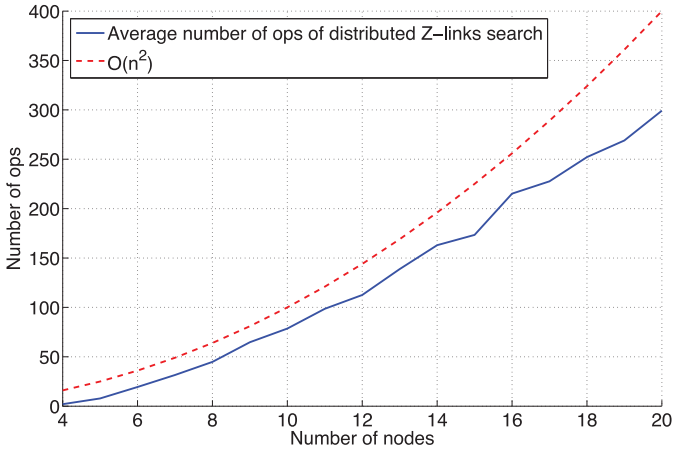


Fig. 12. Average number of ops of decentralized Z-links search, for varying number of nodes.

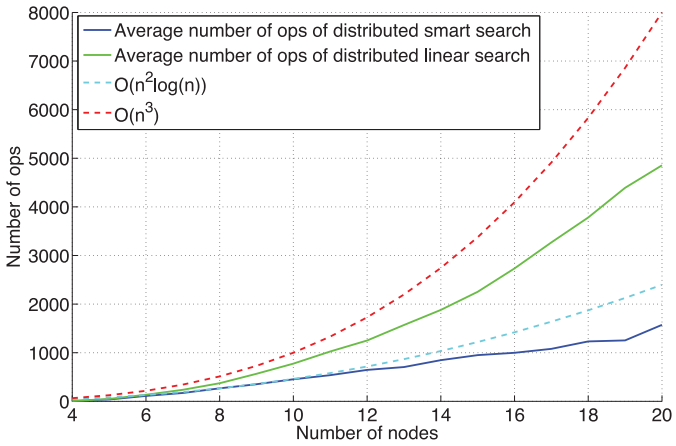


Fig. 13. Comparison of the complexity of the decentralized Algorithm 5, using our proposed smart search heuristics, with a standard DFS as in the centralized Algorithm 1.

*Remark 6:* It is worth noting that our simulation results indicate that a particular cut, if it exists, can be found in sub $O(n^3)$ time, and that the set of all rigidity-preserving cuts can be found in $O(n^3)$ time, by storing all encountered cuts. As a point of comparison, consider that evaluating infinitesimal rigidity of a graph requires spectral techniques which exhibit $O(n^3)$ complexity in a decentralized context [57].

## VI. Conclusion

In this paper, we proposed the conditions under which rigidity-preserving bipartitions are identified and iterative algorithms to perform such an identification. Motivation was derived from the implications of rigid networks for example in formation control and localizability, and the flexibility that splitting can provide for a robotic team. Our methods exploited the previously considered Z-link structure for defining rigid partitions, and a supergraph search mechanism to facilitate the discovery of network Z-links. Decentralization was then achieved through parallelization of Z-link discover, and leader election to distribute the evaluation of potential graph cuts. Finally, simulation results corroborated our

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



claims of algorithm correctness and guaranteed polynomial complexity.

Future work will focus on optimally rigid bipartitioning, multipartitioning, possible application in a 3-D workspace, integration with obstacle avoidance and detection for environment appropriate splitting, and real-world application.

## REFERENCES


[1] G. Shi, Y. Hong, and K. H. Johansson, "Connectivity and set tracking of multi-agent systems guided by multiple moving leaders," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 663–676, Mar. 2012.

[2] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot tree and graph exploration," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 707–717, Aug. 2011.

[3] J. Cortes, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.

[4] X. Wang, S. Li, and P. Shi, "Distributed finite-time containment control for double-integrator multiagent systems," *IEEE Trans. Cybern.*, vol. 44, no. 9, pp. 1518–1528, Sep. 2014.

[5] W. Zeng and M. Chow, "Resilient distributed control in the presence of misbehaving agents in networked control systems," *IEEE Trans. Cybern.*, vol. 44, no. 11, pp. 2038–2049, Nov. 2014.

[6] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 655–661, May 2005.

[7] A. Priolo, A. Gasparri, E. Montijano, and C. Sagues, "A distributed algorithm for average consensus on strongly connected weighted digraphs," *Automatica*, vol. 50, no. 3, pp. 946–951, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109813005852

[8] A. Nedic, A. E. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, Apr. 2010.

[9] Y. Tang, H. Gao, W. Zou, and J. Kurths, "Distributed synchronization in networks of agent systems with nonlinearities and random switchings," *IEEE Trans. Cybern.*, vol. 43, no. 1, pp. 358–370, Feb. 2013.

[10] S. Berman, Q. Lindsey, M. S. Sakar, V. Kumar, and S. C. Pratt, "Experimental study and modeling of group retrieval in ants as an approach to collective transport in swarm robotic systems," *Proc. IEEE*, vol. 99, no. 9, pp. 1470–1481, Sept. 2011.

[11] R. K. Williams, A. Gasparri, and B. Krishnamachari, "Route swarm: Wireless network optimization through mobility," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Chicago, IL, USA, 2014, pp. 3775–3781.

[12] M. M. Zavlanos, A. Ribeiro, and G. J. Pappas, "Network integrity in mobile robotic networks," *IEEE Trans. Autom. Control*, vol. 58, no. 1, pp. 3–18, Jan. 2013.

[13] J. Das *et al.*, "Coordinated sampling of dynamic oceanographic features with underwater vehicles and drifters," *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 626–646, 2012.

[14] A. Gasparri, F. Fiorini, M. Di Rocco, and S. Panzieri, "A networked transferable belief model approach for distributed data aggregation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 391–405, Apr. 2012.

[15] D. Di Paola, A. Gasparri, D. Naso, G. Ulivi, and F. Lewis, "Decentralized task sequencing and multiple mission control for heterogeneous robotic networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, 2011, pp. 4467–4473.

[16] R. K. Williams and G. S. Sukhatme, "Constrained interaction and coordination in proximity-limited multi-agent systems," *IEEE Trans. Robot.*, vol. 29, no. 4, pp. 930–944, Aug. 2013.

[17] M. Dorigo *et al.*, "Swarmanoid: A novel concept for the study of heterogeneous robotic swarms," *IEEE Robot. Autom. Mag.*, vol. 20, no. 4, pp. 60–71, Dec. 2013.

[18] D. Di Paola, A. Gasparri, D. Naso, and F. Lewis, "Decentralized dynamic task planning for heterogeneous robotic networks," *Auton. Robots*, pp. 1–18, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10514-014-9395-y

[19] S. Berman, A. Halasz, M. Hsieh, and V. Kumar, "Optimized stochastic policies for task allocation in swarms of robots," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 927–937, Aug. 2009.

[20] J. Fink, N. Michael, S. Kim, and V. Kumar, "Planning and control for cooperative manipulation and transportation with aerial robots," *Int. J. Robot. Res.*, vol. 30, no. 3, pp. 324–334, Mar. 2011.

[21] I. Mas, S. Li, J. Acain, and C. Kitts, "Entrapment/escorting and patrolling missions in multi-robot cluster space control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, St. Louis, MO, USA, 2009, pp. 5855–5861.

[22] C. Lim, R. Mamat, and T. Braunl, "Market-based approach for multi-team robot cooperation," in *Proc. 4th Int. Conf. Auton. Robots Agents (ICARA)*, Wellington, New Zealand, Feb. 2009, pp. 62–67.

[23] R. Olfati-Saber and R. M. Murray, "Graph rigidity and distributed formation stabilization of multi-vehicle systems," in *Proc. IEEE Conf. Decis. Control*, Las Vegas, NV, USA, Dec. 2002, pp. 2965–2971.

[24] T. Eren, W. Whiteley, A. Morse, P. N. Belhumeur, and B. D. O. Anderson, "Sensor and network topologies of formations with direction, bearing, and angle information between agents," in *Proc. IEEE Conf. Decis. Control*, 2003, pp. 3064–3069.

[25] B. Anderson, C. Yu, B. Fidan, and J. Hendrickx, "Rigid graph control architectures for autonomous formations," *IEEE Control Syst.*, vol. 28, no. 6, pp. 48–63, Dec. 2008.

[26] J. Aspnes *et al.*, "A theory of network localization," *IEEE Trans. Mobile Comput.*, vol. 5, no. 12, pp. 1663–1678, Dec. 2006.

[27] I. Shames, A. N. Bishop, and B. D. O. Anderson, "Analysis of noisy bearing-only network localization," *IEEE Trans. Autom. Control*, vol. 58, no. 1, pp. 247–252, Jan. 2013.

[28] A. R. Berg and T. Jordan, "A proof of Connelly's conjecture on 3-connected circuits of the rigidity matroid," *J. Comb. Theory B*, vol. 88, no. 1, pp. 77–97, 2003.

[29] B. Jackson and T. Jordan, "Connected rigidity matroids and unique realizations of graphs," *J. Comb. Theory B*, vol. 94, no. 1, pp. 1–29, 2005.

[30] B. Hendrickson, "Conditions for unique graph realizations," *SIAM J. Comput.*, vol. 21, no. 1, pp. 65–84, 1992.

[31] G. Laman, "On graphs and rigidity of plane skeletal structures," *J. Eng. Math.*, vol. 4, pp. 331–340, Oct. 1970.

[32] T.-S. Tay and W. Whiteley, "Generating isostatic frameworks," *Struct. Topol.*, vol. 11, pp. 21–69, 1985.

[33] D. J. Jacobs and B. Hendrickson, "An algorithm for two-dimensional rigidity percolation: The pebble game," *J. Comput. Phys.*, vol. 137, no. 2, pp. 346–365, 1997.

[34] L. Krick, M. E. Broucke, and B. A. Francis, "Stabilisation of infinitesimally rigid formations of multi-robot networks," *Int. J. Control*, vol. 82, no. 3, pp. 423–439, 2009.

[35] T. Eren, "Formation shape control based on bearing rigidity," *Int. J. Control*, vol. 85, no. 9, pp. 1361–1379, 2012.

[36] D. Zelazo and F. Allgower, "Growing optimally rigid formations," in *Proc. Amer. Control Conf.*, Montreal, QC, Canada, 2012, pp. 3901–3906.

[37] R. Ren, Y.-Y. Zhang, X.-Y. Luo, and S.-B. Li, "Automatic generation of optimally rigid formations using decentralized methods," *Int. J. Autom. Comput.*, vol. 7, no. 4, pp. 557–564, 2010.

[38] E. Boros *et al.*, "Generating minimal k-vertex connected spanning subgraphs," in *Computing and Combinatorics* (Lecture Notes in Computer Science 4598). Berlin, Germany: Springer, 2007, pp. 222–231.

[39] T. Eren, B. D. Anderson, W. Whiteley, A. S. Morse, and P. N. Belhumeur, "Merging globally rigid formations of mobile autonomous agents," in *Proc. 3rd Int. Joint Conf. Auton. Agents Multiagent Syst.*, vol. 3. New York, NY, USA, 2004, pp. 1260–1261.

[40] T. Eren, B. D. Anderson, A. S. Morse, W. Whiteley, and P. N. Belhumeur, "Operations on rigid formations of autonomous agents," *Commun. Inf. Syst.*, vol. 3, no. 4, pp. 223–258, 2004.

[41] W. Ong, C. Yu, and B. D. O. Anderson, "Splitting rigid formations," in *Proc. 48th IEEE Conf. Decis. Control; 28th Chinese Control Conf. (CDC/CCC)*, Shanghai, China, Dec. 2009, pp. 859–864.

[42] R. Williams, A. Gasparri, A. Priolo, and G. Sukhatme, "Evaluating network rigidity in realistic systems: Decentralization, asynchronicity, and parallelization," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 950–965, Aug. 2014.

[43] H. Gluck, "Almost all simply connected closed surfaces are rigid," in *Geometric Topology*. Berlin, Germany: Springer, 1975, pp. 225–239.

[44] R. K. Williams, A. Gasparri, A. Priolo, and G. S. Sukhatme, "Decentralized generic rigidity evaluation in interconnected systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, Japan, 2013, pp. 5093–5099.

[45] L. Henneberg, *Die Graphische Statik Der Starren Systeme*. New York, NY, USA: Johnson Reprint, 1911.

[46] R. Haas *et al.*, "Planar minimally rigid graphs and pseudo-triangulations," *Comput. Geom.*, vol. 31, no. 12, pp. 31–61, 2005.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



[47] B. Anderson, C. Yu, B. Fidan, and J. Hendrickx, "Rigid graph control architectures for autonomous formations," *IEEE Control Syst.*, vol. 28, no. 6, pp. 48–63, Dec. 2008.
[48] R. K. Williams, A. Gasparri, A. Priolo, and G. S. Sukhatme, "Distributed combinatorial rigidity control in multi-agent networks," in *Proc. IEEE Conf. Decis. Control*, Florence, Italy, 2013, pp. 6061–6066.
[49] B. Schwikowski and E. Speckenmeyer, "On enumerating all minimal solutions of feedback problems," *Discrete Appl. Math.*, vol. 117, nos. 1–3, pp. 253–265, 2002.
[50] L. Khachiyan *et al.*, "Generating cut conjunctions and bridge avoiding extensions in graphs," in *Algorithms and Computation* (Lecture Notes in Computer Science 3827). Berlin, Germany: Springer, 2005, pp. 156–165.
[51] T. W. Mather and M. A. Hsieh, "Distributed robot ensemble control for deployment to multiple sites," in *Proc. Robot. Sci. Syst.*, Los Angeles, CA, USA, 2011.
[52] A. Nedic, "Asynchronous broadcast-based convex optimization over a network," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1337–1351, Jun. 2011.
[53] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis and applications," in *Proc. 24th IEEE INFOCOM Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 3. Miami, FL, USA, 2005, pp. 1653–1664.
[54] R. K. Williams and G. S. Sukhatme, "Topology-constrained flocking in locally interacting mobile networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, 2013, pp. 2002–2007.
[55] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Ann. Oper. Res.*, vol. 14, no. 1, pp. 105–123, 1988.
[56] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, "A distributed auction algorithm for the assignment problem," in *Proc. IEEE Conf. Decis. Control*, Cancun, Mexico, 2008, pp. 1212–1217.
[57] D. Zelazo, A. Franchi, H. H. Bülthoff, and P. R. Giordano. (2013). *Decentralized Rigidity Maintenance Control With Range-Only Measurements for Multi-Robot Systems*. [Online]. Available: http://arxiv.org/abs/1309.0535

**Daniela Carboni** received the master's degree in computer science and automation engineering from the University of Roma Tre, Rome, Italy, in 2011, where she is currently pursuing the Ph.D. degree in informatics and automation from the Doctorate School of Engineering.

Her current research interests include multirobot systems and sensor networks.

**Ryan K. Williams** (S'11) received the B.S. degree in computer engineering from Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2005 and 2014, respectively.

He is currently a Research Affiliate with the Robotic Embedded Systems Laboratory, University of Southern California. His current research interests include control, cooperation, and intelligence in distributed multiagent systems, topological methods in cooperative phenomena, and distributed algorithms for optimization, estimation, inference, and learning. He has been cited by various news outlets, including the *L.A. Times*, and has a patent pending for his research on high-speed autonomous underwater vehicles.

Dr. Williams was the recipient of the Viterbi Fellowship.

**Andrea Gasparri** (M'09) received the *cum laude* degree in computer science and the Ph.D. degree in computer science and automation, both from the University of Roma Tre, Rome, Italy, in 2004 and 2008, respectively.

He has been a Visiting Researcher at several institutions, including the Universite Libre de Bruxelles, Brussel, Belgium, City College of New York, New York, NY, USA, and the University of Southern California, Los Angeles, CA, USA. He is currently an Assistant Professor with the Department of Engineering, University of Roma Tre. His current research interests include mobile robotics, sensor networks, and networked multiagent systems.

Dr. Gasparri was the recipient of the Italian grant FIRB Futuro in Ricerca 2008 for the project Networked Collaborative Team of Autonomous Robots funded by the Italian Ministry of Research and Education (MIUR).

**Giovanni Ulivi** (M'84) received the Laurea degree in electrical engineering degree from the University of Rome La Sapienza, Roma, Italy.

He has been a Full Professor with the Department of Computer Science and Automation, University of Roma TRE, Rome, Italy, since 2000, where he teaches basic automatic control, measures for automation, and robotics in the Engineering courses. From 2004 to 2013, he was the Head of the Department with the same university. His current research interests include mobile robotics and sensory data fusion. In the above fields, he coordinates the work of several Ph.D. students. He has authored over 100 papers and has published in international journals and conferences.

Prof. Ulivi is a member of IFACs TC on Robotics, TC on Autonomous Vehicles, and TC on Instrumentation. He was a member of the Italian Committee for Legal Metrology.

**Gaurav S. Sukhatme** (F'11) received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology Bombay, Mumbai, India, and the M.S. and Ph.D. degrees in computer science from the University of Southern California (USC), Los Angeles, CA, USA.

He is a Professor of Computer Science (joint appointment in Electrical Engineering) with the USC. He is the Co-Director of the USC Robotics Research Laboratory and the Director of the USC Robotic Embedded Systems Laboratory, which he founded in 2000. He has served as a Principal Investigator (PI) for various NSF, DARPA, and NASA grants. He is a Co-PI with the Center for Embedded Networked Sensing, an NSF Science and Technology Center. His current research interests include robot networks with applications to environmental monitoring. He has published extensively in the above and related areas.

Prof. Sukhatme was the recipient of the NSF CAREER Award and the Okawa Foundation Research Award. He is one of the founders of the Robotics: Science and Systems Conference. He was a Program Chair of the 2008 IEEE International Conference on Robotics and Automation. He is a Program Chair of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. He is an Editor-in-Chief of *Autonomous Robots* and has served as an Associate Editor of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, the IEEE TRANSACTIONS ON MOBILE COMPUTING, and on the editorial board of the IEEE Pervasive Computing.