**Preprints of the 19th World Congress**
**The International Federation of Automatic Control**
**Cape Town, South Africa. August 24-29, 2014**

# Identifying Rigidity-Preserving Bipartitions in Planar Multi-Robot Networks ⋆

**Daniela Carboni** * **Ryan K. Williams** ** **Andrea Gasparri** *
**Giovanni Ulivi** * **Gaurav S. Sukhatme** **

* *University of Roma Tre, Roma, 00146, Italy*
*(carboni; gasparri; ulivi)@dia.uniroma3.it*
** *University of Southern California, Los Angeles, CA 90089 USA*
*(rkwillia; gaurav)@usc.edu*

**Abstract:** In this paper, we consider the problem of identifying a bipartition of a planar multi-robot network, such that the resulting two sub-teams are rigid networks. As opposed to approaching the network splitting problem constructively, we instead 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 direction towards which a graph search is applied. Adapting a supergraph search mechanism from the set generation literature, we then provide a methodology for discerning graphs cuts that represent valid rigid bipartitions. Finally, full algorithm details and pseudocode are provided, together with simulation results that verify correctness and demonstrate complexity.

## 1. INTRODUCTION

Collaborative networks of intelligent robotic systems have become a substantial focus for researchers, particularly within recent years. Driving the multi-robot movement are the rapid advancement in computation and communication resources, and the implications that collaborative systems have for fundamentally important applications. Examples of such applications include tracking and coverage [Brass et al., 2011, Cortes et al., 2004], formation control and leader-following [Olfati-Saber and Murray, 2002, Cao et al., 2012], and state consensus and optimization [Ren and Beard, 2005, Nedic et al., 2010]. Additionally, multi-robot systems generally hold promise for significant advantages over single-agent solutions, including heterogeneity in mobility and sensing, redundancy and thus flexibility, and scalability [Di Paola et al., 2011, Williams and Sukhatme, 2013].

In this work, we are concerned with identifying two properties of the graph that describes a *planar* multi-robot network. First, given a single team of connected robots, 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 robotic 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 in the context of task assignment (e.g., [Berman et al., 2009]) is compelling as it would enable task-centric collaboration per-team and also the decoupling of teams and tasks across multiple spatiotemporal scales.

⋆ This work was partially supported by the Italian grant FIRB "Futuro in Ricerca", project NECTAR "Networked Collaborative Team of Autonomous Robots", code RBFR08QWUV, funded by the Italian Ministry of Research and Education (MIUR).

Our second and arguably most important concern is that each partitioned team is *rigid* (which also implies *connected* sub-teams). Graph rigidity has important implications particularly for mission objectives requiring collaboration, e.g., formation stability [Olfati-Saber and Murray, 2002, Anderson et al., 2008] and relative localizability [Eren et al., 2003, Shames et al., 2013]. Rigidity is also a necessary component of *global rigidity* [Hendrickson, 1992], which can further strengthen the guarantees of formation stability and localizability. The general study of rigidity has a rich history in science, mathematics, and engineering [Laman, 1970, Tay and Whiteley, 1985, Hendrickson, 1992, Jacobs and Hendrickson, 1997]. In [Tay and Whiteley, 1985] combinatorial operations are defined which preserve rigidity, with works such as [Olfati-Saber and Murray, 2002, Anderson et al., 2008] extending the ideas to multi-robot splitting and formation control [Krick et al., 2009, Eren, 2012]. In [Zelazo and Allgower, 2012] an algorithm is proposed for generating rigid graphs in the plane based on the Henneberg construction [Tay and Whiteley, 1985]. Similarly, [Ren et al., 2010] 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.*

Thus, we propose in this work an iterative algorithm to determine the existence of a rigidity-preserving bipartition of a planar graph, with guaranteed polynomial complexity. While previous work has provided *constructive* intuition for rigid splitting and rejoining (e.g., [Olfati-Saber and Murray, 2002]), instead in this work 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 par-

titions. Then, we reason on determining through graph search, when a graph cut represents a rigidity-preserving bipartition of the graph. Finally, full algorithm details and pseudocode are provided as well as simulation results and Monte Carlo analysis that verifies our claims of complexity and correctness.

## 2. PRELIMINARIES

Consider a system composed of $n$ robots (agents) indexed by $\mathcal{I} = \{1, \dots, n\}$ operating in $\mathbb{R}^2$, each possessing communication capabilities, denoting by $(i, j)$ a bi-directional communication link between agents $i$ and $j$. The primary concern of this work is the *rigidity* property of the underlying graph $\mathcal{G}$ describing the network topology [1], again given the fundamental guarantees that rigid graphs provide for example in both localizability and formation stability of multi-robot systems [Anderson et al., 2008]. The first *combinatorial* characterization of graph rigidity was described by Laman in [Laman, 1970], and is summarized as follows (also called *generic rigidity*) [2]:

***Theorem** 1.* [Laman, 1970] 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 non-empty subset $\hat{\mathcal{E}} \subseteq \bar{\mathcal{E}}$, we have $|\hat{\mathcal{E}}| \leq 2k - 3$, where $k$ is the number of nodes in $\mathcal{V}$ that are endpoints of $(i, j) \in \hat{\mathcal{E}}$.

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. Further, edges that belong to the edge set $\bar{\mathcal{E}}$ are called *independent*, i.e., the edges necessary to establish a graph's rigidity.

The task of determining the rigidity of $\mathcal{G}$, was originally solved in a centralized manner by [Jacobs and Hendrickson, 1997], with decentralization and parallelization achieved in [Williams et al., 2013a,b]. Our goal in this work 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** 2.1.* A *bipartition* of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a division of the graph into two disjoint components, namely $\mathcal{G}_1(\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2(\mathcal{V}_2, \mathcal{E}_2)$, such that $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ and $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$. We refer to a *k-bipartition* as the case in which we choose $k \in \mathbb{R}_+$ such that $|\mathcal{V}_1| = k$ and $|\mathcal{V}_2| = n - k$, dictating the size of the resulting partitions.

Note that if a graph can be partitioned, it follows that there exists a *cut*:

***Definition** 2.2.* A *cut* $C = \{(i, j) \in \mathcal{E} \,|\, i \in \mathcal{V}_1, j \in \mathcal{V}_2\}$, is a set of edges that must to be removed from $\mathcal{G}$ in order to obtain the disjoint components.

To conclude, we recall two useful results and intuition from [Olfati-Saber and Murray, 2002] which will aid us in determining proper graph partitions.

[1] We provide a brief overview of rigidity theory here. We direct the reader to [Tay and Whiteley, 1985, Laman, 1970] for a technical primer on the subject.

[2] The extension of Laman's conditions to higher dimensions is at present an unresolved problem in rigidity theory.

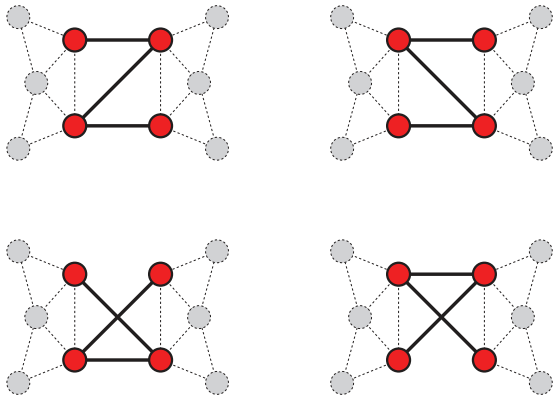

Fig. 1. [Olfati-Saber and Murray, 2002, Figure 7] A Z-link in four possible configuration of edges.

***Definition** 2.3.* [Olfati-Saber and Murray, 2002] We refer to a bipartite graph $\mathcal{K}_{2,2}$ with three edges, shown in Fig. 1, as a Z-link.

***Corollary** 1.* [Olfati-Saber and Murray, 2002] Two minimally rigid graphs that are connected using a Z-link construct a minimally rigid graph

## 3. IDENTIFYING RIGID BIPARTITIONS

We now turn our attention to exploiting the fundamental results of rigidity theory to describe an algorithm that identifies rigid bipartitions in planar networks. First, our target problem is stated formally as follows:

**Problem** 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.

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*. According to the minimal rigidity assumption, we know that

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

Hence we can conclude that $|\mathcal{G}_1| + |\mathcal{G}_2| = 2n - 6$, and thus there must exist 3 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 $(\mathcal{G}_1 \cup \mathcal{G}_2)$. Therefore, we can argue that a feasible cut must be composed of exactly three edges, as formalized by the following theorem. 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** 2.* 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. 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:

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

From Theorem 2, we can argue that given a cut $C$ over minimally rigid graph $\mathcal{G}$, if the cut is composed by 3 edges, then the two induced components, namely $\mathcal{G}_1$ and $\mathcal{G}_2$, are minimally rigid. Among all 3 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. While we present our algorithms (Section 3.1) in a centralized form for clarity, our future goal is to extend the methods to decentralized contexts, and thus our Z-link focus.

Now that we understand the structural elements we seek in order to partition the graph, let us characterize our expectations in searching for such cuts:

***Remark** 1.* An upper bound to the number of Z-links in the 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

$$\binom{|\mathcal{E}|}{3} = \frac{|\mathcal{E}|!}{3!\,(|\mathcal{E}|-3)!} = \frac{|\mathcal{E}|(|\mathcal{E}|-1)(|\mathcal{E}|-2)}{6} = O(|\mathcal{E}|^3)$$

as the graph is minimally rigid, with $|\mathcal{E}| = 2n-3$.

Once a Z-link has been identified it is finally 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 non cuts, 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 2, 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 simplified depth-first search (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:

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

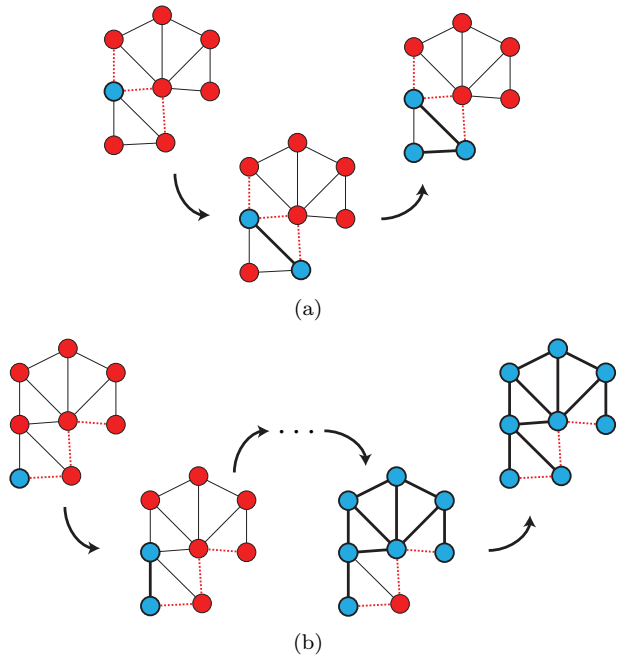


Fig. 2. For this example $n = 8$ and $k = 3$. In red are the Z-link edges. In blue are the visited nodes. (a) At the end of the graph exploration exactly $k$ vertices have been visited, hence the Z-link is a proper cut. (b) All the $n$ vertices have been explored hence the Z-link is not a cut.

This intuition is illustrated in Figure 2. The graph has $n = 8$ vertices and for the bipartition it is chosen $k = 3$. Red edges belong to the Z-link. A vertex is blue if it is visited. In case (a) exactly $k$ vertices can be visited, in fact the selected Z-link is a proper cut. In case (b) all the $n$ vertices are reachable indicating that the Z-link is not a cut.

### *3.1 Algorithm Details and Pseudocode*

The proposed algorithm 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 [Boros et al., 2007], known simply as the $X-e+Y$ method. Briefly, the method is concerned with generating all members of the set $\mathcal{F} = \{X \,|\, X \subseteq \mathcal{E}$ is a minimal set s.t. $\pi(X) = 1\}$, where $\pi(X) : 2^{\mathcal{E}} \rightarrow \{0,1\}$ is a monotone boolean function indicating the satisfaction of some desired property $X$. In our case, we desire *quads*, that is sets of four nodes, which contain at least one Z-link, and thus the $X-e+Y$ method applied in this context will generate all Z-link containing quads in $\mathcal{G}$. 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. 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, as in Algorithm 2. For each node $i \in \mathcal{V}$ the 2-hop neighborhood is considered in order to identify the set of all quads as shown in algorithm 3, in which 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 $|quads| = \binom{p}{4}$.

**Algorithm 1** Partition a graph into two rigid components

```
1: procedure splitGraph(G = (V, E), k)
2:     [quad, zls] ← initialSupernode(G)
3:     for each zlink ∈ zls do
4:         if checkZLink(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, zls] ← swapSetFromQuad(G, quad, v)
14:            for nbrQuad ∈ S do
15:                if nbrQuad ∩ (P ∪ Q) = ∅ then
16:                    for each zlink ∈ zls do
17:                        if checkZLink(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
```

**Algorithm 2** Finds a quad with Z-links

```
1: procedure initialSupernode(G)
2:     for each i ∈ V do
3:         quads ← quadsFromNode(G, i)
4:         for each quad ∈ quads do
5:             zlinks ← zLinksFromQuad(G, quad)
6:             if zlinks = ∅ then
7:                 return {quad, zlinks}
8:             end if
9:         end for
10:    end for
11: end procedure
```

**Algorithm 3** Determine the quads that a given node participates in

```
1: procedure quadsFromNode(G, i)
2:     nodes ← {i} ∪ N(i)
3:     for each j ∈ N(i) do
4:         nodes ← nodes ∪ N(j)
5:     end for
6:     quads ← combosWithoutRepetition(nodes, 4)
7:     return quads
8: end procedure
```

Next, for each $quad \in quads$ the presence of a Z-link is revealed as in Algorithm 4. We know that there are only three ways to partition a quad whose set of nodes is $\{1\ 2\ 3\ 4\}$:

- $\{1, 2\}$ and $\{3, 4\}$
- $\{1, 3\}$ and $\{2, 4\}$
- $\{1, 4\}$ and $\{2, 3\}$

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

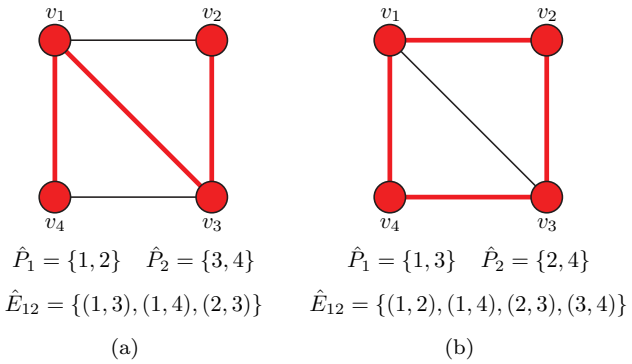


Fig. 3. An example of the conditions stated in Proposition 1. The red edges belong to the set $\mathcal{E}_{12}$ so they are the edges that will be removed. In case (a) a Z-link is found whereas in case (b) it isn't possible since $|\mathcal{E}_{12}| = 4$.

***Proposition** 1.* Let us consider a graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ with 4 vertices. $\hat{\mathcal{G}}$ is a Z-link if there exist 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 $\hat{\mathcal{E}}_{12} = \{(i, j) \in \mathcal{E} : i \in \hat{P}_1, j \in \hat{P}_2\}$, all the following conditions hold:

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

In other words, if a pair has exactly three outgoing edges, where each nodes contributes then there is a Z-link. The conditions introduced in Proposition 1 are explained through an example in Figure 3. Two possible bipartitions of a graph with 4 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 case (a) a Z-link is found by partitioning the graph into $P_1 = \{1, 2\}$ and $P_2 = \{3, 4\}$. In case (b) a Z-link cannot be found since $|\mathcal{E}_{12}| = 4$, in fact the removal of the red edges doesn't lead to a bipartite graph. 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 yield two disjoint components, one with $k$ nodes and the other with $n - k$ nodes. This operation is described in Algorithm 6. As introduced above, a simplified DFS is performed. The depth-first-search is simplified by the fact that only one node is taken into account 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 means 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. To close, we now provide a proof of the expected complexity of our proposed algorithm.

***Theorem** 3.* Consider a minimally rigid planar graph $\mathcal{G}$. 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** For brevity, we provide here a condensed reasoning of algorithm complexity. From [Boros et al., 2007] it follows that the search mechanism illustrated in Algorithm 1 runs

**Algorithm 4** Returns the Z-links in a given set of 4 nodes

```
1: procedure zLinksFromQuad(G, quad, parts)
2:     for each pair ∈ parts do                ▷ pair = {p1, p2}
3:         if |N(quad(p1))| + |N(quad(p2))| = 3 and
   |N(quad(p1))| ≥ 1 and |N(quad(p2))| ≥ 1 then
4:             for each j ∈ N(quad(p1)) do
5:                 zlink ← zlinks ∪ (quad(p1), j)
6:             end for
7:             for each j ∈ N(quad(p2) do
8:                 zlink ← zlinks ∪ (quad(p2), j)
9:             end for
10:            zlinks ← zlink
11:        end if
12:    end for
13:    return zlinks
14: end procedure
```

**Algorithm 5** Generates the nodes which yield Z-links when swapped

```
1: procedure swapSetFromQuad(G, quad, v, parts)
2:     for each u ∈ N(quad(v)) do
3:         nodes ← nodes ∪ N(u)
4:     end for
5:     nodes ← nodes \ quad
6:     for each node ∈ nodes do
7:         swapQuad ← quad
8:         swapQuad(v) ← node
9:         zls ← zLinksFromQuad(G, swapQuad, parts)
10:        if zls ≠ ∅ then
11:            zlinks ← zlinks ∪ zls
12:            swap ← swap ∪ node
13:        end if
14:    end for
15:    return {swap, zlinks}
16: end procedure
```

in polynomial time if and only if the set of reachable quads returned by *swapSetFromQuad* can be generated in polynomial time. 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 4). Thus, we conclude that *swapSetFromQuad* runs in polynomial time, and our result follows. □

## 4. 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. For analyzing complexity, a Monte Carlo analysis for a network with a number of nodes varying from 4 to 20 has been considered. 1000 trials were run for each configuration and average values were taken. Figure 4 shows the average (blue solid line) of the number Z-links found in the network by varying the number of nodes, providing evidence of the scale of the Z-link search problem, and ultimately complexity. In particular, the information concerning the average values is compared with the curve $O(n^2)$ (red dashed line) in order to demonstrate that this is an upper bound for the cardinality of a network's Z-links. Clearly, this working

**Algorithm 6** Check if a Z-link is a proper cut over a graph

```
1: procedure checkZLink(G = (V, E), k, zl)
2:     Ê ← E \ zl
3:     Ĝ ← (V, Ê)
4:     visited← simplifiedDFS(Ĝ, v)    ▷ v is an arbitrary
   node of zl
5:     if |visited| = k then
6:         G1 ← visited
7:         G2 ← V \ visited
8:         return true
9:     elseIf |visited| = n − k
10:        G1 ← V \ visited
11:        G2 ← visited
12:        return true
13:    end if
14: end procedure
```

**Algorithm 7** Explore a graph with a DFS approach

```
1: procedure simplifiedDFS(G = (V, E), u)
2:     for each i ∈ V do
3:         state(i) ← 0
4:     end for
5:     simplifiedDFSVisit(G, u)
6:     visited ← {v ∈ V : state(v) = 1}
7:     return visited
8: end procedure
```

**Algorithm 8** Performs a depth-first visit of a graph node

```
1: procedure simplifiedDFSVisit(G = (V, E), u)
2:     state(u) ← 1
3:     for each v ∈ N_u do
4:         if state(v) = 0 then
5:             simplifiedDFSVisit(G, v)
6:         end if
7:     end for
8: end procedure
```

bound is far superior to the combinatorial bound given previously; revising the theoretical bound is a topic of our current work. Our claim of overall polynomial complexity is then verified by Figure 5 showing the average of the number of operations performed by 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 multi-robot teams.

## 5. CONCLUSIONS AND FUTURE WORK

In this paper we proposed the conditions under which rigidity-preserving bipartitions are identified and an iterative algorithm to perform such identification. Motivation was derived both from the implications of rigid networks for example in formation control and localizability, but also 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. Finally, simulation results corroborated our claims of algorithm correctness and guaranteed polynomial complexity. Future work will focus on extending the algorithm to decentralized domains, considering possible application

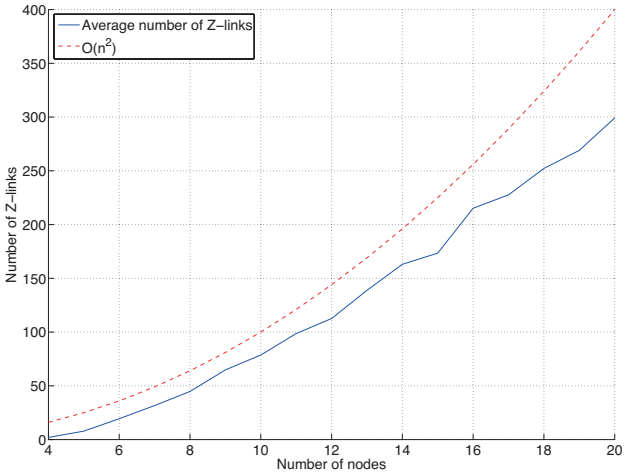


Fig. 4. The average number of Z-links found by varying the number of nodes.

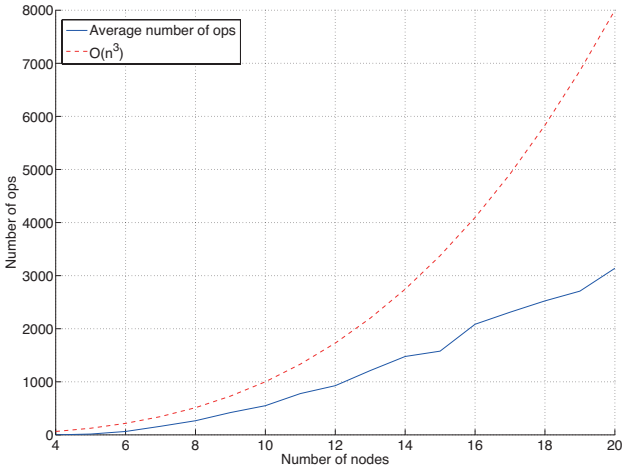


Fig. 5. The average number of operations performed by Algorithm 1, as a function of the number of nodes.

in a three-dimensional workspace, and finally real-world experimentation to solidify our results.

## REFERENCES

B. Anderson, C. Yu, B. Fidan, and J. Hendrickx. Rigid graph control architectures for autonomous formations. *Control Systems, IEEE*, 28(6), 2008.

S. Berman, A. Halasz, M. Hsieh, and V. Kumar. Optimized stochastic policies for task allocation in swarms of robots. *Robotics, IEEE Transactions on*, 25(4):927–937, 2009.

E. Boros, K. Borys, K. Elbassioni, V. Gurvich, K. Makino, and G. Rudolf. Generating minimal k-vertex connected spanning subgraphs. In *Computing and Combinatorics*, volume 4598 of *Lecture Notes in Computer Science*, pages 222–231. Springer Berlin Heidelberg, 2007.

P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, 2011.

Y. Cao, W. Ren, and M. Egerstedt. Distributed containment control with multiple stationary or dynamic leaders in fixed and switching directed networks. *Automatica*, 48(8):1586–1597, 2012.

J. Cortes, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

D. Di Paola, A. Gasparri, D. Naso, G. Ulivi, and F. Lewis. Decentralized task sequencing and multiple mission control for heterogeneous robotic networks. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4467–4473, 2011.

T. Eren. Formation shape control based on bearing rigidity. *International Journal of Control*, 85(9):1361–1379, 2012.

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 *IEEE Conference on Decision and Control*, 2003.

B. Hendrickson. Conditions for unique graph realizations. *SIAM J. Comput.*, 21(1):65–84, 1992.

D. J. Jacobs and B. Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. *J. Comput. Phys.*, 137(2):346–365, 1997.

L. Krick, M. E. Broucke, and B. A. Francis. Stabilisation of infinitesimally rigid formations of multi-robot networks. *International Journal of Control*, 82(3):423–439, 2009.

G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:331–340, 1970. ISSN 0022-0833.

A. Nedic, A. E. Ozdaglar, and P. A. Parrilo. Constrained Consensus and Optimization in Multi-Agent Networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

R. Olfati-Saber and R. M. Murray. Graph rigidity and distributed formation stabilization of multi-vehicle systems. In *IEEE Conference on Decision and Control*, pages 2965–2971, Dec. 2002.

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.*, 7(4):557–564, 2010.

W. Ren and R. W. Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, 2005.

I. Shames, A. N. Bishop, and B. D. O. Anderson. Analysis of Noisy Bearing-Only Network Localization. *IEEE Transactions on Automatic Control*, 58(1), 2013.

T.-S. Tay and W. Whiteley. Generating Isostatic Frameworks. *Structural Topology*, 1985.

R. K. Williams and G. S. Sukhatme. Constrained Interaction and Coordination in Proximity-Limited Multi-Agent Systems. *IEEE Transactions on Robotics*, 2013.

R. K. Williams, A. Gasparri, A. Priolo, and G. Sukhatme. Evaluating Network Rigidity in Realistic Systems: Decentralization, Asynchronicity, and Parallelization. *IEEE Transactions on Robotics (to appear)*, 2013a.

R. K. Williams, A. Gasparri, A. Priolo, and G. S. Sukhatme. Decentralized Generic Rigidity Evaluation in Interconnected Systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013b.

D. Zelazo and F. Allgower. Growing optimally rigid formations. In *American Control Conference*, 2012.