Joint 48th IEEE Conference on Decision and Control and
28th Chinese Control Conference
Shanghai, P.R. China, December 16-18, 2009

Splitting Rigid Formations

Wilson Ong, Changbin Yu, and Brian D.O. Anderson

Abstract— **Consider a network of sensors able to move in 2-dimensional space. We may aim to impose distance constraints between certain sensors to ensure every pair of sensors maintain their distance from one another under any continuous movement. This property is known as rigidity. Rigidity may be required to ensure that no sensor will move out of range of any other sensor during movement. However, there arise situations which require us to decompose a rigid formation into two or more rigid sub-formations, perhaps to avoid an obstacle, to pursue different missions, or to allow merging of part of the original formation with another formation. The paper demonstrates that it is not always possible to decompose a rigid formation into rigid sub-formations without adding new distance constraints. The paper also discusses how to decompose a formation into connected but not necessarily rigid sub-formations (to which edges could be added to ensure rigidity of the sub-formations). We show it is always possible to decompose a rigid formation into two connected sub-formations, one of which has *arbitrary* order, without the addition of any new distance constraints, and we present an algorithm to do this. Although the sub-formations may not be rigid, the connectedness property ensures that no agent or group of agents can deviate too far away from the rest of the agents in the same connected component, and any agent can communicate with any other agent (perhaps via intermediate agents) in the same sub-formation. This will allow rigidity to be recovered within each connected sub-formation by applying existing algorithms.**

I. Introduction

There are situations where we want a particular multi-agent system, modelled by a graph, to preserve the distance between any two vertices during movement (i.e. to move in formation). Consider for example a network of sensors able to move in 2-dimensional space. We may impose distance constraints between certain pairs of sensors (which might be agents such as robots, or perhaps land, underwater or aerial vehicles) to ensure they remain within a fixed range of one another. However it may also be desirable for every other pair of sensors, even those without explicit distance constraints imposed on them, to maintain their distance from one another under any continuous movement of the sensors. This will ensure that no sensor will move out of range of any other sensor during movement. Also, multiple vehicles can be used to synthesize a large antenna for receiving electromagnetic or acoustic signals, allowing better resolution or higher sensitivity than might be possible with an antenna carried by a single vehicle. Accurate knowledge and control of the agents' relative positions of the formation is essential for this application. More examples where these type of multi-agent/sensor formations are required can be found in [1]–[3]. The property that the distance between any two sensors remains constant under any continuous movement of the sensors is called rigidity. Much research has already been done in this area, e.g. [4]–[7].

There may arise situations which require us to decompose the rigid formation into two or more rigid sub-formations. Suppose we need to avoid an obstacle, and there is insufficient space for the rigid formation to go around the obstacle. In this situation we can first decompose the rigid formation into rigid sub-formations, and then merge these sub-formations to form a single rigid formation again once we have bypassed the obstacle. We may also want to decompose a rigid formation into rigid sub-formations to allow one of them to merge with another formation. Alternatively we may need to decompose into rigid sub-formations so each sub-formation can work on a different task, where the tasks require the sub-formations to be rigid. However, we show it is not always possible to decompose a rigid formation into rigid sub-formations without adding new edges. Note that there exist methods in the literature for recovering rigidity within each sub-formation – a 'reduction sequence' algorithm is given in [4], and another method known as 'double patching' is given in [8]; both methods require the addition of new edges within the sub-formations. Though of minor relevance to this paper, we note too that a method for merging the rigid sub-formations is given in [5].

In this paper we investigate whether it is possible to decompose a rigid formation into rigid sub-formations, without adding any new edges. As mentioned, we show this is not always possible. However we show it is always possible to decompose a rigid formation into two connected sub-formations, one of which has *arbitrary* order, without the addition of any new edges. It may be that the addition of new edges cannot be achieved due to the limited capabilities of the agents. However, we may be required to decompose a large formation into smaller connected sub-formations so each sub-formation can perform a separate task that does not require rigidity, but only connectivity. Although the sub-formations may not be rigid, the connectedness property ensures that the agents within each sub-formation are 'linked together'. This ensures that no agent or group of agents can deviate too far away from the rest of the agents in the same connected component, which is important if we wish to keep the agents in a contained area. Connectedness will also ensure that any agent can communicate with any other agent

Wilson Ong is with the School of Mathematics and Statistics, the University of Western Australia, Crawley WA 6009 (though this work was undertaken at the Australian National University as part of a summer research program). E-mail: ongw03@maths.uwa.edu.au

Changbin Yu is with the Australian National University, Canberra ACT 2600. E-mail: brad.yu@anu.edu.au

Brian D.O. Anderson is with the Australian National University and National ICT Australia. E-mail: brian.anderson@anu.edu.au

(perhaps via intermediate agents) in the same sub-formation.

This paper is organised as follows: In section II we review concepts in rigidity theory we require for the results presented in this paper. In section III we investigate decomposing rigid formations into rigid sub-formations. We show that this cannot always be achieved. In section IV we look at decomposing a rigid formation into connected sub-formations. We show we can always decompose a rigid formation into two connected sub-formations, one of which has *arbitrary* order, and present an algorithm for doing this. Section V provides a summary of our results and possible future problems for consideration.

II. Background

In this section we summarise the necessary background for the results presented later in the paper.

We can model a 2-dimensional sensor network with a simple graph. The vertices represent the sensors, and each edge in the graph represents a distance constraint between the vertices it is incident with. Basically a graph is rigid iff for almost all embeddings, any continuous movement of its vertices subject to its edge constraints preserves the distance between every pair of vertices. The following definitions from [6] and [9] will make this idea precise.

An m-dimensional ($m = 2, 3$) *representation* of a graph $G = (V, E)$ is an injective function $r : V \to \mathbb{R}^m$.

We define the distance between two representations r_1 and r_2 of the same graph by

$$d(r_1, r_2) = \max_{i \in V} |r_1(i) - r_2(i)|$$

A *distance set* D for G is a set of distances $d_{ij} > 0$, defined for all edges $ij \in E$. A distance set D is *realizable* if there exists a representation r of the graph for which $|r(i) - r(j)| = d_{ij}$ for all $ij \in E$. Such a representation is then called a *realization* of D. It follows that any representation r is a realization of the distance set defined by $d_{ij} = |r(i) - r(j)|$ for all $ij \in E$.

A realization r of a distance set D is *rigid* if there exists $\epsilon > 0$ such that for all realizations r' of D satisfying $d(r, r') < \epsilon$, there holds $|r'(i) - r'(j)| = |r(i) - r(j)|$ for all $i, j \in V$.

Definition 2.1 (Rigid graph): A graph is rigid in $\mathbb{R}^m$ ($m = 2, 3$) if almost all (i.e. an open dense set of) its realizations in $\mathbb{R}^m$ are rigid.

In this paper we will mostly be dealing with graphs which are rigid in $\mathbb{R}^2$, so we assume $m = 2$ unless otherwise specified. i.e. 'Rigid' will mean rigid in $\mathbb{R}^2$. However most results in section IV are easily extendable to graphs which are rigid in $\mathbb{R}^3$.

Remark 2.2: Strictly speaking, Definition 2.1 is the definition for a *generically rigid* graph. In this paper, and in much of the existing literature such as [6], generic rigidity is simply referred to as 'rigidity'.

Note that in Definition 2.1, we do not require a rigid graph to have all its realizations rigid. The reason for this is explained later in Remark 2.8.

Remark 2.3: By Definition 2.1, the trivial graph (i.e. a graph consisting of a single vertex) is rigid.

Example 2.4: The triangle C_3 is rigid, but C_4 (the cycle graph of order 4) is not.

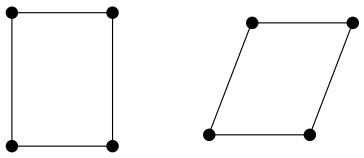

Fig. 1. A 'continuous skew' of C_4.

With any realization of C_4 as a parallelogram, the diagonal lengths are not preserved when C_4 is continuously skewed as shown in Fig. 1.

Definition 2.5 (Minimally rigid graph): A rigid graph is minimally rigid if no edge can be removed from it without losing rigidity.

Theorem 2.6 (Laman's theorem [7]): A non-trivial graph $G = (V, E)$ is minimally rigid iff $|E| = 2|V| - 3$ and for all non-empty $E' \subseteq E$, $|E'| \leq 2|V(E')| - 3$, where $V(E')$ is the set of vertices incident to the edges of E'.

According to [10], all minimally rigid graphs can be constructed by performing a series of *vertex addition* and *edge-splitting* operations on K_2 (the complete graph on 2 vertices with only 1 edge). Both operations add a new vertex v to an existing graph. These two operations are defined as follows:

Performing the operation of *vertex addition* on $G = (V, E)$ yields the graph $G' = (V \cup \{v\}, E \cup \{vi, vj\})$, $i, j \in V$.

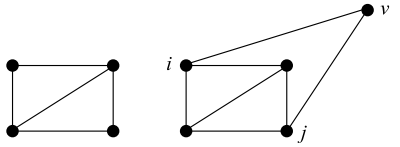

Fig. 2. The vertex addition operation.

Performing the operation of *edge-splitting* on $G = (V, E)$ yields the graph $G' = (V \cup \{v\}, E \cup \{vi, vj, vk\} \setminus \{e\})$, $i, j, k \in V$ with at least two of i, j, k adjacent, and e one of ij, ik, jk.

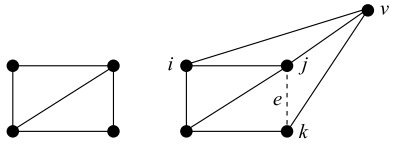

Fig. 3. The edge-splitting operation.

Theorem 2.7 (from [10]): A graph obtained by performing a vertex addition or edge-splitting operation on a minimally rigid graph is minimally rigid.

As a consequence of Theorem 2.7, a graph constructed by performing a series of vertex addition and edge-splitting operations on K_2 is minimally rigid. Such a construction is called a *Henneberg construction*. [10]

Remark 2.8: In Definition 2.1, we define a rigid graph as one where almost all realizations are rigid because some 'rigid' graphs may have non-rigid realizations. Consider the graph shown in Fig. 4.

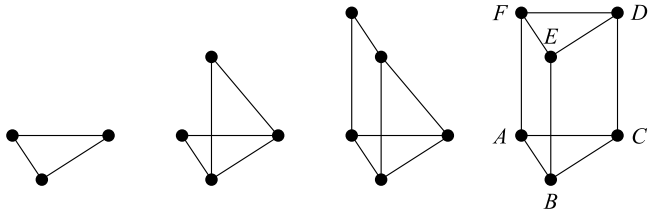

Fig. 4. A non-rigid realization of a rigid graph.

By Theorem 2.7, the final labelled graph is (minimally) rigid because it can be constructed from C_3 by performing two vertex addition operations followed by an edge-splitting operation. The particular realization shown has $\triangle ABC \cong \triangle FED$ and $ACDF$ a parallelogram. However parallelogram $ACDF$ can be continuously skewed so that the distance AD is not preserved.

Definition 2.9 (Split): A split of $G = (V, E)$ is a set $\{G_1, G_2\}$ where $G_i = (V_i, E_i)$ $(i = 1, 2)$ with $\{V_1, V_2\}$ a partition of V, and E_i the edge set induced by V_i. G_1 and G_2 are called the components of the split.

Remark 2.10: A split is simply an edge-cut. In rigidity theory, the term 'split' is conventionally used rather than edge-cut.

III. Splitting into Rigid Components

In this section we investigate decomposing rigid formations into rigid sub-formations. As mentioned, we may need to decompose a rigid graph into two or more rigid components to avoid obstacles, to remove a burden on one or more members of the formation, to merge with another formation, or to assign separate tasks to the different sub-formations. Unfortunately not all rigid graphs can be split into rigid components – for these graphs, we can split them into any arbitrary components first, and then find a 'minimal cover' using the 'reduction sequence' algorithm (see [4]) to recover rigidity within each component by adding new edges. Then [5] gives a method for merging these rigid components to form a single rigid formation again.

Definition 3.1 (Rigid split): A rigid split of $G = (V, E)$ is a split of G where both components of the split are rigid.

We now introduce a lemma which will be used to show that it is not always possible to split a rigid graph into rigid components.

Lemma 3.2: A rigid split on a minimally rigid graph of order at least three removes either two or three edges. Furthermore, only two edges are removed iff one of the components in the split is the trivial graph.

Proof: Let $G = (V, E)$ be a minimally rigid graph of order at least three, so

$$|E| = 2|V| - 3 \tag{1}$$

by Laman's theorem. Suppose G can be split into rigid components $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ by removing r edges. G_1 and G_2 must be minimally rigid since G is.

Case I: G_1 is the trivial graph. Then $|E_1| = 0$ and

$$|E_2| = 2(|V| - 1) - 3 \tag{2}$$

by Laman's theorem. Now $|E| = |E_1| + |E_2| + r$ together with (1) and (2) implies $r = 2$.

Case II: Neither G_1 nor G_2 is the trivial graph. Then $|E_1| = 2|V_1| - 3$ and $|E_2| = 2|V_2| - 3$ by Laman's theorem, and $|E| = |E_1| + |E_2| + r = (2|V_1| - 3) + (2|V_2| - 3) + r = 2|V| - 6 + r$ together with (1) implies $r = 3$. ■

Theorem 3.3: Not all rigid graphs have a rigid split.

Proof: By Theorem 2.7, the complete bipartite graph $K_{3,3}$ is minimally rigid as it can be constructed by performing a vertex addition followed by two edge-splitting operations on the minimally rigid graph C_3.

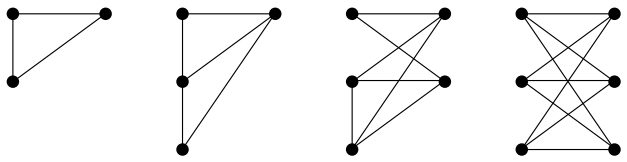

Fig. 5. A Henneberg construction of $K_{3,3}$.

By Lemma 3.2, $K_{3,3}$ cannot have a split into rigid components of orders 1 and 5 because this would require the removal of three edges. It is easily seen that $K_{3,3}$ cannot be split into rigid components of orders 2 and 4, or 3 and 3 since this would require the removal of at least four edges. Hence $K_{3,3}$ does not have a rigid split. ■

Definition 3.4 (Rigid-splittable): We call $G = (V, E)$ rigid-splittable if for all $i = 1, 2, \ldots, |V| - 1$, G has a rigid split into components of orders i and $|V| - i$.

Remark 3.5: Note that the definition of rigid-splittable requires a graph to have a rigid split of all orders, not just the existence of a rigid split. Indeed there exist graphs which have a rigid split but are not rigid-splittable. Consider the rigid graph G shown on the left in Fig. 6, generated by performing two successive vertex addition operations on $K_{3,3}$.

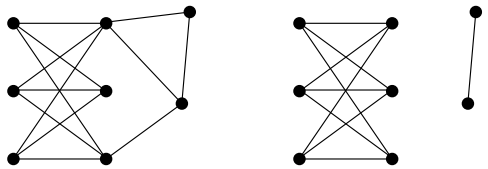

Fig. 6. A rigid split of a non-(rigid-splittable) graph G

G has a split into rigid components of orders 6 and 2, but it is easily seen that there is no split into rigid components of orders 5 and 3.

The following example shows there are some graphs which can be split into rigid components of arbitrary orders.

Example 3.6 (Complete graph): Any complete graph is clearly rigid by definition. The complete graph K_n of order

n is rigid-splittable since any split of K_n decomposes it into rigid components K_i and K_{n-i}.

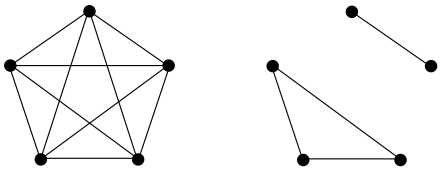

Fig. 7. A rigid split of K_5 into the components K_3 and K_2.

IV. Splitting into Connected Components

By Theorem 3.3, not all rigid graphs can be split into rigid components. But we can always split a rigid graph into two connected components, one of which has *arbitrary* order. In the context of sensor networks, the connectedness property ensures that any sensor can communicate with any other sensor (perhaps via intermediate sensors) in the same component. Connectedness also ensures that no agent or group of agents can deviate too far away from the rest of the agents in the same connected component, which is important if we wish to keep the agents in a contained area.

After splitting a rigid graph into connected components, we can apply the 'reduction sequence' algorithm (see [4]) on each connected component to recover minimal rigidity within each component by adding new edges. Suppose $C = (V, E)$ is one such connected component of the split. Then the algorithm finds a minimally rigid graph $C^* = (V, E \cup E_{\text{new}})$. From Laman's theorem we have $|E \cup E_{\text{new}}| = 2|V|-3$, and from connectedness of C we have $|E| \geq |V| - 1$. This implies $|E_{\text{new}}| \leq |V|-2$, i.e. at most $|V|-2$ new edges need to be added to a connected component to recover rigidity.

Definition 4.1 (Connected split): A connected split of $G = (V, E)$ is a split of G where both components of the split are connected.

Remark 4.2: We call a connected split of a graph of order n into connected components of orders i and $n-i$ a $(i, n-i)$ connected split.

Definition 4.3 (Connected-splittable): We call $G = (V, E)$ connected-splittable if for all $i = 1, 2, \ldots, |V|-1$, G has a $(i, |V| - i)$ connected split.

Remark 4.4: Note that the definition of connected-splittable requires a graph to have a connected split of all orders, not just the existence of a connected split.

Lemma 4.5 (Transfer lemma): Let G be a 2-connected graph of order $n + 1$. Suppose G can be split into two connected components $G_1 = (V_1, E_1)$ of order $i + 1$ and $G_2 = (V_2, E_2)$ of order $n-i$ (with $1 \leq i \leq n-1$). Then there exists a vertex $v_{\text{trans}} \in V_1$ such that the graph $G_1 - \{v_{\text{trans}}\}$ of order i induced by $V_1 \setminus \{v_{\text{trans}}\}$ in G is connected, and the graph $G_2 + \{v_{\text{trans}}\}$ of order $n-i+1$ induced by $V_2 \cup \{v_{\text{trans}}\}$ in G is connected.

Proof: Let G be a 2-connected graph of order $n+1$. By hypothesis, G can be split into two connected components $G_1 = (V_1, E_1)$ of order $i + 1$ and $G_2 = (V_2, E_2)$ of order $n - i$ (with $1 \leq i \leq n - 1$).

Induce a rooted spanning tree $T(G_1)$ from G_1, with any $r \in V_1$ as the root.

Define

$$R = \{v \in V_1 : \exists u \in V_2 \text{ s.t. } vu \in E\}$$

to be the set of all vertices in G_1 adjacent to some vertex to G_2. $R \neq \emptyset$ since G is 2-connected and thus connected.

Let $d_{T(G_1)}(u, v)$ denote the length (number of edges) of the unique path between $u \in V_1$ and $v \in V_1$ in $T(G_1)$. Let

$$v_{\text{trans}} \in \arg\max_{v \in R} d_{T(G_1)}(r, v)$$

be a vertex in R furthest from r in $T(G_1)$.

Clearly the graph $G_2 + \{v_{\text{trans}}\}$ of order $n - i + 1$ induced by $V_2 \cup \{v_{\text{trans}}\}$ in G is connected since the graph induced by V_2 is connected and v_{trans} is adjacent to a vertex in V_2 by construction of R.

It remains to show the graph $G_1 - \{v_{\text{trans}}\}$ of order i induced by $V_1 \setminus \{v_{\text{trans}}\}$ in G is connected.

Let $P_{T(G_1)}(u, v)$ denote the unique path between $u \in V_1$ and $v \in V_1$ in $T(G_1)$.

Define

$$D = \{v \in V_1 : v_{\text{trans}} \in P_{T(G_1)}(r, v)\}$$

to be the set consisting of v_{trans} and its descendants in $T(G_1)$. (See Fig. 8.)

Clearly $T(G_1) - D$ is connected. (To see this, pick two arbitrary vertices u, v in $T(G_1) - D$. Then $P_{T(G_1)}(r, u)$ and $P_{T(G_1)}(r, v)$ do not contain v_{trans}, so both paths are subgraphs of $T(G_1) - D$. Hence there is a path from u to v via r in $T(G_1) - D$.)

Consider the maximally connected component in $G_1 - \{v_{\text{trans}}\}$ containing $T(G_1) - D$ as a subgraph. Call this component M. Suppose, by way of contradiction, that $G_1 - \{v_{\text{trans}}\}$ is not connected. Then $G_1 - \{v_{\text{trans}}\} = M \cup N$ where M and $N \neq \emptyset$ are not adjacent to each other. (i.e. no vertex of M is a neighbour of any vertex of N.) M contains $T(G_1) - D$, so N can only contain vertices in $D \setminus \{v_{\text{trans}}\}$. But

$$\begin{aligned} v \in D \setminus \{v_{\text{trans}}\} &\Rightarrow v_{\text{trans}} \in P_{T(G_1)}(r, v) \text{ and } v \neq v_{\text{trans}} \\ &\Rightarrow d_{T(G_1)}(r, v) > d_{T(G_1)}(r, v_{\text{trans}}) \\ &\Rightarrow v \notin R \end{aligned}$$

So N cannot be adjacent to G_2 in G.

Since N is not adjacent to $M = G_1 - \{v_{\text{trans}}\} - N$ either, N can only be adjacent to v_{trans} in G, implying v_{trans} can be removed to disconnect G. This contradicts G being 2-connected. Hence $G_1 - \{v_{\text{trans}}\}$ is connected.

Therefore G has a $(i, n - i + 1)$ connected split into the connected components $G_1 - \{v_{\text{trans}}\}$ and $G_2 + \{v_{\text{trans}}\}$. ■

The preceding lemma makes no reference to rigidity, and realizations associated with G are not necessarily confined to $\mathbb{R}^2$. We now apply the lemma to graphs where the realizations, at least initially, are in $\mathbb{R}^2$.

Lemma 4.6: Let $G = (V, E)$ be a rigid graph. Then $G - \{v\}$ is connected for any $v \in V$.

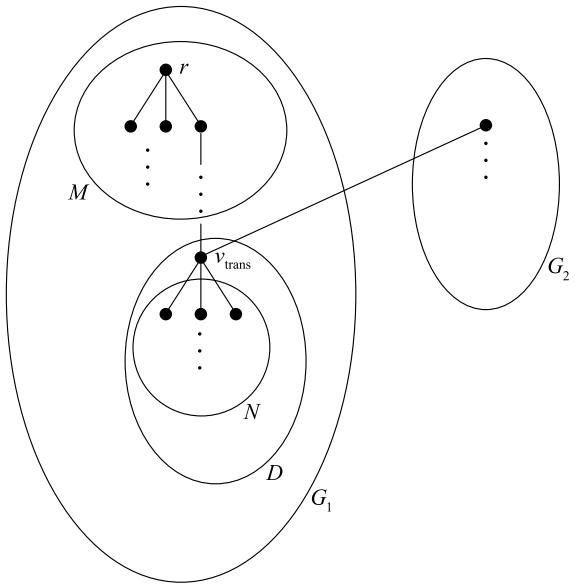

Fig. 8. v_{trans} can be 'transferred' from G_1 to G_2 with connectedness preserved in both components.

Proof: Let $G = (V, E)$ be a rigid graph. Suppose $G - \{v\}$ is not connected for some $v \in V$. Then $G - \{v\} = M \cup N$, $M, N \neq \emptyset$, where M and N are not adjacent to each other (see Fig. 9).

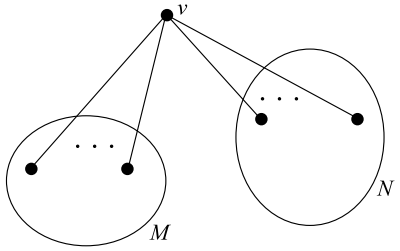

Fig. 9. Situation if $G - \{v\}$ is not connected.

This implies N can rotate about v independently of M (i.e. while M remains stationary in G), a contradiction to G being rigid. Hence $G - \{v\}$ is connected for any $v \in V$. ■

Remark 4.7: Clearly Lemma 4.6 shows any rigid graph is also 2-connected (i.e. it is not possible to remove one vertex to disconnect the graph).

Now we provide a proof to our main theorem.

Theorem 4.8: All rigid graphs are connected-splittable.

Proof: Let G be a rigid graph of order n (and thus G is 2-connected by Remark 4.7). By Lemma 4.6, G has a $(n-1, 1)$ connected split. Now recursing Lemma 4.5, G also has a $(n-2, 2)$ connected split, and thus a $(n-i, i)$ connected split for all $i = 1, 2, \ldots, n-1$. ■

Remark 4.9: Note that the proofs of Lemma 4.5, Lemma 4.6, and consequently Theorem 4.8 are easily extendable to graphs which are rigid in $\mathbb{R}^3$.

A. An Algorithm for finding Connected splits

In this subsection, we provide an algorithm for finding a $(i, n-i)$ connected split of any rigid graph of order n.

Let G be a rigid graph with a connected split into precisely two connected components $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. The ideas used in the proof of Lemma 4.5 give rise to a breadth-first algorithm for finding v_{trans} and transferring it from G_1 to G_2. The following algorithm takes G, G_1 and G_2 as arguments and outputs $G_1^{\text{new}} = G_1 - \{v_{\text{trans}}\}$ (the graph induced by $V_1 \setminus \{v_{\text{trans}}\}$ in G) and $G_2^{\text{new}} = G_2 + \{v_{\text{trans}}\}$ (the graph induced by $V_2 \cup \{v_{\text{trans}}\}$ in G).

```
(G_1^new, G_2^new) = transfer(G, G_1, G_2)
    choose any starting vertex r ∈ G_1
    mark r as visited
    initialise a queue Q := {r}
    while Q non-empty
        choose vertex v from front of Q
            if v is adjacent to G_2 in G
                then v_trans := v
            endif
            for each unmarked neighbour w of v
                mark w as visited
                add w to end of Q
            endfor
        remove v from queue
    endwhile
    G_1^new := G_1 − {v_trans}
    G_2^new := G_2 + {v_trans}
```

The starting vertex r is essentially the root in the proof of Lemma 4.5. The while loop finds v_{trans}. It is easy to see that the above algorithm has running time $O(|V_1| + |E_1|)$.

We can then use the following algorithm (analogous to the proofs of Lemma 4.6 and Theorem 4.8) to obtain a $(i, n-i)$ connected split of a rigid graph $G = (V, E)$ of order n for any $i = 1, 2, \ldots, n-1$. The algorithm takes G and i as arguments and outputs G_1 (the connected component of order $n - i$ in the split) and G_2 (the connected component of order i in the split).

```
(G_1, G_2) = connectedsplit(G, i)
    choose any starting vertex s ∈ G
    G_1 := G − {s}
    G_2 := {s}
    for i − 1 times
        (G_1^new, G_2^new) := transfer(G, G_1, G_2)
        (G_1, G_2) := (G_1^new, G_2^new)
    endfor
```

The algorithm above starts with a $(n-1, 1)$ connected split and transfers one vertex at a time until a $(n-i, i)$ connected split is obtained. Clearly we can always choose $i \leq \lfloor n/2 \rfloor$. So it is clear the above algorithm has worst-case running time $O(n(n + |E|)/2)$.

Remark 4.10: Note that the algorithms above also work for graphs which are rigid in $\mathbb{R}^3$.

B. Connected splits of Minimally Rigid Graphs

This subsection gives a tight upper bound for the number of edges that can be removed from a minimally rigid graph

in a connected split. It serves as a nice check to see whether a particular split can be a connected split.

Lemma 4.11: Any connected split of a minimally rigid graph of order n removes at most $n-1$ edges.

Proof: Let $G=(V,E)$ be a minimally rigid graph. We have $|E|=2|V|-3$ by Laman's theorem. Suppose G can be split into two connected components $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$ by removing $r \geq |V|$ edges. Connectedness of G_1 implies $|E_1| \geq |V_1|-1$ and connectedness of G_2 implies $|E_2| \geq |V_2|-1$. Thus $|E| = |E_1|+|E_2|+r \geq (|V_1|-1)+(|V_2|-1)+|V| = 2|V|-2 > 2|V|-3$, a contradiction. Hence $r<|V|$. ■

Considering the contrapositive of Lemma 4.11, any split of a minimally rigid graph of order n that removes more than $n-1$ edges cannot be a connected split.

Corollary 4.12: All minimally rigid graphs of order n are connected-splittable, and each split removes at most $n-1$ edges.

Proof: Combine Theorem 4.8 with Lemma 4.11. ■

Remark 4.13: Note that the proof of Lemma 4.11 and consequently Corollary 4.12 is not valid for graphs which are rigid in $\mathbb{R}^3$ as Laman's theorem only holds for graphs which are rigid in $\mathbb{R}^2$.

V. Conclusions and Future Works

A. Conclusions

In this paper we examined the problem of splitting rigid graphs without adding new edges. We showed that some, but not all rigid graphs can be split into two rigid components. However any rigid graph can always be split into precisely two connected components, one of which has *arbitrary* order. We gave an algorithm for doing this, and showed that any such split of a minimally rigid graph of order n removes at most $n-1$ edges. Furthermore, it is not possible for both of these components to be rigid if more than three edges are removed in the split.

B. Future Works

We have shown that all rigid graphs are 2-connected and thus connected-splittable, and provided an algorithm for finding a split into two connected components of any desired orders. These results also hold for graphs which are rigid in $\mathbb{R}^3$. One may ask whether there are analogous stronger results for graphs which are rigid in $\mathbb{R}^3$. A graph which is rigid in $\mathbb{R}^3$ is 3-connected, so it would be reasonable to conjecture that graphs which are rigid in $\mathbb{R}^3$ are '2-connected-splittable', that is, they can be split into two 2-connected components, one of which has *arbitrary* order. Future work could focus on proving or disproving this conjecture, and providing an algorithm for finding 2-connected splits of graphs which are rigid in $\mathbb{R}^3$, if such splits do indeed exist.

An algorithm for testing whether a graph is rigid is provided in [11]. Further work could focus on an algorithm for testing whether a rigid graph has a split into rigid components of specific orders, and if so, provide such a rigid split. For the graph in Remark 3.5, the algorithm should be able to identify a split into rigid components of orders 6 and 2, and conclude there is no split into rigid components of orders 5 and 3.

One can also investigate what properties of a graph will ensure that it is rigid-splittable. We know all complete graphs are rigid-splittable, but not all rigid graphs are. Also, what properties will ensure that a graph is connected-splittable? We have provided a partial answer to this question – all 2-connected graphs are connected-splittable, but not all connected graphs are (the star graphs of order $n \geq 4$ do not have a $(2, n-2)$ connected split). Is there a weaker condition than 2-connectedness that will ensure a graph is connected-splittable?

Acknowledgments

W. Ong was supported by the summer research program at the Australian National University. C. Yu is supported by the Australian Research Council through an Australian Postdoctoral Fellowship under DP-0877562. B.D.O. Anderson is supported by DP-0877562 and National ICT Australia – NICTA. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] B.D.O. Anderson, C. Yu, B. Fidan, and J. Hendrickx. Rigid graph control architectures for autonomous formations. *IEEE Control Systems Magazine*, 28(6):48–63, 2008.

[2] R. Olfati-Saber and R. Murray. Graph rigidity and distributed formation stabilization of multi-vehicle systems. *Proceedings of the 41st IEEE Conference on Decision and Control*, 3:2965–2971, 2002.

[3] C. Yu, J.M. Hendrickx, B. Fidan, B.D.O. Anderson, and V.D. Blondel. Three and higher dimensional autonomous formations: Rigidity, persistence and structural persistence. *Automatica*, 43(3):387–402, 2007.

[4] T. Eren, B.D.O. Anderson, A.S. Morse, W. Whiteley, and P.N. Belhumeur. Operations on rigid formations of autonomous agents. *Communications in Information and Systems*, 3(4):223–258, 2004.

[5] L. Henneberg. Die graphische statik der starren systeme, 1911.

[6] J.M. Hendrickx, B.D.O. Anderson, J.C. Delvenne, and V.D. Blondel. Directed graphs for the analysis of rigidity and persistence in autonomous agent systems. *International Journal of Robust and Nonlinear Control*, 17:960–981, 2007.

[7] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4:331–340, 1970.

[8] T.H. Summers, C. Yu, and B.D.O. Anderson. Addressing agent loss in vehicle formations and sensor networks. *International Journal of Robust and Nonlinear Control*. DOI:10.1002/rnc.1400, available online.

[9] W. Whiteley. Rigidity and scene analysis. In J. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 893–916. CRC Press, 1997.

[10] T. Tay and W. Whiteley. Generating isostatic frameworks. *Structural Topology*, 11:21–69, 1985.

[11] C. Moukarzel. An efficient algorithm for testing the generic rigidity of graphs in the plane. *Journal of Physics A: Mathematical and General*, 29:8079–8098, 1996.