2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Decentralized Generic Rigidity Evaluation in Interconnected Systems

Ryan K. Williams, Andrea Gasparri, Attilio Priolo, and Gaurav S. Sukhatme

Abstract— **In this paper, we consider the problem of evaluating the generic rigidity of an interconnected system in the plane, without a priori knowledge of the network's topological properties. We propose the decentralization of the pebble game algorithm of Jacobs et. al., an $O(n^2)$ method that determines the generic rigidity of a planar network. Our decentralization is based on asynchronous inter-agent message-passing and a distributed memory architecture, coupled with consensus-based auctions for electing leaders in the system. We provide analysis of the asynchronous messaging structure and its interaction with leader election, and Monte Carlo simulations demonstrating complexity and correctness. Finally, a novel rigidity evaluation and control scenario in the accompanying media illustrates the applicability of our proposed algorithm.**

I. Introduction

Interconnected systems of intelligent agents have become the recent focus of intense investigation, particularly in the context of autonomous collaboration (such as in multi-robot or sensor systems), affording fundamental advantages in adaptability, scalability, and efficiency compared to classical single-agent solutions. Various impactful applications of such systems have been demonstrated, including monitoring [1], target tracking [2], and dynamic network optimization [3].

Of particular interest in this work is the evaluation of the *rigidity* property of an interconnected system of intelligent agents (e.g. robots, sensors, etc.). A relatively under-explored topic in the area of multi-agent systems, rigidity has important implications particularly for mission objectives requiring collaboration. For example, rigidity is vital for guaranteeing stability in controlling formations of mobile vehicles, when only relative inter-agent information is available [4]–[6]. Further, when a global frame of reference is inaccessible, rigidity becomes a necessary (and under certain conditions sufficient) condition for localization tasks with distance or bearing-only measurements [7]–[9].

The general study of rigidity has a rich history in various contexts of science, mathematics, and engineering [10]–[13]. In [12] combinatorial operations are defined which preserve rigidity, with works such as [4], [6] extending the ideas to multi-robot formations. In [14] an algorithm is proposed for generating rigid graphs in the plane based on the Henneberg construction [12], however from a centralized (or all-to-all) perspective. Similarly, [15] defines decentralized rigid constructions that are edge length optimal, however provide no means of determining an unknown graph's rigidity properties. The work [16] defines a rigidity eigenvalue for infinitesimal rigidity evaluation and control, however such efforts remain centralized and require continuous communication and computational resources. Finally, in our upcoming work [17], we explore distributed combinatorial rigidity control, in contrast by exploiting *local* rigidity information in a rigidity maintenance task.

As opposed to previous work, we propose a *decentralized* method of evaluating *combinatorial* (or *generic*) graph rigidity in the plane, *without* a priori topological information, to our knowledge the first such effort, particularly in a multi-agent context. Our motivation in determining combinatorial rigidity rests on the notion that rigidity is a *generic* property of a network topology [18], eliminating the need to examine all possible realizations (i.e. *infinitesimal rigidity*). Generic rigidity has strong implications in localization [7], [9] and formation control [4], while guaranteeing in almost all cases, infinitesimal rigidity (having similar applications in multi-robot coordination). Thus, our proposition is to decentralize in an *asynchronous* manner the *pebble game* proposed by Jacobs and Hendrickson in [13], an algorithm that determines in $O(n^2)$ time the combinatorial rigidity of a network, and a spanning edge set defining the minimally rigid subcomponent of the graph[1]. Specifically, we propose a leader election procedure based on maximum consensus auctions that manages the sequential nature of the pebble game in a decentralized setting, together with a distributed memory architecture. Further, an asynchronous messaging scheme preserves local-only agent interaction, as well as robustness to delays, failures, etc.

In evaluating a network's rigidity property with leader auctioning, we thus arrive at a *decentralized* means of identifying edge optimal topologies[2] that are rigid, possessing the guarantees in fundamental multi-agent problems implied by rigidity. Such topologies could be leveraged in multi-robot networks to guarantee rigidity through motion control (e.g. by applying [20]), with auction bidding chosen relative to mission objectives (e.g. sensing cost). Beyond topology seeking, our proposed algorithm enables both rigidity maintenance (by defining the edges to retain), and rigidity evaluation, e.g. for initiating rigidity recovery in the case that mission objectives dictate a violation of rigidity. To illustrate our contributions,

R. K. Williams and G. S. Sukhatme are with the Departments of Electrical Engineering and Computer Science at the University of Southern California, Los Angeles, CA 90089 USA (email: rkwillia@usc.edu; gaurav@usc.edu).

A. Gasparri and A. Priolo are with the Department of Engineering, Roma Tre University, Via della Vasca Navale, 79. Roma, 00146, Italy (email: gasparri@dia.uniroma3.it; priolo@dia.uniroma3.it).

This work was partially supported by the ONR MURI program (award N00014-08-1-0693), the NSF CPS program (CNS-1035866), the NSF grant CNS-1213128, a fellowship to R. K. Williams from the USC Viterbi School of Engineering, and partially by the Italian grant FIRB "Futuro in Ricerca", project NECTAR, code RBFR08QWUV, funded by the Italian Ministry of Research and Education (MIUR).

[1] Such an edge set is not obtained by infinitesimal approaches, e.g. [16]

[2] For example, as in relative sensing networks [19].

we provide Monte Carlo analysis of our algorithms to check both correctness and complexity, and describe a novel simulated scenario in the media attachment that encompasses both rigidity evaluation and control.

II. Preliminaries

A. Agent and Network Model

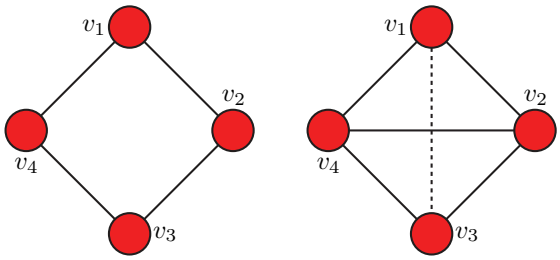
Consider a system composed of n agents indexed by $\mathcal{I} = \{1, \dots, n\}$ operating in $\mathbb{R}^2$, each possessing computation and communication capabilities, denoting by (i, j) a bi-directional communication link between agents i and j. To describe the interconnected system formally, we define *undirected graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, having vertices (nodes) $\mathcal{V} = \{v_1, \dots, v_n\}$ associated with each agent $i \in \mathcal{I}$, and edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ with members (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 with an edge $(i, j) \in \mathcal{E}$ are referred to as *neighbors*, where the set of neighbors for the ith agent is given by $\mathcal{N}_i = \{v_j \in \mathcal{V} \,|\, (i, j) \in \mathcal{E}\}$. Finally, for our purposes, we assume the network topology $\mathcal{G}$ is *connected* for all time to guarantee all agents can participate in rigidity evaluation (Section III). Finally, we make the following systematic assumptions:

Assumption 1. All communication in our system is *asynchronous* (no global clock), every agent is uniquely identifiable, global knowledge of n exists, and $O(n)$ per-agent storage is available.

B. Rigidity Theory

The primary concern of this work is the *rigidity* property of the underlying graph $\mathcal{G}$, specifically as rigid graphs imply guarantees in both localizability and formation stability of multi-robot systems [6]. 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 $p\colon \mathcal{V} \to \mathbb{R}^2$, assigning to each node in $\mathcal{G}$, a location in $\mathbb{R}^2$. The *infinitesimal motion* of $\mathbb{F}_p$ can be described by assigning to the vertices of $\mathcal{G}$, a velocity $\dot{p}_i \in \mathbb{R}^2$ such that edge lengths (inter-agent distance) are preserved over time (i.e. no edge is compressed or stretched over time). The framework is said to undergo a *trivial motion* upon translations and rotations of $\mathbb{R}^2$ itself. If for $\mathbb{F}_p$ all infinitesimal motions are trivial, then $\mathbb{F}_p$ is said to be *infinitesimally rigid*[3] (Fig. 1b). Otherwise, the framework is called *infinitesimally flexible* [12] (Fig. 1a, i.e. v_1 and v_3 can move inward while v_2 and v_4 move outward).

By definition the infinitesimal 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 infinitesimally rigid or flexible (i.e. they form a dense open set in $\mathbb{R}^2$) [18]. Thus, we can treat rigidity from the perspective of $\mathcal{G}$, abstracting away the necessity to check every possible realization, and motivating directly our contributions in this work. The first such *combinatorial* characterization of graph rigidity was described by Laman in [10], and is summarized as follows (also called *generic rigidity*)[4]:



(a) Non-rigid. (b) Generically rigid.

Fig. 1. Graphs demonstrating rigidity. Notice that all solid edges in (a) and (b) are independent. Adding edge $(1, 3)$ in (b) generates a non-minimally rigid graph with redundant edge $(1, 3)$.

Theorem 1 (Graph rigidity, [10])**.** *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ having realizations in $\mathbb{R}^2$ with $n = |\mathcal{V}| \geq 2$ 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 $(i, j) \in \hat{\mathcal{E}}$.*

From Theorem 1 it follows that *every* rigid graph in the plane must then have $|\mathcal{E}| \geq 2n - 3$ edges, with equality holding 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:

Definition 1 (Edge independence, [13])**.** 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 $|\bar{\mathcal{E}}| > 2|\bar{\mathcal{V}}| - 3$. A set of independent edges will be denoted by $\mathcal{E}^*$.

The above conditions imply that all rigid graphs have $|\mathcal{E}^*| = 2n - 3$ independent edges, where edges that do not meet the conditions of Definition 1 are called *redundant* (see Fig. 1 for a depiction of graph rigidity). Thus, in determining the rigidity of $\mathcal{G}$, we can verify the conditions of Definition 1 to discover a suitable set of independent edges $\mathcal{E}^*$.

C. The Pebble Game

To lessen the exponential complexity of the conditions of Theorem 1 (i.e. evaluating *every* subgraph of $\mathcal{G}$), we consider the *pebble game* proposed by Jacobs and Hendrickson in [13], which admits a natural form in the context of interconnected systems. A brief overview of the *centralized* pebble game will be given here, beginning with a useful consequence of Theorem 1 and Definition 1:

Lemma 1 (Edge quadrupling, [13])**.** *Given an independent edge set $\mathcal{E}^*$, an edge $(i, j) \notin \mathcal{E}^*$ is independent of $\mathcal{E}^*$ if and only if the graph formed by* quadrupling *(i, j), i.e. adding 4 copies of (i, j) to $\mathcal{E}^*$, has no subgraph $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$ in which $|\bar{\mathcal{E}}| > 2|\bar{\mathcal{V}}|$.*

[3] Intuitively, if the graph were a bar and joint framework, it would be mechanically rigid.

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

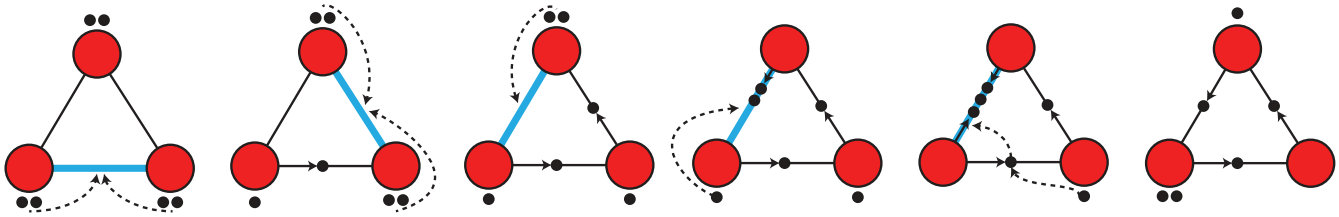


Fig. 2. An example of the pebble game for a rigid graph with $n = 3$ with progression from left to right. Pebbles are given by black dots, quadrupled edges by thick links (blue), pebble shifts by dashed arrows, and local pebble assignment by black arrows. We have here $|\mathcal{E}^*| = 3$.

The generic rigidity test of [13] thus operates by applying Lemma 1 to grow members of $\mathcal{E}^*$ *incrementally*, terminating when $|\mathcal{E}^*| = 2n - 3$. In growing $\mathcal{E}^*$ it can be shown that every independence check of $(i, j) \notin \mathcal{E}^*$ requires only $O(n)$ operations (as there are at most $2n - 3$ members of $\mathcal{E}^*$) [13], and thus the exponential testing of subgraphs has been reduced to simply quadrupling every new edge and checking that the induced subgraphs have a cardinality respecting Lemma 1. A natural simplification to this process is found in the following *pebble game*:

Definition 2 (The pebble game, [13])**.** Considering a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, give to each agent associated with $v_i \in \mathcal{V}$ two pebbles, each of which can be assigned to an edge in $\mathcal{G}$. Our goal in the pebble game is to assign the pebbles in $\mathcal{G}$ such that all edges are covered, i.e. a *pebble covering*. In finding a pebble covering, we allow the assignment of pebbles by agent i *only* to edges incident to v_i in $\mathcal{G}$. Further we allow pebbles to be rearranged only by removing pebbles from edges which have an adjacent vertex with a free pebble, such that the free pebble is shifted to the assigned pebble, freeing the assigned pebble for assignment elsewhere. Thus, if we consider pebble assignments as directed edges exiting from an assigning agent i, when a pebble is needed in the network to cover an edge (i, j), a *pebble search* over a directed network occurs. If a free (unassigned) pebble is found, the local assignment and rearrangement rules then dictate the pebble's return and assignment to (i, j).

It can be shown that if there exists a pebble covering for an independent edge set $\mathcal{E}^*$ with a quadrupled edge $(i, j) \notin \mathcal{E}^*$, there is no subgraph violating the conditions of Lemma 1[5]. Thus, the rigidity evaluation pebble game of [13] operates as follows: every edge $e \in \mathcal{E}$ is quadrupled, and an attempt to expand the current pebble covering for $\mathcal{E}^*$ to each copy of e is made, with success resulting in $\mathcal{E}^* \leftarrow \mathcal{E}^* \cup e$ and termination coming when $|\mathcal{E}^*| = 2n - 3$. The *centralized* pebble game of Jacobs is depicted in Algorithm 1, with an illustration of the quadrupling and pebble search procedure depicted in Fig. 2. Our contribution in this work will be the decentralization of the described pebble game for generic rigidity evaluation, specifically an asynchronous treatment of pebble searching, shifting, and assignment in determining a network's independent edge set.

[5]Intuitively, an agent's pebbles represent its possible commitments to the network's subgraphs, while maintaining the conditions of Lemma 1.

Algorithm 1 The centralized pebble game [13].

procedure PEBBLEGAME($\mathcal{G} = (\mathcal{V}, \mathcal{E})$)
  Assign each v_i two pebbles, $\forall\, i \in \mathcal{I}$
  $\mathcal{E}^* \leftarrow \emptyset$
  for all $(i, j) \in \mathcal{E}$ **do**
    Quadruple (i, j) over $\mathcal{E}^*$
    Search for 4 free pebbles, starting from v_i and v_j
    if *found* **then**
      Rearrange pebbles to cover quadrupled (i, j)
      ▷ Expand independent set, check rigidity:
      $\mathcal{E}^* \leftarrow \mathcal{E}^* \cup (i, j)$
      if $|\mathcal{E}^*| = 2|\mathcal{V}| - 3$ **then** ▷ Check rigidity
        return $\mathcal{E}^*$
      end if
    end if
  end for
end procedure

III. An Asynchronous Decentralized Pebble Game

The primary considerations in decentralizing the pebble game of [13] lie in the sequential nature of edge quadrupling, the storage of $\mathcal{E}^*$ and associated pebble assignments over a distributed network, and the search and rearrangement of pebbles under the asynchronicity of realistic interacting systems. Our proposition in this work is to handle such issues through the election of *leaders* in the network (Section III-A), each expanding $\mathcal{E}^*$ by examining their incident edges, querying the network for free pebbles, and then exchanging leadership when their local neighborhood has been evaluated. The independent edges and pebble assignments are thus localized to each agent, distributing network storage and relying on inter-agent messaging to support pebble searching under asynchronicity (Section III-C). In this way, our contributions in decentralization will be not only the leader election procedure, but the logic by which each leader must operate to accommodate asynchronicity, and the structure of the pebble game over distributed interacting agents.

Specifically, each agent $i \in \mathcal{I}$ possesses pebble assignment set $\mathcal{P}_i$ having at most two edges $\{(i, j) \in \mathcal{E} \,|\, j \in \mathcal{N}_i\}$, that is incident edges (i, j) to which a pebble is associated. Let $p_i = 2 - |\mathcal{P}_i| \in \{0, 1, 2\}$ denote agent i's free pebble count, and use shorthand notation $\mathcal{P}_i(k, l)$ for the lth element of the kth edge in $\mathcal{P}_i$, with $k, l \in \{1, 2\}$. Local independent edge storage is denoted by $\mathcal{E}_i^* = \{(i, j) \in \mathcal{E} \,|\, j \in \mathcal{N}_i\}$, containing

edges for which quadrupling and pebble covering succeeds, where by construction $\mathcal{E}^* = \bigcup_i \mathcal{E}_i^*$.

A. Leader Election

An execution of the decentralized pebble game begins with an agent that initiates the algorithm in response to network conditions or mission objectives (e.g. verifying link deletion in $\mathcal{G}$ with respect to rigidity). The initiating agent begins by triggering an *auction* for electing an agent in the network to become the *lead agent*, where to each agent $i \in \mathcal{I}$ we associate a *bid* $r_i = [i, b_i]$ with $b_i \in \mathbb{R}_{\geq 0}$ indicating the agent's *fitness* in becoming the new leader, with $b_i = 0$ if i has already been 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 leadership auction then operates according the the following auction process:

$$r_i(t^+) = \underset{r_j \in \mathcal{R}_i}{\operatorname{argmax}}(b_j) \tag{1}$$

where the notation t^+ indicates a transition in r_i after all neighboring bids have been collected through asynchronous messaging. As $\mathcal{G}$ is assumed connected for all time, the auction (1) converges in at most $n-1$ steps *uniformly* to the largest leadership bid $r_i = \operatorname{argmax}_{r_j(0)}(b_j(0)),\ \forall\, i, j \in \mathcal{I}$ [21]. After convergence of (1) the winning agent then takes on the leadership role, with the previous leader relinquishing its status. The proposed auction mechanism allows us to decentralize the sequential nature of the pebble game by assigning to each leader the responsibility of locally expanding the independent edge set, noting that previous leaders are never reelected in a given execution due to $b_i = 0$ for such agents[6]. Additionally, as there is only one leader at a time in the network, there is no contention in pebble searching.

It is important to note that the order of leader election (and thus bid selection) is relevant:

Proposition 1 (Initial leader edges)**.** *All incident edges* $\{(i,j) \in \mathcal{E} \,|\, j \in \mathcal{N}_i\}$ *belonging to an initial leader* i *are members of the independent set* $(i,j) \in \mathcal{E}^*$.

Proof. For each edge, a new node $j \neq i$ must be considered as no two edges of i can have the same endpoint and $\mathcal{E}^*$ is empty due to i being the initial leader. Therefore, every subgraph containing the edges and nodes incident to i must have $|\mathcal{E}_s| \leq 2|\mathcal{V}_s| - 3$ edges, where $\mathcal{V}_s$ are the nodes of the considered subgraph and $|\mathcal{E}_s| = |\mathcal{V}_s| - 1$ due to the subgraph's implicit tree structure. Thus, as there exists no subgraph violating Definition 1, the result follows. □

The above reasoning can be extended to subsequent leaders as well, specifically when incident nodes are not endpoints of $(i,j) \in \mathcal{E}^*$, independence is clearly implied. It then follows that leader ordering is closely related to the composition of $\mathcal{E}^*$ and thus the informed selection of bids b_i can regulate directly the resultant rigid subgraph. For example in a sensing regime, bids could reflect the aggregate information value of an agent's incident edges (e.g. [19]), thereby selecting a rigid spanning subgraph with maximal value. The proposed auction technique therefore affords us control over $\mathcal{E}^*$ that goes beyond simply discovering the network's rigidity property[7].

The elected leader maintains the following variables for managing the local expansion of $\mathcal{E}_i^*$: $\mathcal{E}_i = \{(i,j) \in \mathcal{E} \,|\, j \in \mathcal{N}_i\}$ contains the incident edges for quadrupling consideration and $e_i \in \mathcal{E}_i$ is the current edge being quadrupled. After winning auction (1), the new leader i performs initialization

$$\mathcal{E}_i \leftarrow \{(i,j) \in \mathcal{E} \,|\, j \in \mathcal{N}_i \wedge \neg \textit{beenLeader}(j)\} \tag{2}$$

where incident edges (i,j) are considered only when the neighbor $j \in \mathcal{N}_i$ has not yet been a leader, as edges incident to a previous leader j have already been checked. Also, note that each leader receives the current size of the independent edge set $|\mathcal{E}^*(t)|$ in initialization[8], allowing termination when $2n-3$ independent edges are identified.

B. The Leader Algorithm

After election and initialization, the task of the leader i is to continue the expansion of $\mathcal{E}^*$ by evaluating the independence of each $(i,j) \in \mathcal{E}_i$. The leader executes the procedure LEADERRUN depicted in Algorithm 2 to accomplish this task, where we assume such execution occurs relative to the leader's local clock, facilitating edge evaluation. First, recall that in checking independence a pebble covering for each quadrupled edge $e_i \in \mathcal{E}_i$ must be determined. As the pebble assignment information is distributed across the network, the lead agent must therefore *request* pebbles through messaging in an attempt to assign pebbles to e_i (messaging is treated in Section III-C). In making such requests, the lead agent *waits* idly until a response is received, allowing the agent to sequentially handle pebble *responses*; a technique often referred to as *blocking*, applied here to ensure correctness under network asynchronicity.

When there exists no unfulfilled pebble request, the lead agent resumes the quadrupling procedure on the current incident edge $e_i \in \mathcal{E}_i$, in hopes of identifying a pebble covering and establishing the independence of e_i. For each step of the quadrupling (lines 6-14), the leader searches for a pebble to cover e_i, first by looking locally for free pebbles, assigning e_i to $\mathcal{P}_i$ if found. If no local pebbles are available, $p_i = 0 \wedge |\mathcal{P}_i| = 2$, the agent then sends a PEBBLEREQUESTMSG to agent $\mathcal{P}_i(1,2)$, along the first edge to which a pebble is assigned, requesting a free pebble. Note that in sending requests only along $(i,j) \in \mathcal{P}_i$, we properly evaluate independence with respect to $\mathcal{E}^*$, as each $(i,j) \in \mathcal{E}^*$ must have an assigned pebble from previous coverings.

For any edge e_i with a pebble covering, obtained through a combination of local assignment and pebble responses (more on this later), the following actions are taken (lines 16-27). First, we return 3 pebbles to the endpoints of e_i leaving a single pebble on e_i to establish independence, and then add e_i to $\mathcal{E}_i^*$. If in adding e_i, $2n-3$ independent edges have been identified, the leader sends a simple message to the network indicating

[6]Further, the condition $b_i = 0,\ \forall\, i \in \mathcal{I}$ allows termination of the algorithm.

[7]In this paper, we concentrate on the construction of a decentralized pebble game. Our future work focuses on bid selection and related optimality properties.

[8]We can simply embed $|\mathcal{E}^*(t)|$ in the leadership auction to facilitate this transfer.

Algorithm 2 Leader execution logic.

```
1:  procedure LeaderRun(i)
2:      if No Response then                          ▷ Wait for pebble response
3:          return
4:      end if
5:      while e_i ≜ (i, j) do                        ▷ Continue pebble covering
6:          while Quadrupled Copies ≤ 4 do
7:              if p_i > 0 then                      ▷ Assign local pebble
8:                  P_i ← P_i ∪ e_i
9:                  p_i ← p_i − 1
10:             else                                 ▷ Request pebble along first edge
11:                 PebbleRequestMsg(i, P_i(1, 2))
12:                 return
13:             end if
14:         end while
15:         ▷ Quadrupling success, return 3 pebbles:
16:         P_i ← ∅
17:         p_i ← 2
18:         Return 1 pebble to v_j
19:         ▷ Add independent edge and check rigidity:
20:         E*_i ← E*_i ∪ e_i
21:         if |E*| = 2n − 3 then
22:             Send network rigidity notification
23:             return
24:         end if
25:         ▷ Go to next incident edge:
26:         E_i ← E_i − e_i
27:         e_i ← (i, j) ∈ E_i
28:     end while
29:     ▷ All local edges checked:
30:     Initiate leadership transfer auction
31: end procedure
```

that the graph is rigid, and the algorithm terminates. Otherwise, the leader moves to the next member $(i,j) \triangleq e_i$ of $\mathcal{E}_i$. When all members of $\mathcal{E}_i$ have been evaluated, the current leader initiates the leadership auction (1), embedding $|\mathcal{E}^*|$ in the auction for the next elected leader. The process of local edge quadrupling, pebble requests and covering, and independence expansion then continues from leader to leader until either the network is found to be rigid, or every agent i has been a leader, indicating non-rigidity. The described formulation guarantees the *entire* network is evaluated for rigidity, with *no* edge reconsideration (i.e. no edge is checked by multiple agents):

Proposition 2 (Edge consideration)**.** *Disregarding termination when $|\mathcal{E}^*| = 2n-3$, every $(i,j) \in \mathcal{E}$ is eligible to be considered for independence. Further, $\mathcal{E}_i^* \cap \mathcal{E}_j^* = \emptyset$ holds for all $i \neq j \in \mathcal{I}$.*

Proof. These results are a simple consequence of the guaranteed convergence of auction (1), $b_i = 0$ for all *beenLeader*$(i) = 1$ (guaranteeing no reelection), and the initialization of $\mathcal{E}_i$ with edges *not* shared with previous leaders as in (2). □

Algorithm 3 Pebble request handler for agent i.

```
1:  procedure HandlePebbleRequest(from, i)
2:      if Request Not Unique then                   ▷ Already requested
3:          PebbleNotFoundMsg(i, from)
4:          return
5:      end if
6:      if p_i > 0 then                              ▷ Local pebble available
7:          P_i ← P_i ∪ (i, from)                    ▷ Shift free pebble
8:          p_i ← p_i − 1
9:          PebbleFoundMsg(i, from)
10:     else                                         ▷ Request along first assigned edge
11:         PebbleRequestMsg(i, P_i(1, 2))
12:         requester(i) ← from
13:     end if
14: end procedure
```

C. Inter-Agent Messaging

As each leader attempts to expand $\mathcal{E}_i^*$ through quadrupling, free pebbles are needed to establish a pebble covering, implying edge independence. We accommodate the pebble search by defining asynchronous message PebbleRequestMsg, accompanied by response messages PebbleFoundMsg and PebbleNotFoundMsg, indicating the existence of free pebbles.

The reception of a PebbleRequestMsg initiates the handler HandlePebbleRequest depicted in Algorithm 3. We assume that each pebble request is marked with a unique identifier originating from the lead agent, defining the pebble search to which the request is a member (lines 2-5). This allows for any given agent to participate simultaneously in multiple searches[9], and guarantees that each search does not revisit nodes, guaranteeing termination (i.e. the leader *will* receive a response). For unique requests, the agent first attempts to assign local pebbles to the edge connecting the pebble requester, sending a PebbleFoundMsg in response, allowing the requester to free an assigned pebble for either local assignment (a leader) or to itself respond to a pebble request (non-leader). If $p_i = 0$ (lines 10-13) the agent forwards the request to agent $\mathcal{P}_i(1,2)$, the first agent to which a pebble is assigned, recording the requester such that pebble responses can be properly returned.

Remark 1 (Implicit routing)**.** Notice that as opposed to explicit message routing, the local nature of agent interaction is preserved by tracking requesters in the above manner. In particular, routing is exactly dictated by the directed graph associated with local pebble assignments in $\mathcal{G}$.

When the response PebbleFoundMsg to a pebble request is received it triggers the handler HandlePebbleFound depicted in Algorithm 4. Congruent to the shifting action of HandlePebbleRequest, the agent first frees the local pebble assigned to the edge connecting the responder (line 2), and applies the pebble relative to leader status. If receiving

[9] In this work, we do not explicitly investigate such a possibility. This is the focus of future work.

Algorithm 4 Pebble found handler for agent i.

```
1: procedure HandlePebbleFound(from, i)
2:     P_i ← P_i − (i, from)                    ▷ Free local pebble
3:     if isLeader(i) then
4:         P_i ← P_i ∪ e_i                      ▷ Expand covering
5:     else                                     ▷ Give free pebble to requester
6:         P_i ← P_i ∪ (i, requester(i))
7:         PebbleFoundMsg(i, requester(i))
8:     end if
9: end procedure
```

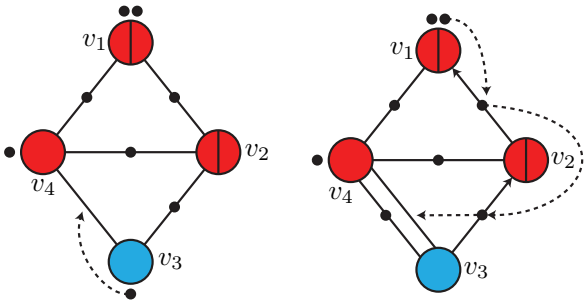
Algorithm 5 Pebble not found handler for agent i.

```
1: procedure HandlePebbleNotFound(from, i)
2:     if Paths Searched < 2 then               ▷ Search other path
3:         PebbleRequestMsg(i, P_i(2, 2))
4:     else                                     ▷ Search failed, no free pebbles
5:         if isLeader(i) then                  ▷ e_i is redundant
6:             Return pebbles assigned to e_i
7:             ▷ Go to next incident edge:
8:             E_i ← E_i − e_i
9:             e_i ← (i, j) ∈ E_i
10:        else
11:            PebbleNotFoundMsg(i, requester(i))
12:        end if
13:    end if
14: end procedure
```

agent i is the current leader (lines 3-4), the freed pebble is assigned locally to e_i, continuing the edge quadrupling process and relieving the request blocking condition. Otherwise, the non-lead agent performs a pebble shift to again free a pebble for the requesting agent, indicating the shift by returning a PebbleFoundMsg to its original pebble requester (lines 5-8).

Finally, the response PebbleNotFoundMsg to a pebble request initiates the handler HandlePebbleNotFound depicted in Algorithm 5. For both leaders and non-leaders the non-existence of a free pebble initiates a further search in the network, along $\mathcal{P}_i(2,2)$ (lines 2-4), i.e. the second edge to which a pebble is assigned. However, if both available search paths are exhausted (lines 5-13), the leadership status of the receiver dictates the action taken. In the simple case of a non-leader (line 11) the response is simply returned to the original requester in order to initiate further search. For the leader, the lack of free pebbles in the network indicates precisely that the conditions of Lemma 1 do not hold, implying the currently considered edge e_i is *redundant*. The edge e_i is removed from consideration by returning all pebbles assigned during the covering attempt to the endpoints of e_i, and the process is moved to the next incident edge (lines 6-9).

Remark 2 (Pebble search)**.** The pebble request and response messaging constructs an event-driven, depth-first traversal of the network, specifically abiding by the rules set forth by Jacobs in [13] (see Fig. 3 for an example of such messaging).



(a) First edge of quadrupling. (b) Second edge of quadrupling.

Fig. 3. Illustration of the first two pebble covering attempts for a quadrupling on edge $(3,4)$. Agents v_1 and v_2 have previously been a leader, while agent v_3 (blue) is the current leader. Pebbles are depicted by solid black dots, requests by inter-agent arrows, and responses and shifts by dashed arrows.

Intuitively, we have illustrated in the context of rigidity evaluation, an asynchronous and distributed graph search, iteratively rooted at each lead agent.

Remark 3 (Complexity)**.** As the pebble game exhibits $O(n^2)$ complexity [13], our pebble messaging scales like $O(n^2)$. In applying leader auction (1) we incur $O(n^2)$ as we expend $O(n)$ auction messaging for $O(n)$ leaders. The per-agent storage complexity scales like $O(n)$, the maximal cardinality of $\mathcal{N}_i$.

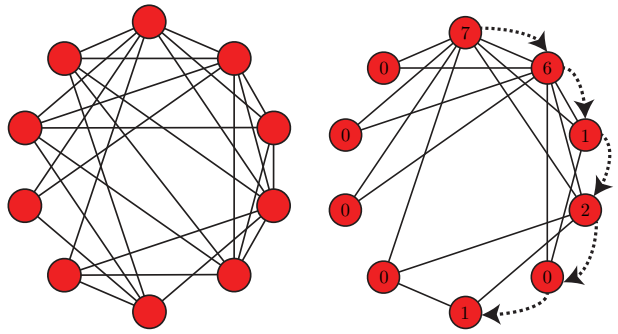
Remark 4 (Failures)**.** If pebble responses are not received in a timely manner, the connecting path can simply be removed from consideration, allowing the algorithm to progress as normal. If a leader experiences failure, a similar reasoning can be applied to initiate the election of a new leader.

IV. Simulation Results

In this section, we present simulation results of our proposed decentralized pebble game algorithm. First, for illustration purposes, we applied our algorithm over the network in Fig. 4a, resulting in the identified independent edge set depicted in Fig. 4b, with the cardinality of the local independent edge sets indicated, together with the leadership transitions. As suggested by Proposition 1, $\mathcal{E}^*$ grows largest with the initial leader, while the edges added by each subsequent leader diminishes as $|\mathcal{E}^*|$ grows towards $2n-3$.

Next, to examine correctness and complexity, we generated 1000 rigid graphs with $n \in [4, 50]$ uniformly. For each graph we compiled communication and storage complexity statistics, as depicted in Fig. 5. The top graph illustrates how the message complexity scales in n, detailing the maximum, minimum, and average per-agent message burden. The bottom graph shows the storage complexity scaling, measured by $|\mathcal{E}_i^*|$. In our testing, the maximum message burden scaled like $O(n^{1.231})$ (black fit, top), while the maximum independent edge storage scaled like $O(n^{0.5646})$ (black fit, bottom).

Remark 5 (Applications)**.** The above Monte Carlo results indicate our decentralization has remained faithful to the original pebble game in terms of complexity, and further illustrates scaling that is amenable to realistic implementation, our ultimate goal in pursuing decentralized rigidity evaluation. As rigidity



(a) Initial rigid graph. (b) Computed Laman subgraph.

Fig. 4. Decentralized pebble game simulation with $n = 10$ and $|\mathcal{E}| = 27$: (a) original non-minimally rigid graph; (b) computed Laman subgraph with leader progression indicated by dashed arrows, and $|\mathcal{E}_i^*|$ denoted.

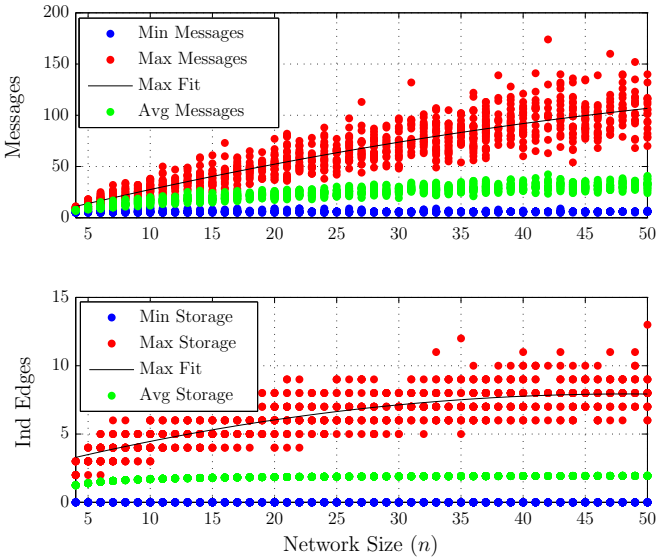


Fig. 5. Monte Carlo simulations over 1000 rigid graphs demonstrating the communication complexity (top) and storage complexity (bottom) of our proposed decentralized pebble game algorithm.

has far reaching implications in multi-robot objectives such as network localization and formation stability, our results here present an opportunity to exploit rigidity theory in real-world robotic systems.

Remark 6 (Media attachment)**.** We direct the reader to the media attachment, illustrating our methods in a novel rigidity evaluation scenario, where we exploit our results here and a topology control framework [20] to yield a dynamic rigidity control and recovery objective.

V. Conclusions and Future Work

In this paper we considered the problem of evaluating the generic rigidity of an interconnected system in the plane, without a priori topological knowledge. The pebble game algorithm for rigidity evaluation was decentralized by means of asynchronous inter-agent message-passing and a distributed memory architecture, coupled with an auction-based leader election procedure. Such auctions allowed the sequential nature of the original pebble game to extend to a decentralized context. Finally Monte Carlo simulations and a novel rigidity evaluation and control scenario in the media attachment demonstrated the promising complexity and applicability of our proposed algorithm.

Directions for future work include parallelization through simultaneous leadership and investigating bid selections for computing optimal minimally rigid spanning graphs with guarantees.

References

[1] S. L. Smith, M. Schwager, and D. Rus, "Persistent Robotic Tasks: Monitoring and Sweeping in Changing Environments," *IEEE Transactions on Robotics*, 2012.

[2] S. Martínez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, 2006.

[3] R. K. Williams, A. Gasparri, and B. Krishnamachari, "Route Swarm: Wireless Network Optimization through Mobility," in *IEEE International Conference on Computer Communications (submitted)*, 2014.

[4] R. Olfati-Saber and R. M. Murray, "Graph rigidity and distributed formation stabilization of multi-vehicle systems," in *IEEE Conference on Decision and Control*, 2002.

[5] T. Eren, P. N. Belhumeur, and A. Morse, "Closing ranks in vehicle formations based on rigidity," in *IEEE Conference on Decision and Control*, 2002.

[6] B. Anderson, C. Yu, B. Fidan, and J. Hendrickx, "Rigid graph control architectures for autonomous formations," *Control Systems, IEEE*, 2008.

[7] 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.

[8] J. Aspnes, T. Eren, D. K. Goldenberg, A. Morse, W. Whiteley, Y. R. Yang, B. D. O. Anderson, and P. N. Belhumeur, "A Theory of Network Localization," *IEEE Transactions on Mobile Computing*, 2006.

[9] I. Shames, A. N. Bishop, and B. D. O. Anderson, "Analysis of Noisy Bearing-Only Network Localization," *IEEE Transactions on Automatic Control*, 2013.

[10] G. Laman, "On graphs and rigidity of plane skeletal structures," *Journal of Engineering Mathematics*, 1970.

[11] L. Lovasz and Y. Yemini, "On Generic Rigidity in the Plane," *Siam Journal on Algebraic and Discrete Methods*, 1982.

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

[13] D. J. Jacobs and B. Hendrickson, "An algorithm for two-dimensional rigidity percolation: the pebble game," *J. Comput. Phys.*, 1997.

[14] D. Zelazo and F. Allgower, "Growing optimally rigid formations," in *American Control Conference*, 2012.

[15] 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.*, 2010.

[16] D. Zelazo, A. Franchi, F. Allgower, H. H. Bulthoff, and P. R. Giordano, "Rigidity Maintenance Control for Multi-Robot Systems," in *Robotics: Science and Systems*, 2012.

[17] R. K. Williams, A. Gasparri, A. Priolo, and G. S. Sukhatme, "Distributed Combinatorial Rigidity Control in Multi-Agent Networks," in *IEEE Conference on Decision and Control (to appear)*, 2013.

[18] H. Gluck, "Almost all simply connected closed surfaces are rigid," in *Geometric Topology*, 1975.

[19] D. Zelazo and M. Mesbahi, "Graph-Theoretic Analysis and Synthesis of Relative Sensing Networks," *IEEE Transactions on Automatic Control*, 2011.

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

[21] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, "A distributed auction algorithm for the assignment problem," in *IEEE Conference on Decision and Control*, 2008.