Decentralized and Parallel Constructions for Optimally Rigid Graphs in $\mathbb{R}^2$

Andrea Gasparri, *Member, IEEE*, Ryan K. Williams, *Student Member, IEEE*, Attilio Priolo, and Gaurav S. Sukhatme, *Fellow, IEEE*

Abstract—In this paper, we address the *decentralized* and *parallel* construction of rigid graphs in the plane that optimize an edge-weighted objective function under cardinality constraints. Two auction-based algorithms to solve this problem in a decentralized fashion are first proposed. Centered around the notion of leader election, the first approach finds an optimal solution through a greedy bidding, while the second approach provides a sub-optimal solution which reduces complexity according to a sliding mode parameter. Then, by exploiting certain local structural properties of graph rigidity, a parallelization to build a portion of the optimal solution in constant time is derived. A theoretical characterization of algorithm performance is provided together with complexity analysis. Finally, simulation results are presented to corroborate the theoretical findings.

Index Terms—Graph Rigidity, Matroid Optimization, Decentralized Systems, Topology Control.

1 Introduction

In recent years, collaborative and decentralized systems have become of interest across various research communities, including robotics, control, sensor networks, etc. This interest is primarily due to their advantages in adaptability, efficiency and scalability with respect to single-node systems [1]. Research interests in the context of such systems range from mutual localization [2], [3] and map-building [4], [5] to the development of decentralized coordination algorithms [6] and the design of decentralized interaction control frameworks [7]. Further interests include issues such as provably fast data collection for sensor networks [8], and joint scheduling and power control in wireless networks [9]. Remarkable capabilities have been demonstrated in several application contexts, including environmental exploration and sensing [10], [11], search and rescue operations [12], and coverage tasks [13].

A popular and important area of study in collaborative systems is the *topology control problem*, where under a wide array of performance metrics, a network's topology is chosen or controlled to be optimal. Examples of topology control are exceptionally vast. For instance, [14] optimizes the tradeoff between node degree and hop stretch under directional communication. In [15] graph geometry is exploited for tunable and decentralized angle-only topology control. Finally, [16] provides schemes that construct degree limited and connected piconets in multi-hop networks.

In this work, we consider the topology optimization problem under a *graph rigidity* constraint. Broadly speaking, rigidity represents an important requirement when the task demands collaboration among teammates. For example, its relevance is clear in the context of controlling formations of mobile nodes when only relative sensing information is available [17], [18]. Specifically, the asymptotic stability of a formation is guaranteed when the graph that defines the formation is rigid by construction. Rigidity becomes a necessary (and in certain settings sufficient) condition for localization tasks with distance or bearing-only measurements [19], [20]. The ability of a network to self-localize is of clear importance across various application contexts. For example in [20] it is shown that if the rigidity conditions for localizability for traditional noiseless systems are satisfied, and measurement errors are small enough, then the network will be approximately localizable, providing a connection between robustness and rigidity. Finally, the flavor of rigidity studied here is also a necessary component of *global rigidity* [21], [22], which can further strengthen the guarantees of formation stability and localizability, as the uniqueness of a given topological embedding is more easily characterized. It is clear then that network rigidity acts as a fundamental precursor to both important spatial behaviors and information-driven objectives, making it a strong motivation of this work.

The literature regarding the study of rigidity is rich and involves various disciplines of mathematics and engineering [23]–[25]. Combinatorial operations to preserve rigidity are defined in [24], while an extension of these ideas to the context of formation control can be

- *A. Gasparri and A. Priolo are with the Department of Engineering, University of "Roma Tre", Via della Vasca Navale, 79. Roma, 00146, Italy (gasparri@dia.uniroma3.it; priolo@dia.uniroma3.it).*
- *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 (rkwillia@usc.edu; gaurav@usc.edu).*

This work was partially supported by the Italian grant FIRB "Futuro in Ricerca", project NECTAR, code RBFR08QWUV, funded by the Italian Ministry of Research and Education (MIUR), and partially by the ONR MURI program (award N00014-08-1-0693), the NSF CPS program (CNS-1035866), and the NSF grant CNS-1213128.

found in [18]. In [26], the authors propose a rigidity maintenance controller for mobile teams, the application of which is mainly limited by the centralized nature of the algorithm and the requirement of continuous communication and computational resources. To render the rigidity control framework tractable from an implementation standpoint, in our previous work [27], we propose a decentralized rigidity controller that preserves the combinatorial rigidity of a dynamic network topology in the plane, through mobility control. Notably, this controller requires (local) communication and computation only during proposed transitions in the network topology, not continuous operation as in [26].

In this work, we assume a utility metric is associated with inter-node connections, allowing us to formulate an optimization problem within the rigidity framework. This is an interesting problem in the context of matroid optimization for which centralized greedy solutions are well-known [28], [29]. In a decentralized context, this problem becomes particularly interesting as the network is built iteratively up until the point of being minimally rigid. Indeed, further optimal edges beyond the point of minimal rigidity can be straightforwardly added as the rigidity condition no longer requires verification. Thus, without loss of generality we will limit ourself to the construction of minimally rigid graphs.

In particular, this work aims at finding in a *decentralized* way the *optimal* rigid subgraph, i.e., a minimally rigid subgraph that maximizes a certain utility function. Such an optimization is attractive as the resulting graph not only holds the guarantees associated with rigidity, but also considers network utility, e.g. for localizable and edge optimal sensor embeddings or mobile networks. Despite its great theoretical and application oriented appeal, this problem has been largely overlooked in the recent literature. In [30], decentralized rigid constructions that are edge length optimal are defined, even though the problem of checking this rigidity property for an arbitrary given graph is not addressed. In [31] an algorithm is proposed for generating optimally rigid graphs based on the Henneberg construction [24], however the proposed algorithm is centralized, generates only locally optimal solutions, and requires certain neighborhood assumptions to ensure convergence.

Notably, a rigid network is also inherently connected. Connectivity is vital in ensuring information flow in a network and in regulating the convergence of agreement processes [32]. For example, the Fiedler eigenvalue is a common metric for multi-node control, i.e., by following a negative gradient to increase connectivity [33]. However, while such methods speed up the convergence of agreement processes, increasing connectivity may have a negative impact on robustness, e.g., noise propagation and time delays [32]. In comparison, relevant notions of robustness such as the $\mathcal{H}_2$ metric [34] can be applied in the context of rigidity. Furthermore, rigidity guarantees connectivity implicitly and most importantly imposes further structure on the network often required for useful guarantees in collaborative systems [27]. In this way, rigid topology control can be seen as a natural evolution of recent work in connectivity control.

In general, *constrained combinatorial* optimization problems are difficult to solve even approximately. However, in light of results in matroid optimization theory [28], [29], it turns out that for our problem a *greedy* edge evaluation will ultimately yield a *provably (sub)optimal* solution. Such a solution is then quite convenient for decentralization, as a simple and locally computable edge ordering dictates algorithm execution. The first major novelties of this work are represented by two decentralized auction-based algorithms to build a (sub)optimal rigid subgraph. Resting on the combinatorial *Laman* conditions which characterize rigidity, and a leader election algorithm to control rigidity evaluation, the first approach finds an optimal solution at the cost of higher communication complexity. The second approach then provides a sub-optimal solution while reducing the computational burden according to a sliding parameter ξ, controlling the balance of optimality and complexity. The second major novelty of this work is represented by a partial parallelization of optimal rigid construction. In particular, taking inspiration from the *Henneberg operations* for rigid graph construction [24] and the greedy optimality of the solution, we derive *local* asynchronous rules which can provably detect a bounded subset of the optimal solution.

A theoretical characterization of the optimality of the first algorithm is provided. Furthermore, a closed form of the maximum gap between the optimal solution and the sub-optimal solution provided by the second algorithm expressed in terms of the tuning parameter ξ is also provided. Then, the correctness of the parallelization is analyzed along with a lower bound on the cardinality of the identified optimal subset. Finally, we close the work by providing a detailed simulation analysis of our algorithms. Notably, Relative Sensing Networks (RSNs) [34] are considered as a realistic application scenario where a mobile network with both spatial and information-based objectives is investigated.

The remainder of the paper is organized as follows. In Section 2, preliminary materials for rigidity and matroid theory are given. The main results are introduced in Section 3, where three decentralized approaches to find an optimal rigid subgraph are presented. Simulations to corroborate the theoretical findings are reported in Section 4. Finally, conclusions are discussed in Section 5.

2 Preliminaries and Background

Consider n nodes, which may be mobile or static, with positions $x_i(t) \in \mathbb{R}^2$ whose network topology is modeled by a weighted undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ is the set of nodes and $\mathcal{E} = \{e_{ij}\} \in \mathcal{V} \times \mathcal{V}$ is the set of edges. Note that, neither self-loops nor multiple-edges between nodes are allowed. Denote with $\mathcal{N}_i = \{j \in \mathcal{V} : e_{ij} \in \mathcal{E}\}$ the set of neighbors of the ith node and with $\mathcal{E}_i = \{e_{ij} \in \mathcal{E} : j \in \mathcal{N}_i\}$ the set of incident edges to node i.

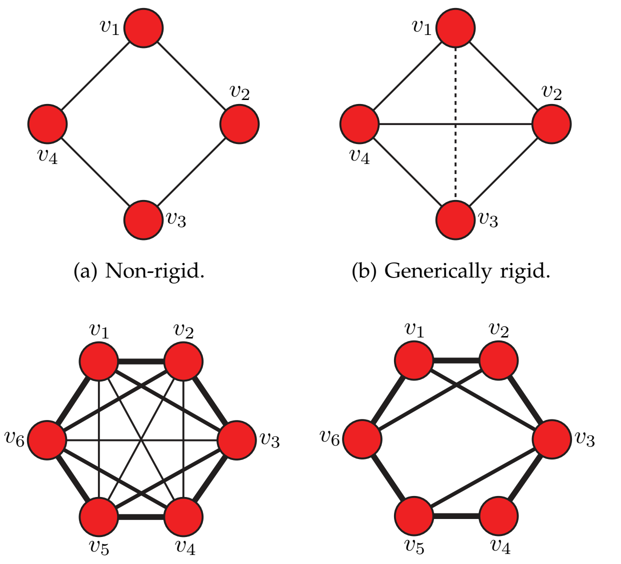
Assume a weight $w(e_{ij}) = \psi(i) + \psi(j) + \phi(i,j)$ to be associated to each edge e_{ij}, where $\psi(i) : \mathcal{V} \to \mathbb{R}^+$ is a utility function associated to each endpoint of the edge and $\phi(i,j) : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$ is a metric associated to e_{ij}. Note that due to the symmetry of $\mathcal{G}(\mathcal{V},\mathcal{E})$, $e_{ij} \in \mathcal{E} \Rightarrow e_{ji} \in \mathcal{E}$, $w(e_{ij}) = w(e_{ji})$. In the rest of the paper, the shorthand notations $\mathcal{G} \triangleq \mathcal{G}(\mathcal{V},\mathcal{E})$ and $w_{ij} \triangleq w(e_{ij})$ will be used for the sake of readability.

Finally, we will assume that node-to-node communication is broadcast-based, in the sense that a single message is sent to a node's neighborhood during a round of communication.

2.1 Graph Rigidity

One of the primary concerns of this work is the *rigidity* property of the underlying graph $\mathcal{G}$ describing the network topology. We provide here a high-level overview of rigidity theory, and we direct the reader to [23], [24], [35], [36] for a full technical review of the subject. To begin, we recall the intuition of how rigidity is recognized for a graph in the plane, following the exposition of [25]. Clearly, graphs with many edges are more likely to be rigid than those with only a few, specifically as each edge acts to constrain the degrees of freedom of motion of the nodes in the graph. In $\mathbb{R}^2$, there are $2n$ degrees of freedom in a network of n nodes, and when we remove the three degrees associated with rigid translation and rotation, we arrive at $2n-3$ degrees of freedom we must constrain to achieve rigidity. Each edge in the graph can be seen as constraining these degrees of freedom, and thus we expect $2n-3$ edges will be required to guarantee a rigid graph. In particular, if a subgraph containing k vertices happens to contain more than $2k-3$ edges, then these edges cannot all be required for constraining the degrees of motion, i.e., they cannot all be *independent*. Our goal in evaluating rigidity is thus to identify the $2n-3$ edges that independently constrain the motion of our nodes.

Notably, the concept of rigidity can alternatively be viewed from a physical standpoint, in the sense that if the graph were a bar-joint framework, it would be mechanically rigid against external and internal forces. That is, the edge lengths (inter-node distance) over $\mathcal{G}$ are preserved in time if the nodes were to move infinitesimally. This mechanical analogy reflects the notion of *infinitesimal rigidity* which accounts for the specific positions of the nodes in the workspace and their motion. On the other hand, combinatorial rigidity introduces the concept of *generic* rigidity by which the rigidity or flexibility of bar-joint frameworks is investigated in terms of the structure of the underlying graph. Interestingly, an important relationship between these two notions of rigidity has been established in $\mathbb{R}^2$. More precisely, it has been shown that the notion of rigidity in $\mathbb{R}^2$ is a *generic* property of $\mathcal{G}$, specifically as *almost all* realizations



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

(c) Full weighted graph. (d) Optimal Laman subgraph.

Fig. 1. Graph rigidity and the Laman subgraph. Notice that all solid edges in (a) and (b) are independent. Adding edge $(1,3)$ in (b) generates a non-minimally rigid graph. In (c) and (d) the fully connected graph and associated optimal Laman subgraph are shown with edge weight depicted by line weight (thicker is better).

of a graph are either infinitesimally rigid or flexible (i.e. they form a dense open set in $\mathbb{R}^2$) [37]. Thus, we can treat rigidity from the perspective of $\mathcal{G}$, abstracting away the necessity to check every possible realization of the graph in $\mathbb{R}^2$. The first such *combinatorial* characterization of graph rigidity was described by Laman in [23], and is summarized as follows (also called *generic rigidity*).

Theorem 1 (Graph rigidity, [23])**.** *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 $\mathcal{E}' \subseteq \bar{\mathcal{E}}$, we have $|\mathcal{E}'| \leq 2k-3$, where k is the number of nodes in $\mathcal{V}$ that are endpoints of $(i,j) \in \mathcal{E}'$.*

We refer to the above as the *Laman conditions*. Note that, at present the extension of Laman's conditions to higher dimensions is an unresolved problem in rigidity theory. We call a graph $\bar{\mathcal{G}} = (\mathcal{V},\bar{\mathcal{E}})$ satisfying Theorem 1 a *Laman subgraph* of $\mathcal{G}$, 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. We reiterate that the (generic) rigidity of a graph in $\mathbb{R}^2$ is purely a topological (combinatorial) property. Given our previous discussion about the power of rigidity in interconnected systems, we will in this work *enforce* the communication topology to possess the rigidity property by satisfying the above Laman conditions under a certain optimality criterion. Fig. 1a and Fig. 1b depict the different flavors of graph rigidity. In Fig. 1a we have a non-rigid graph as the basic $2n-3$ edge condition of Laman is unfulfilled. In adding edge (v_2, v_4) we then generate the minimally rigid graph of Fig. 1b composed of solid edges, as every subgraph of k vertices has at most $2k-3$ edges. Further addition of (v_1, v_3) (dashed) yields a non-minimally rigid graph, precisely as the graph possesses greater than $2n-3$ edges.

2.2 A Decentralized Pebble Game for Graph Rigidity

The Laman conditions given in Theorem 1 state that a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is rigid in $\mathbb{R}^2$ if and only if it possesses $|\bar{\mathcal{E}}| = 2n-3$ *independent* edges, where the residual edges in $\tilde{\mathcal{E}} = \mathcal{E}\backslash\bar{\mathcal{E}}$ are said to be *redundant*. Thus, in determining the rigidity of $\mathcal{G}$, we must verify the Laman conditions to discover a suitable set of independent edges $\bar{\mathcal{E}} \subseteq \mathcal{E}$. The following theorem provides a formal definition of what a set of independent edges is.

Theorem 2 (Edge Set Independence, [25])**.** *The edges of a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ are independent in $\mathbb{R}^2$ if and only if no subgraph $\mathcal{G}' = \{\mathcal{V}', \mathcal{E}'\}$ has more than $2|\mathcal{V}'|-3$ edges.*

A natural simplification to the process of determining sets of independent edges is found in the *pebble game* [25], a decentralization of which we provided in previous work [38] and the results of which we now briefly summarize. The fundamental idea behind the pebble game algorithm is to grow a maximal set of independent edges one at a time. For this reason we now introduce a useful re-characterization of the Laman conditions.

Theorem 3 (Laman restated, [25])**.** *For graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the following statements are equivalent:*

- *All $(i,j) \in \mathcal{E}$ are independent in $\mathbb{R}^2$.*
- *For each $(i,j) \in \mathcal{E}$, the graph formed by* quadrupling *(i,j), i.e., adding 3* virtual *copies of (i,j) to $\mathcal{E}$, has no subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ in which $|\mathcal{E}'| > 2|\mathcal{V}'|$.*

Interestingly, from Theorem 3 a useful condition for the incremental evaluation of edge independence can be derived as the following lemma demonstrates.

Lemma 1 (Incremental Edge Independence, [25])**.** *Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a (possibly empty) set of independent edges $\bar{\mathcal{E}} \subseteq \mathcal{E}$. A new edge e is independent of $\bar{\mathcal{E}}$ if and only if the graph formed by quadrupling e has no induced subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ with too many edges, i.e., $|\mathcal{E}'| > 2|\mathcal{V}'|$, where $\mathcal{V}'$ is the set of vertexes spanned by the set of edges $\mathcal{E}'$.*

Therefore, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to evaluate the independence of an edge e against a (possibly empty) set of independence edges $\bar{\mathcal{E}} \subseteq \mathcal{E}$ we can simply add 3 virtual copies of such an edge e, i.e., *quadrupling* the edge, and evaluate that the resulting graph $\bar{\mathcal{G}} = (\mathcal{V}, \bar{\mathcal{E}} \cup \{e\})$ has no induced subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ with too many edges, i.e., $|\mathcal{E}'| > 2|\mathcal{V}'|$. Notably, Lemma 1 reduces the complexity of independence testing to that of counting edges in subgraphs once the new edge is quadrupled. This intuition is embedded in the pebble game algorithm which we are now briefly going to describe.

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where we associate a node with each $v \in \mathcal{V}$, give to each node two pebbles which can be assigned to an edge in $\mathcal{E}$. 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 node i *only* to edges incident to 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. The adjacent vertex can then use its free pebble to cover the edge, freeing the previously assigned pebble for assignment elsewhere; a so-called *shift* operation. If we consider pebble assignments as directed edges exiting from an assigning node i, when a pebble is needed in the network to cover an edge (i,j), a *pebble search* over a directed network occurs. During the pebble search if a free pebble is found, the rules for pebble shifting then allow the network pebbles to be rearranged in order to assign a free pebble to edge (i,j). If there exists a pebble covering for an independent edge set $\bar{\mathcal{E}}$ with a quadrupled edge $(i,j) \notin \bar{\mathcal{E}}$, it follows that the set $\bar{\mathcal{E}} \cup \{(i,j)\}$ is independent. Rigidity evaluation then operates iteratively as follows: every edge $e \in \mathcal{E}$ is quadrupled, and an attempt to expand the current pebble covering for $\bar{\mathcal{E}}$ to each copy of e is made, with success resulting in $\bar{\mathcal{E}} \leftarrow \bar{\mathcal{E}} \cup e$ and termination coming when $|\bar{\mathcal{E}}| = 2n-3$. The reader is referred to [25] for a detailed description of the *centralized* algorithm.

Our decentralization of the above pebble game in its most simple form introduces leader election to decentralize the consideration of network edges for independence. An execution of the algorithm begins with the initiation of an *auction* for electing a node in the network to become the *leader*. Specifically, to each node $i \in \mathcal{V}$ we associate a *bid* for leadership r_i defined as the pair $r_i = \{i, b_i\}$ with $b_i \in \mathbb{R}_{\geq 0}$ indicating the node's *fitness* in becoming the new leader. As we aim to exploit leader election to achieve optimality, the bids b_i will represent a local utility metric for the ith node, as we will see. Denoting the local bid set by $\mathcal{R}_i = \{r_j \mid j \in \mathcal{N}_i \cup \{i\}\}$, the auction then operates according the following *max-consensus* process:

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

where the notation t^+ indicates a transition in r_i after all neighboring bids have been collected through messaging. As $\mathcal{G}$ would be controlled to be rigid, it will also be connected for all time, and thus (1) converges *uniformly* to the largest leadership bid $r_i = \text{argmax}_{r_j(0)}(b_j(0))$, $\forall\, i,j \in \mathcal{V}$ after some finite time [39].

Remark 1. Note that, in general there may exist different bids with the same value, albeit this is quite rare in the case of continuum weights. In this case, the argmax function should be slightly modified to take into account also the node ID i (which is unique) for the selection. For example, any time two bids b_i and b_j are identical, then the pair $r_i = \{i, b_i\}$ with $i < j$ could be selected.

After convergence of (1) the winning node then takes on the leadership role, with the previous leader releasing its status. The proposed auction mechanism allows us to decentralize the pebble game by assigning to each leader the responsibility of expanding $\bar{\mathcal{E}}$ by evaluating only their incident edges for independence, yielding local sets $\bar{\mathcal{E}}_i$, with $\bar{\mathcal{E}} = \cup_i \bar{\mathcal{E}}_i$. In determining edge independence, pebbles are *queried* from the network through a pebble exchange protocol (as detailed in [38]) in order to cover each copy of a quadrupled incident edge. Leadership then transfers to the next auction winner when the current leader's neighborhood has been exhausted.

As the node with the largest bid is elected, the bids dictate the *order* of elected leaders and thus the edges that constitute the identified rigid subgraph. The proposed auction technique therefore affords us control over $\bar{\mathcal{E}}$ that goes beyond simply discovering the network's rigidity property. Thus, based on our results in [38], we assume that the network can apply a decentralized (and asynchronous) pebble game to determine with $O(n)$ complexity the independence of any edge e_{ij} in $\mathcal{G}$ (represented in pseudocode by $\textsc{PebbleGame}(e_{ij})$). We will then exploit the leader-based mechanism to control the *order* and *cardinality* of considered edges, and ultimately optimality vs. complexity.

2.3 Matroid Theory and Optimization

The study of matroids is an analysis of an abstract theory of dependence. Matroids are combinatorial structures that generalize the notion of linear independence in matrices.

Definition 1. A matroid $\mathcal{M}$ on $\mathcal{E}$ is an ordered pair $(\mathcal{E}, \mathcal{I})$ consisting of a finite set $\mathcal{E}$ (denoted as the *ground set* of $\mathcal{M}$) and a collection $\mathcal{I}$ of subsets of $\mathcal{E}$ (denoted as the *independent sets* of $\mathcal{M}$) satisfying the following three conditions:

i) $\emptyset \in \mathcal{I}$.
ii) If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$.
iii) If I_1 and I_2 are in $\mathcal{I}$ and $|I_1| < |I_2|$, then there is an element e of $I_2 \setminus I_1$ such that $I_1 \cup e \in \mathcal{I}$.

The third condition is called the *independence augmentation axiom*, and it says that if I_1 is independent and there exists a larger independent set I_2 then I_1 can be extended to a larger independent set by adding an element of $I_2 \setminus I_1$. Condition (iii) implies that every maximal (inclusion-wise) independent set is maximum; in other words, all maximal independent sets have the same cardinality. A maximal independent set is called a *base* of the matroid $\mathcal{M}$, while a minimal dependent set is called a *circuit* of the matroid $\mathcal{M}$. The following lemma provides a formalization of the fact all bases share the same cardinality.

Lemma 2 (Lemma 1.2.1, [28])**.** *If B_1 and B_2 are bases of a matroid $\mathcal{M}$, then $|B_1| = |B_2|$.*

Let us denote with $\mathcal{B}$ the collection of bases of a matroid $\mathcal{M}$, for which the following properties hold:

Lemma 3 (Corollary 1.2.5, [28])**.** *Consider a matroid $\mathcal{M}$ on $\mathcal{E}$ and let $\mathcal{B}$ be a set of subsets of the set $\mathcal{E}$. Then $\mathcal{B}$ is the collection of bases of a matroid on $\mathcal{E}$ if and only if it satisfies the following conditions:*

(B1) *$\mathcal{B}$ is non-empty.*
(B2) *If B_1 and B_2 are members of $\mathcal{B}$ and $x \in B_1 \setminus B_2$, then there is an element $y \in B_2 \setminus B_1$ such that $(B_1 \setminus x) \cup y \in \mathcal{B}$.*

In this work, we are interested in a family of matroids denoted as *rigidity matroids*. In particular, we focus on the combinatorial characterization of the rigidity matroid of a graph in $\mathbb{R}^2$. More specifically, consider a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ in $\mathbb{R}^2$, the matroid $R^2(\mathcal{G}) = (\mathcal{E}, \mathcal{I})$ can be defined according to the Laman edge set independence conditions, that is the collection $\mathcal{I}$ of independent sets of edges can be defined as follows:

$$\mathcal{I} = \{\bar{\mathcal{E}} \subseteq \mathcal{E} \,:\, 2\,|\mathcal{V}(\mathcal{E}')| - 3 \geq |\mathcal{E}'|, \forall\, \mathcal{E}' \subseteq \bar{\mathcal{E}}\} \tag{2}$$

where $\mathcal{V}(\mathcal{E}')$ denotes the set of vertices of $\mathcal{V}$ incident to at least one edge in $\mathcal{E}'$. If the graph $\mathcal{G}$ is rigid, then any base B corresponds to a Laman subgraph of $\mathcal{G}$. The reader is referred to [29] for a comprehensive overview of the topic.

A reason of interest for matroids in the field of combinatorial optimization is their association with greedy algorithms [40]. In particular, the following problem definition is considered.

Problem 1. Consider a matroid $\mathcal{M} = (\mathcal{E}, \mathcal{I})$ and let us suppose there exists a weight function $w : \mathcal{M} \to \mathbb{R}^+$ which assigns a weight to each element of the ground set $\mathcal{E}$. The goal is to find a base B of M such that the weight $\sum_{x \in B} w(x)$ is maximized.

A greedy algorithm which iteratively selects at each step k the element $x \in \mathcal{E}$ with the largest weight such that $\mathcal{I}(k-1) \cup \{x\}$ is independent is guaranteed to converge to a maximal independent set. Furthermore, it can be proven the base B obtained with this approach is *optimal*, that is it maximizes the weight function w previously defined. Intuitively, this follows from the independence augmentation axiom in Definition 1, i.e., condition (iii), the fact all bases share the same cardinality as detailed in Lemma 2 and the property B2 of Lemma 3.

3 Optimally Rigid Graph Construction

Let us begin by providing some preliminary assumptions and definitions which are instrumental for the considered optimization problem. Consider a generic utility function $\rho(\mathcal{E}) : \mathcal{E} \to \mathbb{R}_{>0}$ of the form:

$$\rho(\mathcal{E}) = \sum_{e_{ij} \in \mathcal{E}} w_{ij} \tag{3}$$

to be defined over the graph $\mathcal{G}(\mathcal{V},\mathcal{E})$. We are now ready to introduce the definition of an *optimal* subgraph $\mathcal{G}^*$:

Definition 2. A subgraph $\mathcal{G}^*(\mathcal{V},\mathcal{E}^*) \subseteq \mathcal{G}(\mathcal{V},\mathcal{E})$ is optimal if:

$$\nexists\, \mathcal{G}'(\mathcal{V},\mathcal{E}') \subseteq \mathcal{G}(\mathcal{V},\mathcal{E}) : \rho(\mathcal{E}') > \rho(\mathcal{E}^*),$$

with $\mathcal{G}^*$, $\mathcal{G}'$ *minimally* rigid. See Fig. 1c and Fig. 1d for an illustration of an optimal Laman subgraph.

Now, the problem we address in this paper can be characterized as follows.

Problem 2. Consider a rigid graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and a utility function $\rho(\mathcal{E})$ of the form (3). The goal is to find the optimal subgraph given by:

$$\mathcal{G}^* = \underset{\mathcal{G}'(\mathcal{V},\mathcal{E}')\subseteq\mathfrak{R}(\mathcal{G})}{\arg\max} \left\{\rho(\mathcal{E}') : |\mathcal{E}'| \leq \kappa\right\}, \tag{4}$$

where $\kappa = 2n-3$ and $\mathfrak{R}(\mathcal{G})$ denotes the set of all possible rigid subgraphs of $\mathcal{G}$.

We reiterate that in a decentralized context, Problem 2 becomes particularly interesting for the case of minimally rigid graphs. Optimal edges beyond the point of minimal rigidity, i.e., $\kappa > 2n-3$, no longer require the rigidity conditions to be verified. Indeed, at that point optimality is merely a function of the marginal utility of each edge. Thus, without loss of generality we will limit ourself to the construction of minimally rigid graphs, that is $\kappa = 2n-3$. It is important to note however, that our proposed algorithms will work for any $\kappa \geq 2n-3$, simply by considering the edge independence constraint only until a minimally rigid graph is constructed.

3.1 Optimal Greedy Algorithm

The Optimal Greedy algorithm (OG) proposed in this work extends the decentralized version of the pebble game originally proposed by the authors in [38]. Briefly speaking, major modifications are made to the bidding process for leader election and the number of edges each leader can check for independence at each iteration.

The pseudo-code of the proposed OG algorithm is depicted in Algorithm 1. Generally speaking, the algorithm works as follows. In steps 2–3, each node initializes the local independent set $\mathcal{E}_i^*$ and the local set of edges to check, $\mathcal{E}_i$. Then, the algorithm runs until the cardinality of the independent set is $|\mathcal{E}^*(k)| = 2n-3$, that is an optimal Laman subgraph has been found at the k^{th} iteration. In particular, the following steps are executed. First, the bidding process for the leader election detailed in Algorithm 2 is run (step 6). Then, the chosen leader selects (and removes) from the local set of candidates the edge with maximum weight and checks its independence by running the decentralized pebble game according to Algorithm 3. This edge is then added to the local independent edge set if the check succeeds, or it is removed otherwise. Let us now further detail the three algorithms mentioned above.

Algorithm 1 The Optimal Greedy Algorithm.

```
 1: procedure OptGreedy($\mathcal{G} = (\mathcal{V},\mathcal{E})$)
 2:     $\mathcal{E}_i^*(0) = \{\emptyset\},\ \forall i \in \mathcal{I}$
 3:     $\mathcal{E}_i(0) \leftarrow \text{Sort}\left(\{e_{ij} : j \in \mathcal{N}_i\}\right),\ \forall i \in \mathcal{I}$
 4:     $k \leftarrow 1$
 5:     repeat
 6:         $\triangleright$ Compute the bids for each $i$:
 7:         $\{b_i(k), e_{ij}^{\max}(k)\} \leftarrow$ComputeBid($i$, $\mathcal{E}_i(k-1)$)
 8:         $\triangleright$ Auction for leader with maximum bid:
 9:         $l \leftarrow \arg\max_i b_i(k)$
10:         $\triangleright$ Leader checks $e_{lj}^{\max}(k)$ independence:
11:         $\mathcal{E}_l^*(k) \leftarrow$LeaderRun($l$, $\mathcal{E}_l^*(k-1), e_{lj}^{\max}(k)$)
12:         $\triangleright$ Remove already considered edges:
13:         $\mathcal{E}_l(k) = \mathcal{E}_l(k-1) \setminus \{e_{lj}^{\max}(k)\}$
14:         $\mathcal{E}_j(k) = \mathcal{E}_j(k-1) \setminus \{e_{jl}^{\max}(k)\}$
15:         $k \leftarrow k+1$
16:     until $|\mathcal{E}^*(k)| = |\cup_i \mathcal{E}_i^*(k)| = 2n-3$
17: end procedure
```

Algorithm 2 computes the bids for each node before the leader election step. The value of the bid is determined according to the cardinality of the set $\mathcal{E}_i(k)$, i.e., the set of candidate edges at step k. If this set is empty, i.e., all the incident edges have been checked, zero is assigned to the bid (step 6). As will be explained in the following, a leader is elected by the algorithm if it submits the largest bid. Therefore, a bid of zero prevents node i from becoming a leader. If $|\mathcal{E}_i(k)| \neq 0$, then the maximum weight among the candidates edges is computed and used as the bid (step 3). Using the largest weight as a bid guarantees that the solution is incrementally built considering the graph edges in a decreasing order with respect to their weights. The edge corresponding to the maximum weight in $\mathcal{E}_i(k)$ is stored by each node i in $e_{ij}^{\max}(k)$ because if i is elected as the next leader, then $e_{ij}^{\max}(k)$ is the edge whose independence has to be checked (step 4). Algorithm 3 describes the leader execution. In step 2, the decentralized pebble game originally introduced by the authors in [38] is used for checking the independence of $e_{lj}^{\max}(k)$. If the check succeeds, then the edge is added to the i^{th} local set of independent edges in step 4.

In light of the matroid theory briefly described in Section 2.3, we now demonstrate the optimality and complexity properties of the proposed distributed OG algorithm.

Theorem 4 (Greedy Algorithm Optimality)**.** *The solution $\mathcal{G}^*(\mathcal{V},\mathcal{E}^*) \subseteq \mathcal{G}(\mathcal{V},\mathcal{E})$ built incrementally according to Algorithm 1 is optimal.*

Proof: Let us consider the first iteration of the algorithm, i.e, $k = 1$. The bid choice in Algorithm 1 guarantees that the edge $e_{lj}^{\max}(1)$ with maximum weight is selected and added to the independent set $\mathcal{E}_l^*(1)$ with l

Algorithm 2 Compute bid and maximum weight edge.

```
1: procedure ComputeBid($i, \mathcal{E}_i(k-1)$)
2:     if $\mathcal{E}_i(k-1) \neq \{\emptyset\}$ then
3:         $b_i(k) = \max_{e_{ij} \in \mathcal{E}_i(k-1)} w_{ij}$
4:         $e_{ij}^{\max}(k) = \arg\max_{e_{ij} \in \mathcal{E}_i(k-1)} w_{ij}$
5:     else
6:         $b_i(k) = 0$
7:     end if
8: end procedure
```

Algorithm 3 The leader execution.

```
1: procedure LeaderRun($l$, $\mathcal{E}_l^*(k-1)$, $e_{lj}^{\max}(k)$)
2:     $ind$ ←PebbleGame($e_{lj}^{\max}(k)$)
3:     if $ind$ is true then
4:         $\mathcal{E}_l^*(k) = \mathcal{E}_l^*(k-1) \cup \{e_{lj}^{\max}(k)\}$
5:     end if
6: end procedure
```

the index of the current leader. Therefore the value of the objective function computed using $\mathcal{E}^*(1)$ is maximum. Let us now consider the generic k^{th} iteration, with $k > 1$. From line 2 of Algorithm 3 it follows that two outcomes are possible, that is either $e_{lj}^{\max}(k)$ is independent or $e_{lj}^{\max}(k)$ is redundant. If the edge is independent, then it must be $|\mathcal{E}^*(k-1)| < 2n-3$, otherwise the edge would be redundant. Thus, the insertion of $e_{lj}^{\max}(k)$ (line 4) guarantees that the objective function computed using the edges in $\mathcal{E}^*(k) = \mathcal{E}^*(k-1) \cup \{e_{lj}^{\max}(k)\}$ is maximum because of the $e_{lj}^{\max}(k)$ definition. On the other hand, if the edge $e_{lj}^{\max}(k)$ is redundant we cannot simply add this edge to the current independent set $\mathcal{E}^*(k-1)$. Recall that according to Problem 2 the solution must be a Laman subgraph. Thus as $\mathcal{E}^*(k-1) \cup \{e_{lj}^{\max}(k)\}$ is not a Laman subgraph, $e_{lj}^{\max}(k)$ must be discarded. Alternatively, according to property B2 of Lemma 3, we could exchange the edge $e_{lj}^{\max}(k)$ with any element $e_{ij} \in \mathcal{E}^*(k-1)$. However, this would not result in any improvement of the objective function as $e_{lj}^{\max}(k) \leq e_{ij}$, $e_{ij} \in \mathcal{E}^*(k-1)$. Notice that in the case of equality, allowing the exchange simply would result in an alternative optimal solution, which is not of interest, as in this work all optimal solutions are considered to be equivalent. □

In the following proposition, the communication complexity of the algorithm, in terms of number of exchanged messages, is characterized.

Proposition 1 (Greedy Algorithm Complexity). *Algorithm 1 exhibits a O(n^3) worst case per-node communication complexity for the leader election process.*

Proof: In the leader election process each node sends O(n) messages for each auction [41, Theorem 1]. Now, since in the worst case, all the edges of the network are analyzed, then O(n^2) auctions must be executed. Therefore, the overall messaging complexity per-node becomes O(n^3). □

3.2 Heuristic Sub-Optimal Algorithm

In this section we introduce a modified algorithm, denoted as the SO algorithm, by which the complexity of the OG algorithm can be mitigated. To this end, we modify the OG algorithm by introducing a tuning parameter ξ which represents the number of edges to be checked for independence at each leader election. As in the previous section, let us assume that each node i is able to sort its incident edges $\mathcal{E}_i$ according to the associated weights w_{ij}. Before the algorithm execution, a parameter $\xi \in [1, \ldots, n-1]$ is chosen. It represents the maximum number of edges that an elected leader is allowed to check for independence. This parameter is used by modified versions of ComputeBid and LeaderRun algorithms previously described. Briefly, these functions are extended to deal with *sets* of edges rather than a *single* edge as in the optimal algorithm.

In the modified bid computation algorithm, the bid for the leader auction process is computed by each node summing up the first ξ edge weights and the edges themselves are stored in $\mathcal{E}_i^{\xi}$. This represents a conservative approach to maximize the increase of the solution at step $k+1$, even though this may lead to a sub-optimal solution. Likewise, in the modified leader run algorithm, the edges in the set $\mathcal{E}_i^{\xi}$ are checked for independence. Each edge is inserted into the i^{th} portion of the independent set after a successful independence check, it is removed otherwise.

In allowing each elected leader to insert ξ edges we may deviate from the greedy edge ordering that leads us to the optimal rigid subgraph. The parameter ξ yields what is effectively a *lazy* auction controlling the evaluation of edge independence. We expect that the network size n and the queue size ξ will factor into the optimality gap experienced by the SO algorithm. Let us denote with $\mathcal{E}^*$ the optimal solution built by the OG algorithm and $\hat{\mathcal{E}}^*(k)$ the sub-optimal set built up to the k-th auction by the SO-algorithm. Similarly to [40], the following inequality holds for any auction k:

$$\rho(\mathcal{E}^*) \leq \sum_{(i,j) \in \hat{\mathcal{E}}^*(k)} w_{ij} + \sum_{(i,j) \in \mathcal{E}^* \setminus \hat{\mathcal{E}}^*(k)} w_{ij}. \tag{5}$$

Now, let us consider k to be the auction for which a complete suboptimal solution is built. Then by applying (5) we have:

$$\rho(\mathcal{E}^*) \leq \rho(\hat{\mathcal{E}}^*) + |\tilde{\mathcal{E}}|\,(\xi - 1)\,\rho(\hat{\mathcal{E}}^*)$$

where $\tilde{\mathcal{E}} = \mathcal{E} \setminus \mathcal{E}^*$ denotes the set of redundant edges for which $|\tilde{\mathcal{E}}| = (|\mathcal{E}| - 2n + 3)$. Finally we can conclude with the following bound on the quality of our suboptimal solution:

$$\frac{\rho(\hat{\mathcal{E}}^*)}{\rho(\mathcal{E}^*)} \geq \frac{1}{1 + |\tilde{\mathcal{E}}|\,(\xi - 1)} \tag{6}$$

Like [40], we point out that while this bound is in general not tight, its soundness is corroborated by the fact the when either $\tilde{\mathcal{E}} \to 0$ and $\xi \to 1$, the suboptimal solution correctly approaches the optimal one.

Let us now analyze the messaging complexity exhibited by the algorithm:

Proposition 2 (Lazy Algorithm Complexity)**.** *The modified algorithm exhibits a $O(n^3/\xi)$ worst case per-node communication complexity for the leader election process.*

Proof: In the leader election process each node sends $\mathrm{O}(n)$ messages for each auction. In the case of a fully connected graph, $O(n^2)$ edges must be inspected. Then, $\mathrm{O}(n^2/\xi)$ auctions must be executed. Therefore, the overall messaging complexity per-node becomes $\mathrm{O}(n^3/\xi)$. □

Remark 2. It is worthy to note that if $\xi \to 1$, the per-node messaging complexity is $\mathrm{O}(n^3)$ as the sub-optimal algorithm collapses to the greedy optimal version. Instead, if $\xi \to n-1$ the per-node complexity becomes quadratic, i.e., $\mathrm{O}(n^2)$. Thus ξ behaves exactly like a sliding mode control, trading off complexity with optimality.

3.3 Optimal Parallel Algorithm

While the complexity of the OG and SO algorithms is reasonable for scalable implementation (i.e., polynomial in network size), it is clear that execution is inherently serial. The nature of edge independence requires such a serial implementation in general; the independence of a given edge must be determined *relative* to those edges previously considered. However, we would like to better exploit our decentralized setting by allowing the optimal independent edge set to be built, at least partially, in *parallel*.

Despite the global scope of the rigidity property, we can make two observations that guarantee optimal edge independence by examining only *local* information. First, taking inspiration from the *Henneberg operations* for rigid graph construction [24], it follows that *any* rigid graph must possess nodes with at least degree two. This implies that any edge incident to a node with degree less than two *must* be independent by construction. Second, as a consequence of rigid matroid optimization, a greedy edge ordering leads us to the optimal solution, and thus we can conclude that every optimal edge must be within the first two highest weighted edges for either (or both) of its endpoints. Although we formalize the preceding reasoning later, these simple rules are sufficient to construct an algorithm that builds in parallel a portion of the optimal independent edge set.

Let us now outline our Optimal Parallel Algorithm (OP). First, we assume execution and messaging operates according to an *asynchronous* model of time (e.g., as in [42]), and that each node acts according to their own local (and potentially unsynchronized) clock. Further, we assume that nodes handle messages in a first-in-first-out way, and that all local execution and message handling occurs atomically (i.e., without race conditions). The true complexity of the algorithm is dealing with coherence in an asynchronous setting. Specifically, in order to enforce our local rules for optimal independence, we associate with each node the following variables: *committed*$(i) \in \mathbb{Z}_{\geq 0}$, $M_i \in \mathbb{Z}_{\geq 0}$, and *best*$(i)$. The *commitment* variable stores the cardinality of i as an endpoint of edges in the distributed independent edge set, or more formally, the node degree of v_i in the graph $\mathcal{G}^* = (\mathcal{V}, \cup_{i \in \mathcal{V}} \mathcal{E}_i^*)$. Further, for each node i we keep a counter of the number of maximal edges that have been checked, denoted by M_i. Finally, the two maximal edges incident to node i are in the set *best*(i).

Algorithm 4 Parallel execution logic for node i.

```
1: procedure PARALLELRUN($i$)
2:     if $e_i \triangleq (i, j) \neq \mathbf{0} \wedge M_i < 2$ then        ▷ Next edge
3:         EDGEREQUESTMSG($i$, $j$)
4:         $requestedFrom(i) \leftarrow j$
5:         return
6:     end if
7:     ▷ All local edges checked:
8:     $idle(i) \leftarrow$ Yes
9: end procedure
```

As opposed to the leader-based execution of the OG and SO algorithms, here every node executes concurrently according to Algorithm 4. The precondition for execution is that the incident edge set $\mathcal{E}_i$ is sorted in descending order, the currently considered edge e_i is set to the maximum for node i, *committed*$(i) = 0$, and $M_i = 0$, for all i. As reflected in lines 2-6, each node simply iterates through their sorted incident edge set, sending for each edge a request message EDGEREQUESTMSG(i, j), blocking execution until a response is received. The request message indicates node i's desire to add an incident edge to the optimal independent set, and thus allows *agreement* between edge endpoints on the rules for optimal independence. The checking of incident edges stops when either the set $\mathcal{E}_i$ is empty (denoted by $e_i = \mathbf{0}$), or node i (or one of its neighbors) has checked both of the edges in *best*(i), i.e., $M_i = 2$. At this point, node i simply enters an idle state and waits until all nodes have completed execution.

The remaining logic for optimal independent edge selection and rule checking resides in the message handlers for EDGEREQUESTMSG(i, j), depicted in Algorithms 5 and 6. The reception of an edge request message by node i from neighbor $j \in \mathcal{N}_i$ triggers the message handling logic HANDLEEDGEREQUEST(i, j). As all nodes are trying to add incident edges to the independent edge set simultaneously, it may be the case that two nodes sharing an edge conflict on which node takes the edge, specifically as both cannot. Thus, receiving node i first determines whether a request represents a conflict over edge (i, j). Conflicts are resolved in lines 3-7 of Algorithm 5, where we make the basic assumption

Algorithm 5 Parallel request handler for node i.

```
1: procedure HandleEdgeRequest($i$, $j$)
2:     $response \leftarrow committed(i)$
3:     if $requestedFrom(i) = j$ then                ▷ Edge contention
4:         if $M_i < 2 \wedge (i < j \vee M_j \geq 2)$ then
5:             $response \leftarrow -1$            ▷ Ensure $i$ wins edge
6:         end if
7:     end if
8:     EdgeResponseMsg($i$, $j$, $response$)
9:     ▷ Count independent edge commitments:
10:    if $response \neq -1 \wedge committed(i) < 2$ then
11:        $committed(i) \leftarrow committed(i) + 1$
12:    end if
13:    ▷ Count maximally weighted edges checked:
14:    if $w_{ij} \in best(i) \wedge response \neq -1$ then
15:        $M_i \leftarrow M_i + 1$
16:    end if
17:    ▷ Do not double check $(i, j)$:
18:    $\mathcal{E}_i \leftarrow \mathcal{E}_i - (i, j)$
19: end procedure
```

Algorithm 6 Parallel response handler for node i.

```
1: procedure HandleEdgeResponse($i$, $j$, $response$)
2:     if $response \neq -1 \wedge$
3:     $(response < 2 \vee committed(i) < 2) \wedge M_i < 2$ then
4:         $\mathcal{E}_i^* \leftarrow \mathcal{E}_i^* \cup (i, j)$
5:         $committed(i) \leftarrow committed(i) + 1$
6:     end if
7:     ▷ Count maximally weighted edges checked:
8:     if $w_{ij} \in best(i) \wedge response \neq -1$ then
9:         $M_i \leftarrow M_i + 1$
10:    end if
11:    ▷ Go to next largest incident edge:
12:    $\mathcal{E}_i \leftarrow \mathcal{E}_i - (i, j)$
13:    $e_i \leftarrow \arg\max_{(i,j) \in \mathcal{E}_i}(w_{ij})$
14: end procedure
```

that node label is *one* of the factors in determining a conflict winner. Note however that node label alone is insufficient, as a node cannot win a conflict when it has exhausted its local maximally weighted edge set. Thus, we introduced the extra logic in the conflict resolution (line 4). Now, after handling potential conflicts, receiver i then responds to j via EdgeResponseMsg$(i, j, response)$ indicating its current commitment to $\mathcal{E}^*$, increases its own commitment and maximally weighted edge count (M_i) to guarantee optimal independence, and removes (i, j) from $\mathcal{E}_i$ to avoid double checking (lines 8-18).

In coherence with the edge request logic, in receiving an EdgeResponseMsg$(i, j, response)$, a node i acts according to Algorithm 6. As an edge response conveys node j's commitment to $\mathcal{E}^*$, it is validated against the previously specified rules (lines 2-3), with success resulting in the assignment of (i, j) to $\mathcal{E}_i$, and an incrementing of $committed(i)$ (lines 4-5). After independence preserving assignment, node i increments its maximally weighted edge count appropriately (lines 8-10), and moves to consider its next incident edge (lines 12-13).

As we show in the sequel, the parallel algorithm identifies a portion of the independent set for a given input graph. Thus, to identify a complete independent set, we must pass the results of the parallel algorithm to the OG or SO algorithm, yielding a composite rigidity evaluation. We omit the basic details of this composite algorithm, and direct the reader to our previous work [38], which discusses rigidity parallelization in a general context.

We are now going to demonstrate the correctness and complexity of the parallel algorithm. In this regard, the following proposition is instrumental for the optimality analysis.

Proposition 3 (Parallel Mutual Exclusivity)**.** *Consider the parallel algorithm given by Algorithms 4, 5 and 6 applied to a (rigid) graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Then the following holds:*

$$\bigcap_{i \in \mathcal{V}} \mathcal{E}_i^* = \emptyset \tag{7}$$

Proof: In order to prove the proposition, it is sufficient to show that when an edge $e_{ij} \in \mathcal{E}$ is added to the optimal local set $\mathcal{E}_i^*(k)$ of a node i at a time instant k, the following two conditions are enforced:

$$\underbrace{e_{ij} \notin \bigcup_{r \in \mathcal{V}} \mathcal{E}_r^*(k^-)}_{\text{(a)}} \quad \text{and} \quad \underbrace{e_{ij} \notin \bigcup_{r \in \mathcal{V} \setminus \{i\}} \mathcal{E}_r^*(k^+)}_{\text{(b)}} \tag{8}$$

with $k^- \in [0, k)$ and $k^+ \in [k, \infty)$. Condition (a) follows directly from the incremental consideration of local edges (lines 2-5 of Algorithm 4 and lines 12-13 of Algorithm 6), the conflict resolution (lines 3-7 of Algorithm 5) and the removal of local edges on edge request (line 18 of Algorithm 5). Condition (b) follows directly from the removal of local edges on edge request (line 18 of Algorithm 5) and the removal of local edges on edge response (line 12-13 of Algorithm 6). □

We now define a structural property of the optimal solution which turns out to provide a convenient condition for local and yet optimal choices for edge independence.

Proposition 4 (Local Optimal Independence)**.** *Consider Problem 2 having an optimal solution $\mathcal{G}^*$. Define a subset of independent edges $\hat{\mathcal{E}} \subseteq \mathcal{E}$, for which every $e_{ij} \in \hat{\mathcal{E}}$ is within the first two maximally weighted edges for at least one of the two endpoints i and j. Then it follows that any such subset $\hat{\mathcal{E}}$ is itself a subset of the optimal solution $\mathcal{E}^*$, i.e., $\hat{\mathcal{E}} \subseteq \mathcal{E}^*$.*

Proof: In order to prove the proposition, we first note that the optimal solution $\mathcal{E}^*$ must span the graph, which follows directly from the Laman conditions of Theorem 1. Furthermore, in a Laman subgraph any node by construction must have at least two incident edges. And thus since the optimal solution follows a greedy edge consideration, as detailed in Theorem 4, the result follows. □

We are now ready to state our main result on the parallel algorithm, that is by means of the parallel execution we can identify a subset of the optimal solution.

Proposition 5 (Parallel Algorithm Optimality)**.** *Consider the parallel algorithm given by Algorithms 4, 5 and 6 applied to a (rigid) graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$*. Then the union of the optimal local sets is a subset of the optimal solution, that is* $\bigcup_{i\in\mathcal{V}} \mathcal{E}_i^* \subseteq \mathcal{E}^*$*.*

Proof: In order to prove the proposition it is sufficient to notice that by construction the parallel algorithm builds the sets $\mathcal{E}_i^*$ by limiting the addition of an edge e_{ij} to a local independent set $\mathcal{E}_i^*$ only if such an edge is within the first two maximally weighted edges for at least one of the two endpoints i and j (see lines 2-6 of Algorithm 6). Then as the union $\bigcup_{i\in\mathcal{V}} \mathcal{E}_i^*$ follows the property of the subset $\hat{\mathcal{E}}$ of Proposition 4 the proposed condition holds. □

The following proposition identifies how much of the independent set we can identify in a parallel fashion.

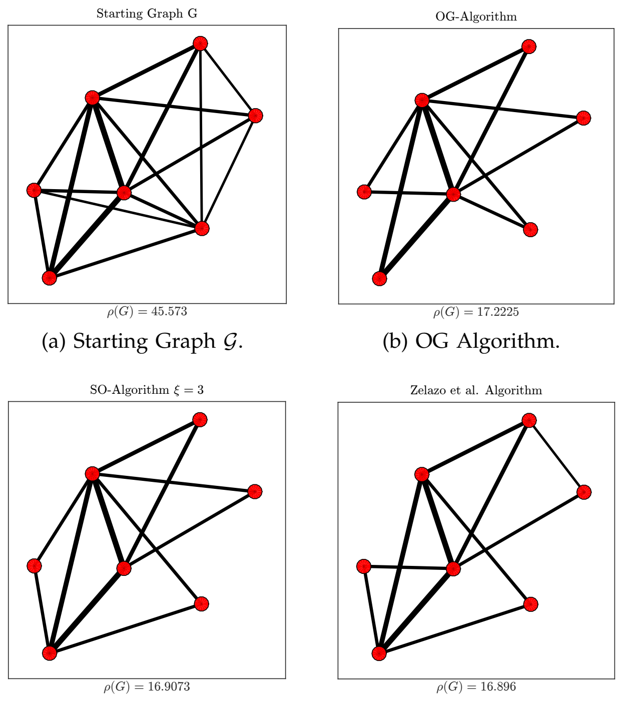
Proposition 6 (Parallel Algorithm Performance)**.** *Consider the parallel algorithm given by Algorithms 4, 5 and 6 applied to a rigid graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$*. Then:*

$$\left| \bigcup_{i\in\mathcal{V}} \mathcal{E}_i^* \right| \geq n. \tag{9}$$

Proof: In order to prove the proposition it is enough to notice that once the parallel execution terminates, the set $\mathcal{E}_{\mathbb{P}}^* = \bigcup_{i\in\mathcal{V}} \mathcal{E}_i^*$ by construction is such that the resulting graph $\mathcal{G}_{\mathbb{P}}^* = \{\mathcal{V}, \mathcal{E}_{\mathbb{P}}^*\}$ has the property that $|\mathcal{N}_i(\mathcal{G}_{\mathbb{P}}^*)| \geq 2$ for all i. Therefore, since in the worst case scenario these edges may be shared between the endpoints the lower bound on the number of identified independent edges is equal to n. □

Remark 3. Notably, by making an analogy with competitive analysis (see [43]), this can be seen as building a subset of the optimal solution with a "competitive ratio" given by $|\mathcal{E}^*|/|\mathcal{E}_{\mathbb{P}}^*| = (2n-3)/n = 2 - 3/n$. Indeed, the ratio tends to 2 as n goes to infinity, meaning that in the worst case scenario we identify at least half of the optimal set of edges, as demonstrated in Fig. 4b.

Remark 4. We reiterate that an important reason of interest for matroids in the field of combinatorial optimization is their association with greedy algorithms. Therefore, generally speaking the problem formulation adopted in this work could be easily and effectively applied to all cost functions for which greedy algorithms are known to perform well. In this regard, we mention an important family of functions denoted as *submodular set functions*, which have important implications in the canonical problem of sensor placement [44]. Note that, in this case our algorithms would achieve suboptimal solutions with theoretical approximation guarantees.



(a) Starting Graph $\mathcal{G}$. (b) OG Algorithm. (c) SO Algorithm. (d) Zelazo Algorithm.

Fig. 2. Rigid network composed of 7 vertices running the OG algorithm, the SO algorithm with $\xi = \lceil N/2 \rceil$ and the centralized algorithm proposed in [31], where edge thickness denotes normalized utility.

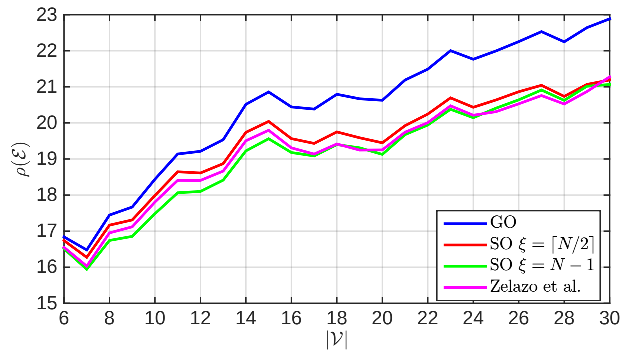
4 Simulation Results

In this section we present simulation results that demonstrate the correctness, complexity and performance of the proposed algorithms.

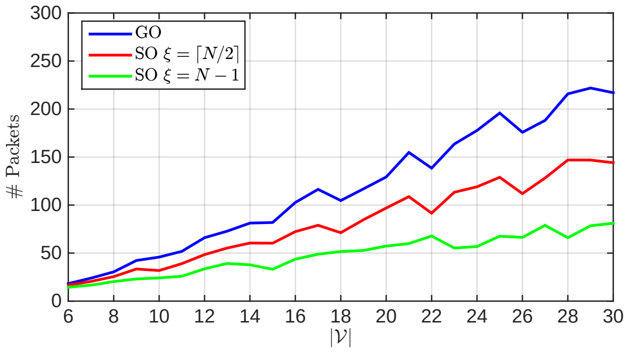
4.1 Minimally Rigid Network Optimization

In [31], a *centralized* algorithm for generating *locally* optimal rigid graphs according to a certain optimality criterion is provided. Briefly, this algorithm consists of a variation of the Henneberg construction for generating rigid graphs in the plane by adding performance requirements and sensing constraints, yielding the optimal vertex addition and edge splitting algorithms. We demonstrate how our algorithm can extend the state of the art both through decentralization and optimality.

For the purpose of Monte Carlo analysis, we compare the algorithm of [31] to our OG and SO algorithms over randomly generated non-minimally rigid graphs according to Problem 2. Figure 2 depicts a typical run of the algorithms for a rigid network composed of 7 vertices where the thickness of the edge denotes its normalized utility. It can be noticed that the OG algorithm provides the solution with the largest value of the optimization function $\rho(\mathcal{E}) = 17.225$, while the centralized algorithm given in [31] provides (as expected) a locally optimal solution $\rho(\mathcal{E}) = 16.896$, and finally the SO-algorithm with $\xi = \lceil N/2 \rceil$ which allows to reduce the communication cost also provides a sub-optimal solution $\rho(\mathcal{E}) = 16.9073$.

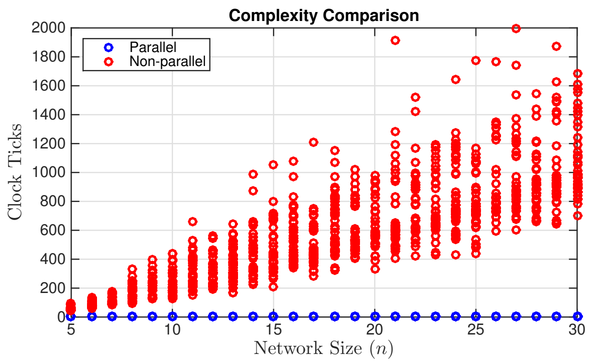


(a) Objective Function Comparison.

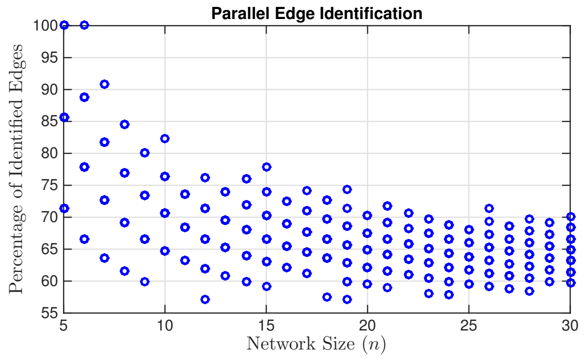


(b) Communication Cost Comparison.

Fig. 3. Monte Carlo Analysis for a network with a number of nodes ranging from 6 to 30. (a) shows a comparison of the average objective function for the OG algorithm, the SO algorithm with $\xi = \lceil N/2 \rceil$, the SO algorithm with $\xi = N - 1$, the centralized algorithm described in [31], respectively. (b) shows the average communication cost for the OG algorithm, the SO algorithm with $\xi = \lceil N/2 \rceil$, and the SO algorithm with $\xi = N - 1$, respectively.



(a) Complexity Comparison.



(b) Parallel Edge Identification.

Fig. 4. (a) Monte Carlo analysis of complexity for the decentralized OG algorithm, and the OP algorithm, for networks sizes $n \in [5, 30]$. Complexity is measured in asynchronous clock ticks, assuming zero communication delay. (b) Monte Carlo analysis of the percentage of optimal edges identified by the OP algorithm, for networks sizes $n \in [5, 30]$.

Furthermore, we compare the performance of the proposed SO algorithm against the centralized algorithm in [31] for different values of the tunable parameter ξ. In particular, Figure 3 depicts the results of Monte Carlo analysis for a network with a number of nodes ranging from 6 to 30. For a fixed network size, we consider 50 different graph topology realizations with random (positive) weights. Fig. 3a depicts the average of the objective function for the OG algorithm, the SO algorithm with $\xi = \lceil N/2 \rceil$, the SO algorithm with $\xi = N - 1$, and the centralized algorithm described in [31], respectively. It can be noticed that the OG algorithm as expected provides always the best (optimal) solution. Notably, the SO algorithm with different ξ values, although sub-optimal still provides satisfactory results, comparable with the (locally) optimal solution given by the centralized algorithm described in [31]. Fig. 3b shows a comparison of the average communication cost in terms of number of packets exchanged for the OG algorithm, the SO algorithm with $\xi = \lceil N/2 \rceil$, and the SO algorithm with $\xi = N-1$, respectively. It can be noticed that, while providing sub-optimal solutions, the SO-algorithm scales better in terms of the number of packets exchanged compared to the OG-algorithm.

4.2 Parallel Algorithm Evaluation

In this section, we again use Monte Carlo to demonstrate the effectiveness of our parallel extension to the OG algorithm. Random non-minimally rigid graphs are again the basis of comparison in the context of Problem 2. Fig. 4 depicts the outcome of the Monte Carlo analysis. In particular, Fig. 4a describes the complexity for the decentralized optimal algorithm, and the parallel optimal algorithm, for networks sizes $n \in [5, 30]$. Note that, complexity is measured in asynchronous clock ticks, assuming zero communication delay. Furthermore, Fig. 4b shows the percentage of optimal edges that are identified by the parallel optimal algorithm, for networks sizes $n \in [5, 30]$.

As expected, it can be noticed that the time complexity of the parallelization is $O(1)$. Clearly, this follows directly from the local nature of the rules by which each node i builds the local independent sets $\mathcal{E}_i^*$ with $i = 1, \ldots, n$. That is, the two maximum incident edges are considered for each node and thus the local algorithm execution does not depend upon the size of the network. Thus allowing to efficiently identify a significant portion of the optimal independent set. In addition, the fact that synchronization is not required renders the implementation of the overall algorithm practical.

Finally, the Monte Carlo analysis also corroborates the parallelization performance bound we derived in Proposition 6. In particular, our results suggest that we can indeed identify at least half of the optimal independent subset with constant complexity. Interestingly, the asymptotic nature of the competitive ratio pointed out in Remark 3 is also demonstrated in Fig. 4b. In particular, it appears that for smaller networks a larger portion of the optimal set is identified, while the lower bound n is met as the network size increases.

4.3 Relative Sensing: A Case Study

In this section, Relative Sensing Networks (RSNs) are considered as a realistic application scenario [34]. By assuming nodes to have linear (possibly heterogeneous) dynamics, a useful metrics is proposed in [34], namely the $\mathcal{H}_2$ norm. It allows to investigate the role of the underlying connection topology on the system norms mapping the exogenous inputs to the relative sensed output. In this way a characterization of the robustness of the network against external disturbances is derived. In this context, we consider a *mobile* network with both spatial and information-based objectives. For the spatial objective, we assume that the network is required to disperse in the environment, which is a primitive behavior often found in applications such as coverage, monitoring or tracking. For the information-based objective, we assume that the network is required to sense and exchange information to reach a common objective. As an example, we choose the *consensus* objective as it represents a fundamental building-block of various important information-based applications. Examples include distributed estimation and filtering, global localization and clock synchronization. Finally, we require rigidity in the network to guarantee localizability while also ensuring connectivity as a by-product. That is, our simulated system will achieve dispersion while maintaining the optimal underlying rigid topology for minimizing the impact of noise on consensus according to the $\mathcal{H}_2$ metrics. The reader is referred to [7] and [32] for a comprehensive overview of topology-constrained mobility control and the consensus problem, respectively.

Now, similarly to the mathematical framework for RSNs proposed in [34], [45], we begin deriving the $\mathcal{H}_2$ metrics by consider the following open-loop consensus model:

$$\begin{aligned} \dot{x}(t) &= u(t) \\ y(t) &= E(\mathcal{G})^T x(t) + v(t) \end{aligned} \tag{10}$$

where $x(t) \in \mathbb{R}^n$ is an internal state vector, $E(\mathcal{G}) \in \mathbb{R}^{n\times|\mathcal{E}|}$ is the incidence matrix with arbitrary orientation and $v(t) \in \mathbb{R}^{|\mathcal{E}|}$ is the gaussian zero mean white noise modeling the channel quantization effects [46], with σ_{ij}^2 the variance for each edge $e_{ij} \in \mathcal{E}$ and $\sigma_{ij} = \sigma_{ji}$. We point out that the considered quantization model is simplified for the sake of the illustration. Indeed, if our concern were quantization, there would be more sophisticated modeling to more effectively approach the problem.

Consider now the following output-feedback control $u(t) = -E(\mathcal{G})y(t)$ from which the following closed loop is then obtained:

$$\Sigma : \begin{cases} \dot{x}(t) = -L(\mathcal{G})x(t) - E(\mathcal{G})v(t) \\ z(t) = E(\mathcal{G})^T x(t) \end{cases} \tag{11}$$

where $L(\mathcal{G})$ is the symmetric Laplacian matrix commonly used to describe multi-agent interaction (see [32]). Notice that, $x(t)$ is the *consensus*-variable which could represent for example a state for clock synchronization or a shared variable for distributed estimation. Thus, $x(t)$ is not explicitly position-dependent, although the spatial position of the nodes dictates the network topology over which consensus operates (due to proximity limitations), and on which the $\mathcal{H}_2$ metric is dependent.

According to the developments of [45], by means of a coordinate transformation, the $\mathcal{H}_2$ norm for the system Σ is:

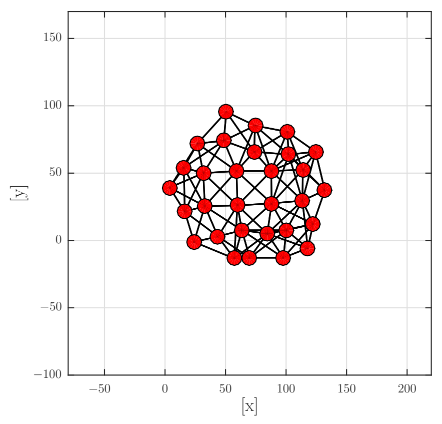
$$\|\Sigma\|_2^2 = \sum_{(i,j)\in\mathcal{E}} \sigma_{ij}^2. \tag{12}$$

Then, by assuming $w_{ij} = \sigma_{ij}^2$ we obtain the formulation of Problem 2 where the utility function is defined as:

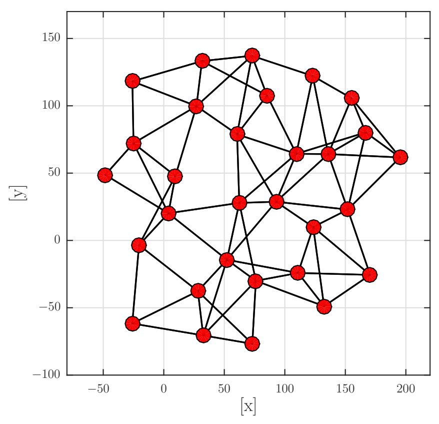
$$\rho(\mathcal{E}) = -\sum_{(i,j)\in\mathcal{E}} \sigma_{ij}^2. \tag{13}$$

Fig. 5 depicts a typical run for a spatially interacting mobile RSN composed of 30 nodes performing a dispersive behavior while preserving rigidity of the network topology and minimizing the $\mathcal{H}_2$ norm for consensus. In particular, the network is initialized in a densely connected topology where we generate random zero mean white noise with variance σ_{ij}^2 for each edge $e_{ij} \in \mathcal{E}$. Before motion begins, the network computes with the known noise the optimal topology using our OG algorithm, and motion is constrained to retain edges belonging to the optimal topology using [7]. It can be noticed in the figure how the network dispersion is constrained by the underlying optimally rigid topology for consensus. Indeed, an unconstrained dispersion would yield uniform relative distances at convergence.

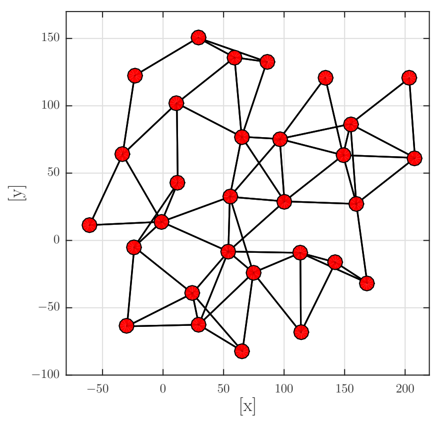
Fig. 6 corroborates the fact that maintaining the optimal topology for consensus yields superior performance in terms of noise rejection, even while performing an additional spatial objective. In particular, the blue line represents the error in the consensus value due to the effect of noisy edges in the case of the optimal topology, while the red line represents the error in the case of maintaining a randomly chosen minimally rigid topology. It is important to note that, the results are not skewed by the number of edges in the chosen topologies as they are both minimally rigid. Instead, it is our optimization that identifies the superior topology.



(a) Initial configuration at time $t = 1$.



(b) Intermediate configuration at time $t = 100$.



(c) Final configuration at time $t = 500$.

Fig. 5. Spatially interacting mobile RSN composed of 30 nodes performing a dispersive behavior while preserving rigidity of the network topology and maximizing the negative $\mathcal{H}_2$ norm.

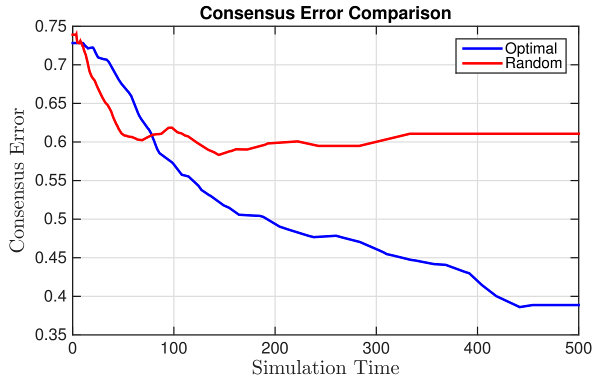


Fig. 6. Consensus error over time, comparing the optimal $\mathcal{H}_2$ topology to a randomly chosen rigid topology.

5 Conclusions

In this work, the problem of building an optimal rigid subgraph has been considered. In particular, three decentralized approaches were presented. The first approach iteratively built an optimal minimally rigid graph by electing leaders to check the independence of at most a single incident edge. To mitigate the messaging complexity required by this algorithm, a modified version was then introduced which allows to scale the complexity by choosing a parameter ξ governing the number of weights incorporated in the bids and the number of edges considered for the independence check. Finally, a parallelization was presented which was shown to identify a significant portion of the optimally rigid graph in constant time. Simulation results were provided to corroborate the theoretical findings.

References

[1] T. Arai, E. Pagello, and L. E. Parker, "Editorial: Advances in multi-robot systems," *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002.

[2] A. Gasparri, S. Panzieri, and F. Pascucci, "A spatially structured genetic algorithm for multi-robot localization," *Intelligent Service Robotics*, vol. 2, no. 1, pp. 31–40, 2009.

[3] Z. Yang and Y. Liu, "Understanding node localizability of wireless ad hoc and sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 8, pp. 1249–1260, Aug 2012.

[4] C. Carletti, M. Di Rocco, A. Gasparri, and G. Ulivi, "A distributed transferable belief model for collaborative topological map-building in multi-robot systems," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Oct 2010, pp. 554–560.

[5] P. Chand and D. A. Carnegie, "Mapping and exploration in a hierarchical heterogeneous multi-robot system using limited capability robots," *Robotics and Autonomous Systems*, 2013.

[6] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 427–438, 2013.

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

[8] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast Data Collection in Tree-Based Wireless Sensor Networks," *Mobile Computing, IEEE Transactions on*, 2012.

[9] T. ElBatt and A. Ephremides, "Joint scheduling and power control for wireless ad hoc networks," *Wireless Communications, IEEE Transactions on*, 2004.

[10] R. K. Williams and G. S. Sukhatme, "Probabilistic Spatial Mapping and Curve Tracking in Distributed Multi-Agent Systems," in *IEEE International Conference on Robotics and Automation*, 2012.

[11] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin, "PermaSense: Investigating Permafrost with a WSN in the Swiss Alps," in *Proc. of the 4th Workshop on Embedded Networked Sensors*, 2007.

[12] J. Gancet, E. Motard, A. Naghsh, C. Roast, M. Arancon, and L. Marques, "User interfaces for human robot interactions with a swarm of robots in support to firefighters," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[13] A. Gasparri, B. Krishnamachari, and G. S. Sukhatme, "A framework for multi-robot node coverage in sensor networks," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2-4, pp. 281–305, Apr. 2008.

[14] E. Gelal, G. Jakllari, S. V. Krishnamurthy, and N. E. Young, "Topology Management in Directional Antenna-Equipped Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, 2009.

[15] S. Poduri, S. Pattem, B. Krishnamachari, and G. S. Sukhatme, "Using Local Geometry for Tunable Topology Control in Sensor Networks," *IEEE Transactions on Mobile Computing*, 2009.

[16] M. Li, I. Stojmenovic, and Y. Wang, "Partial Delaunay triangulation and degree limited localized Bluetooth scatternet formation," *Parallel and Distributed Systems, IEEE Transactions on*, 2004.

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

[18] B. Anderson, C. Yu, B. Fidan, and J. Hendrickx, "Rigid graph con-

trol architectures for autonomous formations," *Control Systems, IEEE*, 2008.

[19] 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.

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

[21] B. Jackson and T. Jordan, "Connected rigidity matroids and unique realizations of graphs," *J. Comb. Theory Ser. B*, 2005.

[22] B. Hendrickson, "Conditions for unique graph realizations," *SIAM J. Comput.*, 1992.

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

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

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

[26] 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.

[27] 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*, 2013.

[28] J. G. Oxley, *Matroid Theory*, ser. Oxford Graduate Texts in Mathematics. Oxford University Press, 1997.

[29] M. Develin, J. Martin, and V. Reiner, "Rigidity theory for matroids," *Comment. Math. Helv.*, vol. 82, no. 1, pp. 197–233, 2007.

[30] 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.

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

[32] R. Olfati-Saber, J. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[33] M. De Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *Decision and Control, 2006 45th IEEE Conference on*, Dec 2006, pp. 3628–3633.

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

[35] B. Roth, "Rigid and Flexible Frameworks," *The American Mathematical Monthly*, 1981.

[36] L. Asimow and B. Roth, "The rigidity of graphs, II," *Journal of Mathematical Analysis and Applications*, 1979.

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

[38] 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*, 2013.

[39] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research*, 1988.

[40] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions?i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[41] A. Fagiolini, M. Pellinacci, G. Valenti, G. Dini, and A. Bicchi, "Consensus-based distributed intrusion detection for multi-robot systems," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008.

[42] A. Nedic, "Asynchronous Broadcast-Based Convex Optimization over a Network," *IEEE Transactions on Automatic Control*, 2010.

[43] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. New York, NY, USA: Cambridge University Press, 1998.

[44] A. Krause and D. Golovin, *Tractability (Practical Approaches to Hard Problems)*. Cambridge, United Kingdom: Cambridge University Press, 2014, ch. Submodular function maximization.

[45] D. Zelazo and M. Mesbahi, "Edge agreement: Graph-theoretic performance bounds and passivity analysis," *Automatic Control, IEEE Transactions on*, vol. 56, no. 3, pp. 544–555, March 2011.

[46] B. Widrow, I. Kollar, and M.-C. Liu, "Statistical theory of quantization," *Instrumentation and Measurement, IEEE Transactions on*, vol. 45, no. 2, pp. 353–361, Apr 1996.

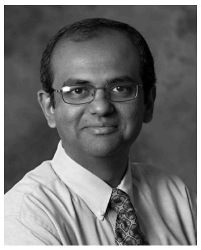
Andrea Gasparri received the *cum laude* degree in computer science and the Ph.D. degree in computer science and automation, both from the University of Roma Tre, Rome, Italy, in 2004 and 2008, respectively.

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

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

Ryan K. Williams received the B.S. degree in computer engineering from Virginia Polytechnic Institute and State University in 2005 and the Ph.D. degree in electrical engineering from the University of Southern California in 2014. He is currently a research affiliate at the Robotic Embedded Systems Laboratory. His research interests include control, cooperation, and intelligence in distributed multi-node systems, topological methods in cooperative phenomena, and distributed algorithms for optimization, estimation, inference, and learning. Ryan K. Williams is a Viterbi Fellowship recipient, has been featured by various news outlets, including the L.A. Times, and has a patent pending for his work on high-speed AUVs.

Attilio Priolo received the Master degree in Computer Science and Automation Engineering in 2009 and the Ph.D degree on Computer Science and Automation Engineering in 2013, both at Roma Tre University, in Rome, Italy. Currently, he is employed as Unmanned Vehicle Engineer in the Research and Innovation Department at Info Solution S.p.A.



Gaurav S. Sukhatme is a Professor of Computer Science (joint appointment in Electrical Engineering) at the University of Southern California (USC). He received his undergraduate education at IIT Bombay in Computer Science and Engineering, and M.S. and Ph.D. degrees in Computer Science from USC. He is the co-director of the USC Robotics Research Laboratory and the director of the USC Robotic Embedded Systems Laboratory which he founded in 2000. His research interests are in robot networks with applications to environmental monitoring. He has published extensively in these and related areas. Sukhatme has served as PI on numerous NSF, DARPA and NASA grants. He is a Co-PI on the Center for Embedded Networked Sensing (CENS), an NSF Science and Technology Center. He is a fellow of the IEEE and a recipient of the NSF CAREER award and the Okawa foundation research award. He is one of the founders of the Robotics: Science and Systems conference. He was program chair of the 2008 IEEE International Conference on Robotics and Automation and is program chair of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. He is the Editor-in-Chief of Autonomous Robots and has served as Associate Editor of the IEEE Transactions on Robotics and Automation, the IEEE Transactions on Mobile Computing, and on the editorial board of IEEE Pervasive Computing.