2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Decentralized Algorithms for Optimally Rigid Network Constructions

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

Abstract— **In this paper, we address the construction of optimally rigid networks that minimize an edge-weighted objective function over a planar graph. We propose two auction-based algorithms to solve this problem in a fully *decentralized* way. The first approach finds an optimal solution at the cost of high communication complexity; the second approach provides a sub-optimal solution while reducing the computational burden according to a sliding mode parameter ζ, yielding a tradeoff between complexity and optimality. A theoretical characterization of the optimality of the first algorithm is provided, and a closed form for the maximum gap between the optimal solution and the sub-optimal solution is also given. Simulation results are presented to corroborate the theoretical findings.**

I. Introduction

In recent years, multi-robot systems have gained interest in the robotics community due to their advantages in adaptability, efficiency and scalability with respect to single-robot systems. Research interests in the context of multi-robot systems range from mutual localization [1], to the development of decentralized coordination algorithms [2], and the design of distributed interaction control frameworks [3]. Remarkable capabilities have been demonstrated in several application contexts, including environmental exploration [4], [5], search and rescue operations [6], agricultural foraging [7] or sweeping operations [8].

Broadly speaking, rigidity represents an important requirement when the task of a multi-robot system demands tight collaboration among teammates. For example, its relevance is clear in the context of controlling formations of mobile vehicles when only relative sensing information is available [9]–[11]. Rigidity becomes a necessary (and under certain conditions sufficient) condition for localization tasks with distance or bearing-only measurements [12], [13], and in split/rejoin maneuvers [11]. The literature regarding the study of rigidity is rich and involves various disciplines of mathematics and engineering [14]–[16]. Combinatorial operations to preserve rigidity are defined in [15], while an extension of these ideas to the context of multi-robot formation control can be found in [9], [11]. In [17], the authors propose an interesting rigidity maintenance gradient controller for a multi-robot team, the application of which is mainly limited by the requirement of continuous communication and computational resources and the potential for multiplicity in the rigidity eigenvalue. In recent work [18], we propose a distributed rigidity controller that preserves the combinatorial rigidity of a dynamic network topology in the plane, requiring communication only during proposed transitions in the network topology.

In this work, a metric measuring the communication performance among the robots is associated with inter-robot connections, thus allowing the formulation of an optimization problem within the rigidity framework. In particular, this work aims at finding in a *decentralized* way the optimal rigid subgraph embedded in the graph encoding agent interactions and link metrics. Despite its great theoretical and application oriented appeal, this problem has been largely overlooked in the recent literature. In [19], 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 [20] an algorithm is proposed for generating optimally rigid graphs based on the Henneberg construction [15], however the proposed procedure is centralized, generates only locally optimal solutions, and requires neighborhood assumptions to ensure convergence.

The major novelty of this work is represented by two decentralized auction-based algorithms to find the optimal rigid subgraph embedded in a rigid graph. By electing leaders to control rigidity evaluation, the first approach finds greedily an optimal solution at the cost of high communication complexity. The second approach provides a sub-optimal solution while reducing the computational burden according to a sliding parameter ζ, controlling the balance of optimality and complexity. 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 ζ is also provided.

II. Graph Rigidity

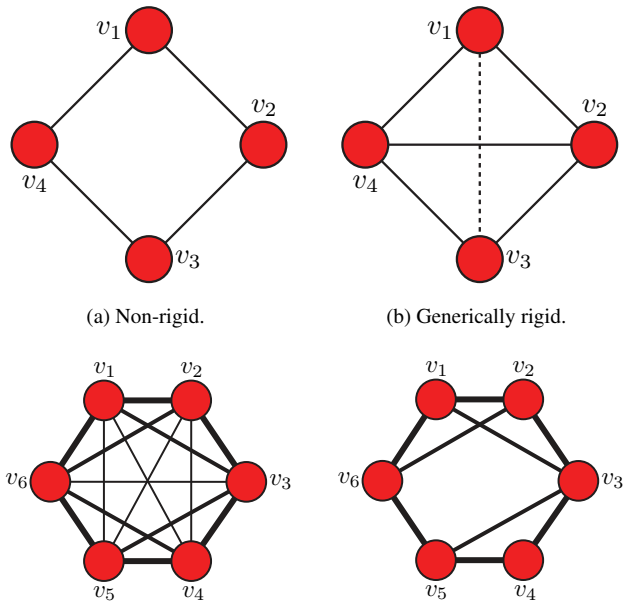
The primary concern of this work is the *rigidity* property of the underlying graph $\mathcal{G}$ describing the network topology[1], specifically as rigid graphs imply guarantees for example in both localizability and formation stability of multi-robot systems [11]. Intuitively, the concept of rigidity can be thought of in a physical way, that is if the graph were a bar and joint framework, it would be mechanically rigid against external and internal forces. That is, the edge lengths (inter-agent distance)

A. Gasparri and A. Priolo are with the Department of Engineering, Roma Tre University, 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 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]We provide a brief overview of rigidity theory here. We direct the reader to [14], [15], [21] for an in depth review of the subject.



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

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

Fig. 1. Graphs demonstrating 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 with redundant edge $(1,3)$. 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).

over $\mathcal{G}$ are preserved in time during infinitesimal motions of the agents, or no edge is compressed or stretched over time. Although there exists notions of graph rigidity that account for the specific positions of the agents in the workspace (*infinitesimal rigidity*), 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 $\mathbb{R}^2$) [22]. 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 [14], and is summarized as follows (also called *generic rigidity*)[2]:

Theorem 1 (Graph rigidity, [14])**.** *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with realizations in $\mathbb{R}^2$ having $n \geq 2$ nodes is rigid if and only if there exists a subset $\bar{\mathcal{E}} \subseteq \mathcal{E}$ consisting of $|\bar{\mathcal{E}}| = 2n - 3$ edges satisfying the property that for any non-empty subset $\hat{\mathcal{E}} \subseteq \bar{\mathcal{E}}$, we have $|\hat{\mathcal{E}}| \leq 2k - 3$, where k is the number of nodes in $\mathcal{V}$ that are endpoints of $(i,j) \in \hat{\mathcal{E}}$.*

We refer to the above as the *Laman conditions*, and denote by $\mathcal{G}_{\mathbb{R}}$ the set of all rigid graphs in $\mathbb{R}^2$. We call the graph $\bar{\mathcal{S}} = (\mathcal{V}, \bar{\mathcal{E}})$ 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. 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:

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

Definition 1 (Edge independence, [16])**.** 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 $\mathcal{E}^*$, while the graph over $\mathcal{E}^*$ is denoted by $\mathcal{G}^*$.

The above conditions imply that a graph is rigid in $\mathbb{R}^2$ if and only if it possesses $|\mathcal{E}^*| = 2n - 3$ independent edges, where edges that do not meet the conditions of Definition 1 are called *redundant* (see Figure 1 for a depiction of graph rigidity). Thus, in determining the rigidity of $\mathcal{G}$, we must verify the Laman conditions to discover a suitable set of independent edges $\mathcal{E}^*$. A natural simplification to the process of determining edge independence is found in the *pebble game* of [16], a decentralization of which we provided in previous work [23]. Thus based on our results in [23], we assume that the network can apply a decentralized (and asynchronous) pebble game to determine with $O(n^2)$ complexity the independence of any edge e_{ij} in $\mathcal{G}$. The algorithm of [23] utilizes leaders elected in the network to decentralize the evaluation of edge independence; a feature we exploit here to control the *order* and *cardinality* of considered edges, and ultimately optimality vs. complexity. Under the assumption of weighted graph edges, we therefore seek *optimal* Laman subgraphs to construct cost effective rigid networks in a fully decentralized way.

III. Optimally Rigid Graph Construction

Consider n mobile robots 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 $\mathcal{V} = \{v_i\}$ is the set of nodes/robots indexed by $\mathcal{I} = \{i\} = \{1, \ldots, n\}$ and $\mathcal{E} = \{e_{ij}\} \in \mathcal{V} \times \mathcal{V}$ is the set of edges. Denote with $\mathcal{N}_i = \{v_j \in \mathcal{V} : e_{ij} \in \mathcal{E}\}$ the set of neighbors of the i^{th} robot and with $\mathcal{E}_i = \{e_{ij} \in \mathcal{E} : v_j \in \mathcal{N}_i\}$ the set of incident edges to robot i.

Assume a weight $w(e_{ij}) = \xi(v_i) + \xi(v_j) + d(v_i, v_j)$ to be associated to each edge e_{ij}, where $\xi(v_i) : \mathcal{V} \to \mathbb{R}^+$ is a cost function associated to each endpoint of the edge and $d(v_i, v_j) : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$ 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 $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E})$, $w_{ij} = w(e_{ij})$, $\xi_i = \xi(v_i)$ and $d_{ij} = d(v_i, v_j)$ will be used for the sake of readability.

Let us now provide some preliminary assumptions and definitions which are instrumental for the considered optimization problem. Consider a generic objective function $\rho(\mathcal{G}) : \mathcal{G} \to \mathbb{R}^+$ of the form:

$$\rho(\mathcal{G}) = \sum_{e_{ij} \in \mathcal{E}} w_{ij} \tag{1}$$

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

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

$$\nexists\, \hat{\mathcal{G}}(\mathcal{V}, \hat{\mathcal{E}}) \subseteq \mathcal{G}(\mathcal{V}, \mathcal{E}) : \rho(\hat{\mathcal{G}}) < \rho(\bar{\mathcal{G}}),$$

with $\bar{\mathcal{G}}$, $\hat{\mathcal{G}}$ rigid.

The identification of an *optimal* subgraph $\bar{\mathcal{G}}$ can be expressed in terms of a minimization problem as follows:

$$\bar{\mathcal{G}} = \underset{\mathcal{G}' \subseteq \mathcal{G}}{\arg\min}\ \rho(\mathcal{G}'). \tag{2}$$

According to Definition 2 a naive approach to find an optimal rigid subgraph would be to inspect all the possible subgraphs of $\mathcal{G}$, thus leading to an impractical computational complexity. However, it has been proven in [20] that the optimal solution to (2) is a Laman subgraph $\bar{\mathcal{S}}(\mathcal{V}, \bar{\mathcal{E}})$. Therefore in this work we propose a decentralized approach to build an optimal Laman subgraph $\bar{\mathcal{S}}(\mathcal{V}, \bar{\mathcal{E}}) \subseteq \mathcal{G}(\mathcal{V}, \mathcal{E})$ based on pairwise collaboration among the robots. More specifically, we propose two alternative algorithms: the first approach finds an optimal solution to (2) at the cost of communication complexity; the second approach provides a sub-optimal solution while reducing the computational burden according to a design parameter ζ.

A. Optimal Greedy Algorithm

The Optimal Greedy algorithm (OG) proposed in this work exploits the distributed version of the pebble game originally proposed by the authors in [23]. 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 reader is referred to [23] for a comprehensive overview of such a distributed pebble game algorithm; here we denote this algorithm by PEBBLEGAME in our pseudocode implementation.

Algorithm 1 The Optimal Greedy Algorithm.

1: **procedure** OPTGREEDY($\mathcal{G} = (\mathcal{V}, \mathcal{E})$)
  ▷ Init the data structures
2: $\{\mathcal{E}_i^*(0), \mathcal{E}_i(0)\} \leftarrow$INITSTRUCTS($i$, $\mathcal{G}$), $\forall i \in \mathcal{I}$
3: $k \leftarrow 1$
4: **repeat**
5: ▷ Compute the bids for each i:
6: $\{b_i(k), e_{ij}^{\min}(k)\} \leftarrow$COMPUTEBID($i$,$\mathcal{E}_i(k-1)$)
7: ▷ Auction to select the leader with minimum bid:
8: $l \leftarrow \underset{i}{\arg\min}\, b_i(k)$
9: ▷ Leader checks $e_{lj}^{\min}(k)$ independence:
10: $\mathcal{E}_l^*(k) \leftarrow$LEADERRUN($l, \mathcal{E}_l^*(k-1), e_{lj}^{\min}(k)$)
11: ▷ Remove already considered edges:
12: $\mathcal{E}_l(k) = \mathcal{E}_l(k-1) \setminus e_{lj}^{\min}(k)$
13: $\mathcal{E}_j(k) = \mathcal{E}_j(k-1) \setminus e_{jl}^{\min}(k)$
14: $k \leftarrow k+1$
15: **until** $|\mathcal{E}^*(k)| = |\cup_i \mathcal{E}_i^*(k)| = 2n-3$
16: **end procedure**

The pseudo-code of the proposed OG algorithm is depicted in Algorithm 1. Generally speaking, the algorithm works as follows. In step 2, each agent initializes according to Procedure 2. 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[3]. In particular, the following steps are executed. First, the bidding process for the leader election detailed in Procedure 3 is run (step 6). Then, the chosen leader selects (and removes) from the local set of candidates the edge with minimum weight and checks its independence by running the decentralized pebble game according to Procedure 4. 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 procedures mentioned above.

Procedure 2 performs the initialization of the required data structure. In particular, in step 2 the portion of the independent set $\mathcal{E}_i^*$ owned by the i^{th} robot is initialized along with the set of edges $\mathcal{E}_i$ originating/ending with the i^{th} robot and sorted according to the chosen metric.

Procedure 2 Data structures initialization.

1: **procedure** INITSTRUCTS
  Input: i, $\mathcal{G}$
  Output: $\mathcal{E}_i^*(0)$, $\mathcal{E}_i(0)$
2: $\mathcal{E}_i^*(0) = \{\emptyset\}$, $\forall i \in \mathcal{I}$ ▷ Init independent set portion
3: ▷ Sort the incident edges set:
4: $\mathcal{E}_i(0) \leftarrow \{e_{ij} : v_j \in \mathcal{N}_i\}$
5: Sort $(\mathcal{E}_i(0))$, $\forall i \in \mathcal{I}$
6: **end procedure**

Procedure 3 computes the bids for each robot 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, an arbitrary large value is assigned to the bid (step 6). As it will be explained in the following, a leader is elected by the algorithm if it submits the smallest bid. Therefore, a large value in the bid prevents robot i from becoming a leader. If $|\mathcal{E}_i(k)| \neq 0$, then the minimum weight among the candidates edges is computed and used as the bid (step 3). Using the smallest weight as bid guarantees that the solution is incrementally built considering the graph edges in an increasing order with respect to their weights. The edge corresponding to the minimum weight in $\mathcal{E}_i(k)$ is stored by each robot i in $e_{ij}^{\min}(k)$ because if i is elected as the next leader, then $e_{ij}^{\min}(k)$ is the edge whose independence has to be checked (step 4).

Procedure 4 describes the leader execution. In step 2, the distributed pebble game originally introduced by the authors in [23] is used for checking the independence of $e_{lj}^{\min}(k)$. If the check succeeds, then the edge is added to the i^{th} local set of independent edges in step 4.

We now provide a brief reasoning as to the optimality of Algorithm 1:

Theorem 2. *The solution $\bar{\mathcal{S}}(\mathcal{V}, \bar{\mathcal{E}}) \subseteq \mathcal{G}(\mathcal{V}, \mathcal{E})$ 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

[3]Note that to compute $|\mathcal{E}^*(k)|$, after each leadership auction, the robots update a global counter embedded in the auction packages, decentralizing the process.

Procedure 3 Bid and minimum weight edge computation.

1: **procedure** COMPUTEBID
Input: $i, \mathcal{E}_i(k-1)$
Output: $b_i(k)$, $e_{ij}^{min}(k)$
2: **if** $\mathcal{E}_i(k-1) \neq \{\emptyset\}$ **then**
3: $b_i(k) = \min_{e_{ij} \in \mathcal{E}_i(k-1)} w_{ij}$
4: $e_{ij}^{\min}(k) = \arg\min_{e_{ij} \in \mathcal{E}_i(k-1)} w_{ij}$
5: **else**
6: $b_i(k) = \infty$
7: **end if**
8: **end procedure**

Procedure 4 The leader execution.

1: **procedure** LEADERRUN
Input: l, $\mathcal{E}_l^*(k-1)$, $e_{lj}^{\min}(k)$
Output: $\mathcal{E}_l^*(k)$
2: $res \leftarrow$PEBBLEGAME($e_{lj}^{\min}(k)$)
3: **if** res is **true then**
4: $\mathcal{E}_l^*(k) = \mathcal{E}_l^*(k-1) \cup e_{lj}^{\min}(k)$
5: **end if**
6: **end procedure**

edge $e_{lj}^{\min}(1)$ with minimum weight is selected and added to the independent set $\mathcal{E}_l^*(1)$ with l the index of the current leader. Therefore the value of the objective function computed using $\mathcal{E}^*(1)$ is minimum. Let us now consider the generic k^{th} iteration, with $k > 1$. Two cases are possible:

C1 $e_{lj}^{\min}(k)$ is independent,
C2 $e_{lj}^{\min}(k)$ is redundant.

Let us analyze **C1**. First of all, note that $|\mathcal{E}^*(k-1)| < 2n-3$, otherwise the edge would be redundant. Then, the insertion of $e_{lj}^{\min}(k)$ guarantees that the objective function computed using the edges in $\mathcal{E}^*(k) = \mathcal{E}^*(k-1) \cup e_{lj}^{\min}(k)$ is minimum because of the $e_{lj}^{\min}(k)$ definition.

Consider now **C2**. The redundancy of $e_{lj}^{\min}(k)$ together with the Laman conditions implies that a *single* edge e_{ij} could be removed from the current optimal subgraph and replaced with $e_{lj}^{\min}(k)$. However, the replacement of a generic edge e_{ij} in the optimal subgraph would increase the value of the solution precisely due to the assumed optimality at the $k-1$ step. Therefore, at step k the edge $e_{lj}^{\min}(k)$ should not be inserted into $\mathcal{E}_i^*(k)$. Iterating this reasoning until $|\mathcal{E}^*(k)| = 2n-3$, it turns out that the optimality of the solution is preserved at each step. □

As previously stated, the optimality of the solution comes at a price. In the leader election process, $\mathrm{O}(n)$ messages are expended for each auction. In the worst case, all the edges of the network are analyzed, incurring $\mathrm{O}(n^2)$ in executions. Therefore, the overall messaging complexity becomes $\mathrm{O}(n^3)$.

B. Sub-Optimal Algorithm

In this section we introduce a modified algorithm, denoted as the SO algorithm, by which the computational complexity of the OG algorithm can be mitigated. To this end, we modify the OG algorithm by introducing a tuning parameter ζ which represents the number of edges to be checked for independence at each iteration. It will be shown that in this way, the communication complexity can be reduced up to $O(n^2)$. In addition, we provide a closed form of the maximum gap between the optimal solution and the sub-optimal solution provided by this algorithm expressed in terms of the tuning parameter ζ.

Let us now detail the major differences concerning the modified SO algorithm and the OG algorithm given in Algorithm 1. As in the previous section, let us assume that each robot i is able to sort its incident edges $\mathcal{E}_i$ according to the associated weights w_{ij}. Let us denote with w_{ij}^q the q^{th} weight among the weights associated to robot i edges after the sorting, such that $w_{ij}^1 \leq w_{ij}^q \leq w_{ij}^{|\mathcal{E}_i|}$, where $q \in [1, \ldots, |\mathcal{E}_i|]$ and with $e_{ij}^q \in \mathcal{E}_i$ the edge corresponding to w_{ij}^q. Before the algorithm execution, a parameter $\zeta \in [1, \ldots, n-1]$ is chosen. It represents the number of edges that an elected leader is allowed to check for independence. This parameter is used by the modified COMPUTEBID and LEADERRUN procedures depicted in Procedures 5 and 6, respectively.

In Procedure 5, the bid for the leader auction process is computed by each robot summing up the first ζ edges weights and the edges themselves are stored in $\mathcal{E}_i^\zeta$ (steps 3-4). This represents a conservative approach to minimize the increase of the solution at step $k+1$, even though this might lead to a sub-optimal solution.

In Procedure 6, the edges in the set $\mathcal{E}_l^\zeta$ are checked for independence (step 3). Each edge is inserted into the the i^{th} portion of the independent set after a successful independence check, it is removed otherwise.

In the following proposition, the role of $\delta_i^q = w_{ij}^q - w^{\min}$, with $w^{\min} = \min_{e_{ij} \in \mathcal{E}} w_{ij}$, as an upper bound for a single edge insertion error is given:

Proposition 1. *Assume that the i^{th} robot inserts an edge with weight w_{ij}^q into its portion of the independent set. Then, the increase in the objective function with respect to the optimal solution is upper bounded by $\delta_i^q = w_{ij}^q - w^{\min}$.*

Proof. Adding an edge to the independent set could prevent the insertion of an edge of the optimal solution, denoted with $\bar{e}_{ij}$, whose weight $\bar{w}_{ij} \geq w^{\min}$, where $\bar{w}_{ij}$ is the weight associated to $\bar{e}_{ij}$. Therefore, $\delta_i^q \geq w_{ij}^q - \bar{w}_{ij}$. □

Denote with $\Delta_l^\zeta(k) = b_l(k) - \zeta \min_{e_{ij} \in \{\cup_i \mathcal{E}_i(k)\}} w_{ij}$ the difference between the leader bid and ζ times the minimum weight among all the not considered edges at step k. In the following proposition, an upper bound on the error occurring when ζ edges are inserted by the leader is given.

Proposition 2. *The error occurring by inserting ζ edges into the leader portion of the independent set at step k is upper bounded by $\Delta_l^\zeta(k)$.*

Procedure 5 Modified bid and minimum weight edge computation.

procedure MODCOMPUTEBID
Input: $i, \mathcal{E}_i(k-1), \zeta$
Output: $b_i(k)$, $\mathcal{E}_i^{\zeta}(k)$
  if $\mathcal{E}_i(k-1) \neq \{\emptyset\}$ **then**
    $b_i(k) = \sum_{q=1}^{\zeta} w_{ij}^q$
    $\mathcal{E}_i^{\zeta}(k) = \{e_{ij}^q \in \mathcal{E}_i(k) : q \leq \zeta\}$
  else
    $b_i(k) = \infty$
  end if
end procedure

Procedure 6 The modified leader execution.

procedure MODLEADERRUN
Input: l, $\mathcal{E}_l^*(k-1)$, $\mathcal{E}_l^{\zeta}(k)$
Output: $\mathcal{E}_l^*(k)$
  for all $e_{ij} \in \mathcal{E}_l^{\zeta}$ **do**
    $res \leftarrow$PEBBLEGAME(e_{ij})
    if res is **true then**
      $\mathcal{E}_l^*(k) = \mathcal{E}_l^*(k-1) \cup e_{ij}$
    end if
  end for
end procedure

Proof. The proof can be derived by adapting Proposition 1 when ζ edges are inserted. □

To conclude, the messaging complexity exhibited by the algorithm can be shown to scale like $O(n^3/\zeta)$, our expected sliding mode behavior. Briefly, this is a consequence of the worst case of $(n-1)/\zeta$ elections for $O(n)$ leaders, where $O(n)$ messages are sent during each leader election step.

IV. SIMULATIONS

In this section, simulations results involving $n = 12$ robots are proposed to corroborate the theoretical findings. A fully connected graph $\mathcal{G}$ is considered for the simulations. This represents a convenient choice as it describes the worst case scenario for the evaluation of the communication burden (it has the largest number of edges) while at the same time it ensures the rigidity condition of the graph. For each simulation, a random symmetric weight matrix composed of integers is generated and associated to the graph edges.

In Figure 2, a comparison between the OG algorithm and SO algorithm in terms of the objective function (2a) and the amount of exchanged messages (2b) is depicted where the parameter ζ is ranged within the interval $[1, \ldots, n-1]$. For each value of the parameter ζ, both algorithms are executed in a Monte Carlo fashion. Furthermore, the maximum, minimum and mean cost of the objective function obtained for each value of the parameter ζ are shown for both algorithms. It is worth noticing that if $\zeta = 1$, then the algorithms performances are identical. As expected, it can be seen that the larger the value of ζ, the greater the gap between the optimal and suboptimal solutions. Similarly, the larger the value of ζ, the smaller the amount of exchanged messages. Indeed, ζ represents a trade-off between optimality and complexity where the optimal solution is achieved with $\zeta = 1$ and the best communication performance is achieved with $\zeta = n-1$. The solid green line in Figure 2a describes the gap between the optimal solution of the OG algorithm and the suboptimal solution of the SO algorithm. Note that, this line is computed by summing up to the value of the optimal solution the upper bound computed for each value of the parameter ζ. Interestingly, this curve assumes a sigmoid-like shape, that is as ζ increases a saturation is experienced for the cost of the objective function; a direct consequence of the saturation of the Laman conditions at $2n-3$ independent edges.

Figure 3 depicts a typical outcome of the two algorithms with $\zeta = 11$. In particular, Figure 3a represents the outcome of the SO algorithm. It can be noticed that a large value of ζ induces a large cardinality for both $\bar{\mathcal{E}}_8, \bar{\mathcal{E}}_{10}$, i.e., $|\bar{\mathcal{E}}_8| = 11$, $|\bar{\mathcal{E}}_{10}| = 11$. Figure 3b represents the outcome of the OG algorithm. It can be noticed that in this case the cardinality of the incidence edges is more balanced compared to the solution previously described, which reflects the randomized nature of our Monte Carlo initial conditions.

V. CONCLUSIONS AND FUTURE WORK

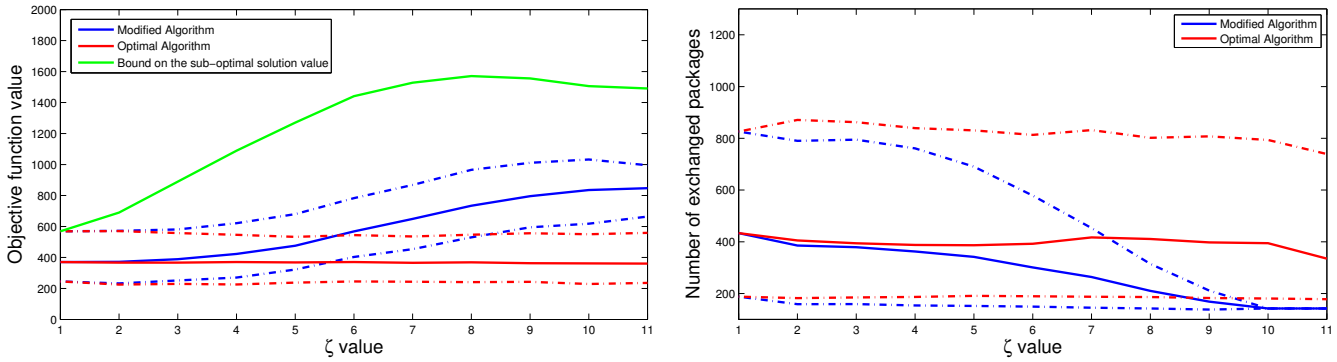
In this work, the problem of building an optimal rigid subgraph exploiting the distributed version of the pebble game introduced by the authors in [23] has been considered. In particular, two decentralized approaches have been presented. The first approach iteratively builds the optimal minimally rigid graph by assigning the minimum edge weights to the bids used by the robots in a leadership auction and by allowing each leader to check the independence of at most a single edge. To mitigate the messaging complexity required by this algorithm, a modified version has been introduced. In particular, by choosing a parameter ζ governing the number of weights incorporated in the bids and the number of edges considered for the independence check, the messaging complexity has been scaled by a factor $1/\zeta$. This comes at a price, i.e. the modified approach generates a sub-optimal minimally rigid graph. However, a closed form to upper-bound the difference between the two algorithms solutions has been presented. Future work will be mainly focused on investigating more complex bidding strategies for applications in wireless sensor networks.

REFERENCES

[1] 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.

[2] 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.

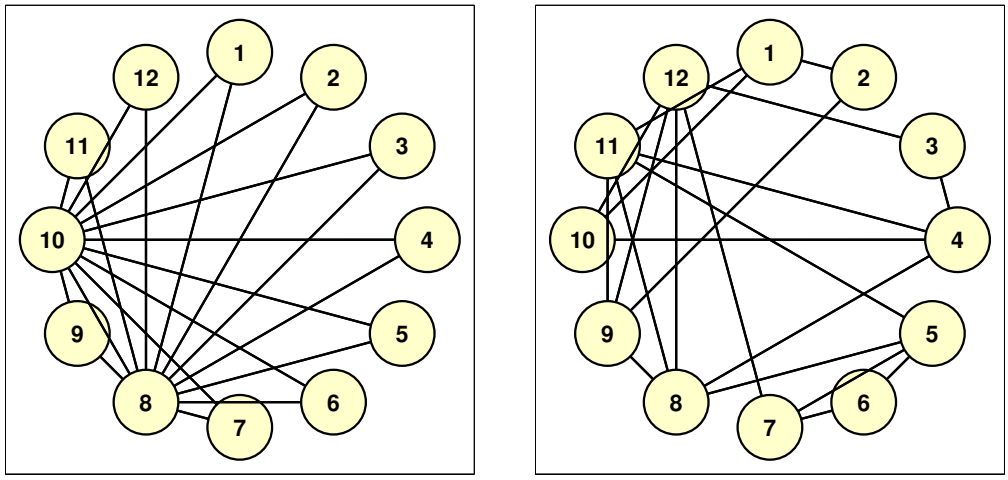
[3] R. K. Williams and G. S. Sukhatme, "Constrained Interaction and Coordination in Proximity-Limited Multi-Agent Systems," *IEEE Transactions on Robotics*, 2013.



(a) Objective functions comparison.

(b) Exchanged messages comparison.

Fig. 2. Illustration of the comparison between the optimal algorithm and the sub-optimal one in terms of objective function and amount of exchanged messages.



(a) Sub Optimal Solution.

(b) Optimal Solution.

Fig. 3. Illustration of the solutions found by a single run of the SO and the OG approach with $\zeta = 11$.

[4] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot tree and graph exploration," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 707 –717, aug. 2011.

[5] 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.

[6] 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),*, may 2010.

[7] D. Shell and M. Mataric, "On foraging strategies for large-scale multi-robot systems," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, oct. 2006, pp. 2717 –2723.

[8] S. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics,*, vol. 28, no. 2, pp. 410 –426, april 2012.

[9] 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.

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

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

[12] 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.

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

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

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

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

[17] 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.

[18] 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.

[19] 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.

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

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

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

[23] 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.