Proceedings of the 34th Chinese Control Conference
July 28-30, 2015, Hangzhou, China

Distributed optimization on proximity network rigidity via robotic movements

Zhiyong Sun[1,2], Changbin Yu[1,2] and Brian D. O. Anderson[1,2]

1. Shandong Computer Science Center (National Supercomputer Center in Jinan), Shandong Provincial Key Laboratory of Computer Networks, Jinan, Shandong, China

2. Research School of Engineering, The Australian National University, Canberra ACT 0200, Australia
E-mail: zhiyong.sun@anu.edu.au, brad.yu@anu.edu.au, brian.anderson@anu.edu.au

Abstract: This paper considers a rigidity optimization problem for mobile robotic teams modeled in a proximity network with state-dependent network topology. The aim is to move all robots' positions to reach a configuration such that the worst-case rigidity metric can be maximized. Key properties of a Gramian matrix involving a weighted rigidity matrix are discussed for solving this optimization problem. We design a decentralized algorithm to update all robots' positions to maximize the eigenvalue function, which requires local information from each robot itself and its neighbors. Furthermore, a distributed eigenvector estimation scheme based on inverse shifted power iteration method and averaging consensus algorithm is devised to allow each robot to estimate the global eigenvector information. Simulation results are also provided to demonstrate the effectiveness of the estimation and optimization scheme.

Key Words: Distributed optimization, graph rigidity, robotic network

1 Introduction

Networked mobile robots have the desirable capacity of performing spatially distributed tasks like large area surveillance, underwater exploration, target detection, etc., while these tasks often cannot be achieved by a single robot. To achieve the cohesiveness of such robotic systems, certain fundamental requirements relating to network quality need to be guaranteed. One aspect is the connectivity property of the dynamic network, in which the algebraic connectivity [1] is frequently used to describe the connectivity quality of the robotic network. There have been great efforts reported in the literature on how to control and/or maximize the algebraic connectivity for a multi-robot network via robotic mobility by taking into consideration of different constraints and control requirements; see e.g. [2–6].

In robotic network study, another closely related concept which has equal importance to network connectivity is formation rigidity [7]. It has been shown that rigid graph theory plays a key role in analyzing formation performance [7] and network localization [8]. Formation rigidity can be particularly important in the formation shape control for multi-robot systems, which enables the distance-based formation design without a centralized controller or a global coordinate reference frame being required for all agents [9]. The favorable property of a rigid framework, therefore, motivates people to consider the problem of controlling, preserving and optimizing robotic network rigidity when coordinating robot teams with planned motions. Some recent works in this direction include [10] which introduced the concept of quantitative rigidity and [11] on rigidity maintenance control, and [12] on rigid network design via submodular function optimization. The mentioned papers all considered an algebraic basis on analyzing the rigidity property by using spectrum analysis of the rigidity matrix. Another direction has been reported in [13], which focused on a combinatorial aspect for rigidity evaluation and optimization.

In this paper the graph rigidity is described and evaluated via a weighted rigidity matrix, similar to the concept of a weighted Laplacian [14]. The aim is to design a *distributed* algorithm to update all the agents' positions to reach a maximum level of proximity network rigidity. Proximity networks capture the notion of physical communication constraints between agents, in that as agent separations increase, interagent communication becomes problematic or impossible. The problem to be discussed in this paper can be seen as a natural extension of the algebraic connectivity optimization problem, while the algorithm design here is quite challenging as the rigidity matrix involves both *network topology* and *agents' position information*. The optimization framework considered here adopts a similar concept to that of using a *smallest* eigenvalue of a symmetric positive definite matrix to quantify the worst-case rigidity level, which serves as a natural index to indicate how far away a framework is from being infinitesimally flexible. We also show several novel and important results on the Gramian matrix involving the weighted rigidity matrix, which prove to be useful for solving the optimization problem. Furthermore, in order to implement the algorithm in a distributed way, we then aim to devise an eigenvector estimation scheme to estimate the global information from a particular eigenvector of the rigidity Gramian matrix [1] (a concept similar to the Fiedler vector for graph Laplacian [1]). The results in this paper will provide further insights on how to achieve the cohesiveness of multi-robotic systems with desired fundamental requirements relating to network rigidity.

This paper is organized as follows. Section 2 reviews basic concepts on graph theory and rigidity theory. In Section

This work was supported by NICTA Ltd, Overseas Expert Program of Shandong Province, a grant from Shandong Academy of Science Development Fund for Science and Technology, the Pilot Project for Science and Technology in Shandong Academy of Science, and the National Natural Science Foundation of China (61375072). Z. Sun was supported by the Prime Ministers Australia Asia Incoming Endeavour Postgraduate Award. C. Yu was supported by a Queen Elizabeth II Fellowship under DP110100538. B. D. O. Anderson was also supported by the ARC under grant DP110100538 and DP130103610.

[1] Some results are posted online in an unpublished preprint [15].

3, we propose the problem formulation, and show several results involving a weighted rigidity matrix and its Gramian. Algorithm design study is discussed in Section 4, where we show a decentralized implementation of the algorithm is possible. Section 5 focuses on distributed eigenvector estimation to eliminate the requirement of global eigenvector information in the algorithm. In Section 6 simulation results are provided, followed by some concluding remarks which end this paper.

2 Preliminaries

2.1 Notations

The notations used in this paper are fairly standard. $\mathbb{R}^n$ denotes the n-dimensional Euclidean space. $\mathbb{R}^{m\times n}$ denotes the set of $m\times n$ real matrices. If M is a vector or matrix, its transpose is denoted by M^T. The rank, image and null space of matrix M are denoted by $rank(M)$, $Im(M)$ and $null(M)$, respectively. For a symmetric matrix M, its i-th smallest eigenvalue is denoted by $\lambda_i(M)$. The notation $\mathrm{diag}\{x\}$ denotes a (block) diagonal matrix with the (block) vector x on its diagonal. $span\{v_1, v_2, \cdots, v_k\}$ represents the subspace spanned by a set of vectors $v_1, v_2, \cdots, v_k$. I_n is the $n\times n$ identity matrix, and $\mathbf{1}_n$ denotes a n-tuple column vector of all ones. For two symmetric matrices M and N, we write $M \succ N$ (resp. $M \succeq N$) to indicate that $M-N$ is positive definite (resp. positive semidefinite). The symbol $\otimes$ denotes the Kronecker product.

2.2 Graph theory

Consider an undirected graph with m edges and n vertices, denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V} = \{1, 2, \cdots, n\}$ and edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. The vertex set represents the robots (and we may use the word *agent* interchangeably in the context) and the edge set represents the communication links between different robots. Two matrices, the incidence matrix $H = \{h_{ij}\} \in \mathbb{R}^{m\times n}$ and adjacency matrix $A(\mathcal{G}) \in \mathbb{R}^{n\times n}$ whose definitions can be found in [14], are used frequently to describe the relationship of nodes and edges in graphs. Another important matrix representation of a graph $\mathcal{G}$ is the Laplacian matrix $L(\mathcal{G})$, which is defined as $L(\mathcal{G}) = H^T H = \mathrm{diag}\{A\mathbf{1}\} - A$. By defining an $m\times m$ diagonal matrix W whose diagonal entries are the weights for each edge, one can construct the weighted Laplacian matrix $L(\mathcal{G}_w) = H^T W H$. In modeling *proximity networks* with dynamic state-dependent topology, the edge weights w_k are usually some functions of the distances between pairs of neighboring agents, which represent the communication range constraints or communication strength between robots [14].

2.3 Rigidity theory and rigidity matrix

We embed the graph $\mathcal{G}$ into the 2-D space [2]. Let $p_i = [p_{ix}, p_{iy}]^T \in \mathbb{R}^2, i \in \{1, 2, \cdots, n\}$ denote the position of node (i.e. robot) i. The stacked vector $p = [p_1^T, p_2^T, \cdots, p_n^T]^T$ represents the position configuration for all the n robots. By introducing the matrix $\bar{H} := H \otimes I_2 \in \mathbb{R}^{2m\times 2n}$, one can construct the edge space as an image of $\bar{H}$ from the position vector p:

$$z = \bar{H} p \tag{1}$$

with $z_i = [z_{i,x}, z_{i,y}]^T \in \mathbb{R}^2$ being the relative position vector for the vertex pair defined by the i-th edge. In the following, two notations, z_k and $z_{k_{ij}}$ will be used interchangeably to denote the k-th edge which links agent i and agent j.

The rigidity function $r_{\mathcal{G}}(p) : \mathbb{R}^{2n} \to \mathbb{R}^m$ associated with the framework $(\mathcal{G}, p)$ is defined as:

$$r_{\mathcal{G}}(p) = \frac{1}{2}\left[\cdots, \|p_i - p_j\|^2, \cdots\right]^T \tag{2}$$

where the norm is the standard Euclidean norm, and the k-th component in $r_{\mathcal{G}}(p)$, $\|p_i - p_j\|^2$, corresponds to the square length of edge z_k. One useful tool to characterize the rigidity property of a framework is the rigidity matrix [7], which is defined as $R(p) = \frac{\partial r_{\mathcal{G}}(p)}{\partial p}$. There is a simple expression for the rigidity matrix which involves both the network topology and position configuration, which will be shown below (also see e.g. [7, 9]). The definition of the rigidity function shows that it is a map from the relative position vectors z to the squared edge lengths. Thus we can redefine the rigidity function, $g_{\mathcal{G}}(z) : Im(\bar{H}) \to \mathbb{R}^m$ as $g_{\mathcal{G}}(z) = \frac{1}{2}\left[\|z_1\|^2, \|z_2\|^2, \|z_3\|^2, \cdots, \|z_m\|^2\right]^T$. From (1) and (2), one can obtain the following simple and useful form for the rigidity matrix

$$\begin{aligned} R(p) = \frac{\partial r_{\mathcal{G}}(p)}{\partial p} &= \frac{\partial g_{\mathcal{G}}(z)}{\partial z}\frac{\partial z}{\partial p} \\ &= \begin{pmatrix} z_1^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & z_m^T \end{pmatrix} \bar{H} \\ &= Z^T \bar{H} \end{aligned} \tag{3}$$

where Z is a block diagonal matrix $Z = \mathrm{diag}\{z_1, z_2, \cdots, z_m\}$.

The framework $(\mathcal{G}, p)$ is said to be infinitesimally rigid if the rank of the rigidity matrix R equals $2n-3$, which is the maximum rank. Also, if $(\mathcal{G}, p)$ is infinitesimally rigid, so is $(\mathcal{G}, p')$ for a generic (open and dense) set of p'. Generally speaking, infinitesimal rigidity implies rigidity, but the converse is not true. In the rest of this paper, we will use the rank condition of the rigidity matrix to determine whether a configuration is infinitesimally rigid.

According to the concept of rigidity, the set of all infinitesimal displacements caused by the rigid body motions forms the null space of the rigidity matrix R. In fact, a set of linearly independent vectors which span the null space of R can be calculated as (also see [16])

$$v_1 = \mathbf{1} \otimes [1, 0]^T = [1, 0, 1, 0, \cdots, 1, 0]^T \tag{4}$$

$$v_2 = \mathbf{1} \otimes [0, 1]^T = [0, 1, 0, 1, \cdots, 0, 1]^T \tag{5}$$

$$v_3 = [p_{1y}, -p_{1x}, p_{2y}, -p_{2x}, \cdots, p_{ny}, -p_{nx}]^T \tag{6}$$

In the following, we will denote $\mathcal{F} := span\{v_1, v_2, v_3\}$, i.e. $\mathcal{F}$ is a subspace in $\mathbb{R}^{2n}$ of dimension three, spanned by the three vectors.

Following the definition of the weighted Laplacian matrix $L(\mathcal{G}_w)$, we construct a new matrix E for the proximity network in a similar way as follows:

$$\boxed{L(\mathcal{G}_w) = H^T W H} \to \boxed{E(\mathcal{G}_w, p) = R^T W R = \bar{H}^T Z W Z^T \bar{H}} \tag{7}$$

[2] In this paper we mostly focus on the analysis in the 2-D space. The generalization to the 3-D space will be treated in the extended version of this paper.

where the weight matrix W represents the physical constraints in the network, and the weight function is usually chosen as some decreasing function of the inter-agent distances. For discussions on the edge weight function for modeling proximity network, see e.g. [2] and [14].

3 Rigid network optimization

3.1 Problem formulation

The problem considered in this paper is to drive all the agents to reach a rigid network with *maximum* rigidity. We will use the smallest positive eigenvalue $\lambda_4(E)$ to describe the measure of rigidity, a strategy similar to [10, 11]. In fact, by considering a *weighted* rigidity matrix $\tilde{R}$ in the form $\tilde{R} = W^{\frac{1}{2}}R$, the matrix E is rewritten as $E = \tilde{R}^T\tilde{R}$. Hence, $\sqrt{\lambda_4(E)}$ is a singular value of the modified rigidity matrix $\tilde{R}$ which takes into account the *weights* of each edge [3]. According to the definition of infinitesimal rigidity and the implication of the smallest positive singular value of a matrix, $\lambda_4(E)$ actually measures how far away a framework is from being non-rigid. Similar to the problem formulation on algebraic connectivity optimization (i.e. maximizing $\lambda_2(L)$) as considered in e.g. [2, 17], the network rigidity optimization can be formulated as

$$\mathscr{P}: \max_{p} \ \lambda_4(E(p)) \tag{8}$$

with the constraint that $\lambda_4(E(p)) > 0, \forall p \in \mathbb{R}^{2n}$. To remove this constraint, we assume that the initial network is infinitesimally rigid. Thus, the problem is to find a position configuration p for all the agents modeled in the proximity network, such that $\lambda_4(E)$ can be maximized in the workspace.

Remark 1. *There exist other metrics in the literature on characterizing network rigidity, most of which employ the spectrum information of the matrix R^TR. In a recent paper [12], Shames and Summers term the two matrix products, R^TR and RR^T, as edge rigidity Gramian and vertex rigidity Gramian, respectively, and consider some submodular functions (e.g. the trace of the Gramian) in the rigid network design. Note that these two Gramians have the same non-zero eigenvalues. Besides the smallest positive singular value considered here and also in [10, 11] (termed worst-case rigidity level), other metrics include the average rigidity level involving a trace function [10], or eigenvalue function using a reduced-order rigidity matrix [16]. For the metric using the smallest positive singular value, we note that [11] provides an excellent study on maintaining robotic formation rigidity via using continuous-time dynamical systems. Currently we are considering other metrics for describing the rigidity level for which one can establish distributed algorithms for the associated optimization problem.*

3.2 Structure and properties of E

By comparing with the definition and structure of $L(\mathscr{G}_w)$, one can see that the matrix E can be regarded as a position- and distance-weighted Laplacian matrix for the framework $(\mathscr{G}, p)$, where the weights are described by a *diagonal block matrix* ZWZ^T involving the position and inter-agent distance information. The block diagonal matrix ZWZ^T is expressed by

$$ZWZ^T = \text{diag}\{w_1z_1z_1^T, w_2z_2z_2^T, \cdots, w_mz_mz_m^T\} \tag{9}$$

where w_i is a scalar and $z_iz_i^T$ is a 2×2 block:

$$z_iz_i^T = \begin{pmatrix} z_{i,x}^2 & z_{i,x}z_{i,y} \\ z_{i,x}z_{i,y} & z_{i,y}^2 \end{pmatrix} \tag{10}$$

The structure of E also resembles that of the Laplacian matrix L, but E has twice the dimension compared to L. In fact, it is more convenient to consider the 2×2 block entries: $E = [E_{ij}]_{1\le i,j\le n} \in \mathbb{R}^{2n\times 2n}$:

$$E_{ij} = \begin{cases} \sum_{l\in\mathscr{N}_i} w_{il}(z_{k_{il}}z_{k_{il}}^T), & \text{if } i = j \\ -w_{ij}(z_{k_{ij}}z_{k_{ij}}^T), & \text{if } i \neq j \text{ and } \{i,j\} \in \mathscr{E} \\ \mathbf{0}_{2\times 2}, & \text{if } i \neq j \text{ and } \{i,j\} \notin \mathscr{E} \end{cases} \tag{11}$$

The following result shows that E shares the same null space with the rigidity matrix R.

Lemma 1. *$null(E(\mathscr{G}, p)) = null(R^TR) = null(R) = \mathscr{F}$.*

Proof. It is obvious that $null(R^TR) = null(R)$. The weight matrix W is invertible as it is a diagonal matrix with positive diagonal entries. Thus $rank(E(\mathscr{G}, p)) = rank(R^TR)$ and the eigenspace corresponding to the zero eigenvalues of E is the same as that of R^TR. □

The above result will be used in Section 4 to construct a modified matrix based on the above vectors in the null space of R for the distributed eigenvector estimation of E.

3.3 Useful lemmas for the optimization

Before presenting the optimization algorithm, we show several lemmas which will be used later.

Proposition 1. *The following constraint (which holds if and only if the framework $(\mathscr{G}, p)$ is infinitesimally rigid)*

$$\lambda_4(E) > 0$$

is equivalent to

$$P^TEP \succ 0 \tag{12}$$

where $P \in \mathbb{R}^{2n\times(2n-3)}$ is a matrix whose range is the orthogonal complement of $\mathscr{F}$; that is, $Im(P) = \mathscr{F}^\perp$.

Proof. The above result is a consequence of the Courant-Fischer theorem [18]. The range of P is the eigenspace spanned by the eigenvectors corresponding to the eigenvalues $\lambda_4(E), \lambda_5(E), \cdots, \lambda_{2n}(E)$ according to Lemma 1. □

Three further lemmas are in order.

Lemma 2. *For any non-zero vector $x \in \mathscr{F}^\perp$, there holds that $x^TEx > 0$ if and only if $\lambda_4(E) > 0$.*

Lemma 3. *Choose three orthonormal vectors in $\mathscr{F}$ and denote them as $\bar{v}_1, \bar{v}_2, \bar{v}_3$. Then there holds*

$$E + \sum_{i=1}^{3} \vartheta \bar{v}_i\bar{v}_i^T \succeq min\{\lambda_4, \vartheta\}I_{2n}$$

where ϑ is a positive constant.

[3] The measure we adopt is one that, without the introduction of weighting to penalize agent separation, would have the property that more separation would correspond to a higher index, which is unrealistic for practical robotic network. Hence weighting terms are introduced which penalize separation in some way, in accordance with the concept of a proximity network [14].

Lemma 4. *$\lambda_4(E) > 0$ guarantees that $\lambda_2(L(\mathscr{G}_w)) > 0$. That is, the maintenance of network rigidity preserves the network connectivity.*

The above lemmas are the consequences of symmetric matrices' spectral theorem with application to Lemma 1 and Proposition 1. The proofs are omitted here due to space limit and will be presented in the extended version of this paper.

4 Algorithm design

In this section we show several steps on algorithm design, which involves the (super-)gradient function on $\lambda_4(E)$ and the gradient function $\nabla_p\lambda_4(E)$ for optimization direction, in which a decentralized property will be established.

4.1 Calculation of a (super-)gradient on $\lambda_4(E)$

Since E is a symmetric matrix, we can obtain the following standard inequality:

$$\lambda_4(E)x^Tx \le x^TEx, \forall x \in \mathscr{F}^\perp \Rightarrow \lambda_4(E) = \inf_{\forall x\in\mathscr{F}^\perp}(\frac{x^TEx}{x^Tx}) \tag{13}$$

Denote v_4 as a normalized eigenvector corresponding to the eigenvalue $\lambda_4(E)$. Then $G := v_4v_4^T$ is a supergradient of $\lambda_4(E)$ [18]. Thus we can update the matrix E for maximizing $\lambda_4(E)$ using the following updating law

$$E^{(k+1)} = E^{(k)} + \alpha^{(k)}G^{(k)} \tag{14}$$

where $\alpha^{(k)}$ is a suitably chosen step size in the step k. [4]

The above results can be seen as parallel ones for the problem of connectivity maximization [19] involving the Fiedler eigenvector. If $\lambda_4(E)$ is single, then G is exactly the gradient of the eigenvalue function. In the following, we will assume that $\lambda_4(E)$ is non-repeated to conduct and facilitate most analysis [5]. Note that in the case of repeated eigenvalue, most analytical results in the following text still hold by employing the convergence results of supergradient/subgradient algorithm [20].

4.2 Decentralized property of the gradient function

It has been shown above that the gradient of λ_4 w.r.t. E is $v_4v_4^T$. Note that the entries of E depend nonlinearly on p. In this subsection we aim to calculate the gradient $\nabla_p\lambda_4(E)$. Since $\|v_4\| = 1$ and E is a symmetric matrix, one has $\lambda_4(E) = v_4^TEv_4$. Thus, there holds,

$$\nabla_p\lambda_4(E) = \frac{\partial(v_4^TEv_4)}{\partial p} = v_4^T\frac{\partial E}{\partial p}v_4$$

where the second equality is a non-trivial result in matrix calculus theory [18, P. 565] due to the symmetry of E.

Proposition 2. *The gradient vector $\nabla_{p_i}\lambda_4$ for agent i is decentralized in that it only involves information from agent i itself and its neighbors.*

Proof. Denote $v_4 = [v_{4,1}^T, v_{4,2}^T, \cdots, v_{4,n}^T]^T \in \mathbb{R}^{2n}$ with $v_{4,i} = [v_{4,ix}, v_{4,iy}]^T$. According to the structure of the matrix E in (11), one has

$$\begin{aligned} v_4^TEv_4 &= \sum_{\{i,j\}\in\mathscr{E}} v_{4,i}^TE_{ij}v_{4,j} \\ &= \sum_{\{i,j\}\in\mathscr{E}} w_{ij}z_{k_{ij}}^T(v_{4,i}-v_{4,j})(v_{4,i}-v_{4,j})^Tz_{k_{ij}} \end{aligned} \tag{15}$$

while the ij-th block entries of E are non-zero if and only if $\{i,j\} \in \mathscr{E}$ as shown in (11). Thus, it can be concluded that the gradient function $\nabla_{p_i}\lambda_4$ involves only agent i and its neighbors' information, which proves the decentralized property of the gradient. □

The above result enables us to design distributed optimization algorithm, such that local information with respect to neighboring agents is sufficient for implementing the algorithm.

4.3 Movement algorithm for each agent

The optimization law for updating all robots' positions are designed as

$$p^{(k+1)} = p^{(k)} + \alpha^{(k)}\nabla_p\lambda_4(E^{(k)}) \tag{16}$$

where the gradient $\nabla_p\lambda_4(E^{(k)})$ provides the updating direction [6], and in the following we will write $\nabla_p\lambda_4$ in short. For agent i, the updating law should be

$$p_i^{(k+1)} = p_i^{(k)} + \alpha^{(k)}\nabla_{p_i}\lambda_4 \tag{17}$$

where

$$\begin{aligned} \nabla_{p_i}\lambda_4 &= v_4^T\frac{\partial E}{\partial p_i}v_4 \\ &= 2\sum_{j\in\mathscr{N}_i} w_{ij}\frac{\partial z_{ij}^T}{\partial p_i}(v_{4,i}-v_{4,j})(v_{4,i}-v_{4,j})^Tz_{k_{ij}} \\ &\quad + \sum_{j\in\mathscr{N}_i}\frac{\partial w_{ij}}{\partial p_i}z_{k_{ij}}^T(v_{4,i}-v_{4,j})(v_{4,i}-v_{4,j})^Tz_{k_{ij}} \\ &= 2\sum_{j\in\mathscr{N}_i} w_{ij}(v_{4,i}-v_{4,j})(v_{4,i}-v_{4,j})^Tz_{k_{ij}} \\ &\quad + \sum_{j\in\mathscr{N}_i}\frac{\partial w_{ij}}{\partial p_i}z_{k_{ij}}^T(v_{4,i}-v_{4,j})(v_{4,i}-v_{4,j})^Tz_{k_{ij}} \end{aligned} \tag{18}$$

which is obviously decentralized (assuming that $v_{4,i}$ can be accessed by agent i and $v_{4,j}$ can be obtained locally from its neighbors), which also confirms statements in Proposition 2.

We note that the above position update law, which although has a decentralized structure, involves the entries $v_{4,i}$ of the eigenvector v_4 of the matrix E, which is essentially global information. For the implementation of (17), one can postulate that there exists a centralized processor, whose task is to collect the states of all the agents, to calculate the eigenvector v_4 of E, and then to broadcast it to all the agents. However, this is not desirable for distributed control framework for robotic networks. Hence, we will make efforts to eliminate this centralized requirement by designing distributed eigenvector algorithms, which will be discussed in next section.

[4] In our later simulation study, we take constant step sizes.

[5] In our calculations and experiments we rarely find the case of repeated eigenvalue for λ_4. We conjecture this is due to the fact that the entries of matrix E involve agents' relative positions, while the non-collocated position values ensure the very low possibility of repeated λ_4.

[6] The algorithm can guarantee the convergence to a local maximum but cannot guarantee the convergence to the global maximum due to the *non-concavity* of $\lambda_4(E(p))$ as a function of p. Note that although $\lambda_4(E)$ is a concave function of E, E depends nonlinearly on p and the composite function $\lambda_4(p)$ is non-concave.

5 Distributed eigenvector estimation via consensus algorithm

This section aims to design distributed eigenvector estimation method, via *shifted inverse power iteration* [21] combined with other typical distributed algorithms, which include the consensus algorithm for averaging [22] and the Jacobi overrelaxation method for iteratively solving linear equations in the form $Ax = b$ [23]. The reason for choosing the inverse power iteration method is to improve the convergence speed of the estimation process, which is a desired property for robotic systems in a *dynamic* environment. Also for this reason, the inverse power iteration has been discussed in a recent work [24] for the constrained connectivity control. Note that we use $\text{Ave}(\cdot)$ to denote the average operation via *dynamic average consensus*; for details, see [22].

5.1 Distributed eigenvector estimation using shifted inverse power iteration

Let us consider a matrix $\bar{E} := E + Q$ where $Q = \vartheta v_1 v_1^T + \vartheta v_2 v_2^T + \vartheta v_3 v_3^T$ and ϑ is some sufficiently large positive constant. Note that Q is a rank-three symmetric matrix, which has three positive eigenvalues and the rest are all zero. Denote the positive eigenvalues of Q as η_1, η_2, η_3. The value of ϑ is chosen such that $min\{\eta_1, \eta_2, \eta_3\} > \lambda_4$. According to Lemma 3 and matrix spectral theorem, the eigenstructure of the matrix $\bar{E}$ is as follows (note that the list is not necessarily in an ascending order and the vectors v_1, v_2, v_3 are not necessarily orthonormal):

$$\text{Eigenvalues of } \bar{E} : \lambda_4,\ \lambda_5, \cdots, \lambda_{2n},\ \eta_1,\ \eta_2,\ \eta_3 \tag{19}$$

with associated eigenvectors (or eigenspace)

$$v_4,\ v_5, \cdots, v_{2n},\ span\{v_1,\ v_2,\ v_3\} \tag{20}$$

Define a new matrix $(\bar{E} - \mu I_{2n})^{-1}$ where μ is a positive number. It follows that the set of eigenvectors of $(\bar{E} - \mu I_{2n})^{-1}$ are the same as those of $\bar{E}$, with the eigenvalues listed below:

$$(\lambda_4 - \mu)^{-1}, \cdots, (\lambda_{2n} - \mu)^{-1}, (\eta_1 - \mu)^{-1}, (\eta_2 - \mu)^{-1}, (\eta_3 - \mu)^{-1} \tag{21}$$

By choosing μ close to λ_4, the dominant eigenvalue of $(\bar{E} - \mu I_{2n})^{-1}$ will be much larger in magnitude than other eigenvalues listed above. Thus, by doing the power iteration method on the matrix $(\bar{E} - \mu I_{2n})^{-1}$, the convergence rate can be greatly improved [21].

Another issue is to obtain in a distributed way the inverse of the matrix $(\bar{E} - \mu I_{2n})$. Instead of performing the matrix inversion operation, we would like to solve the following linear equation:

$$(\bar{E} - \mu I_{2n}) r = \widetilde{v}_4^{(s)} \tag{22}$$

where $\widetilde{v}_4^{(s)}$ is the estimate of v_4 at the s-th step. The Jacobi overrelaxation iteration method [23] is employed to solve it in a distributed way:

$$\begin{aligned} r_i^{(t+1)} &= (1-\gamma) r_i^{(t)} \\ &\quad - \frac{\gamma}{\bar{E}_{ii} - \mu} \left(-\widetilde{v}_{4,i}^{(s)} - \sum_{j \in \mathcal{N}_i} E_{ij} r_j^{(t)} + \sum_{j \neq i} Q_{ij} r_j^{(t)} \right) \end{aligned} \tag{23}$$

where Q_{ij} is the ij-th block of the matrix Q:

$$Q_{ij} = \vartheta \begin{pmatrix} p_{iy} p_{jy} + 1 & -p_{iy} p_{jx} \\ -p_{ix} p_{jy} & p_{ix} p_{jx} + 1 \end{pmatrix} \tag{24}$$

The iteration step in (23) involves mostly local communication except that the calculation of the last term $\sum_{j \neq i} Q_{ij} r_j^{(t)}$ requires global information. Note that $\sum_{j \neq i} Q_{ij} r_j^{(t)} = n\text{Ave}(\sum_j Q_{ij} r_j^{(t)}) - Q_{ii} r_i^{(t)}$, where $\text{Ave}(\sum_j Q_{ij} r_j^{(t)})$ is the average information which can be computed by using the local average estimator

$$\vartheta \begin{pmatrix} p_{iy}\text{Ave}(p_{iy} r_{i,x}^{(t)}) + \text{Ave}(r_{i,x}^{(t)}) - p_{iy}\text{Ave}(p_{ix} r_{i,y}^{(t)}) \\ -p_{ix}\text{Ave}(p_{iy} r_{i,x}^{(t)}) + p_{ix}\text{Ave}(p_{ix} r_{i,y}^{(t)}) + \text{Ave}(r_{i,y}^{(t)}) \end{pmatrix} \tag{25}$$

Suppose after $\bar{t}$ steps the iteration of the solution to (22) converges with specified stopping criteria. The following normalization step can also be implemented in a distributed way by using the average consensus estimator:

$$\widetilde{v}_{4,i}^{(s+1)} = \frac{r_i^{(\bar{t})}}{\sqrt{n\text{Ave}((r_{i,x}^{(\bar{t})})^2 + (r_{i,y}^{(\bar{t})})^2)}} \tag{26}$$

Further suppose that after $\bar{s}$ steps, the convergence of the eigenvector estimation is achieved. Then the optimization algorithm in (17) can be modified by replacing the relevant entries of v_4 with their estimates.

The distributed implementation of the algorithm is elaborated in the following algorithm table [7].

Algorithm 1: Distributed eigenvector and eigenvalue estimation

Data: agent i's own position p_{ix}, p_{iy}, agent i's neighbors' positions: $p_{jx}, p_{jy}, j \in \mathcal{N}_i$, the dynamic average consensus measurement via $\text{Ave}(\cdot)$ operation

Result: agent i's estimate of the eigenvalue $\widetilde{\lambda}_{4,i}$ and the eigenvector $\widetilde{v}_{4,i}^{(\bar{s})}$

Initialization: agent i chooses the initial value for $\widetilde{v}_{4,i}^{(0)}$

for $s = 1$ **to** $\bar{s}$ **do**
  for $t = 1$ **to** $\bar{t}$ **do**
    Iteratively solve $(\bar{E} - \mu I_{2n}) r = \widetilde{v}_4^{(s)}$ using Jacobi overrelaxation method: (23) and (25);
  end
  Normalization step: calculate the normalized $\widetilde{v}_{4,i}^{(s+1)}$ using (26);
end
return *The local estimate of* $\widetilde{v}_{4,i}$.

5.2 Conditions for convergence

When the Jacobi overrelaxation iteration method is used to solve the linear equation related to the matrix $(\bar{E} - \mu I_{2n})$, the condition for convergence is that $\gamma > 0$ is sufficiently small, and the matrix $(\bar{E} - \mu I_{2n})$ should be symmetric and positive definite [23, P. 154]. The parameter γ can be adjusted in

[7] The normalization can also be done when necessary, but not at every $\bar{t}$ step. This can reduce the calculation burden.

the implementation process. In order to ensure the positive definiteness of $(\bar{E}-\mu I_{2n})$, we can choose ϑ and μ such that $min\{\eta_1,\eta_2,\eta_3\}>\lambda_4$ and $0<\mu<\lambda_4$. The latter condition also ensures that $(\lambda_4-\mu)^{-1}$ is the dominant eigenvalue of the matrix $(\bar{E}-\mu I_{2n})^{-1}$.

5.3 Convergence rate of the eigenvector estimation

The convergence rate of the estimation of the eigenvector is controlled by the ratio $|\frac{\lambda_4-\mu}{\lambda_5-\mu}|$ [21]. Thus, by choosing μ closer to λ_4, the convergence rate will be faster. If one fixes μ in each optimization step, due to $\lambda_4(k)\geq\lambda_4(k-1)\geq\cdots\geq\lambda_4(0)$, the convergence rate for eigenvector estimation will be decreasing with the increasing of update step k. Thus an adaptive choice of μ will be desirable to improve the convergence when step k becomes large. We propose an idea of using the updated $\lambda_4(k)$ for adjusting μ in each succeeding step. Since the eigenvector is estimated in each step, the eigenvalue can be estimated by using the Rayleigh quotient

$$\widetilde{\lambda}_4=(\widetilde{v}_4^{(\bar{s})})^T E(\widetilde{v}_4^{(\bar{s})}) \tag{27}$$

However, the actual value $\widetilde{\lambda}_4$ cannot be computed by each agent as the information of the estimated normalized eigenvector $\widetilde{v}^{(\bar{s})}$ cannot be accessed by all the agents. Nevertheless, the local estimation of the eigenvalue can still be computed by using again the average consensus procedure. The initial input for the average consensus is

$$\chi_i(0)=(\widetilde{v}_{4,i}^{(\bar{s})})^T\sum_{j\in\mathcal{N}_i}E_{ij}(\widetilde{v}_{4,i}^{(\bar{s})}-\widetilde{v}_{4,j}^{(\bar{s})}) \tag{28}$$

Thus agent i can calculate its local estimation of the eigenvalue by

$$\widetilde{\lambda}_{4,i}=n\mathrm{Ave}(\chi_i) \tag{29}$$

All the agents can then run a min-max consensus to reach an agreement on the *smallest estimation* of the eigenvalue and to choose $\mu(k)$ satisfying $0<\mu(k)\leq\widetilde{\lambda_4(k)}_{min}<\lambda_4(k)$ such that $\mu(k)$ is close to $\lambda_4(k)$ but still smaller than it. Note that the accurate value of the min-consensus can be reached in *finite steps*.

6 Simulation results

In this section we give some simulation results obtained by using Matlab/Simulink. Suppose the system has $n=5$ robots in the plane with communication topology being modeled by a proximity network. We consider the following edge weight function

$$w_{ij}=\begin{cases} e^{-\|z_{k_{ij}}\|^2/(2\sigma^2)}, & \text{if } \|z_{k_{ij}}\|\leq\kappa \\ 0, & \text{otherwise}\end{cases} \tag{30}$$

which is a very popular choice for modeling proximity networks; see e.g. [3,4,6]. One can choose the scalar parameter σ to satisfy a threshold condition $e^{-\kappa^2/(2\sigma^2)}=\sigma'$, with σ' being a small predefined threshold [8]. In the simulation we set the communication radius as $\kappa=10$ and the threshold as $\sigma'=0.01$.

[8]The definition of the weights introduces a non-smooth term in the optimization, which can be avoided by introducing a smooth bump function as suggested in [6]. Another remedy in our discrete-time optimization setting is to choose σ' as a suitably small number to reduce the effect of discontinuity, which is the approach used in our simulation. Smooth nonlinear function for modelling the weight is also possible, see [17].

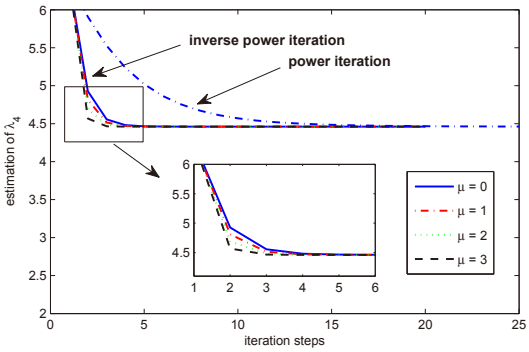

Fig. 1: Convergence speed of the eigenvalue estimation between power iteration and inverse power iteration method.

6.1 Distributed eigenvector(eigenvalue) estimation

We simulate using randomly-generated positions for the robots in a 10×10 square area (position data omitted here), which in one case generates a matrix E with the spectrum $[0,0,0,4.46,8.49,11.18,18.56,22.85,32.74,38.93]$. By using the distributed inverse power iteration method, the desired eigenvector has been estimated by each agent (simulation results not shown here). We consider the power iteration method discussed in [4,6,11] and modify a discrete-time version for the comparison. To compare the convergence speed, all the assumptions and initial conditions of the estimates are the same. The results are shown in Fig.1. It is obvious that the distributed inverse power iteration method proposed in this paper displays superior performance over the distributed power iteration method considered in [4,6,11].

Also from Fig.1 one can observe that, when the initial guess μ is chosen closer to the true value of λ_4, the convergence will be much faster, which can be achieved by only a few iteration steps. This property is quite favorable for distributed large-scale robot network control.

6.2 Distributed optimization on network rigidity

The simulation settings are the same to the above subsection. Initially the five robot group form a rigid framework (indicated by the red dashed lines in Fig. 2), while the rigidity is at a low level. After running the algorithm (via a small constant step size $\alpha=0.02$) with distributed estimation of the global eigenvector information, all the agents iteratively move to a final framework so that the rigidity index is maximized. Fig. 3 shows the evolution of the rigidity level for the dynamic network which finally reaches a constant value. The final shape with a maximum rigidity shown in Fig. 2 comes at no surprise, which is illustrated by a complete graph, but note that each edge is weighted with an optimum inter-agent distance so that an optimal rigidity can be achieved. Also note the edge weight is a function of distances, and the final configuration depends strongly on the weight function that models the weighted proximity network.

7 Conclusions

In this paper we have discussed the rigidity optimization problem for multi-robot systems modeled in a proximity network. An algorithm for updating robots' positions is proposed to maximize the worst-case rigidity level from a rigid-

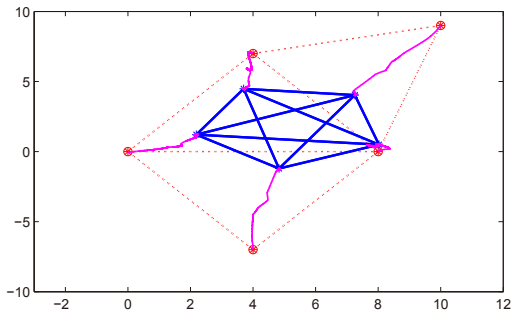

Fig. 2: The initial and final configurations of the 5-robot team, and their trajectories in the rigidity maximization process. The initial positions are denoted by circles, with initial weighted edge denoted by dashed red line. The trajectories are shown in pink lines, and the final configuration is shown in blue.

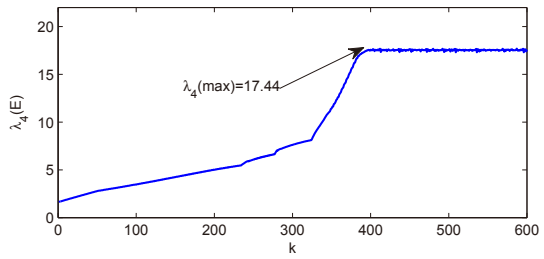

Fig. 3: The evolution of the worst case rigidity level characterized by the eigenvalue λ_4. The maximum value is shown after the iteration and remains constant.

ity Gramian matrix, constructed by a weighted rigidity matrix which captures the property of proximity networks. To implement the algorithm in a totally distributed way, we devise an eigenvector and eigenvalue estimation approach via the inverse power iteration method in combination with distributed consensus algorithm as an inner loop. The convergence rate of the distributed estimation can be greatly improved by suitably choosing the shift parameter (as discussed in Section 5.3). The analytical results and effectiveness of the proposed distributed algorithms are validated by simulations. Future work will include the consideration of certain mobility constraints with distributed algorithm, and generalizations to more realistic robot models.

References

[1] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[2] Y. Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph laplacian," *Automatic Control, IEEE Transactions on*, vol. 51, no. 1, pp. 116–120, 2006.

[3] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[4] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390–396, 2010.

[5] P. R. Giordano, A. Franchi, C. Secchi, and H. H. Bülthoff, "A passivity-based decentralized strategy for generalized connectivity maintenance," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 299–323, 2013.

[6] L. Sabattini, C. Secchi, N. Chopra, and A. Gasparri, "Distributed control of multirobot systems with global connectivity maintenance," *Robotics, IEEE Transactions on*, vol. 29, no. 5, pp. 1326–1332, 2013.

[7] B. D. O. Anderson, C. Yu, B. Fidan, and J. M. Hendrickx, "Rigid graph control architectures for autonomous formations," *Control Systems Magazine, IEEE*, vol. 28, no. 6, pp. 48–63, 2008.

[8] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. O. Anderson, and P. N. Belhumeur, "A theory of network localization," *Mobile Computing, IEEE Transactions on*, vol. 5, no. 12, pp. 1663–1678, 2006.

[9] Z. Sun, S. Mou, M. Deghat, B. D. O. Anderson, and A. S. Morse, "Finite time distance-based rigid formation stabilization and flocking," in *Proc. of the 19th IFAC Congress*, pp. 9183–9189, 2014.

[10] G. Zhu and J. Hu, "Link resource allocation for maximizing the rigidity of multi-agent formations," in *Proc. of the 50th Conference on Decision and Control*, pp. 2920–2925, IEEE, 2011.

[11] D. Zelazo, A. Franchi, H. H. Bülthoff, and P. R. Giordano, "Decentralized rigidity maintenance control with range-only measurements for multi-robot systems," *The International Journal of Robotics Research, article in press*, 2014.

[12] I. Shames and T. H. Summers, "Rigid network design via submodular set function optimization," *submitted to IEEE Transactions on Network Science and Engineering, available at http://control.ee.ethz.ch/ tsummers/papers/TNSE2014.pdf*, 2014.

[13] R. K. Williams, A. Gasparri, A. Priolo, and G. S. Sukhatme, "Evaluating network rigidity in realistic systems: Decentralization, asynchronicity, and parallelization," *Robotics, IEEE Transactions on*, vol. 30, no. 4, pp. 950–965, 2014.

[14] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[15] Z. Sun, C. Yu, and B. D. O. Anderson, "Distributed estimation and control for preserving formation rigidity for mobile robot teams," *arXiv preprint arXiv:1309.4850*, 2013.

[16] I. Shames, B. Fidan, and B. D. O. Anderson, "Minimization of the effect of noisy measurements on localization of multi-agent autonomous formations," *Automatica*, vol. 45, no. 4, pp. 1058–1065, 2009.

[17] A. Simonetto, T. Keviczky, and R. Babuška, "Constrained distributed algebraic connectivity maximization in robotic networks," *Automatica*, vol. 49, no. 5, pp. 1348–1357, 2013.

[18] D. A. Harville, *Matrix algebra from a statistician's perspective*. Springer, 2008.

[19] M. C. DeGennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *Proc. of the 45th Conference on Decision and Control*, pp. 3628–3633, IEEE, 2006.

[20] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *lecture notes of EE392, Stanford University, Autumn Quarter*, 2003.

[21] L. N. Trefethen and D. Bau III, *Numerical linear algebra*, vol. 50. SIAM, 1997.

[22] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.

[23] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[24] R. K. Williams and G. S. Sukhatme, "Locally constrained connectivity control in mobile robot networks," in *Proc. of International Conference on Robotics and Automation (ICRA)*, pp. 901–906, IEEE, 2013.