J Intell Robot Syst (2009) 56:99–126
DOI 10.1007/s10846-009-9328-8

Distributed Motion Constraints for Algebraic Connectivity of Robotic Networks

Michael Schuresko · Jorge Cortés

Received: 9 June 2008 / Accepted: 30 March 2009 / Published online: 29 April 2009

Abstract This paper studies connectivity maintenance of robotic networks that communicate at discrete times and move in continuous space. We propose a distributed coordination algorithm that allows the robots to decide whether a desired collective motion breaks connectivity. We build on this procedure to design a second coordination algorithm that allows the robots to modify a desired collective motion to guarantee that connectivity is preserved. These algorithms work under imperfect information caused by delays in communication and the robots' mobility. Under very outdated information, the proposed algorithms might prevent some or all of the robots from moving. We analyze the correctness of our algorithms by formulating them as games against a hypothetical adversary who chooses system states consistent with observed information. The technical approach combines tools from algebraic graph theory, linear algebra, and nonsmooth analysis.

Keywords Robotic networks · Cooperative control · Graph connectivity · Flocking

Mathematics Subject Classifications (2000) 93C85 · 05C50 · 05C40 · 68W15

1 Introduction

Network connectivity is a critical issue in cooperative robotics. In many applications, connectivity is needed in order to guarantee the successful completion of a desired

M. Schuresko (✉)
Department of Applied Mathematics and Statistics,
University of California, Santa Cruz, CA 95064, USA
e-mail: mds@soe.ucsc.edu

J. Cortés
Department of Mechanical and Aerospace Engineering,
University of California, San Diego, CA 92093, USA
e-mail: cortes@ucsd.edu

coordination task. Examples include rendezvous at a point and distributed sensor fusion. In sensor fusion, distributed agreement protocols have convergence rates which depend on the degree of connectivity of the underlying communication network. Since connectivity is a global property, it is difficult to maintain it in a distributed manner. The objective of this paper is to develop a distributed approach to preserving network connectivity that allows for flexibility of individual robot motions and, at the same time, does not impose a heavy communication burden on the operation of the overall network.

Literature Review We classify previous work on connectivity of robotic networks into two main categories. The first category deals with how to design the network motion so that some desired measure of connectivity is maximized under a given set of position constraints. In [1], convex optimization is used to solve a connectivity problem in the presence of convex constraints on the strength of each inter-agent communication link. A solution to a related problem with nonconvex constraints is presented in [11]. An extension of similar methods to provide a distributed algorithm when the strength of each communication link is a convex function of inter-robot distance is presented in [5]. Potential fields are used in [24] to maximize algebraic connectivity. The second category deals with a measure of the connectivity of the interaction graph, a connectivity threshold, and some coordination task. In this category, algorithms are designed so that the robots' motion achieve the task subject to the value of the measure of connectivity never crossing the threshold. A solution to such a problem is proposed in [22]. This solution allows for a general range of agent motions, but is not distributed. A distributed approach to maintaining connectivity from a hybrid systems perspective is proposed in [23]. A distributed solution that makes agents with second-order dynamics maintain a fixed set of edges appears in [17]. A distributed solution which allow for varying set of edges to be preserved is presented in [19]. Connectivity problems have been studied also in the context of formation control. In [20], connectivity-preserving motions between pairs of formations are generated. Control laws based on the Laplacian matrix of the interconnection graph are designed in [10] to solve formation control problems while preserving connectivity.

Statement of Contributions In this paper, our approach starts by considering a measure of the connectivity of the interaction graph based on its Laplacian matrix. The Laplacian matrix of a graph, G, is an analog to the Laplacian operator over G; its second smallest eigenvalue, λ_2, determines many connectivity properties of the graph G. Given a pre-specified (arbitrary) threshold on λ_2, and a proposed instantaneous direction of physical motion, we set out to solve the following version of the connectivity maintenance problem: how can the robots cooperatively decide which proposed motions can safely be taken without causing the measure of connectivity (λ_2) to cross below the threshold? An added difficulty is the fact that the gradient of the second smallest eigenvalue of the Laplacian matrix is a nonsmooth function of the edge weights of the underlying graph. Our solution uses an information dissemination algorithm to compute upper and lower bounds on the Laplacian matrix of the interaction graph computed individually by each robot based on partial information. Each robot then plays a game against an opponent who picks graphs consistent with the information available to that robot. A given robot wins the game if it moves in a direction close to the proposed motion which is guaranteed not to

decrease λ_2 whenever λ_2 is below the given threshold. Each robots is free to pick "stand still" as an allowable motion should it fail to find an actual motion which satisfies the criteria for winning the game. The proposed coordination algorithm works under imperfect information caused by delays in communication and the robots' mobility, and has the added advantage of allowing for nonconvex mappings from inter-robot distance to edge weights. We provide correctness guarantees for the proposed coordination algorithm and show several simulations when combined with algorithms that implement random motion, trajectory following, and flocking.

Organization The paper is organized as follows. Section 2 introduces basic notation and notions from nonsmooth analysis and graph theory. Section 3 formally defines the robotic network model and the connectivity problem we address. Section 4 formulates this problem as a game against a world-picking opponent and presents algorithmic solutions to it. This section also introduces the communication protocol used by the network for position information dissemination. Section 5 ties all the ingredients together to propose a distributed coordination algorithm for connectivity maintenance and presents various simulation results. Finally, Section 6 presents our conclusions and ideas for future work.

Notation Throughout the paper, $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{R}_{>0}$ denote the sets of real, nonnegative real, and positive real numbers, respectively. For a set S, $\mathbb{F}(S)$ denotes the collection of all finite subsets of S. Whenever we provide algorithm pseudo-code, we use $a \leftarrow b$ to mean "a is assigned a value of b." We denote by $\mathbb{R}^{m\times n}$ the space of matrices of size $m \times n$, and by $\mathrm{Sym}(n)$ the space of symmetric square matrices of size n. The *Frobenius inner product* of $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times n}$ is defined by

$$A \bullet B = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{i,j}B_{i,j}.$$

The 2-norm of $M \in \mathbb{R}^{n\times n}$, denoted $\|M\|_2$, is the norm induced by the Frobenius inner product, i.e., $\|M\|_2 = \sqrt{M \bullet M}$. The strong norm of $M \in \mathbb{R}^{n\times n}$, denoted $\|M\|_{\mathrm{s}}$, is $\max_{x\in\mathbb{S}^n}\{x^T Mx\}$. Note that $\|M\|_{\mathrm{s}} \leq \|M\|_2$. For convenience, we introduce the "vectorization" $\mathrm{vec} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n^2}$ of a matrix defined by $\mathrm{vec}(M)_{in+j} = M_{i,j}$. Note that $(\mathrm{vec}(A))^T\mathrm{vec}(B) = A \bullet B$. Finally, we denote $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^n$ and $\mathbf{0} = (0, \ldots, 0)^T \in \mathbb{R}^n$.

2 Preliminaries

This section presents preliminary notions on algebraic graph theory, proximity graphs, and nonsmooth analysis. As we illustrate later, nonsmooth analysis is needed in order to characterize the smoothness properties of the algebraic connectivity function associated to the robotic network.

2.1 The Graph Laplacian and its Spectrum

An (undirected) graph $G = (V, \mathcal{E})$ consists of a vertex set V and an edge set $\mathcal{E} \subset V \times V$ of unordered pairs of vertexes, i.e., $(i, j) \in \mathcal{E}$ implies that $(j, i) \in \mathcal{E}$. A weighted graph is an undirected graph where each edge $(i, j) \in \mathcal{E}$ has an associated weight

$w_{i,j} \in \mathbb{R}_{\geq 0}$. For a weighted graph $G = (V, \mathcal{E})$, the (weighted) adjacency $A(G) \in \mathbb{R}^{n \times n}$ and the Laplacian $L(G) \in \mathbb{R}^{n \times n}$ are given by

$$A(G)_{i,j} = w_{i,j}$$

$$L(G)_{i,j} = \begin{cases} \sum_{k \neq i} w_{i,k} & i = j, \\ -w_{i,j} & i \neq j. \end{cases}$$

When the specific graph is clear from the context, we simply use A and L. Note that both matrices are symmetric. For convenience, we denote by $\Lambda : \mathrm{Sym}(n) \to \mathrm{Sym}(n)$ the linear map that transforms an adjacency matrix A into the Laplacian L defined by

$$\Lambda(A) = \mathrm{diag}(A\mathbf{1}) - A = L.$$

Properties of the Laplacian matrix include [7]: the vector $\mathbf{1} \in \mathbb{R}^n$ is an eigenvector with eigenvalue 0; $L(G)$ is positive semidefinite; and the dimensionality of the null space of $L(G)$ is equal to the number of connected components of G. As a consequence of these properties, an undirected graph is connected if and only if the second smallest eigenvalue of its Laplacian is greater than zero. Another convenient property of Laplacians is that adding weight to an edge is guaranteed not to decrease any of its eigenvalues [21].

2.2 Proximity Graphs and Proximity Functions

We use proximity graphs as an abstraction of network connectivity among spatially distributed robots. A proximity graph is an association of a set of positions with a weighted graph. Let $\mathcal{P} = (p_1, \ldots, p_n) \in (\mathbb{R}^d)^n$ be a vector of n robot positions, where each robot evolves in $\mathbb{R}^d$. Let $\mathbb{G}(n)$ be the set of weighted graphs whose vertex set is the set of integers between 1 and n, denoted by $\{1, \ldots, n\}$. Then, we have the following definition [9].

Definition 1 A *proximity graph* $\mathcal{G} : (\mathbb{R}^d)^n \to \mathcal{G}(n)$ associates to $\mathcal{P} \in (\mathbb{R}^d)^n$ a graph with vertex set $\{1, \ldots, n\}$, edge set $\mathcal{E}_G(\mathcal{P})$, where $\mathcal{E}_G : (\mathbb{R}^d)^n \to \{1, \ldots, n\} \times \{1, \ldots, n\}$, and weights $w_{i,j} \in \mathbb{R}_{>0}$ for all $(i, j) \in \mathcal{E}_G(x)$. A proximity graph must satisfy that $G(p_{\sigma(1)}, \ldots, p_{\sigma(n)})$ is isomorphic to $G(p_1, \ldots, p_n)$ for any n-permutation σ and $(p_1, \ldots, p_n) \in (\mathbb{R}^d)^n$.

We refer the reader to [4] for several examples of proximity graphs. For a given proximity graph, we often use the associated *proximity function* $(\mathbb{R}^d)^n \to \mathrm{Sym}(n)$ that maps a tuple $\mathcal{P} \in (\mathbb{R}^d)^n$ to the adjacency matrix $A(G(\mathcal{P})) \in \mathrm{Sym}(n)$. Note that a proximity graph can be alternatively defined by specifying a proximity function.

Remark 1 Examples of proximity functions include the following:

(i) the r-disk proximity function,

$$f_{r\text{-disk}}(p_1, \ldots, p_n)_{i,j} = \begin{cases} 1, & \|p_i - p_j\| \leq r, \\ 0, & \text{otherwise}, \end{cases}$$

(ii) the exponentially-decaying proximity function,

$$f_{\exp}(p_1, \ldots, p_n)_{i,j} = \exp(-\|p_i - p_j\|),$$

(iii) the approximate r-disk graph, for a sharpness value $k \in \mathbb{R}$,

$$f_{r\text{-disk-cont}}(p_1, \ldots, p_n)_{i,j} = \frac{1}{1 + \exp(k(\|p_i - p_j\| - r))}.$$

(iv) the spline graph, with $0 < r_{\min} < r_{\max}$, an approximation of $f_{r\text{-disk}}$, with $r \in [r_{\min}, r_{\max}]$. The (i, j) entry $f_{\text{spline}}(p_1, \ldots, p_n)_{i,j}$ is given by

$$\begin{cases} 0, & \|p_i - p_j\| < r_{\min}, \\ 1 - 3\left(\dfrac{\|p_i - p_j\| - r_{\min}}{r_{\max} - r_{\min}}\right)^2 + 2\left(\dfrac{\|p_i - p_j\| - r_{\min}}{r_{\max} - r_{\min}}\right)^3, & r_{\min} \le \|p_i - p_j\| \le r_{\max}, \\ 1, & s = \|p_i - p_j\| > r_{\max}, \end{cases}$$

These examples are particular classes of a larger class of proximity functions defined by $f(p_1, \ldots, p_n)_{i,j} = g_{\text{wgt}}(\|p_i - p_j\|)$, with $g_{\text{wgt}} : \mathbb{R}_{\ge 0} \to \mathbb{R}$. For this paper we consider proximity functions of this form with the added restrictions that g_{wgt} is $\mathcal{C}^2$ and monotonically decreasing, like in examples (ii)–(iv). Some properties of our proposed algorithms will require that the second derivative of g_{wgt} is bounded and g_{wgt} has zero derivative at 0, like in example (iv).

2.3 Elements of Nonsmooth Analysis

It is possible to define a notion of gradient for locally Lipschitz functions [3]. Let $f : \mathbb{R}^d \to \mathbb{R}$ be locally Lipschitz at $x \in \mathbb{R}^d$. For any $v \in \mathbb{R}^d$, the *generalized directional derivative of f at x in the direction v*, denoted $f^\circ(x; v)$, is

$$f^\circ(x; v) = \limsup_{y \to x, t \downarrow 0} \frac{f(y + tv) - f(y)}{t}.$$

In contrast, the *one-sided directional derivative, of f at x in the direction v*, denoted $f'(x; v)$, is

$$f'(x; v) = \lim_{t \to 0} \frac{f(y + tv) - f(y)}{t}.$$

The generalized directional derivative has the property of always being well-defined, whereas the one-sided directional derivative might not exist in some cases. The *generalized gradient of f* at $x \in X$, denoted $\partial f(x)$, is the subset

$$\partial f(x) = \left\{\xi \in X | f^\circ(x; v) \ge \xi^T v \text{ for all } v \text{ in } X\right\}.$$

If f is continuously differentiable at x, then $\partial f(x) = \{\nabla f(x)\}$.

2.4 Nonsmooth Analysis of the Algebraic Connectivity Function

Here we specify our scalar measure of network connectivity. Denote the (not necessarily distinct) eigenvalues of $M \in \text{Sym}(n)$ by $\lambda_1(M) \le \lambda_2(M) \le \cdots \le \lambda_n(M)$.

We denote by $f_{\lambda_i} : \mathrm{Sym}(n) \to \mathbb{R}$ the function that maps the matrix M to $\lambda_i(M)$. Given a proximity function $f : (\mathbb{R}^d)^n \to \mathrm{Sym}(n)$, we let

$$f_{i-\mathrm{conn}} = f_{\lambda_i} \circ \Lambda \circ f : (\mathbb{R}^d)^n \to \mathbb{R}. \tag{1}$$

We refer to $f_{2-\mathrm{conn}}$ as the *algebraic connectivity function.*

Next, we analyze the smoothness properties of the functions $f_{i-\mathrm{conn}}$, for $i \in \{1, \ldots, n\}$. We are particularly interested in $f_{2-\mathrm{conn}}$, but the same results are valid for any $f_{i-\mathrm{conn}}$, and therefore we present them in general.

Lemma 1 *For $i \in \{1, \ldots, n\}$, the function f_{λ_i} is globally Lipschitz with Lipschitz constant* 1.

Since the composition of Lipschitz functions is also Lipschitz, we have the following corollary.

Corollary 1 *For $i \in \{1, \ldots, n\}$ and a locally Lipschitz proximity function f, the connectivity function f_{i-conn} is also locally Lipschitz.*

The following result [12] gives gradients of functions f_{λ_i}.

Theorem 1 *For $i \in \{1, \ldots, n\}$, the generalized directional derivative (in the direction $X \in Sym(n)$) and the generalized gradient of f_{λ_i} at $M \in Sym(n)$ are given by*

$$f^{\circ}_{\lambda_i}(M; X) = \max_{\{v \in \mathbb{S}^n \mid Mv = \lambda_i v\}} vv^T \bullet X,$$

$$\partial f_{\lambda_i}(M) = co_{\{v \in \mathbb{S}^n \mid Mv = \lambda_i v\}} \{vv^T\}.$$

The next result is a consequence of (1) and the nonsmooth chain rule [3, Theorem 2.3.10].

Theorem 2 *Given a continuously differentiable proximity function, $f : (\mathbb{R}^d)^n \to Sym(n)$, we have at $\mathcal{P} \in (\mathbb{R}^d)^n$, and $L = \Lambda(f(\mathcal{P}))$,*

$$\partial f_{i-conn}(\mathcal{P}) \subseteq (\mathrm{vec}(\partial f_{\lambda_i}(L)))^T (\nabla \mathrm{vec}(L)).$$

3 Robotic Network Model and Problem Formulation

Here, we describe our assumptions on the robotic network and state the problem we address in this paper.

3.1 Robotic Network Model

In this section, we informally describe our robotic network model and its operation under a coordination algorithm. A more formal description would be possible, for instance within the modeling framework introduced in [14], but here we have chosen to keep the presentation simpler. We consider a network of n physical agents moving in $\mathbb{R}^d$ according to the first-order dynamics

$$\dot{p}_i = u_i, \quad i \in \{1, \ldots, n\}. \tag{2}$$

We assume this model for simplicity, although the algorithms proposed later work equally for any controllable agent model with bounded inputs. We let $\mathcal{P} = (p_1, \ldots, p_n) \in (\mathbb{R}^d)^n$ refer to the vector of all robot positions. Robots communicate over a spatially-induced proximity graph at discrete-time instants. The proximity graph is defined via a proximity function as introduced in Section 2.2. A coordination algorithm specifies a set of time instants when communication takes place. At these time instants, each robot sends a finite number of real numbers to each of its neighbors, performs computation on those numbers as specified by the algorithm, and stores the results. The set $\mathbb{F}(\mathbb{R} \cup \{\text{true}, \text{false}\})$ is the space of possible collections of stored variables at each robot. In between communication rounds, the motion of each agent motion is governed by a control law specified by the coordination algorithm. This law specifies robot motion as a function of physical state and the variables stored during the discrete-time communication and computation. Finally, an evolution of the robotic network under a coordination algorithm is the iterative execution of discrete-time communication, discrete-time computation, and continuous-time motion from a valid initial state.

3.2 Problem Formulation

Assume we are given a specific algorithm which achieves a coordination task. Our objective is to design a procedure that modifies the directions of motion specified by the given algorithm as little as possible while preserving network connectivity. Let us start by formalizing this idea.

Definition 2 An Underlying Control Law for n robots in $\mathbb{R}^d$ is a specification, for each network configuration $\mathcal{P}$, of a control input $u_{\text{goal}-i}$ for each robot $i \in \{1, \ldots, n\}$, a bound $\theta_{\max-i}$ on the angle by which the true motion of the ith robot is allowed to deviate from $u_{\text{goal}-i}$, and a time step $\delta T > 0$ over which $u_{\text{goal}-i}$ and $\theta_{\max-i}$ are valid. A set of inputs $(u_i)_{i\in\{1,\ldots,n\}}$ is *compatible with* the Underlying Control Law if and only if the following two conditions hold for all $i \in \{1, \ldots, n\}$,

$$\|u_i\| \leq \|u_{\text{goal}-i}\|, \qquad |\angle(u_{\text{goal}-i}, u_i)| \leq \theta_{\max-i}.$$

The first problem we address is that of deciding when a proposed motion can be made while safely maintaining connectivity of the robotic network.

Problem 1 (Spectral Connectivity Decision Problem) Given a control input, u_i known to the ith agent, for $i \in \{1, \ldots, n\}$, a control bound, $v_{\max}$, known to all agents, such that each agent's control input, u_i must always satisfy $\|u_i\| \leq v_{\max}$, a time interval $[t_0, t_0 + \delta T]$, and $[\lambda_-, \lambda_+] \subset \mathbb{R}_{>0}$, Spectral Connectivity Decision Problem consists of providing a (distributed) procedure which, for each robot i returns a value, $f_{\text{safe}} \in \mathbb{R}$ having $f_{\text{safe}} \geq 0$ only if the following hold for all $t \in [t_0, t_0 + \delta T]$ for all $\{\tilde{u}_j, \|\tilde{u}_j\| \leq v_{\max}\}_{j\in\{1,\ldots,n\}\setminus\{i\}}$ and for all network configurations, $\mathcal{P}$, consistent with the information available to the ith robot.

- $f_{2-\text{conn}}(\mathcal{P}(t)) \notin [\lambda_-, \lambda_+]$, or
- $f^{\circ}_{2-\text{conn}}(\mathcal{P}(t); [\mathbf{0}, \ldots, u_i^T, \ldots, \mathbf{0}]^T) \geq 0$.

where $\mathcal{P}(t)$ denotes the network evolution under control $(\tilde{u}_1, \ldots, \tilde{u}_{i-1}, u_i, \tilde{u}_{i+1}, \ldots, \tilde{u}_n)$ starting from $\mathcal{P}$.

We follow with an informal description of Definition 1

[Informal description:] Our goal is to determine (in a distributed manner) whether a given motion could, potentially, cause λ_2 to drop below the threshold λ_+. Under ideal conditions with exact computation, it would suffice to ensure that no motion causes the time derivative of λ_2 to be less than zero whenever λ_2 is exactly equal to λ_+. Because exact computation is, at best, infeasible (and at worst impossible), we instead ensure that the time derivative of λ_2 is never less than zero whenever λ_2 inhabits a range around λ_+, in this case $[\lambda_-, \lambda_+]$ (while we have not characterized an optimal value for λ_- given λ_+, in practice $\lambda_- = 0.0$ works just fine, and elegantly handles cases where the initial connectivity is lower than planned for). We further specify that our solution answer this decision problem by returning a number, f_{safe}, which is greater than or equal to zero only if we can guarantee that our proposed robot motion is safe.

The second problem we address is the problem of determining directions of motion that are compatible with the given algorithm and preserve connectivity.

Problem 2 (Spectral Connectivity Problem) Given an Underlying Control Law, a bound, $v_{\max}$, on agent control input, a time interval, $[t_0, t_0 + \delta T]$, and an interval $[\lambda_-, \lambda_+] \subset \mathbb{R}_{>0}$, Spectral Connectivity Problem consists of providing a procedure which, for each robot, $i \in \{1, \ldots, n\}$ finds an input u_i, having $\|u_i\| \leq v_{\max}$, compatible with the Underlying Control Law such that Spectral Connectivity Decision Problem returns a value $f_{\text{safe}} \geq 0$ when provided with $[t_0, t_0 + \delta T]$, $[\lambda_-, \lambda_+]$, and u_i.

For clarity, we also present an informal description of Definition 2

[Informal description:] In Definition 2 we are describing the process of, given a procedure to solve Spectral Connectivity Decision Problem, determine whether we can find a set of robot motions close to the motions specified by the underlying control law which are allowed by our solution to Spectral Connectivity Decision Problem. In our case, "close" means "the direction taken by each robot is close in angle to the direction proposed by the underlying control law." This is somewhat like expressing Spectral Connectivity Decision Problem as a function from "angle of motion" to f_{safe}, and searching for roots of f_{safe}.

4 Eigenvalue Games and Information Dissemination

In this section we introduce the main algorithmic components of our solution to the problems presented in Section 3.2. In Section 4.1, we reformulate Spectral Connectivity Decision Problem as a game, termed Graph Picking Game, which can be played with out-of-date information on the state of the network and in Section 4.2 we study the properties of its space of solutions. Next, we present in Section 4.3 a distributed procedure that allows network agents to decide whether an intended motion wins Graph Picking Game. The other algorithmic component of our solution is a distributed information dissemination algorithm, presented in Section 4.4, which provides each robot with the information needed to play Graph Picking Game.

4.1 Graph Picking Game

We are interested in characterizing the rates of change of Laplacian matrices arising from instantaneous robot motions which solve Spectral Connectivity Decision Problem. To do this, we reformulate this problem as a game and study the properties of the solutions to the game.

In order to present a clean formulation, let us introduce some notation. Let

$$\mathrm{LAP}_{\pm}(n) = \{M \in \mathrm{Sym}(n) \mid M\mathbf{1} = \mathbf{0}\},$$

$$\mathrm{LAP}(n) = \{M \in \mathrm{LAP}_{\pm}(n) \mid M_{i,j} \leq 0 \text{ for all } i \neq j\}.$$

Note that, given $M \in \mathrm{LAP}(n)$, it is possible to define a graph, G, with Laplacian M, by assigning to each edge $(i, j) \in \mathcal{E}$ the weight $-M_{i,j}$. We consider the following partial order in $\mathrm{LAP}_{\pm}(n)$. For $A, B \in \mathrm{LAP}_{\pm}(n)$, we write $A <_{\mathrm{LAP}} B$ if and only if $A_{i,j} > B_{i,j}$ for all $i \neq j \in \{1, \ldots n\}$. Likewise, $A \leq_{\mathrm{LAP}} B$ if and only if $A_{i,j} \geq B_{i,j}$ for all $i \neq j \in \{1, \ldots n\}$. For $A \leq_{\mathrm{LAP}} B$, we define the interval

$$[A, B]_{\mathrm{LAP}} = \{L \in \mathrm{LAP}_{\pm}(n) \mid A \leq_{\mathrm{LAP}} L \leq_{\mathrm{LAP}} B\}.$$

Note that $A, B \in \mathrm{LAP}(n)$ and $L \in [A, B]_{\mathrm{LAP}}$ imply $L \in \mathrm{LAP}(n)$. The following result provides more properties of the matrices in the interval $[A, B]_{\mathrm{LAP}}$.

Lemma 2 *Let $A, B \in LAP(n)$ and $L \in [A, B]_{LAP}$. Then,*

(i) $f_{\lambda_2}(L) \in [f_{\lambda_2}(A), f_{\lambda_2}(B)]$,
(ii) $vv^T \bullet L \in [vv^T \bullet A, vv^T \bullet B]$ *for* $v \in \mathbb{R}^n$.

Proof Fact (i) follows from the monotonicity of $\lambda_2(G)$ on the edge weights of G. To prove fact (ii), note that $vv^T \bullet L = v^T L v$ for any $L \in \mathrm{Sym}(n)$ and any $v \in \mathbb{R}^n$. Because any graph Laplacian is positive semidefinite, and $L - A$, $B - L \in \mathrm{LAP}(n)$, we have

$$vv^T \bullet (L - A) = vv^T \bullet L - vv^T \bullet A \geq 0,$$

$$vv^T \bullet (B - L) = vv^T \bullet B - vv^T \bullet L \geq 0,$$

and the result follows. □

We can now formulate the core question we need to solve to obtain a solution to Spectral Connectivity Decision Problem as a game played against a graph-picking opponent.

Definition 3 (Graph Picking Game) Given $A, B \in \mathrm{LAP}(n)$ with $A \leq_{\mathrm{LAP}} B$, we pick $Y \in [A, B]_{\mathrm{LAP}}$. Our opponent then selects $L \in [A, B]_{\mathrm{LAP}}$. We win if either of the following conditions hold

- $f_{\lambda_2}(L) \notin [\lambda_-, \lambda_+]$, or
- $f^{\circ}_{\lambda_2}(L; Y) \geq 0$.

Our objective is to characterize the choices Y that ensure that Graph Picking Game is won (specifically, we characterize the choices X such that all Y having $X \leq_{\mathrm{LAP}} Y$ win Graph Picking Game). This is what we tackle next.

4.2 Bounds on Matrices which Win Graph Picking Game

A direction that a robot can take in physical space induces an instantaneous rate of change of the Laplacian matrix of the underlying communication graph of the network. Given out-of-date information on the state of the network, each robot can produce bounds on the actual Laplacian of the graph. In this section we answer the following question: given a matrix lower bound $A \in \mathrm{LAP}(n)$ and a matrix upper bound $B \in \mathrm{LAP}(n)$ on the Laplacian matrix of the communication graph and a range of possible instantaneous rates of change of the Laplacian matrix due to a proposed physical motion, can we guarantee that the proposed motion will not decrease the second smallest eigenvalue of the graph Laplacian? We do this by answering the related question: given the information listed above, and a range of "unsafe" eigenvalues, $[\lambda_-, \lambda_+]$, can we guarantee the proposed motion will not decrease the second smallest eigenvalue of the Laplacian matrix whenever the said eigenvalue is outside of the range $[\lambda_-, \lambda_+]$?

More formally, we bound the union of all possible gradients of f_{λ_2} evaluated at $L \in [A, B]_{\mathrm{LAP}}$. Consider $L \in [A, B]_{\mathrm{LAP}}$ such that $f_{\lambda_2}(L) \in [\lambda_-, \lambda_+]$. Following the formula for the gradient of f_{λ_2} in Theorem 1, we examine the vectors $w \in \mathbb{S}^n$ such that $Lw = \lambda_2(L)w$. For such vectors, we have $L \bullet (ww^T) = w^T L w = f_{\lambda_2}(L)$, and therefore, using Lemma 2, it follows that $A \bullet (ww^T) \leq \lambda_+$. Our strategy is then to bound the set of $w \in \mathbb{R}^n$ which satisfy $A \bullet (ww^T) \leq \lambda_+$. The fact that the non-smooth gradient of f_{λ_2} is actually the convex closure of such ww^T is addressed in Proposition 1.

Let $\{u_1, \ldots, u_m\}$ be the m eigenvectors of A corresponding to eigenvalues $\lambda_j \leq \lambda_+$ and let $\{u_{m+1}, \ldots, u_n\}$ be the $n-m$ eigenvectors of A corresponding to eigenvalues $\lambda_j > \lambda_+$. Given $\tilde{m} \geq m$, define

$$\epsilon_A(\tilde{m}) = \sqrt{\frac{\lambda_+ - \lambda_2(A)}{\lambda_{\tilde{m}+1}(A) - \lambda_2(A)}},$$

$$\mathbf{u}_{\text{span-}A}(\tilde{m}) = \mathrm{span}\{u_1, \ldots, u_{\tilde{m}}\},$$

$$U_A(\tilde{m}) = \left\{ w \in \mathbb{S}^n \mid \exists u \in \mathbf{u}_{\text{span-}A}(\tilde{m}) \text{ with } \|u\| = 1 \text{ such that } w \in B(u, \epsilon_A(\tilde{m})) \right\}.$$

We have chosen $U_A(\tilde{m})$ to contain the elements w satisfying $ww^T \in \partial f_{\lambda_2}(L)$ for any $\tilde{m} \geq m$, as we show next.

Proposition 1 *Let $A, B \in LAP(n)$, $L \in [A, B]_{LAP}$, $f_{\lambda_2}(L) \leq \lambda_+$, $w \in \mathbb{S}^n$, $\tilde{m} \geq m$. If $w \notin U_A(\tilde{m})$, then $ww^T \notin \partial f_{\lambda_2}(L)$.*

Proof If $w \notin U_A(\tilde{m})$, then the component of w outside $\mathrm{span}(v_1(A), \ldots, v_{\tilde{m}-1}(A))$ has magnitude at least $\epsilon_A(\tilde{m})$. Since $w \in \mathbb{S}^n$, the remaining component has magnitude at most $\sqrt{1 - \epsilon_A^2(\tilde{m})}$, and therefore we can deduce

$$\begin{aligned} w^T A w &> \left(\sqrt{1 - \epsilon_A(\tilde{m})^2}\right)^2 f_{\lambda_2}(A) + \epsilon_A(\tilde{m})^2 f_{\lambda_{\tilde{m}+1}}(A) \\ &= f_{\lambda_2}(A) + \epsilon_A(\tilde{m})^2 (f_{\lambda_{\tilde{m}+1}}(A) - f_{\lambda_2}(A)) = \lambda_+. \end{aligned}$$

Since $L \in [A, B]_{\mathrm{LAP}}$, $w^T L w \geq w^T A w > \lambda_+$. By $\lambda_+ > f_{\lambda_2}(L)$, $w \notin \{v \in \mathbb{S}^n \mid v^T L v = f_{\lambda_2}(L)\}$. To show $ww^T \notin \partial f_{\lambda_2}(L)$ we recall that $\partial f_{\lambda_2}(L) = \mathrm{co}_{\{v \in \mathbb{S}^n \mid Lv = f_{\lambda_2}(L)v\}}\{vv^T\} =$

$\text{co}_{\{v\in\mathbb{S}^n \mid v^T L v = f_{\lambda_2}(L)\}}\{vv^T\}$. Noting that there is only one combination of vectors in $\{vv^T \mid v \in \mathbb{S}^n\}$ which can have a convex combination of ww^T for $w \in \mathbb{S}^n$, namely the singleton set $\{ww^T\}$, we deduce that $w \notin \{v \in \mathbb{S}^n \mid v^T L v = f_{\lambda_2}(L)\}$ implies $ww^T \notin \text{co}_{\{v\in\mathbb{S}^n \mid v^T L v = f_{\lambda_2}(L)\}}\{vv^T\} = \partial f_{\lambda_2}(L)$. □

The bound induced by $U_A(\tilde{m})$ works for any $\tilde{m} \geq m$. Our idea is to check the bounds induced by all such $\tilde{m} \geq m$, in the hope of finding one which verifies that our proposed motion is allowable.

The following result is a consequence of Proposition 1 and Theorems 1 and 2.

Corollary 2 *Any instantaneous change in robot positions, $(u_i)_{i\in\{1,\ldots,n\}}$, inducing an instantaneous rate of change of the Laplacian $Y \in LAP_{\pm}(n)$ satisfying $Y \bullet (uu^T) \geq 0$ for all $u \in U_A(\tilde{m})$ for some $\tilde{m} \geq m$, satisfies $f^{\circ}_{2-conn}(\mathcal{P}; (u_i)_{i\in\{1,\ldots,n\}}) \geq 0$.*

Given some $\tilde{m} \geq m$, we can conclude from Proposition 1 and Corollary 2 that any Y satisfying $Y \bullet (ww^T) \geq 0$ for all $w \in U_A(\tilde{m})$ wins Graph Picking Game on $A, B, \lambda_-, \lambda_+$. To determine whether a given Y satisfies this property, it is sufficient to find the vector $u_{\min} \in \mathbf{u}_{\text{span-}A}(\tilde{m})$ which minimizes $Y \bullet (uu^T) = u^T Y u$ (and then tack a fudge factor based on $\epsilon_A(\tilde{m})$ onto this minimum). We justify this in the following.

Let $M_{u(\tilde{m})} \in \mathbb{R}^{n\times\tilde{m}}$ be a matrix whose column vectors are an orthonormal basis of $\mathbf{u}_{\text{span-}A}(\tilde{m})$. Note that any vector in $\mathbf{u}_{\text{span-}A}(\tilde{m}) \cap \mathbb{S}^n$ can be expressed in the form $M_{u(\tilde{m})}x$ for some $x \in \mathbb{S}^{\tilde{m}}$. Likewise any $x \in \mathbb{S}^{\tilde{m}}$ satisfies $M_{u(\tilde{m})}x \in \mathbf{u}_{\text{span-}A}(\tilde{m}) \cap \mathbb{S}^n$.

Proposition 2 *Finding the vector $u \in \mathbf{u}_{span\text{-}A}(\tilde{m}) \cap \mathbb{S}^n$ which minimizes $Y \bullet (uu^T)$ is equivalent to finding the vector $x \in \mathbb{S}^{\tilde{m}}$ which minimizes $x^T M^T_{u(\tilde{m})} Y M_{u(\tilde{m})} x$. Since $M^T_{u(\tilde{m})} Y M_{u(\tilde{m})}$ is symmetric, minimization is achieved when $u^T Y u$ equals the smallest eigenvalue of $M^T_{u(\tilde{m})} Y M_{u(\tilde{m})}$.*

Proof Let $u_{\min}$ be the vector in $\mathbf{u}_{\text{span-}A}(\tilde{m}) \cap \mathbb{S}^n$ that minimizes $u^T Y u$. Since there exists $x_{\min} \in \mathbb{S}^{\tilde{m}}$ having $M_{u(\tilde{m})}x_{\min} = u_{\min}$, we have $x^T_{\min} M^T_{u(\tilde{m})} Y M_{u(\tilde{m})} x_{\min} \leq \min_{u\in\mathbf{u}_{\text{span-}A}(\tilde{m})\cap\mathbb{S}^n}(u^T Y u)$, and hence $\min_{\lambda\in\text{eigs}(M^T_{u(\tilde{m})} Y M_{u(\tilde{m})})}(\lambda) \leq u^T_{\min} Y u_{\min}$. Since each $x \in \mathbb{S}^{\tilde{m}}$ satisfies $M_{u(\tilde{m})}x \in \mathbf{u}_{\text{span-}A}(m) \cap \mathbb{S}^n$, then $x^T M^T_{u(\tilde{m})} Y M_{u(\tilde{m})} x \geq u^T_{\min} Y u_{\min}$ for all $x \in \mathbb{S}^{\tilde{m}}$. Thus $\min_{\lambda\in\text{eigs}(M^T_{u(\tilde{m})} Y M_{u(\tilde{m})})}(\lambda) \geq u^T_{\min} Y u_{\min}$. We conclude that $\min_{\lambda\in\text{eigs}(M^T_{u(\tilde{m})} Y M_{u(\tilde{m})})}(\lambda) = u^T_{\min} Y u_{\min}$. □

The next results provides a sufficient criterion to check if a matrix is a winning solution to Graph Picking Game.

Proposition 3 $(1 - \epsilon_A(\tilde{m})^2) Y \bullet (uu^T) + \epsilon_A(\tilde{m})^2 \min(\min(eigs(Y)), 0) \geq 0$ *for all* $u \in \mathbf{u}_{span\text{-}A}(\tilde{m})$ *implies that* $Y \bullet (ww^T) \geq 0$ *for all* $w \in U_A(\tilde{m})$.

Proof Any $w \in U_A(\tilde{m})$ can be decomposed into $\alpha u + \sqrt{1-\alpha^2}v$ for $u \in \mathbf{u}_{\text{span-}A}(\tilde{m})$ and $v \in \text{complement}(\mathbf{u}_{\text{span-}A}(\tilde{m}))$ where $\sqrt{1-\alpha^2} \leq \epsilon_A(\tilde{m})$. Since Proposition 2 gives us $Y \bullet (vv^T) \geq \min(\text{eigs}(Y))$, we have $Y \bullet (\sqrt{1-\alpha^2}^2 vv^T) \geq \epsilon^2_A(\tilde{m}) \min(\text{eigs}(Y))$ if $\min(\text{eigs}(Y)) \leq 0$ and $Y \bullet (\sqrt{1-\alpha^2}^2 vv^T) \geq \min(\text{eigs}(Y)) \geq 0$ if $\min(\text{eigs}(Y)) \geq 0$. Thus

$Y \bullet (\sqrt{1-\alpha^2}^2 vv^T) \geq \epsilon_A^2(\tilde{m}) \min(\min(\text{eigs}(Y)), 0)$ and $Y \bullet (ww^T) \geq (1-\epsilon_A(\tilde{m})^2) Y \bullet (uu^T) + \epsilon_A(\tilde{m})^2 \min(\min(\text{eigs}(Y)), 0) \geq 0$. □

4.3 Direction Checking Algorithm

We introduce Direction Checking Algorithm in Table 1. Given $A, B \in \text{LAP}(n)$, and a lower bound, $X \in \text{LAP}_{\pm}(n)$ of the candidate instantaneous rate of change of the Laplacian matrix, $Y \in \text{LAP}_{\pm}(n)$, $Y \geq_{\text{LAP}} X$, the algorithm returns a value $S_{\text{check}} \geq 0$ if it can verify that any $Y \geq_{\text{LAP}} X$ wins Graph Picking Game on A and B, and returns $S_{\text{check}} < 0$ otherwise.

Lemma 3 Direction Checking Algorithm *returns* $S_{check} \geq 0$ *only if* X *satisfies* $X \bullet M \geq 0$ *for every* $M \in \partial f_{\lambda_2}(L)$ *with* $L \in [A, B]_{LAP}$.

Proof If Direction Checking Algorithm returns $S_{\text{check}} \geq 0$, Proposition 2 implies that there must be some $\tilde{m} \geq m$ having $\min_{\{ww^T \mid w \in B(u, \epsilon_A(\tilde{m})), u \in \mathbb{S}^n \cap \mathbf{u}_{\text{span-}A}(\tilde{m})\}}(X \bullet (uu^T)) \geq 0$ thus, by Proposition 3, $X \bullet ww^T \geq 0$ for all $w \in U_A(\tilde{m})$ and, by

Table 1 Direction checking algorithm

Name:	Direction Checking Algorithm
Goal:	Let Y be the (unknown) instantaneous rate of change of the Laplacian matrix of the communication graph of a robotic network. Given X (known) such that $Y - X$ is known to be positive semidefinite, determine whether Y can be proved to win Graph Picking Game on A and B and eigenvalue bounds λ_- and λ_+
Inputs:	• Matrices $A, B \in \text{LAP}(n)$ • Eigenvalue bounds $\lambda_- \leq \lambda_+ \in \mathbb{R}$ • Lower bound, $X \in \text{LAP}_{\pm}(n)$, on candidate direction in matrix space, $Y \in \text{LAP}_{\pm}(n)$
Outputs:	$S_{\text{check}} \in \mathbb{R}$. $S_{\text{check}} \geq 0$ means each $Y \geq_{\text{LAP}} X$ wins Graph Picking Game on A, B and $[\lambda_-, \lambda_+]$

Let $\lambda_+ \leftarrow \min(\lambda_+, \lambda_2(B))$
Let $\lambda_- \leftarrow \max(\lambda_-, \lambda_2(A))$
if $\lambda_- > \lambda_+$ **then**
  return 0
end if
Let $\lambda_{\min} \leftarrow \min(\text{eigs}(X))$
Let $m_{\min} \leftarrow \min\{m \mid \lambda_m \in \text{eigs}(A),\ \lambda_m > \lambda_+\}$
Initialize $S_{\text{check}} \leftarrow -1$.
for all $\tilde{m} \in \{m_{\min} - 1, \ldots, n\}$ **do**
  if $\tilde{m} < n$ **then**
    Let $\epsilon_A(\tilde{m}) \leftarrow \sqrt{\frac{\lambda_+ - \lambda_2(A)}{\lambda_{\tilde{m}+1}(A) - \lambda_2(A)}}$ and $\mathbf{u}_{\text{span-}A}(\tilde{m}) \leftarrow \text{span}(u_j,\ j \in \{1, \ldots, m\})$
  else
    Let $\epsilon_A(\tilde{m}) \leftarrow 0$
  end if
  Let $M_{u(\tilde{m})} \in \mathbb{R}^{n \times \tilde{m}}$ whose columns are orthogonal basis of $\mathbf{u}_{\text{span-}A}(\tilde{m})$
  Let $S \leftarrow (1 - \epsilon_A(\tilde{m})^2) \min\left(\text{eigs}\left(M_{u(\tilde{m})}^T X M_{u(\tilde{m})}\right)\right) + \epsilon_A(\tilde{m})^2 \min(\lambda_{\min}, 0)$ /*Does current $\tilde{m}$ verify X is safe?*/
  Let $S_{\text{check}} \leftarrow \max(S, S_{\text{check}})$ /*Does any $\tilde{m}$ checked so far verify X is safe?*/
end for
return S_{check} /*Does any $\tilde{m}$ verify X is safe?*/

Proposition 1, $X \bullet ww^T \geq 0$ for all $w \in \mathbb{S}^n$ having $ww^T \in \partial f_{\lambda_2}(L)$. Since any $M \in \partial f_{\lambda_2}(L)$ is the convex combination of some set of $ww^T \in \partial f_{\lambda_2}(L)$, $X \bullet M$ must be the sum of $X \bullet (ww^T)$ for such a set of ww^T and thus $X \bullet M \geq 0$. □

The following result shows that Direction Checking Algorithm is successful in determining if we win Graph Picking Game.

Theorem 3 Direction Checking Algorithm *returns* $S_{check} \geq 0$ *only when each* Y *having* $Y \geq_{LAP} X$ *satisfies* $Y \bullet M \geq 0$ *for* $M \in \partial f_{\lambda_2}(L)$ *with* $L \in [A, B]_{LAP}$.

Proof The conditions $S_{\text{check}} \geq 0$ holds only if

$$\min(\text{eigs}(M_{u(\tilde{m})}^T X M_{u(\tilde{m})})) + \epsilon_A(\tilde{m}) \min(\lambda_{\min}(X), 0) \geq 0,$$

for some $\tilde{m} > m$. For any Y having $Y \geq_{\text{LAP}} X$, $Y - X$ is positive semidefinite, thus each eigenvalue of Y is greater than or equal to the corresponding eigenvector of X, making $\lambda_{\min}(Y) > \lambda_{\min}(X)$. Likewise, we can express $M_{u(\tilde{m})}^T Y M_{u(\tilde{m})}$ as $M_{u(\tilde{m})}^T X M_{u(\tilde{m})} + M_{u(\tilde{m})}^T (Y - X) M_{u(\tilde{m})}$ where $Y - X$, and therefore $M_{u(\tilde{m})}^T (Y - X) M_{u(\tilde{m})}$ as well, are each positive semidefinite. This means that each eigenvalue of $M_{u(\tilde{m})}^T Y M_{u(\tilde{m})} = M_{u(\tilde{m})}^T X M_{u(\tilde{m})} + M_{u(\tilde{m})}^T (Y - X) M_{u(\tilde{m})}$ is greater than or equal to the corresponding eigenvalue of $M_{u(\tilde{m})}^T X M_{u(\tilde{m})}$. So $\min(\text{eigs}(M_{u(\tilde{m})}^T X M_{u(\tilde{m})})) + \epsilon_A(\tilde{m}) \min(\lambda_{\min}(X), 0) \leq \min(\text{eigs}(M_{u(\tilde{m})}^T Y M_{u(\tilde{m})})) + \epsilon_A(\tilde{m}) \min(\lambda_{\min}(Y), 0)$, thus $S_{\text{check}} > 0$ only if

$$\min\left(\text{eigs}\left(M_{u(\tilde{m})}^T Y M_{u(\tilde{m})}\right)\right) + \epsilon_A(\tilde{m}) \min(\lambda_{\min}(Y), 0) > 0,$$

for some $\tilde{m} > m$. By Lemma 3 this holds only if Y satisfies $Y \bullet M \geq 0$ for $M \in \partial f_{\lambda_2}(L)$ with $L \in [A, B]_{\text{LAP}}$. □

4.4 Information Dissemination of Robot Positions

In order to execute Direction Checking Algorithm, robots first need information about the past states of the network to come up with reasonable bounds on the Laplacian matrix. Before specifying the protocol to disseminate information about each node throughout the network, we first address what it means for each node to hold information which is consistent with the real world. A formal definition is given next.

Definition 4 (Consistency of stored network information) Let $\mathcal{P}_{\text{truth}} \in \mathbb{R}^{n \times n}$ be the actual position of the robots at time t_{curr}, and let $v_{\max}$ be a bound on the maximum velocity of each individual robot. A tuple, $(\mathcal{P}, T, D)$, $\mathcal{P} \in \mathbb{R}^{d \times n}$, $T \in \mathbb{R}^n$, $D \in \mathbb{R}^{n \times n}$, is called consistent with $\mathcal{P}_{\text{truth}}$ at time t_{curr} if the following hold:

(i) For each $i \in \{1, \ldots, n\}$, $\mathcal{P}_i \in B(\mathcal{P}_{\text{truth}i}, (t_{\text{curr}} - T_i) v_{\max})$.
(ii) For each $i, j \in \{1, \ldots, n\} \times \{1, \ldots, n\}$, $\|\mathcal{P}_{\text{truth}i} - \mathcal{P}_{\text{truth}j}\| \in [D_{i,j} - v_{\max}(t_{\text{curr}} - T_i + t_{\text{curr}} - T_j),\ D_{i,j} + v_{\max}(t_{\text{curr}} - T_i + t_{\text{curr}} - T_j)]$.

In other words, a set of information is consistent with an actual state of the world, (i) if the position of each robot, i, in the actual state of the world is within the range it could have reached by traveling with speed $v_{\max}$ starting from position $\mathcal{P}_i$ for time

$t_{\text{curr}} - T_i$ and (ii) $D_{i,j}$ stores the distance between $\mathcal{P}_i$ and $\mathcal{P}_j$ and relates to the actual distance in the natural way.

Next we provide an algorithm which correctly disseminates CONSISTENT information on the state of the network to all robots. We start with the bookkeeping data necessary for this task. Each robot $i \in \{1, \ldots, n\}$ holds the following data structures

- For each other robot, j, a position, $\mathcal{P}_j^{[i]} \in \mathbb{R}^d$ and a time $T_j^{[i]} \in \mathbb{R}$ since it was last known that the position of j was $\mathcal{P}_j^{[i]}$.
- For each other robot, j, $S_{k,j}(i) \in \{\text{true}, \text{false}\}$ is true if and only if the most recent copy of $\left(\mathcal{P}_k^{[i]}, T_k^{[i]}\right)$ needs to be sent from i to j.

The ALL-TO-ALL BROADCAST ALGORITHM is described in Table 2.

Table 2 All-to-all broadcast algorithm

Name:	ALL-TO-ALL BROADCAST ALGORITHM
Goal:	Disseminate information about robot positions throughout the network
Inputs:	• $t_{\text{curr}} \in \mathbb{R}$ indicating the current time
Messages from neighbors:	• $j \in \{1, \ldots, n\}$ is identifier of the robot from which the signal originated • $t_{\text{valid}} \in \mathbb{R}$ indicating the time at which message from j originated • $P_j \in \mathbb{R}^d$, the position of j at time t_{valid}
Sensor Data	• $P_{\text{id}} \in \mathbb{R}^d$, current position of this robot, acquired via sensing
Persistent data:	• id $\in \{1, \ldots, n\}$, current robot's unique identifier • $T \in \mathbb{R}^n$, array of last recorded time information • $\mathcal{P} \in \mathbb{R}^{d \times n}$, array of last recorded position information • $S \in \{\text{true}, \text{false}\}^{n \times n}$ where $S_{i,j}$ indicates whether the most up-to-date information about i needs to be sent to j (value true) or not (value false) • $D \in \mathbb{R}^{n \times n}$, matrix of approximate inter-robot distances

Let $\mathcal{P}_{\text{id}} \leftarrow P_{\text{id}}$ and $T_{\text{id}} \leftarrow t_{\text{curr}}$ /*Update self*/
for all $m \in \{1, \ldots, n\} \setminus \{\text{id}\}$ **do**
  Let $S_{\text{id},m} \leftarrow$ true /*Sending phase*/
end for
for all $k \in \mathcal{N}$ **do**
  Randomly select j from elements having $S_{j,k} =$ true
  Push $(j, T_j, \mathcal{P}_j)$ onto the queue of items to be sent to agent k
  Push (id, $T_{\text{id}}, \mathcal{P}_{\text{id}}$) onto the queue of items to be sent to agent k
end for
Send two items from each queue to corresponding destination
for all $k \in \mathcal{N}$ **do**
  for all message $\{1, 2\}$ /*Each neighbor sent two messages previously*/ **do**
    Receive j, t_{valid}, P_j from k. /*Receiving phase*/
    if $t_{\text{valid}} > T_j$ **then**
      for all $m \in \{1, \ldots, n\} \setminus \{\text{id}\}$ **do**
        Let $S_{j,m} \leftarrow$ true
      end for
      Let $S_{j,k} \leftarrow$ false, $T_j \leftarrow t_{\text{valid}}$, and $\mathcal{P}_j \leftarrow P_j$
      for all $i \in \{1, \ldots, n\} \setminus \{j\}$ **do**
        Let $D_{i,j} \leftarrow \|\mathcal{P}_i - \mathcal{P}_j\|$ and $D_{j,i} \leftarrow \|\mathcal{P}_i - \mathcal{P}_j\|$
      end for
    end if
  end for
end for

Note that, in lines 7:-8: of ALL-TO-ALL BROADCAST ALGORITHM, each robot pushes its own updated information in its queue, as well as that of a randomly selected robot, at each communication round. While this is not necessary for our proof, in practice it leads to much better bounds on the instantaneous rates of change of the Laplacian due to individual robot motions, which primarily depend on accurate information on each agent's immediate neighbors.

Because this algorithm is randomized, we discuss its expected performance. In the following result, let $\text{path}_{j,k}(t)$ denote the shortest path between robots j and k at time t.

Lemma 4 *For $j \in \{1,\ldots,n\}$, let $k \in \{1,\ldots,n\}$ be the robot that maximizes $t_{curr} - T_j^{[k]}$ at time t_{curr}, in other words, k's estimate of j is the most out of date estimate on the network. If the communication graph is connected between rounds at time t and $t + \delta T$, then the expectation of $\sum_{i \in path_{j,k}(t)} T_j^{[i]}$, increases by at least $\frac{1}{n}\left|T_j^{[k]} - T_j^{[j]}\right|$ between t and $t + \delta T$ for any $j \in \{1,\ldots,n\}$. Likewise, if t_{curr} is the current time, the expectation of $\sum_{i \in path_{j,k}(t)}(t_{curr} - T_j^{[i]})$ (the sum of the amount by which the timestamps for j are out of date along the path from j to k) decreases by at least $\frac{1}{n}\max_{k,l\in\{1,\ldots,n\}}\left(\left|T_j^{[k]} - T_j^{[l]}\right|\right) - |path_{j,k}(t)|\delta T$ between t and $t + \delta T$ for any $j \in \{1,\ldots,n\}$.*

Proof Let i_{less} and i_{greater} be the two agent identities adjacent to an edge, e, having $T_j^{[i_{\text{less}}]} \leq T_j^{[i_{\text{greater}}]}$. With probability at least $\frac{1}{n}$ agent i_{greater} will broadcast its estimate of j to agent i_{less} increasing $T_j^{[i_{\text{less}}]}$ by $\left|T_j^{[i_{\text{greater}}]} - T_j^{[i_{\text{less}}]}\right|$. Broadcasting j the other direction does not affect any agents state estimate. Since $\mathcal{G}(\mathcal{P}(T))$ is connected for $T \in [t, t+\delta T]$, there must be at least one path between the agents with the least and greatest value of $T_j^{[i]}$, and a set of edges must exist along this path for which the sum of the differences of $T_j^{[i_{\text{greater}}]} - T_j^{[i_{\text{less}}]}$ must exceed $\max_{k\in\{1,\ldots,n\}}\left(\left|T_j^{[k]} - T_j^{[j]}\right|\right)$. The second half of the statement follows from the first. □

The next result characterizes the expected time by which the information held by each node may be out of date.

Corollary 3 *For any robot, $j \in \{1,\ldots,n\}$, the average expectation, over all robots, $i \in \{1,\ldots,n\}$ of $(t_{curr} - T_j^{[i]})$ never exceeds $(n^2+1)\delta T$. Likewise the expected maximum, over all $i \in \{1,\ldots,n\}$ of $t_{curr} - T_j^{[i]}$ never exceeds $n(n^2+1)\delta T$, and the expected maximum, over all i, $j \in \{1,\ldots,n\}$ of $t_{curr} - T_j^{[i]}$ never exceeds $n(n^2+1)$.*

Proof By Lemma 4, $\left(\frac{1}{n}\sum_{i\in 1,\ldots n}\left(t_{\text{curr}} - T_j^{[i]}\right)\right)$ decreases whenever

$$\frac{1}{n}\max_{k,l\in\{1,\ldots,n\}}\left|T_j^{[k]} - T_j^{[l]}\right| - n\delta T > 0.$$

Since at least one node (j) has the fresh value of j $\left(T_j^{[j]} = t_{\text{curr}}\right)$, the expectation of $\max_{k,l\in\{1,\ldots,n\}}\left(\left|T_j^{[k]} - T_j^{[l]}\right|\right)$ is at least the average (over i) of the expectation of $t_{\text{curr}} - T_j^{[i]}$ and the average expectation of $t_{\text{curr}} - T_j^{[i]}$ decreases whenever the expectation of $\max_{k,l\in\{1,\ldots,n\}}\left(\left|T_j^{[k]} - T_j^{[l]}\right|\right)$ is above $n^2\delta T$. The expectation of the average over i

of $t_{\text{curr}} - T_j^{[i]}$, being a lower bound of $\max_{k,l\in\{1,\ldots,n\}} \left(\left|T_j^{[k]} - T_j^{[l]}\right|\right)$, must decrease if it exceeds $n^2\delta T$. This quantity goes up by at most δT each round and thus never exceeds $n^2\delta T + \delta T$. Since the expectation of $t_{\text{curr}} - T_j^{[i]}$ is never below 0, the expectation of the maximum must be at most n times the average, thus it never exceeds $n(n^2+1)\delta T$.

For the last part of the statement, the average over all i of the expectation of $t_{\text{curr}} - T_j^{[i]}$ never exceeds $(n^2+1)\delta T$ for any fixed j, and so the average over all i, j of the expectation of $t_{\text{curr}} - T_j^{[i]}$ also never exceeds $(n^2+1)\delta T$. Since the maximum over all $i, j \in \{1,\ldots,n\}$ of $t_{\text{curr}} - T_j^{[i]}$ is at most n^2 times the average over all i, j of $t_{\text{curr}} - T_j^{[i]}$, this value never exceeds $n^2(n^2+1)\delta T$. □

Finally, we establish that the information stored by the network is consistent with the actual robot positions in the sense of Definition 4.

Theorem 4 *Assume each robot moves with velocity at most v_{max}. At all times, each robot holds values of T, $\mathcal{P}$, D which are* CONSISTENT*, in the sense of Definition* 4*, with the state of the network at time t_{curr}.*

Proof Each broadcast message, as sent in ALL-TO-ALL BROADCAST ALGORITHM, contains a position of a robot, and the time at which that position was valid. Since the time and the position are both stored, the results always verify condition 1 of Definition 4. Condition 2 of Definition 4 holds for any $(\mathcal{P}, T, D)$ where condition 1 holds for $\mathcal{P}$ and T, and $D_{i,j} = \|\mathcal{P}_i - \mathcal{P}_j\|$ for all i and j. □

5 Algorithmic Solutions to the Connectivity Problems

In this section, we combine the algorithmic procedures developed in Section 4 to decide if a proposed network motion is safe for connectivity maintenance and to disseminate position information across the network. This allows us to synthesize the MOTION TEST ALGORITHM and the MOTION PROJECTION ALGORITHM to solve the SPECTRAL CONNECTIVITY DECISION PROBLEM and the SPECTRAL CONNECTIVITY PROBLEM, respectively.

5.1 MOTION TEST ALGORITHM

Here we combine the DIRECTION CHECKING ALGORITHM and the ALL-TO-ALL BROADCAST ALGORITHM to synthesize a motion coordination algorithm that solves the SPECTRAL CONNECTIVITY DECISION PROBLEM. First, we provide the MATRIX BOUND GENERATOR in Table 3 to compute lower $A \in \text{LAP}(n)$ and upper $B \in \text{LAP}(n)$ bounds on the Laplacian matrix of the communication graph from the data disseminated via the ALL-TO-ALL BROADCAST ALGORITHM.

Lemma 5 *Let t_{curr} be the current time. Given a proximity graph induced by a monotonic function $g_{wgt} : \mathbb{R} \to \mathbb{R}$, for $(\mathcal{P}, T, D)$* CONSISTENT *with the current set of robot positions,* MATRIX BOUND GENERATOR *returns two matrices which bound the Laplacian of the graph induced by $\mathcal{P}_{truth}$ at any time between t_{curr} and $t_{curr} + \delta T$.*

Table 3 Matrix bound generator

Name:	MATRIX BOUND GENERATOR
Goal:	Compute bounds $A, B \in \mathrm{LAP}(n)$ of Laplacian of communication graph
Inputs:	• Current time $t_{\mathrm{curr}} \in \mathbb{R}$ • Maximum velocity of any robot, $v_{\max}$ • Maximum time between communication rounds, δT
Persistent data:	• $T \in \mathbb{R}^n$, array of last recorded time information • $\mathcal{P} \in \mathbb{R}^{d\times n}$, array of last recorded position information • $D \in \mathbb{R}^{n\times n}$, matrix of approximate inter-robot distances
Outputs:	$A, B \in \mathrm{LAP}(n)$, are the matrix bounds

1: **for all** $i \in \{1,\ldots,n\}$ **do**
2: **for all** $j \in \{1,\ldots,n\}$ **do**
3: Let $A_{i,j} \leftarrow g_{\mathrm{wgt}}(D_{i,j} - v_{\max}((t_{\mathrm{curr}} - T_i + \delta T) + (t_{\mathrm{curr}} - T_j + \delta T)))$
4: Let $B_{i,j} \leftarrow g_{\mathrm{wgt}}(D_{i,j} + v_{\max}((t_{\mathrm{curr}} - T_i + \delta T) + (t_{\mathrm{curr}} - T_j + \delta T)))$
5: **end for**
6: **end for**
7: Let $A \leftarrow \mathrm{diag}(\mathbf{1}^T A) - A$
8: Let $B \leftarrow \mathrm{diag}(\mathbf{1}^T B) - B$

Proof Let $t \in [t_{\mathrm{curr}}, t_{\mathrm{curr}} + \delta T]$. By monotonicity of g_{wgt} and consistency of $(\mathcal{P}, T, D)$, the distance between i and j is between $D_{i,j} - v_{\max}((t_{\mathrm{curr}} - T_i + \delta T) + (t_{\mathrm{curr}} - T_j + \delta T))$ and $D_{i,j} + v_{\max}((t_{\mathrm{curr}} - T_i) + (t_{\mathrm{curr}} - T_j + \delta T))$, yielding the (i, j)-th element of the adjacency matrix in the interval $[g_{\mathrm{wgt}}(D_{i,j} - v_{\max}((t_{\mathrm{curr}} - T_i + \delta T) + (t_{\mathrm{curr}} - T_j + \delta T))), g_{\mathrm{wgt}}(D_{i,j} - v_{\max}((t_{\mathrm{curr}} - T_i + \delta T) + (t_{\mathrm{curr}} - T_j + \delta T)))]$. Since all the off-diagonal elements of the Laplacian matrices, A and B, are defined this way, and the diagonal elements are CONSISTENT with the definition of $\mathrm{LAP}_{\pm}(n)$, the actual Laplacian matrix of the graph is in $[A, B]_{\mathrm{LAP}}$ for all $t \in [t_{\mathrm{curr}}, t_{\mathrm{curr}} + \delta T]$. □

Next, we characterize the expected gap between the off-diagonal elements of A and B generated by MATRIX BOUND GENERATOR.

Lemma 6 *Let* $g_{max} = \max_{x\in\mathbb{R}_{\geq 0}}(|g'_{wgt}(x)|)$. *For* $i \neq j \in \{1,\ldots,n\}$, *the expected gap,* $B_{i,j} - A_{i,j}$ *never exceeds* $4g_{max}v_{max}(n(n^2+2)\delta T)$. *The expected maximum over* $i \neq j, i \in \{1,\ldots,n\}, j \in \{1,\ldots,n\}$ $B_{i,j} - A_{i,j}$ *never exceeds* $4g_{max}v_{max}(n^2(n^2+2)\delta T)$.

Proof The expectations of $t_{\mathrm{curr}} - T_i$ and $t_{\mathrm{curr}} - T_j$ never exceed $(n(n^2+1)\delta T)$ by Corollary 3. Since $A_{i,j} = g_{\mathrm{wgt}}(D_{i,j} - v_{\max}((t_{\mathrm{curr}} - T_i + \delta T) + (t_{\mathrm{curr}} - T_j + \delta T)))$ and $B_{i,j} = g_{\mathrm{wgt}}(D_{i,j} - v_{\max}((t_{\mathrm{curr}} - T_i + \delta T) + (t_{\mathrm{curr}} - T_j + \delta T)))$, their difference never exceeds $2g_{\max}v_{\max}(n(n^2+1+1)\delta T + n(n^2+1+1)\delta T)$.

By similar reasoning, the expected maximum over all $i \neq j$ of $B_{i,j} - A_{i,j}$ never exceeds $4g_{\max}v_{\max}(n^2(n^2+1+1)\delta T)$ since this is at most twice $2g_{\max}v_{\max}$ times $\max_{i\in\{1,\ldots,n\}}(t_{\mathrm{curr}} - T_i + \delta T)$. □

Finally, we combine the DIRECTION CHECKING ALGORITHM which verifies winning solutions to GRAPH PICKING GAME with the ALL-TO-ALL BROADCAST ALGORITHM and the MATRIX BOUND GENERATOR which provide position information to the network robots. This combination allows us to synthesize a solution to SPECTRAL CONNECTIVITY DECISION PROBLEM. This solution, MOTION TEST ALGORITHM, is presented in Table 4.

Table 4 Motion test algorithm

Name:	Motion Test Algorithm
Goal:	Solve Spectral Connectivity Decision Problem.
Inputs:	• Current time $t_{\text{curr}} \in \mathbb{R}$ • Maximum velocity of any robot, $v_{\max}$ • Maximum time between communication rounds, δT • Proposed direction of motion, v • Eigenvalue bounds $\lambda_- \leq \lambda_+ \in \mathbb{R}$
Persistent data:	• $T \in \mathbb{R}^n$, an array of last recorded time information • $\mathcal{P} \in \mathbb{R}^{d\times n}$, an array of last recorded position information • $D \in \mathbb{R}^{n\times n}$, a matrix of rough inter-robot distances • $A, B \in \text{LAP}(n)$ • id $\in \{1, \ldots, n\}$, unique identifier of current robot
Outputs:	• $f_{\text{safe}} \in \mathbb{R}$ such that $f_{\text{safe}} \geq 0$ if, for any time $t \in [t_{\text{curr}}, t_{\text{curr}} + \delta T]$, the instantaneous change in the Laplacian matrix due to motion in the direction v wins Graph Picking Game at time t

1: Initialize $X_{\text{upper}} \leftarrow \mathbf{0}$
2: Initialize $X_{\text{lower}} \leftarrow \mathbf{0}$
3: **for all** $i \in \{1, \ldots, n\}$ **do**
4: $\quad X_{\text{lower id},i} \leftarrow -\min_{p\in B(\mathcal{P}_i, v_{\max}(t-T_i+\delta T))}(g'_{\text{wgt}}(p, \mathcal{P}_{\text{id}}; v, \mathbf{0}))$ /**Compute bounds on direction matrix**/
5: $\quad$ Let $X_{\text{upper id},i} \leftarrow -\max_{p\in B(\mathcal{P}_i, v_{\max}(t-T_i+\delta T))} g'_{\text{wgt}}(p, \mathcal{P}_{\text{id}}; v, \mathbf{0})$
6: $\quad$ Let $X_{\text{upper id,id}} \leftarrow X_{\text{upper id,id}} - X_{\text{upper id},i}$
7: $\quad$ Let $X_{\text{lower id,id}} \leftarrow X_{\text{lower id,id}} - X_{\text{lower id},i}$
8: **end for**
9: Let $\lambda_- \leftarrow \max(\lambda_-, \lambda_2(A))$
10: Let $\lambda_+ \leftarrow \min(\lambda_+, \lambda_2(B))$
11: **if** $\lambda_- \geq \lambda_+$ **then**
12: $\quad$ return 0 /**There are no possible matrices with eigenvalues in the disallowed range**/
13: **end if**
14: Let $f_{\text{safe}} \leftarrow$ Direction Checking Algorithm on $A, B, X_{\text{lower}}, \lambda_-, \lambda_+$
15: return f_{safe}

The next result shows that Motion Test Algorithm returns a value of $f_{\text{safe}} \geq 0$ only if the instantaneous change in the Laplacian due to motion in direction v wins Graph Picking Game.

Theorem 5 *Assuming that each robot moves with velocity at most v_{max},* Motion Test Algorithm *solves* Spectral Connectivity Decision Problem.

Proof By Theorem 4 the information received by each robot is consistent with the state of the network. Lemma 5 shows, given consistent data, that the matrices A and B properly bound the graph Laplacian of the communication graph. The motion in matrix space induced by the proposed instantaneous motion in $\mathbb{R}^{d\times n}$ is bounded from below by X_{lower}, i.e., it is a member of $\{Y \in \text{LAP}_{\pm}(n) \mid Y \geq_{\text{LAP}} X_{\text{lower}}\}$. Finally Theorem 3 and Corollary 2 show that Direction Checking Algorithm called on line `14:` of Motion Test Algorithm returns $f_{\text{safe}} \leq 0$ only when the proposed direction of motion wins Graph Picking Game, and hence is allowable under Spectral Connectivity Decision Problem. □

The next result shows that solutions to the Spectral Connectivity Decision Problem keep the algebraic connectivity of the network above the desired threshold.

Corollary 4 *If each robot runs an algorithm which solves* Spectral Connectivity Decision Problem*, then the algebraic connectivity* λ_2 *of the network never drops below* λ_+.

Proof Since each robot's individual motion solves Spectral Connectivity Decision Problem, whenever $\lambda_2 \leq \lambda_+$, we know $f^{\circ}_{2-\text{conn}}(\mathcal{P}(t);[\mathbf{0},\ldots,u_i^T,\ldots,\mathbf{0}]^T) \geq 0$, for $i \in \{1,\ldots,n\}$, for all times t. Since $[u_1,\ldots,u_n]^T = \sum_{i=1}^n [\mathbf{0},\ldots,u_i^T,\ldots,\mathbf{0}]^T$, by [3, Proposition 2.3.3],

$$f^{\circ}_{2-\text{conn}}\left(\mathcal{P}(t);[u_1,\ldots,u_n]^T\right) \subseteq \left[\sum_{i=1}^{n} \min\left(f^{\circ}_{2-\text{conn}}\left(\mathcal{P}(t);\left[\mathbf{0},\ldots,u_i^T,\ldots,\mathbf{0}\right]^T\right)\right), \times \sum_{i=1}^{n} \max\left(f^{\circ}_{2-\text{conn}}\left(\mathcal{P}(t);\left[\mathbf{0},\ldots,u_i^T,\ldots,\mathbf{0}\right]^T\right)\right)\right],$$

and thus $f^{\circ}_{2-\text{conn}}(\mathcal{P}(t);[u_1,\ldots,u_n]^T) \geq 0$. □

5.2 Analysis of Motion Test Algorithm Under Perfect Information

We wish to show that Motion Test Algorithm exhibits reasonable behavior as δT becomes small. To do so, we compare it to an idealized variant of Motion Test Algorithm under which each robot has perfect information.

We let Idealized Motion Test Algorithm be the algorithm defined by executing Motion Test Algorithm in continuous time, with $\delta T = 0$, and with perfect information about the state of the network available to each robot. We expound on how this is an idealized variant of Motion Test Algorithm in Lemma 7, which shows that Idealized Motion Test Algorithm allows any collective motion such that no individual robot's motion instantaneously decreases λ_2 unless $\lambda_2 > \lambda_+$.

Lemma 7 *Under* Idealized Motion Test Algorithm*, a direction proposed by robot* j *is accepted if and only if it does not decrease* λ_2 *when taken by itself or if* $\lambda_2 > \lambda_+$.

Proof Consider a direction of motion, u_j, which induces an instantaneous rate of change, $X \in \text{LAP}_{\pm}(n)$ of the Laplacian matrix $L \in \text{LAP}(n)$. Consider the case when $X \bullet (vv^T) < 0$ for some $vv^T \in \delta\lambda_2(L)$ and $\lambda_+ \geq \lambda_2(L)$. Line `1:` of Direction Checking Algorithm ensures that λ_2 is used in place of λ_+ in picking $m_{\min}$. Since each such (vv^T) satisfies $Lv = \lambda_2(L)$, $v \in \mathbf{u}_{\text{span-}L}(m)$ for any m having $\lambda_{m+1}(L) > \lambda_2(L)$. Thus the eigenvector calculation in line `17:` of Direction Checking Algorithm returns a value less than 0 and the proposed motion is rejected. If $\lambda_+ \geq \lambda_2(L)$ and the direction, X, does not satisfy $X \bullet (vv^T) < 0$ for any $vv^T \in \delta\lambda_2(L)$, then there is no basis element, u, in $\mathbf{u}_{\text{span-}L}(m_{\min} - 1)$ having $u^T X u < 0$, so there is at least one m which produces a value greater than or equal to zero in line `17:` of Direction Checking Algorithm. In the case in which $\lambda_+ < \lambda_2(L)$, lines `3:` and `4:` of Direction Checking Algorithm force the direction to be allowed. □

The following lemma establishes various useful facts that hold with high probability as δT approaches zero.

Lemma 8 *If the second derivative of g_{wgt} is bounded, and g_{wgt} has zero derivative at 0, then for any configuration, for any $k \in \mathbb{R}$ and any proposed physical motion of robot j, u_j, there exists a time step δT such that, with high probability, the following are true:*

(i) *The actual Laplacian, $L \in LAP(n)$ is within k of the estimated lower and upper bounds on the Laplacian ($A, B \in LAP(n)$), i.e., $\|A - L\|_2 < k$ and $\|B - L\|_2 < k$.*
(ii) *The actual instantaneous direction of motion, $Y \in LAP_{\pm}(n)$, is within k of the lower bound on the direction of motion, $X \in LAP_{\pm}(n)$, i.e., $\|Y - X\|_2 < k$.*
(iii) *For some m, $\epsilon_A(m) < k$.*

Proof Fact (i) follows from Lemma 6. Corollary 3 allows us to bound the expected time by which the information robot i holds about robot j is out of date. The bound is a decreasing function of δT. This bound is linearly related to the bound on the radius of a sphere known to contain robot j, which induce decreasing bounds on both the range of angles i can be relative to j and the range of distances i can be from j. If the second derivative of g_{wgt} is also bounded, these induce a bound on the computations in lines `4:` and `5:` of Matrix Bound Generator. Since the results of these computations form upper and lower bounds for Y, we can deduce that the distance from the lower bound to Y can be bounded by a decreasing function of δT, thus showing fact (ii). Regarding fact (iii), note that $g'_{\text{wgt}}(0) = 0$, thus the proximity graph is smooth with bounded second derivative even where two robots are coincident. Assume $\lambda_n(L) \neq \lambda_2(L)$. Let l be the index such that $\lambda_l(A) > \lambda_{l-1}(A) = \lambda_2(A)$. From part 1, δT can be picked such that, with high probability, $\|(B - L)\|_2 < \lambda_+ - \lambda_2(L)$. Thus we can replace the expression in line `11:` of Direction Checking Algorithm with $\sqrt{\frac{\lambda_2(B)-\lambda_2(A)}{\lambda_l(A)-\lambda_2(A)}}$ There exists a δT such that the maximum eigenvalue of the expected difference between A and B is less than any constant, thus allow $\epsilon_A(l)$ to be chosen to be less than any given constant. If $\lambda_n(L) = \lambda_2(L)$, let $l = n + 1$, and note that $\epsilon_A(n) = 0$. □

The next result shows that, as δT approaches zero, the behavior of Motion Test Algorithm approaches that of Idealized Motion Test Algorithm.

Theorem 6 *For any configuration, and any proposed direction of motion, v, for robot j, which is permitted under* Idealized Motion Test Algorithm*, there exists a time step, δT, such that when communication happens every δT time units, with high probability robot j will be allowed to move in direction v.*

Proof Let Y be the instantaneous change in the Laplacian matrix induced by v. Let $-3k > \min_{\{v \mid Lv = \lambda_2 v\}}(Y \bullet (vv^T))$ Let l be defined as in Lemma 8, i.e., $\lambda_l(L) > \lambda_{l-1}(L) = \lambda_2(L)$ or $l = n + 1$ and $\lambda_{l-1}(L) = \lambda_n(L)$. By Lemma 8 there is a δT such that $\epsilon_A(l-1) < k$ and $|\lambda_{\min} - \min(\text{eigs}(Y))| < k$. Likewise, by Lemma 8, we can pick X and A sufficiently close to Y and L respectively that $|(Y - X) \bullet A)| + |Y \bullet (L - A)| \leq k$ thus guaranteeing that $M^T_{u((l-1))} X M_{u((l-1))}(1 - \epsilon^2_A(l-1)) + \epsilon^2_A(l-1) \min(\lambda_{\min}, 0) \geq 0$. □

5.3 Motion Projection Algorithm

In this section, we introduce Motion Projection Algorithm to solve the Spectral Connectivity Problem, see Table 5. For this algorithm, we set the dimensionality of physical space, d, to be 2. In other words, each robot lives in $\mathbb{R}^2$. Roughly speaking, Motion Projection Algorithm is a root-finder procedure wrapped around Motion Test Algorithm. The idea is to find the minimum angle by which to deviate a proposed direction of motion so that Motion Test Algorithm returns $f_{\text{safe}} \geq 0$ when executed with the resulting projected direction. If this is the case, then Theorem 5 guarantees that the change in the Laplacian due to motion in the projected direction wins Graph Picking Game.

Let us start by introducing some useful notation. Given a vector $v = [v_1, v_2]^T \in \mathbb{R}^2$, let $f_{\text{motion}} : \mathbb{R} \mapsto \mathbb{R}$ be defined as follows: for each $\theta \in \mathbb{R}$, $f_{\text{motion}}(\theta)$ is the result of evaluating Motion Test Algorithm with the direction of motion $[v_1 \cos(\theta) -$

Table 5 Motion projection algorithm

Name:	Motion Projection Algorithm
Goal:	Solve Spectral Connectivity Problem
Inputs:	• Current time $t_{\text{curr}} \in \mathbb{R}$ • Maximum velocity of any robot, $v_{\max}$ • Maximum time between communication rounds, δT • Proposed direction of motion, v • Eigenvalue bounds $\lambda_- \leq \lambda_+ \in \mathbb{R}$
Persistent data:	• $T \in \mathbb{R}^n$, an array of last recorded time information • $\mathcal{P} \in \mathbb{R}^{2\times n}$, an array of last recorded position information • $D \in \mathbb{R}^{n\times n}$, a matrix of approximate inter-robot distances • $A, B \in \text{LAP}(n)$ • id $\in \{1, \ldots, n\}$, unique identifier of current robot • Maximum angle deflection, $\theta_{\max-\text{id}}$ • $x_{\text{incr}} \in \mathbb{R}$, step-size for root finder.
Outputs:	• $\tilde{v} \in \mathbb{R}^2$, safe projected direction. • $\theta \in S_1$, angle by which to rotate v to get safe direction

Let $D \leftarrow$ call All-to-all Broadcast Algorithm on t_{curr}
Let $(A, B) \leftarrow$ Matrix Bound Generator on t_{curr}, $v_{\max}$ and δT
Let $v_\perp \leftarrow [-v_1, v_2]^T$ and $\theta \leftarrow 0$ */*Perpendicular direction, for computing rotations*/*
while $\theta \leq \theta_{\max-\text{id}}$ **do**
  for all $x_{\text{sgn}} \in \{-1, 1\}$ **do**
    Let $\tilde{v} \leftarrow v\cos(\theta) + x_{\text{sgn}} v_\perp \sin(\theta)$ */*Rotated direction*/*
    Let $S_{\text{check}} \leftarrow$ Motion Test Algorithm on t_{curr}, $v_{\max}$, δT, $\tilde{v}$
    if $x_{\text{sgn}} = -1$ or $|S_{\text{check}}| < k$ **then**
      Let $k \leftarrow |S_{\text{check}}|$ */*For stepsize computation*/*
    end if
    if $S_{\text{check}} \geq 0$ **then**
      return $(\tilde{v}, x_{\text{sgn}}\theta)$ */*Found good direction*/*
    end if
  end for
  Let $\theta \leftarrow \theta + \max(x_{\text{incr}}, \frac{k}{n v_{\max} \max_{s\in\mathbb{R}}(g'_{\text{wgt}}(s))})$ */*Step θ*/*
end while */*At this point, no safe direction has been found.*/*
return $([0, 0]^T, 0)$

$v_2 \sin(\theta), v_2 \cos(\theta) + v_1 \sin(\theta)]^T$. The following result provides an upper bound on how f_{motion} changes with θ.

Lemma 9 *f_{motion} is globally Lipschitz with Lipschitz constant $nv_{max} \max_{s\in\mathbb{R}}(g'_{wgt}(s))$.*

Proof We begin by noting that $f_{\text{motion}}(\theta)$ is the minimum, over $\tilde{m} > m$, of

$$\min\left(\text{eigs}\left(M_{u(\tilde{m})}^T X M_{u(\tilde{m})}\right)\right) + \epsilon_A(\tilde{m}),$$

where $M_{u(\tilde{m})}$ and $\epsilon_A(\tilde{m})$ do not depend on θ. Each off-diagonal element of X has a Lipschitz constant bounded by $v_{\max} \max_{s\in\mathbb{R}}(g'_{\text{wgt}}(s))$. From here on, let $v_{\text{edg}} = v_{\max} \max_{s\in\mathbb{R}}(g'_{\text{wgt}}(s))$. Let $L_{\text{clique}}(n) \in \text{LAP}(n)$ satisfy $L_{\text{clique}}(n)_{i,j} = -1$ for each $i \neq j$. The change in X due to a change in θ of $\Delta\theta$ lives in $\text{LAP}_{\pm}(n)$ and is bounded from above by $v_{\text{edg}} L_{\text{clique}}(n)\Delta\theta$ and from below by $-v_{\text{edg}} L_{\text{clique}}(n)\Delta\theta$. Let $\Delta expression$ be the change in the value of *expression* due to a change in θ of $\Delta\theta$. Since $M_{u(\tilde{m})}$ is defined by a subset of the columns of an orthogonal basis, each element of $\text{eigs}\left(M_{u(\tilde{m})}^T X\right)$ has a Lipschitz constant contained in

$$\left[-v_{\text{edg}} \max\left(\text{eigs}\left(L_{\text{clique}}(n)\right)\right), v_{\text{edg}} \max\left(\text{eigs}\left(L_{\text{clique}}(n)\right)\right)\right] = [-v_{\text{edg}} n, v_{\text{edg}} n].$$

Because

$$\min(\text{eigs}(A + B)) \in [\min(\text{eigs}(A) + \min(\text{eigs}(B)), \min(\text{eigs}(A)) + \max(\text{eigs}(B))],$$

we deduce that

$$\begin{aligned}
&\Delta \min\left(\text{eigs}\left(M_{u(\tilde{m})}^T X M_{u(\tilde{m})}\right)\right) \\
&\quad \in \left[\min\left(\text{eigs}\left(M_{u(\tilde{m})}^T \Delta X M_{u(\tilde{m})}\right)\right), \max\left(\text{eigs}\left(M_{u(\tilde{m})}^T \Delta X M_{u(\tilde{m})}\right)\right)\right]
\end{aligned}$$

and therefore $\Delta \min\left(\text{eigs}(M_{u(\tilde{m})}^T X M_{u(\tilde{m})})\right) \in [-v_{\text{edg}} n, v_{\text{edg}} n]$. □

The following result establishes that the root finder embedded in Motion Projection Algorithm uses a reasonable step size.

Lemma 10 *Let $v_{edg} = v_{max} \max_{s\in\mathbb{R}}(g'_{wgt}(s))$. If $f_{motion}(\theta) < -k$ for some $k \in \mathbb{R}_{\geq 0}$, $f_{motion}\left(\theta + \frac{k}{nv_{edg}}\right) < 0$.*

Proof The bound that the Lipschitz constant for $f_{\text{motion}}(\theta)$ lies in $[-nv_{\text{edg}}, nv_{\text{edg}}]$ holds for all θ. So the change in f_{motion} from θ to $\theta + \frac{k}{nv_{\text{edg}}}$ is bounded within $\frac{k}{nv_{\text{edg}}}[-nv_{\text{edg}}, nv_{\text{edg}}]$. □

Finally, we are ready to show that Motion Projection Algorithm is a solution to Spectral Connectivity Problem.

Theorem 7 *The* Motion Projection Algorithm *solves the* Spectral Connectivity Problem. *If there is an interval, $[\theta_-, \theta_+] \subseteq S_1$, such that $f_{motion}(\alpha) \geq 0$ for all $\alpha \in [\theta_-, \theta_+]$ and $\theta_+ - \theta_- \geq x_{incr}$ then* Motion Projection Algorithm *will return a direction other than* $[0, 0]^T$.

Proof The outer loop in Motion Projection Algorithm checks possible directions and evaluates Motion Test Algorithm on them. If one of these directions results in Motion Test Algorithm returning a non-negative value, Motion Projection Algorithm returns that direction, otherwise it returns $[0, 0]^T$. Let the current angle be θ. If θ steps by $\frac{k}{nv_{\max}\max_{s\in\mathbb{R}}(g'_{\text{wgt}}(s))}$, where k is the max of f_{motion} over $\{\theta, -\theta\}$, then f_{motion} will be less than zero for the next θ. If θ steps by more than this, then θ steps by x_{incr}. To pass the boundaries of the region $[\theta_-, \theta_+]$, θ must step by x_{incr}, and must, therefore land in $[\theta_-, \theta_+]$. Because each step of θ is at least x_{incr}, θ covers the entire region from 0 to $\theta_{\max-\text{id}}$ in finite time. □

5.4 Analysis of the Communication Complexity

At each communication round, the above solutions to the Spectral Connectivity Decision Problem and the Spectral Connectivity Problem require each robot to perform computations whose memory requirements are polynomial in the number of robots and whose time complexity is polynomial in the number of robots times the time required to calculate the eigenspace of an $n \times n$ matrix.

As is typically the case with any coordination algorithm, multiple communication rounds are required in order to complete the assigned task. Therefore, it is also important to study the complexity of our algorithm in terms of the required rate of communication to achieve a desired performance. This is related, in some sense, to the notion of communication complexity, commonly discussed in the literature on distributed algorithms [2, 13, 16], in which one studies the number of messages that need to be sent during an algorithm execution in order to achieve a given task, usually written as a function of the size of the network.

In addition to the size of the network, there are additional factors that need to be considered to accurately characterize the required rate of communication, including the required performance (represented here as the probability that an agent will move, $p_{\text{move}} \in (0, 1)$), the amount we expect $\lambda_2(\mathcal{G})$ to be above the threshold λ_+, the maximum velocity with which the robots move, $v_{\max}$, the time between communication rounds, δT, and a bound, $g_{\max}$, on the magnitude of the gradient of the proximity function.

We begin by characterizing a property of the matrices A and B generated by the Matrix Bound Generator which we will use to bound the gap between $\lambda_2(A)$ and $\lambda_2(L(\mathcal{G}))$.

Lemma 11 *For any $p_{move} \in (0, 1)$, the following holds*

$$\Pr\left(\max_{i,j\in\{1,\ldots,n\}}(B_{i,j} - A_{i,j})\right) \leq \frac{4}{1 - p_{move}} g_{max} v_{max}\left(n^2\left(n^2 + 2\right)\delta T\right) \leq p_{move}.$$

Proof Follows from Markov's inequality, $\Pr(|X| > \alpha) \leq \frac{E(|X|)}{\alpha}$ and Lemma 6. □

We next proceed to combine this result with some general properties of graph Laplacians to bound the transmission rate necessary for each robot to be allowed to move in any direction with high probability (and thus be allowed to move in the direction specified by the underlying control algorithm).

Theorem 8 *Let* $g_{max} = \max_{x \in \mathbb{R}_{\geq 0}}(|g'_{wgt}(x)|)$. *If the time between communication rounds is such that* $\delta T \leq \frac{(1-p_{move})(\lambda_2(L(\mathcal{G}))-\lambda_+)}{4g_{max}v_{max}(n^2(n^2+2)}$, *then each robot will move on each timestep with probability at least* p_{move}.

Proof By Lemma 11, with probability at least p_{move}, $\max_{i \neq j, i \in \{1,\ldots,n\}, j \in \{1,\ldots,n\}} B_{i,j} - A_{i,j}$ never exceeds $\frac{4}{1-p_{\text{move}}} g_{\max} v_{\max}(n^2(n^2+2)\delta T)$. Thus, with probability at least p_{move}, $B - A$ satisfies $B - A \leq_{\text{LAP}} \frac{4}{1-p_{\text{move}}} g_{\max} v_{\max}(n^2(n^2+2)\delta T) L_{\text{unit}}$ where $L_{\text{unit}} \in \text{LAP}(n)$ has off-diagonal entries consisting solely of 0 and 1. The maximum possible eigenvalue of such a matrix is n, as shown in [6], and $\lambda_2(B) \leq \lambda_2(A) + \lambda_n(B - A)$, thus $\lambda_2(B) - \lambda_2(A) \leq \lambda_2(B - A)$ Whenever $\lambda_2(L)$ satisfies $\lambda_2(L(\mathcal{G})) \geq \lambda_+ + \delta\lambda$, setting $\delta T \leq \frac{(1-p_{\text{move}})\delta\lambda}{4g_{\max}v_{\max}(n^2(n^2+2)}$, or equivalently, setting the transmission rate to approximately $\frac{8}{1-p_{\text{move}}} g_{\max} v_{\max} n^4$ real numbers per time unit allows the robots to move at each timestep with probability at least p_{move}. □

We note that this result does not fully characterize the communication complexity of our solutions because, among other things, does not account explicitly for the proposed direction of motion of the robots. We discuss this topic among our ideas for future research later in Section 6.

5.5 Simulations

In this section, we present simulations of the Motion Projection Algorithm to further validate the results presented in the previous sections. We have developed a custom Java-based platform [18] for the simulation of algorithms running on networks of robotic agents following the modeling framework proposed in [14]. The platform has a user interface layer which allows the simulations to be graphically displayed in real time in a Java applet.

In all simulations, Motion Projection Algorithm is implemented with the r-disk graph for the actual communication network, and the nonconvex weight function of the spline graph defined in Remark 1, where $r_{\max} \in \mathbb{R}$ was chosen to be slightly less than r and $r_{\min} \in \mathbb{R}$ was slightly bigger than zero. Note that the function g_{wgt} satisfies the conditions of Lemma 8. The maximum velocity of each individual robot is $v_{\max} = 0.125$ and the time step is $\delta T = 0.125$.

We consider four sets of underlying control laws. In the first simulation, shown in Fig. 1, all robots attempt to follow a Laplacian-based flocking algorithm. Each robot moves at unit speed and, at each time step, updates its heading to its own heading plus a scalar (0.1) times the average of the differences between its own heading and those of its neighbors. This is in fact a discrete-time implementation of the Laplacian-based averaging consensus [15], see also [8]. Each robot tries to move in the direction of its own heading, subject to maintaining connectivity.

In the second simulation, shown in Fig. 2, all robots attempt to follow a Laplacian-based rendezvous algorithm. Each robot moves at unit speed and, at each time step, moves towards the average of its neighbors positions, while maintaining connectivity.

In the third simulation, shown in Fig. 3, five agents with different conflicting control laws attempt to follow their own directives while maintaining connectivity. Note that the simulations in which the control directives naturally align with maintaining connectivity tend to require far fewer iterations to converge than those in which this is not the case (particularly those with random or conflicting motion).

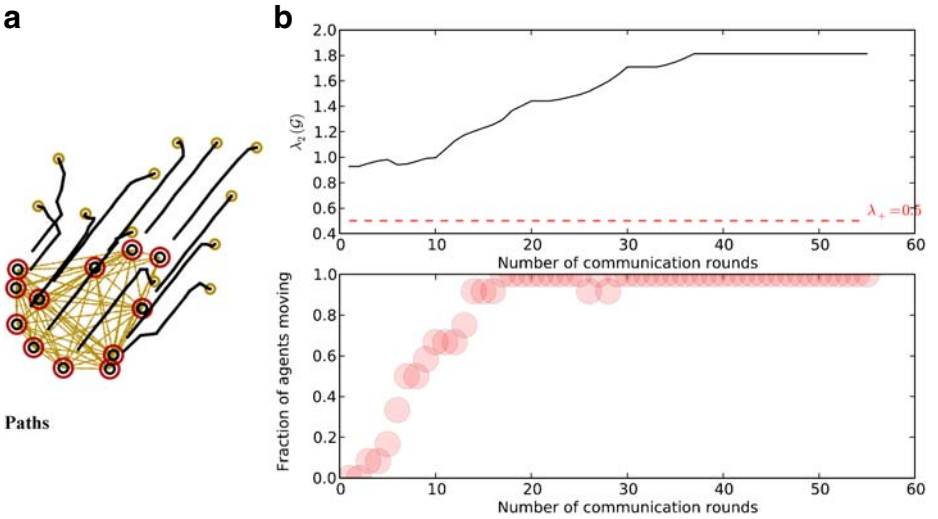

Fig. 1 Execution of Motion Projection Algorithm with 12 robotic agents. The underlying control law is determined by the robots moving at unit speed and running a Laplacian-based consensus to update its heading. Plot (**a**) shows the paths taken by the robots and plot (**b**) shows the evolution of the algebraic connectivity and the proportion of robots actively moving as functions of the communication round. The angle of motion each robot is allowed to deviate from is 0.95π

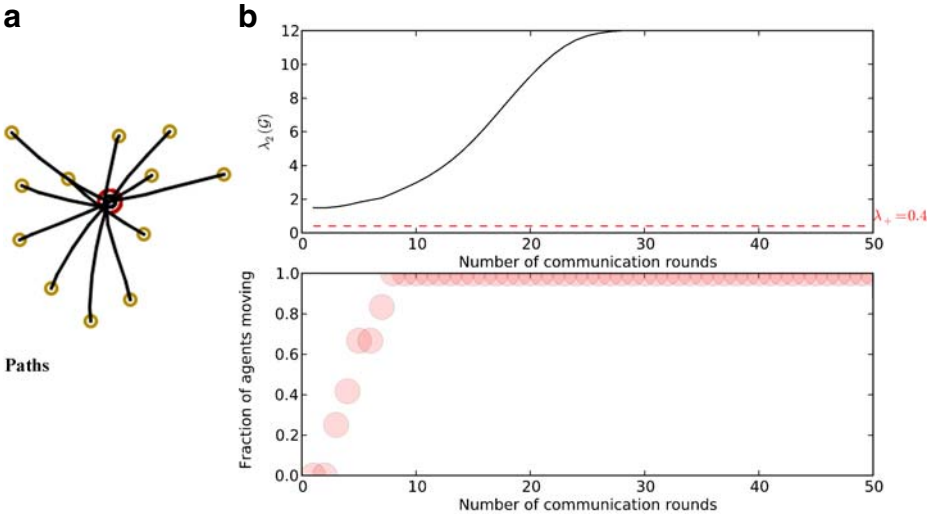

Fig. 2 Execution of Motion Projection Algorithm with 12 robotic agents. The underlying control law is determined by the robots moving at unit speed and running a Laplacian-based consensus to update its target position. Plot (**a**) shows the paths taken by the robots and plot (**b**) shows the evolution of the algebraic connectivity and the proportion of robots actively moving as functions of the communication round. The angle of motion each robot is allowed to deviate from is π

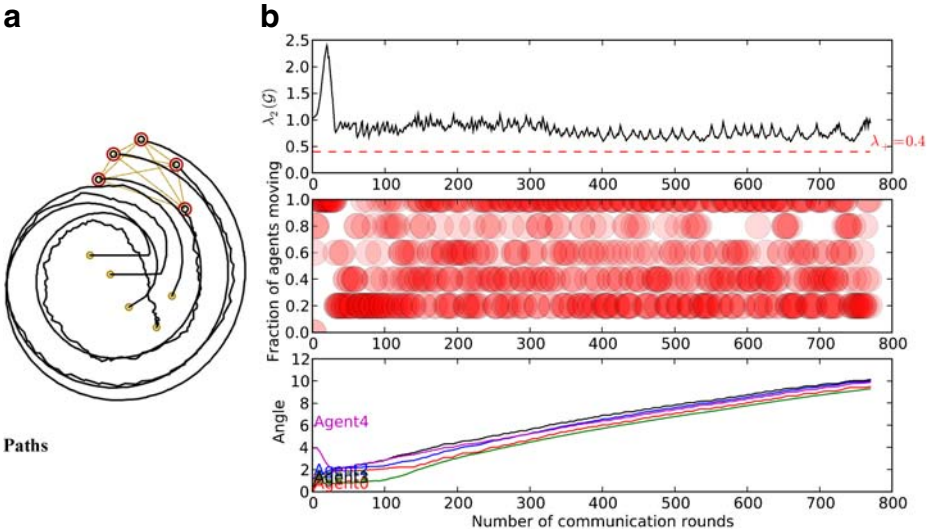

Fig. 3 Execution of Motion Projection Algorithm with 5 robotic agents. The underlying control law corresponds to following scenario: Four leaders each attempt to follow different control laws each of which converges on a different fixed trajectory. The remaining agent moves randomly. Plot (**a**) shows the paths taken by the robots and plot (**b**) shows the evolution of the algebraic connectivity, the fraction of robots moving at each round, and the evolution of the angle of each robot's position relative to the origin

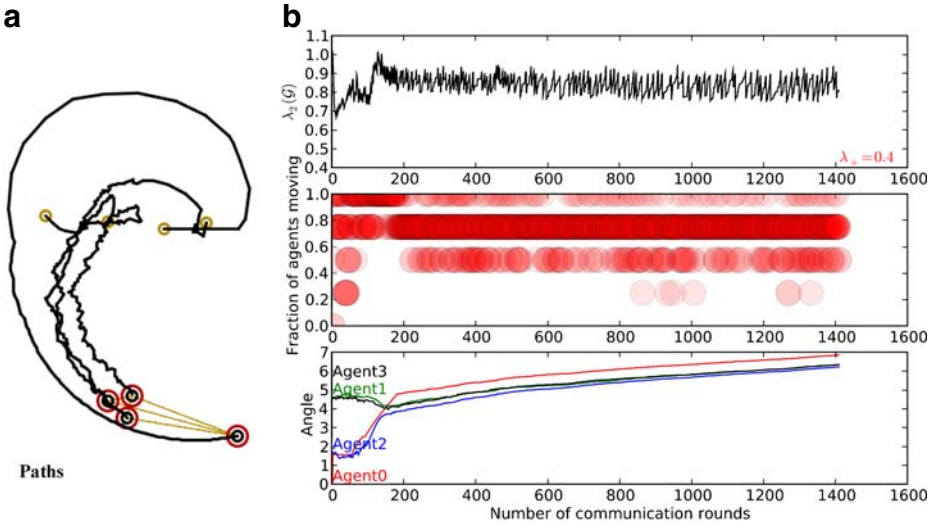

Fig. 4 Execution of Motion Projection Algorithm with 4 robotic agents. The underlying control law corresponds to one leader following a fixed trajectory and the remaining agents moving randomly. Plot (**a**) shows the paths taken by the robots and plot (**b**) shows the evolution of the algebraic connectivity, the fraction of robots moving at each round, and the evolution of the angle of each robot's position relative to the origin

In the fourth simulation, shown in Fig. 4, one leader robot attempts to follow a fixed trajectory while the remaining robots try to move randomly subject to the constraint of maintaining connectivity. For each robot, we specify the threshold $\theta_{\max-i} = 0.2$.

These simulations validate the preliminary results in Section 5.4 linking the communication complexity of the algorithm to the difference between λ_2 and λ_+. In particular, we observe that the fraction of agents moving at any given time is correlated to the difference between λ_2 and λ_+, and that the fraction of agents moving together with $v_{\max}$ affect the time required for the network to complete the given task. We also find that, in situations where λ_2 stays well above λ_+, the complexity is reasonable. On the other hand, the complexity degrades as the algorithm begins to push λ_2 up against the threshold. Adding a single agent whose underlying motion is random may help to maintain connectivity, but assigning random underlying motion to the majority of the agents, as in the simulation shown in Fig. 4, may slow down collective agent motion.

6 Conclusions and Future Work

We have studied the problem of connectivity maintenance in robotic networks performing spatially-distributed tasks. In our approach, the edge weights of the connectivity graph are not necessarily convex functions of the inter-robot distances. We have proposed a distributed procedure to synthesize motion constraints on the individual robots so that the algebraic connectivity of the overall network remains above a desired threshold. The algorithm works even though individual robots only have partial information about the network state due to communication delays and network mobility. We have shown that as the communication rate increases, the performance of the proposed algorithm approaches that of the ideal centralized solution of Spectral Connectivity Decision Problem.

Future work will evaluate the communication complexity of the proposed coordination algorithm. We are especially interested in calculating lower bounds on the communication complexity required by the computation of the gradient of the algebraic connectivity function in a distributed way. We also plan to study the relationship between the rate of information transmission and the rate of robot motion in terms of the number of robots and the exact value of $f_{2-\text{conn}}$. Finally, we will explore the combination of the proposed approach with algorithms for deployment and exploration.

Acknowledgement This research was supported in part by NSF CAREER Award ECS-0546871.

References

1. Boyd, S.: Convex optimization of graph Laplacian eigenvalues. In: Proceedings of the International Congress of Mathematicians, pp. 1311–1319, Madrid, August 2006

2. Bullo, F., Cortés, J., Martínez, S.: Distributed Control of Robotic Networks. Applied Mathematics Series. Princeton University Press, September. Manuscript under contract. http://www.coordinationbook.info (2008)
3. Clarke, F.H.: Optimization and Nonsmooth Analysis. Canadian Mathematical Society Series of Monographs and Advanced Texts. Wiley, New York (1983)
4. Cortés, J., Martínez, S., Bullo, F.: Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. IEEE Trans. Automat. Contr. **51**(8), 1289–1298 (2006)
5. de Gennaro, M.C., Jadbabaie, A.: Decentralized control of connectivity for multi-agent systems. In: IEEE Conf. on Decision and Control, pp. 3628–3633, San Diego, December 2006
6. Fiedler, M.: Algebraic connectivity of graphs. Czech. Math. J. **23**, 298–305 (1973)
7. Godsil, C.D., Royle, G.F.: Algebraic graph theory. Graduate Texts in Mathematics, vol. 207. Springer, New York (2001)
8. Jadbabaie, A., Lin, J., Morse, A.S.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans. Automat. Contr. **48**(6), 988–1001 (2003)
9. Jaromczyk, J.W., Toussaint, G.T.: Relative neighborhood graphs and their relatives. Proc. IEEE **80**(9), 1502–1517 (1992)
10. Ji, M., Egerstedt, M.: Distributed control of multiagent systems while preserving connectedness. IEEE Trans. Robot. **23**(4), 693–703 (2007)
11. Kim, Y., Mesbahi, M.: On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian. IEEE Trans. Automat. Contr. **51**(1), 116–120 (2006)
12. Lewis, A.S.: Nonsmooth analysis of eigenvalues. Math. Program. **84**, 1–24 (1999)
13. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann, San Francisco (1997)
14. Martínez, S., Bullo, F., Cortés, J., Frazzoli, E.: On synchronous robotic networks - Part I: models, tasks, and complexity. IEEE Trans. Automat. Contr. **52**(12), 2199–2213 (2007)
15. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. Proc. IEEE **95**(1), 215–233 (2007)
16. Peleg, D.: Distributed Computing. A Locality-Sensitive Approach. Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia (2000)
17. Savla, K., Notarstefano, G., Bullo, F.: Maintaining limited-range connectivity among second-order agents. SIAM J. Control Optim. **48**, 187–205 (2009)
18. Schuresko, M.D.: CCLsim. a simulation environment for robotic networks. http://www.soe.ucsc.edu/~mds/cclsim (2008)
19. Schuresko, M.D., Cortés, J.: Safe graph rearrangements for distributed connectivity of robotic networks. In: IEEE Conf. on Decision and Control, pp. 4602–4607, New Orleans, 12–14 December 2007
20. Spanos, D.P., Murray, R.M.: Motion planning with wireless network constraints. In: American Control Conference, pp. 87–92, Portland, June 2005
21. Wu, C.W.: Algebraic connectivity of directed graphs. Linear Multilinear Algebra **53**(3), 203–223 (2005)
22. Zavlanos, M.M., Pappas, G.J.: Controlling connectivity of dynamic graphs. In: IEEE Conf. on Decision and Control and European Control Conference, pp. 6388–6393, Seville, December 2005
23. Zavlanos, M.M., Pappas, G.J.: Distributed connectivity control of mobile networks. In: IEEE Conf. on Decision and Control, New Orleans, 12–14 December 2007
24. Zavlanos, M.M., Pappas, G.J.: Flocking while preserving network connectivity. In: IEEE Conf. on Decision and Control, New Orleans, 12–14 December 2007