Proceedings of the 45th IEEE Conference on Decision & Control
Manchester Grand Hyatt Hotel
San Diego, CA, USA, December 13-15, 2006

Decentralized Control of Connectivity for Multi-Agent Systems

Maria Carmela De Gennaro, Ali Jadbabaie

Abstract— **In this paper we propose a decentralized algorithm to increase the connectivity of a multi-agent system. The connectivity property of the multi-agent system is quantified through the second smallest eigenvalue of the state dependent Laplacian of the proximity graph of agents. An exponential decay model is used to characterize the connection between agents. A supergradient algorithm is then used in conjunction with a recently developed decentralized algorithm for eigenvector computation to maximize the second smallest eigenvalue of the Laplacian of the proximity graph. A potential based control law is utilized to achieve the distances dictated by the supergradient algorithm. The algorithm is completely decentralized, where each agent receives information only from its neighbors, and uses this information to update its control law at each step of the iteration. Simulations demonstrate the effectiveness of the algorithm.**

I. INTRODUCTION

Cooperative control of multi-agent systems is a very active research area of control theory. In the past few years problems such as flocking, consensus, coverage and pattern formation have been studied. The study is generally focused on the development of distributed control laws in order to reach a global objective [1]–[7].

One interesting problem recently analyzed regards the connectivity maintenance of the distance-dependent graph of the network. In such a graph, known also as R-disk graph [7], there is an edge between two nodes if their Euclidean distance is less than or equal to a pre specified number R. The difficulty in connectivity maintenance stems from the fact that connectivity is an inherently global property and a complicated function of the motion of the nodes. Other attempts to model changes in topology, such as [1], ignore the dependence of switching on motion. Several attempts have been made in the wireless networking literature to follow local rules that guarantee connectivity. One example is the "sector rule" which guarantees connectivity of the R-disk graph on the plane if each agent has at least one neighbor in every sector of 120 degrees [8]. Another interesting solution to the connectivity problem is given by the circumcenter algorithm, which increases gradually the degree of each agent and constraints the motion of the agents to avoid the lost of previously present connections [7]. Many authors in the control theory literature have also made progress on this problem [9]–[12].

Dr. Maria Carmela De Gennaro is with the Università del Sannio, Dipartimento di Ingegneria, piazza Roma 21, Benevento, Italy. Email:degennaro@unisannio.it

Dr. Ali Jadbabaie is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA. Email: jadbabai@seas.upenn.edu

In this paper, our goal is to maximize the second smallest eigenvalue of the Laplacian matrix (called *algebraic connectivity*) of the state dependent proximity graph of agents, similarly to [11], but with a decentralized approach.

The starting point of our work is the observation that the algebraic connectivity is a concave function of the Laplacian matrix. Thus we can easily find a supergradient for it. Motivated by results in [13], we demonstrate in this paper that a supergradient direction for the algebraic connectivity is function of the corresponding eigenvector, the Fiedler vector. This eigenvector is computed in a decentralized way by using the algorithm in [14]. In this way we blend the results of [13] and [14] to realize a decentralized supergradient algorithm which allows each agent to maximize the algebraic connectivity by working only with one row of the Laplacian matrix.

However, contrary to [13], the supergradient cannot modify directly the entries of the Laplacian matrix, because in our framework the Laplacian is a state dependent matrix. This means that each iteration of the supergradient algorithm only provides agents with set-points, that have to be used to generate the actual control. The control law used is a simple potential-based flow [15].

The paper is organized as follows. In Section II, we show the framework of our problem. In Section III, we describe the optimization problem and the proposed algorithm, with convergence properties. In Section IV, we present the control law for the agents, with convergence properties. In Section V, we explain some simulation results. The paper concludes with a summary of the proposed work.

II. PROBLEM FORMULATION

We consider a multi-agent system composed by N agents, that move on the plane. The state of the agents is defined by the vector $\mathbf{z} = (z_1, z_2, ...z_N) \in \mathbb{R}^{2N}$. The dynamic of each agent i is:

$$\dot{z}_i = u_i, \tag{1}$$

where $u_i \in \mathbb{R}^2$ is the control action of agent i.

The link among each pair of agents is assumed to be dependent on their distance: if the distance is at most equal to a fixed *connection radius* R, then the agents are said to be connected, otherwise they are not connected. Moreover the strength of the connection decreases smoothly with distance (see [6], [11], [16]). If the distance between two agents is less than a threshold ρ, the agents are "strongly" connected. Otherwise the connection is exponentially weakened as the distance increases, until the distance R, where they practically lose the connection. The strength dependence on the

1-4244-0171-2/06/$20.00 ©2006 IEEE.

distance is due to physical reasons: in a multi-robot system each robot is equipped with sensors whose resolution is decaying exponentially with the distance to the object to observe.

A. Graph representation

The multi-agent system can be represented by a graph [17] $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes (agents) and $\mathcal{E}$ is the set of edges (connections). An edge is a pair (i, j) of distinct nodes of $\mathcal{V}$, which are the representation of agents whose mutual distance is at most equal to the connection radius R. When the connection is specified as above, the node i is said to be connected to all the nodes that are in the circular neighborhood of i, which is a subset of $\mathcal{V}$ defined as:

$$C_i \triangleq \{j \in \mathcal{V}, j \neq i : \|r_{ij}\| = \|z_i - z_j\| \leq R\}. \quad (2)$$

We assume that R has the same value for all the agents, thus the graph $\mathcal{G}$ is undirected. The graph $\mathcal{G}$ is state dependent, in the sense that it evolves in time with its connectivity governed by the dynamics of the agents.

B. Laplacian matrix

The graph can be also represented using the Laplacian matrix:

$$\mathbf{L}(\mathbf{z}) = \mathbf{\Delta}(\mathbf{z}) - \mathbf{A}(\mathbf{z}), \quad (3)$$

where $\mathbf{A}(\mathbf{z})$ is the adjacency matrix, whose entries A_{ij} are positive if $j \in C_i$, and zero otherwise:

$$A_{ij} = \begin{cases} 1 & \|r_{ij}\| < \rho; \\ e^{\frac{-5(\|r_{ij}\|-\rho)}{(R-\rho)}} & \rho \leq \|r_{ij}\| \leq R; \\ 0 & \|r_{ij}\| > R; \end{cases} \quad (4)$$

and $\mathbf{\Delta}(\mathbf{z})$ is a diagonal matrix, with elements $\Delta_i = \sum_{j=1\ldots N} A_{ij}$ along the diagonal.

The entry A_{ij} represents the strength of the connection among agents (i, j), which decays exponentially with the distance[1]. Figure 1 shows the shape of A_{ij}.

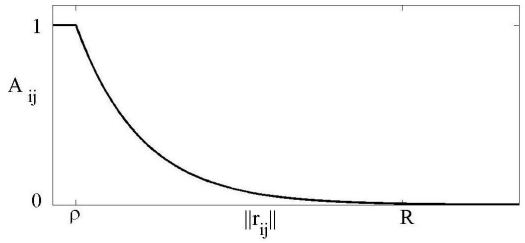

Fig. 1. Adjacency matrix element A_{ij} as a function of the distance between agents i and j. At the distance $\|r_{ij}\| = R$, the agents lose the connection. Function A_{ij} is not continuous in the point R, but the value that it assumes at R can be considered a good approximation of zero.

The eigenvalues of the Laplacian matrix capture some interesting properties of the underlying graph, thus it is interesting to take a look on their values. Since $\mathbf{L}(\mathbf{z})$ is positive semi-definite and symmetric, its eigenvalues are all nonnegative. By ordering the eigenvalues in a increasing way, we have:

$$0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N.$$

The eigenvector corresponding to the first eigenvalue is always $\mathbf{1}$. The second eigenvalue λ_2 is called *algebraic connectivity* [17] of the system, and it is an indicator of how much the graph is connected. The value of λ_2 is zero if the graph is not connected, and it increases when the connectivity of the graph increases. The maximum value of λ_2 is equal to N, and it is reached when the entries (i, j) of the adjacency matrix are all equal to 1, that is the graph is completely connected (all the possible edges are present in it) and the distances among agents in the original system are all at most ρ.

The relation between λ_2 and the graph connectivity can be used to find a control action that preserves the connectivity in time. Generally speaking, λ_2 is function of the state of the entire system, thus we can write it as $\lambda_2(\mathbf{L}(\mathbf{z}))$. What we want to show is how to increase the value of λ_2 with a decentralized control action, in which each agent knows only information about its neighbors, while it doesn't know the current value of λ_2 because it is function of the entire Laplacian matrix.

The optimization problem to solve is:

$$\max_{\mathbf{z}} \lambda_2(\mathbf{L}(\mathbf{z})).$$

It can be decomposed in two subproblems, which consider the dependance on the external and the internal unknowns, $\mathbf{L}$ and $\mathbf{z}$. Firstly, the dependance on the Laplacian matrix is represented in the external optimization problem:

$$\max_{\mathbf{L}} \lambda_2(\mathbf{L}), \quad (5)$$

where the Laplacian $\mathbf{L}$ has to satisfy the constraints defined in (3)-(4). This optimization is realized in open loop, because the dependance on the state of the agents $\mathbf{z}$ is not included. The solution found is called $\mathbf{L}^*$. In Section III we will show the iterative algorithm of the decentralized supergradient that solve this problem.

The closed loop optimization is realized when each agent uses $\mathbf{L}^*$ as a reference for its controller, and moves according to the control action. At the equilibrium, the Laplacian $\mathbf{L}(\mathbf{z})$ is updated with the current value of the state. This updated Laplacian matrix is assumed to be the starting point for the new iteration of the supergradient. In Section IV we will describe the control action applied by each agent, with the convergence properties of the supergradient.

III. Maximization of $\lambda_2(\mathbf{L})$

First, we note that $\lambda_2(\mathbf{L})$ is a concave function of $\mathbf{L}$ in the space $\mathbf{1}^\perp$, in fact it is the infimum of a set of linear functions in $\mathbf{L}$:

$$\lambda_2(\mathbf{L})\mathbf{v}^T\mathbf{v} \leq \mathbf{v}^T\mathbf{L}\mathbf{v}, \forall \mathbf{v} \in \mathbf{1}^\perp \Rightarrow \lambda_2(\mathbf{L}) = \inf_{\mathbf{v}\in\mathbf{1}^\perp} \{\frac{\mathbf{v}^T\mathbf{L}\mathbf{v}}{\mathbf{v}^T\mathbf{v}}\}. \quad (6)$$

[1]The coefficient 5 in the expression of A_{ij} is due to the convergence property of the exponential function: when $\|r_{ij}\| = R$, $A_{ij} = e^{-5} \approx 0$.

The optimization of λ_2 has been performed by the SDP [11], but this method is not decentralized. Since we want to solve the problem in a decentralized framework, we have to choose a different optimization algorithm. One that is suitable for decentralization is the supergradient algorithm [13].

Now, we recall the notion of supergradient for a concave function.

Definition: Let $f : \mathbb{R}^N \rightarrow \mathbb{R}$ be a concave function. The vector $\mathbf{g}$ is a supergradient of f in the point $\mathbf{x}$ if for all $\mathbf{y} \neq \mathbf{x}$ the following inequality holds:

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \mathbf{g}^T(\mathbf{y} - \mathbf{x})$$

To maximize the function f, the updating rule of the supergradient at the step k is:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{g}^{(k)},$$

where $\alpha^{(k)} > 0$ is the step-size to choose.

Using the notion of supergradient, it is simple to find a supergradient matrix for $\lambda_2(\mathbf{L})$. We start from the following inequality:

$$\lambda_2(\tilde{\mathbf{L}})\mathbf{v}_2^T\mathbf{v}_2 \leq \mathbf{v}_2^T(\tilde{\mathbf{L}})\mathbf{v}_2, \tag{7}$$

where $\tilde{\mathbf{L}} \neq \mathbf{L}$, and $\mathbf{v}_2 \in \mathbf{1}^{\perp}$ is the unit eigenvector of $\mathbf{L}$ corresponding to $\lambda_2(\mathbf{L})$. The right side of the previous equation can be rewritten as:

$$\begin{aligned} \mathbf{v}_2^T(\tilde{\mathbf{L}})\mathbf{v}_2 &= \mathbf{v}_2^T\mathbf{L}\mathbf{v}_2 + \mathbf{v}_2^T(\tilde{\mathbf{L}} - \mathbf{L})\mathbf{v}_2 \\ &= \mathbf{v}_2^T\lambda_2(\mathbf{L})\mathbf{v}_2 + \mathbf{v}_2^T(\tilde{\mathbf{L}} - \mathbf{L})\mathbf{v}_2 \\ &= \lambda_2(\mathbf{L}) + < \mathbf{v}_2\mathbf{v}_2^T, (\tilde{\mathbf{L}} - \mathbf{L}) > \end{aligned} \tag{8}$$

Composing (7) and (8) we obtain that:

$$\lambda_2(\tilde{\mathbf{L}}) \leq \lambda_2(\mathbf{L}) + < \mathbf{v}_2\mathbf{v}_2^T, (\tilde{\mathbf{L}} - \mathbf{L}) >, \tag{9}$$

which shows that the matrix $\mathbf{G} = \mathbf{v}_2\mathbf{v}_2^T$ is a supergradient for $\lambda_2(\mathbf{L})$. The updating rule for the $\mathbf{L}$ matrix is:

$$\mathbf{L}^{*(k+1)} = \mathbf{L}^{*(k)} + \alpha^{(k)}\mathbf{G}^{(k)}. \tag{10}$$

From [18]–[20] it is known that if the step-size $\alpha^{(k)}$ is the coefficient of a not summable but square summable series, the supergradient method converges to the optimal value.

The decentralized computation of the supergradient matrix $\mathbf{G}$ is discussed in [13], [19]; here we apply the procedure to find the supergradient of the Laplacian matrix.

A. Decentralized supergradient algorithm

Let's define a vector $\mathbf{p} = (p_{ij})$ of elements of the matrix $\mathbf{L}$, for all the possible connections of agents (i, j):

$$p_{ij} = L_{ij}, i = 1, ..., N, j > i \tag{11}$$

The dimension of $\mathbf{p}$ is $N(N-1)/2$, that is the maximum number of links in the graph.

Then we define a matrix $\mathbf{E_{ij}}$ for each pair (i, j):

$$E_{\mathbf{ij}_{ij}} = E_{\mathbf{ij}_{ji}} = 1, E_{\mathbf{ij}_{ii}} = E_{\mathbf{ij}_{jj}} = -1, \tag{12}$$

with 0 in all the other entries; thus we can now write:

$$\mathbf{L} = \sum_{i=1,...,N, j>i} \mathbf{E_{ij}}p_{ij}. \tag{13}$$

The statement of the optimization problem (5) becomes:

$$\begin{aligned} &\max_{\mathbf{p}} \lambda_2\Big(\sum_{i=1,...,N, j>i} \mathbf{E_{ij}}p_{ij}\Big) \\ &s.t.\ -1 \leq p_{ij} \leq 0 \quad \forall(i,j) \end{aligned} \tag{14}$$

where the feasibility constraints derive from (3)-(4).

The function $\lambda_2(\mathbf{p})$ is concave in $\mathbf{p}$:

$$\begin{aligned} \lambda_2(\tilde{\mathbf{p}}) &\leq \lambda_2(\mathbf{p}) + \sum_{i=1,...,N, j>i} < \mathbf{v}_2\mathbf{v}_2^T, \mathbf{E_{ij}}(\tilde{p}_{ij} - p_{ij}) > \\ &= \lambda_2(\mathbf{p}) + \sum_{i=1,...,N, j>i} (\mathbf{v}_2^T\mathbf{E_{ij}}\mathbf{v}_2)(\tilde{p}_{ij} - p_{ij}). \end{aligned} \tag{15}$$

From the last equality we can observe that the supergradient vector for $\mathbf{p}$ is:

$$\mathbf{g} = (\mathbf{v}_2^T\mathbf{E_{12}}\mathbf{v}_2, ..., \mathbf{v}_2^T\mathbf{E_{ij}}\mathbf{v}_2, ..., \mathbf{v}_2^T\mathbf{E_{N-1,N}}\mathbf{v}_2); \tag{16}$$

and for each component of p_{ij} the relative supergradient is:

$$g_{ij} = \mathbf{v}_2^T\mathbf{E_{ij}}\mathbf{v}_2 = -(v_{2_i} - v_{2_j})^2. \tag{17}$$

The updating rule for each element of the vector $\mathbf{p}$ is:

$$p_{ij}^{*(k+1)} = p_{ij}^{*(k)} + \alpha^{(k)}g_{ij}^{(k)} \tag{18}$$

where $\alpha^{(k)}$ and $g_{ij}^{(k)}$ are respectively the coefficient of the supergradient method and the supergradient at the step k.

Then the updated components $p_{ij}^{*(k+1)}$ have to be projected on the feasible set defined in (14). Since $\alpha^{(k)}g_{ij}^{(k)} \leq 0$ always, the update ensures that $p_{ij}^{*(k+1)} \leq p_{ij}^{*(k)}$. The projection on the feasible set requires only to satisfy the condition $p_{ij}^{*(k+1)} \geq -1$, which can be done by solving a least square problem, whose solution is:

$$p_{ij}^{*(k+1)} = \max(-1, p_{ij}^{*(k+1)}). \tag{19}$$

Equation (19) gives the update values of $L_{ij}^{*(k+1)}$. The values of $L_{ii}^{*(k+1)}$ are computed for all $i = 1, ..., N$ by:

$$L_{ii}^{*(k+1)} = -\sum_{j=1, j\neq i}^{N} L_{ij}^{*(k+1)}. \tag{20}$$

The decentralized supergradient updates each entry $L_{ij}^{*(k+1)}$ separately, by knowing the components $v_{2i}^{(k)}$ and $v_{2j}^{(k)}$ of the second eigenvector $\mathbf{v}_2$ of the Laplacian.

This means that each agent, in order to apply the supergradient, needs to know the components of the eigenvector $\mathbf{v}_2$ from its neighbors. Once this information is obtained, each agent uses the supergradient method to update the corresponding row of the Laplacian matrix. The step k of the supergradient algorithm applied by agent i is the following:

- compute $v_{2i}^{(k)}$;
- receive from the neighbors the updated values of $v_{2j}^{(k)}$, and send $v_{2i}^{(k)}$ to them;
- update the entries $g_{ij}^{(k)}$ using (17) for all $j \in C_i$;
- update the entries $L_{ij}^{*(k+1)}$ using (18) for all $j \in C_i$;
- project the updated $L_{ij}^{*(k+1)}$ on the feasible set using (19);

- compute $L_{ii}^{*(k+1)}$ using (20);

At the step $k = 0$, agent i receives information about the positions of the neighbors, and compute the row $\mathbf{L}_i^{(0)}$ with (3)-(4).

Next section describes the algorithm for the decentralized computation of the eigenvectors of the $\mathbf{L}$ matrix as given in [14].

B. Decentralized Computation of the eigenvectors of $\mathbf{L}$

The decentralized computation of the eigenvectors of $\mathbf{L}$ follows the Decentralized Orthogonal Iteration Algorithm (**DOI**) [14]. Let $\mathbf{w}_i$ be the row vector associated with agent i:

$$\mathbf{w}_i = [v_{1i}, v_{2i}, ..., v_{Ni}], \tag{21}$$

whose entries are the i-th component of all the eigenvectors of $\mathbf{L}$. Let $\mathbf{W}$ be the matrix:

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ ... \\ ... \\ \mathbf{w}_N \end{bmatrix} = \begin{bmatrix} v_{11} & v_{21} & ... & v_{N1} \\ v_{12} & v_{22} & ... & v_{N2} \\ ... & ... & ... & ... \\ ... & ... & ... & ... \\ v_{1N} & v_{2N} & ... & v_{NN} \end{bmatrix}, \tag{22}$$

whose columns are the eigenvectors of $\mathbf{L}$.

Assume that agent i receives the row $\mathbf{w}_j$ from each agent j connected with it. Agent i updates $\mathbf{w}_i$ by following the **DOI** algorithm:

- initialize $\mathbf{w}_i^{(0)}$ with a random vector;
- $\mathbf{e}_i^{(s+1)} = \mathbf{L}_i^{(k)}\mathbf{W}^{(s)}$, where $\mathbf{e}_i^{(s+1)}$ is the update of the row $\mathbf{w}_i$ at the step $s+1$ of the algorithm, $\mathbf{L}_i^{(k)}$ is the i-th row of the Laplacian matrix; since each element $L_{ij}^{(k)}$ is nonzero only for connected agents, the product $\mathbf{L}_i^{(k)}\mathbf{W}^{(s)}$ requires only the rows of the matrix $\mathbf{W}$ related with the agents connected with i;
- compute $\bar{e}_2^{(s+1)}$, that is the average value of the second eigenvector of $\mathbf{L}$;
- project the second eigenvector in the space $\mathbf{1}^{\perp}$: $e_{2i}^{(s+1)} = e_{2i}^{(s+1)} - \bar{e}_2^{(s+1)}$;
- $\mathbf{w}_i^{(s+1)}$=orthonormalization($\mathbf{e}_i^{(s+1)}$);
- repeat the loop until convergence of the vector $\mathbf{w}_i$.

Two steps of the **DOI** require attention: the computation of the average $\bar{e}_2^{(s+1)}$ and the orthonormalization. Each of them is carried out by a nested iteration. The two steps are analyzed in the following. Note that in the computation of the eigenvectors, $\mathbf{L}$ has index k, that is the index of the supergradient algorithm, while s is the index of the nested iteration necessary to compute the eigenvectors.

1) Computation of the average $\bar{e}_2^{(s+1)}$*:* it is necessary to project the second eigenvector of the Laplacian in the space $\mathbf{1}^{\perp}$. Agent i uses an heuristic averaging algorithm to compute the average $\bar{e}_2^{(s+1)}$:

- initialize $\bar{e}_i^{(0)} = e_{2i}^{(s+1)}$;
- $\bar{e}_i^{(r+1)} = \sum_{j=1}^{N} M_{ij}^{(k)} \bar{e}_j^{(r)}$;
- repeat the loop until convergence of each $\bar{e}_i$ to the average $\bar{e}_2^{(s+1)}$;

The matrix $\mathbf{M}$ is weighted, and its entries are given by:

$$\begin{aligned} M_{ij} &= \begin{cases} \frac{1}{\max(n_i, n_j)} & j \in C_i; \\ 0 & \text{otherwise;} \end{cases} \\ M_{ii} &= 1 - \textstyle\sum_{j \in C_i} M_{ij} \end{aligned} \tag{23}$$

where n_i and n_j are respectively the valencies of agents i and j, that is the number of agents connected respectively with agents i and j.
The convergence of the method to the real average among all agents is studied in [21].

2) Decentralized Orthonormalization: it derives from the centralized orthonormalization. In the centralized approach, the orthonormalization is typically performed by the factorization of the matrix $\mathbf{W}$ in the form: $\mathbf{E} = \mathbf{WH}$, where the rows of the matrix $\mathbf{E}$ are the not orthonormal eigenvectors $\mathbf{e}_i$ (the same that are computed by the **DOI** from each agent), and $\mathbf{H}$ is a upper triangular matrix to find. To find $\mathbf{H}$, we define the matrix $\mathbf{S} = \mathbf{E}^T\mathbf{E}$, and compute the product:

$$\mathbf{E}^T\mathbf{E} = \mathbf{H}^T\mathbf{W}^T\mathbf{WH} = \mathbf{H}^T\mathbf{H}.$$

The last equality is valid because $\mathbf{W}$ is orthonormal. From that equality the matrix $\mathbf{S}$ can be found by the Cholesky factorization:

$$\mathbf{S} = \mathbf{H}^T\mathbf{H} \Rightarrow \mathbf{H} = chol(\mathbf{S}).$$

In the decentralized case agent i does not know the entire $\mathbf{S}$ matrix. But we can observe that $\mathbf{S}$ can be written as:

$$\mathbf{S} = \mathbf{E}^T\mathbf{E} = \sum_{i=1}^{N} \mathbf{e}_i^T\mathbf{e}_i = \sum_{i=1}^{N} \mathbf{S}_i. \tag{24}$$

Since agent i computes $\mathbf{e}_i$, it knows the matrix $\mathbf{S}_i$ and it receives the matrices $\mathbf{S}_j$ from its neighbors. Thus, by using once again the heuristic averaging algorithm, an average of the $\mathbf{S}$ matrix is computed, and the real value of $\mathbf{S}$ is simply obtained by multiplying the average for the number N of the agents in the graph. The proposed algorithm is:

- initialize $\mathbf{S}_i^{(0)} = \mathbf{e}_i^T\mathbf{e}_i$;
- $\mathbf{S}_i^{(r+1)} = \sum_{j=1}^{N} M_{ij}^{(k)}\mathbf{S}_j^{(r)}$;
- repeat the loop until convergence;
- compute $\mathbf{S} = \mathbf{S}_i N$.

IV. Control action for each agent

The computation of the supergradient and the update of the optimum Laplacian $\mathbf{L}^{*(k)}$ at the k-th step is an open loop computation. Since the agents move on the plane and their motion causes the variation of the state dependent Laplacian $\mathbf{L}(\mathbf{z})$, there is at each step of the supergradient algorithm an error between the current Laplacian $\mathbf{L}(\mathbf{z})$ and the optimum value $\mathbf{L}^{*(k)}$. This error can be minimized by applying to each agent a decentralized control action that drives the group toward a configuration corresponding to the optimal Laplacian $\mathbf{L}^{*(k)}$.

The optimization problem can be formulated for the agent i as:

$$\min_{z_i} \|\mathbf{L}_i(\mathbf{z}) - \mathbf{L}_i^{*(k)}\|_2^2, \tag{25}$$

where $\mathbf{L}_i(\mathbf{z})$ is the row i of the Laplacian matrix, as function of the state of the agents, and $\mathbf{L}_i^{*(k)}$ is the row of the optimal Laplacian found by agent i at the step k of the supergradient. The minimization problem is solved by using potential functions.

From each $L_{ij}^{*(k)}$, by using (3) and by reversing (4), we obtain the desired distance $\delta_{ij}^{(k)}$ between connected agents i, j:

$$\delta_{ij}^{(k)} = A_{ij}^{-1}(L_{ij}^{*(k)}). \tag{26}$$

For each pair of connected agents (i, j) a quadratic potential function $V_{ij}(\|r_{ij}\|)$ is defined as:

$$V_{ij} = \begin{cases} (\|r_{ij}\| - \delta_{ij}^{(k)})^2 & \|r_{ij}\| \leq R; \\ (R - \delta_{ij}^{(k)})^2 & \|r_{ij}\| > R. \end{cases} \tag{27}$$

The potential function $V_{ij}(\|r_{ij}\|)$ is positive definite, it is zero when the distance $\|r_{ij}\|$ is equal to the desired value $\delta_{ij}^{(k)}$, and it becomes constant when the distance $\|r_{ij}\|$ is greater than the connection radius R.

The control action for agent i is defined by solving the optimization problem, equivalent to (25):

$$\min_{z_i} \sum_{j \in C_i} V_{ij}, \tag{28}$$

from which the control action on agent i is defined as the sum of the negative gradients of the potentials V_{ij} for all $j \in C_i$:

$$u_i = -\sum_{j \in C_i} \nabla_{z_i} V_{ij}. \tag{29}$$

Each agent i moves according to the control action (29), until an equilibrium for the group is reached. At the equilibrium, agent i stops and sends the information about its current position $z_i(t)$ to the neighbors, and receives their current positions. With this information, agent i updates the row i of the Laplacian matrix $\mathbf{L}(\mathbf{z})$. This updated row will be used from agent i at the step $k+1$ of the supergradient.

The potential based control applied by the agents does not ensure the minimization of (25), because it presents local minima if the graph of connections of the group is not a tree. Thus, an analysis of the convergence of the supergradient algorithm is necessary to show that the error on the Laplacian matrix does not affect the convergence to the optimum λ_2.

A. Convergence of the supergradient algorithm

As we did in Section III-A, we call $\mathbf{p}^{*(k+1)}$ the vector of components of the updated Laplacian $\mathbf{L}^{*(k+1)}$ at the step $k+1$ of the supergradient. The application of the potential control to the agents move them to the a new configuration on the plane corresponding to a different vector, called $\tilde{\mathbf{p}}^{(k+1)}$. The difference between the two vectors is an error vector:

$$\tilde{\mathbf{p}}^{(k+1)} = \mathbf{p}^{*(k+1)} + \varepsilon^{(k+1)}. \tag{30}$$

If we call the error at the step $k+1$ of the supergradient, between the vector $\mathbf{p}^{*(k+1)}$ and the final optimal vector $\mathbf{p}^*$, corresponding to the maximum value of λ_2:

$$\mathbf{e}^{(k+1)} = \mathbf{p}^{*(k+1)} - \mathbf{p}^*, \tag{31}$$

then the error between the real vector $\tilde{\mathbf{p}}^{(k+1)}$ and the optimal vector is:

$$\tilde{\mathbf{p}}^{(k+1)} - \mathbf{p}^* = \mathbf{p}^{*(k+1)} + \varepsilon^{(k+1)} - \mathbf{p}^* = \mathbf{e}^{(k+1)} + \varepsilon^{(k+1)} \tag{32}$$

The error can be limited in norm (see [19] for a similar proof):

$$\begin{aligned} \|\tilde{\mathbf{p}}^{(k+1)} - \mathbf{p}^*\|_2^2 &= \|\mathbf{e}^{(k+1)} + \varepsilon^{(k+1)}\|_2^2 = \|\mathbf{e}^{(k+1)}\|_2^2 \\ &+ \|\varepsilon^{(k+1)}\|_2^2 + 2\mathbf{e}^{(k+1)T}\varepsilon^{(k+1)}; \end{aligned} \tag{33}$$

The last term of the previous relation can be rewritten as:

$$\begin{aligned} 2\mathbf{e}^{(k+1)T}\varepsilon^{(k+1)} &= \|\varepsilon^{(k+1)}\|_2^2 + \|\mathbf{e}^{(k+1)}\|_2^2 \\ &- \|\varepsilon^{(k+1)} - \mathbf{e}^{(k+1)}\|_2^2 \\ &\leq \|\varepsilon^{(k+1)}\|_2^2 + \|\mathbf{e}^{(k+1)}\|_2^2; \end{aligned} \tag{34}$$

By substituting (34) in (33), using (18) and the definition of supergradient, the error becomes:

$$\begin{aligned} \|\tilde{\mathbf{p}}^{(k+1)} - \mathbf{p}^*\|_2^2 &\leq 2\|\varepsilon^{(k+1)}\|_2^2 + 2\|\mathbf{e}^{(k+1)}\|_2^2 \\ &\leq 2\|\varepsilon^{(k+1)}\|_2^2 + 2[\|\tilde{\mathbf{p}}^{(k)} - \mathbf{p}^*\|_2^2 \\ &+ {\alpha^{(k)}}^2\|\mathbf{g}^{(k)}\|_2^2 + 2\alpha^{(k)}(\lambda_2(\tilde{\mathbf{p}}^{(k)}) - \lambda_2^*)]; \end{aligned} \tag{35}$$

From the last expression, since $\|\tilde{\mathbf{p}}^{(k+1)} - \mathbf{p}^*\|_2^2 \geq 0$, the error between the current value of λ_2 and the optimal λ_2^* can be bounded:

$$\begin{aligned} 2\alpha^{(k)}(\lambda_2^* - \lambda_2(\tilde{\mathbf{p}}^{(k)})) &\leq \|\varepsilon^{(k+1)}\|^2 + \|\tilde{\mathbf{p}}^{(k)} - \mathbf{p}^*\|^2 \\ &+ {\alpha^{(k)}}^2\|\mathbf{g}^{(k)}\|_2^2. \end{aligned} \tag{36}$$

By considering the first k steps of the supergradient, a bound on the error between the optimal value $\lambda_{2_{best}}$ reached with the supergradient, and the final optimal value λ_2^* can be found:

$$\begin{aligned} \lambda_2^* - \lambda_{2_{best}} &\leq \frac{\sum_{i=1}^k \|\varepsilon^{(i+1)}\|_2^2}{2\sum_{i=1}^k \alpha^{(i)}} + \frac{\sum_{i=1}^k {\alpha^{(i)}}^2\|\mathbf{g}^{(k)}\|_2^2}{2\sum_{i=1}^k \alpha^{(i)}} \\ &+ \|\tilde{\mathbf{p}^{(1)}} - \mathbf{p}^*\|_2^2. \end{aligned} \tag{37}$$

If the series of the error norms $\|\varepsilon^{(k)}\|_2^2$ is not divergent, or diverges with a rate less than the series of $\alpha^{(k)}$, and the norm of the supergradient matrix is bounded at each step k, $\|\mathbf{g}^{(k)}\|_2^2 \leq G$, then, by choosing the series of $\alpha^{(k)}$ as square summable, the error is bounded and the supergradient converges.

V. Simulation results

In this section some simulations will be shown to test the effectiveness of the proposed algorithm. The multi-agent system is composed by 6 agents on the plane. The connection radius is $R = 30$. The maximum theoretical value of $\lambda_2(\mathbf{L})$ is equal to N, and it is reached if each agent reaches a distance at most equal to ρ with all the others. The chosen step-size for the supergradient algorithm is $\alpha^{(k)} = 1/(k+1)$, thus ensuring the convergence of the algorithm.

In the first simulation the minimum distance is $\rho = 2$. Figure 2 shows respectively, from left to right, and from up to down, the value of λ_2, the initial configuration, an

intermediate configuration, and the final configuration of the agents on the plane. The circles are the agents, the lines are the connections among agents. The graph of connections in the intermediate configuration is completely connected, and it corresponds to the iteration 60 of the supergradient, where $\lambda_2 \approx 4$. The final configuration, corresponding to $\lambda_2 = 5$, has one agent, at the center of the formation, at distances equal to ρ with all others, and all the other distances greater than ρ.

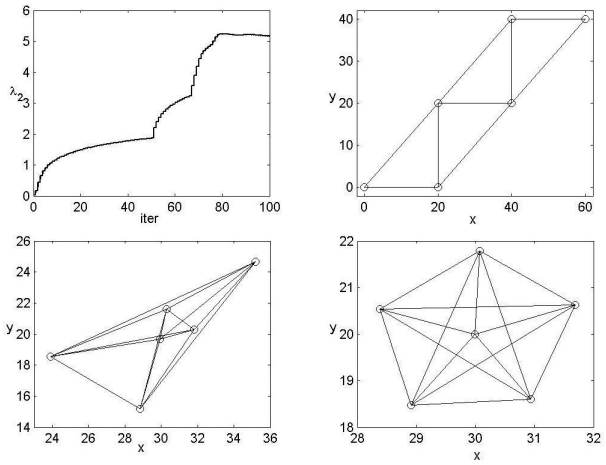

Fig. 2. From left to right, from up to down: value of λ_2 during the iterations, initial, intermediate and final configuration of the agents on the plane. Agents are represented by 'o' and links by lines.

In the second simulation we have $\rho = 0$, thus the maximum λ_2 is obtained when agents reach the same point in the plane (rendezvous). Since the control action for the agents does not include the collision avoidance, it is possible for the agents to reach the rendezvous, as shown in Figure 3.

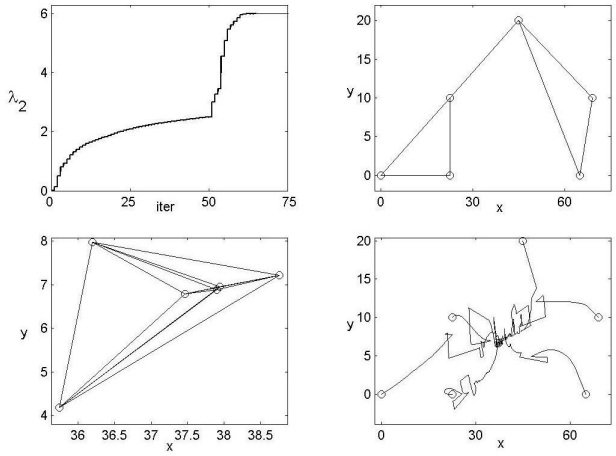

Fig. 3. From left to right, from up to down: value of λ_2 during the iterations, initial and intermediate configuration of the agents on the plane, motion of the agents on the plane.

VI. Conclusions

In this paper we proposed an iterative decentralized algorithm for the connectivity control of a multi-agent system. The algorithm maximizes the second smallest eigenvalue of the Laplacian matrix by a supergradient method, in conjunction with a potential based control that drives the agents toward a formation defined at each step of the supergradient. The convergence of the supergradient is shown analytically and by simulations.

References

[1] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[2] N. L. P. Ogren, E. Fiorelli, "Cooperative control of mobile sensing networks: Adaptive gradient climbing in a distributed environment," *IEEE Transaction on Automatic Control*, vol. 49(8), pp. 1292–1302, August 2004.

[3] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, February 2004.

[4] R. O. Saber, "Flocking in multiagent dynamic systems: Algorithms and theory," *IEEE Trans. on Automatic Control*, vol. 51, no. 3, Mar 2006.

[5] E. Justh and P. Krishnaprasad, "Equilibria and steering laws for planar formations," *Systems and Control letters*, vol. 52, no. 1, pp. 25–38, May 2004.

[6] F. Cucker and S. Smale, "Emergent behavior in flocks." [Online]. Available: http://math.berkeley.edu/ smale/

[7] J. Cortes, S. Martinez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1289–1298, August 2006.

[8] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, "Distributed topology control for wireless multihop ad-hoc networks," in *INFOCOM*, 2001, pp. 1388–1397. [Online]. Available: citeseer.ist.psu.edu/wattenhofer01distributed.html

[9] A. Muhammad and M. Egerstedt, "Connectivity graphs as models of local interactions," *Proceedings of the 43th IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas*, December 2004.

[10] D. Spanos and R. M. Murray, "Robust connectivity of networked vehicles," *Proceedings of the 43th IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas*, December 2004.

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

[12] M. M. Zavlanos and G. J. Pappas, "Controlling connectivity of dynamic graphs," *Proceedings of the 44th IEEE Conference on Decision and Control, Seville, Spain*, December 2005.

[13] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis and applications," *Proceedings of IEEE Infocom 2005, vol. 3, pp. 1653-1664, Miami*, March 2005.

[14] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," *Proceedings of STOC 2004, Chicago, Illinois, USA*, June 2004.

[15] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. on Robotics and Automation, vol. 8, no. 5, pp. 501-518*, 1992.

[16] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Transaction on Automatic Control*.

[17] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer, 2001.

[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2003.

[19] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *Lecture Notes*, October 2003.

[20] D. Bertsekas, *Nonlinear Programing*. Athena Scientific, 1999.

[21] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, Hawaii, USA*, December 2003.