Distributed connectivity control in a dynamic network

Distributed Connectivity Control in A Dynamic Network

Ren Zhong and Shigeru Yamamoto

Abstract— **In this paper, we propose a distributed algorithm to control connectivity of mobile agents in a constrained dynamic network. The connectivity property of the multi-agent system is quantified by the second smallest eigenvalue of the state dependent Laplacian of the proximity graph of agents. Motivated by k-regular graph characteristics, we consider a discrete-time model for autonomous agents. We explore an approach based on tentative overlay as a condition mapping from interrelation and interaction between each agent, which equipped with local sensing and wireless communication capabilities. It thereby shows that load balancing can be controlled with network's connectivity, while other than nearest neighbor information, our approach assumes no knowledge of global network topology. Simultaneously, a new collision avoidance method motivated by vertex coloring is presented for our distributed approach. In addition, simulations are provided that demonstrate the effectiveness of our theoretical results, for which we show a distributed dynamic programming of multi-agent system.**

Index Terms— **Distributed control, Dynamic networks, Multi-agent systems**

I. INTRODUCTION

The advances of distributed coordination in dynamic multi agents have attracted several researchers in recent years. Particularly, achieving a global coordinated objective while only using local information gives us a great new challenge that relies on algebraic graph theory, matrix theory, and control theory. The broad applications of multi agents include formation control [1], flocking [2]-[4], consensus problems [5]-[9], mobile peer-to-peer networks [10], and so on. Comparing to conventional centralized control, distributed control of multi-agent systems provide increased efficiency, performance, scalability, and robustness.

A control mechanism of a group of mobile agents to form a designated formation while flocking within a constrained environment is seen in [1]. It is motivated by Reynolds' flocking model [2], which has a mechanism for achieving velocity synchronization and regulation of relative distances within a group of agents. In [2], the aggregate motion of a flock of birds was simulated for animation industry. In [3], a discrete-time model of agents was proposed. Each agent's heading is updated using a local rule based on the average of its own heading plus the headings of its neighbors. Moreover, there has been a tremendous amount of renewed attracting on flocking like [4]. Since the aggregate motion is applied in many different areas, the emphasis in this note is not only on decentralized coordination method, but also interesting in consensus problems and avoided collision.

Closely related to the topics discussed in this paper is also work in solving consensus problems. Two consensus protocols are introduced by [5], in which disagreement functions are also introduced for convergence analysis of consensus protocols that a Lyapunov function for the disagreement network dynamics. From [6], the authors provide a convergence analysis of linear and nonlinear protocols for undirected networks in presence or lack of communication time-delays. The work in [7] focuses on attitude alignment on undirected graphs in which the agents have simple dynamics motivated by the model used in [3].

Following the above footsteps, a distributed algorithms that can be used by multiple agents to align their estimates with a particular value over a network with time-varying connectivity is investigated in [8]. To highlight the effects of constraints, a constrained consensus problem is considered and a distributed algorithm is presented in which agents combine their local averaging operation with projection on their individual constraint sets. Furthermore, in [9], a semidefinite programming approach is used for reaching average-consensus, which partially relies on the work in [6]. And especially, the recent rise of mobile peer to peer networks as [10] in which builds up a common applicable theoretical framework can relied on our research and get more challenges.

In this paper, our analysis relies on several tools mainly coming from algebraic graph theory [11]-[13]. Especially, [14] and [15] give us a great deal of motivation for our intensive study. We coordinate connectivity, performance and dispersion of agents by using distributed control in network. The outline of this paper is as follows. In Section II, we give our model under algebraic graph theory and our problem formulation. In Section III, a topological distributed algorithm is given in our constrained dynamic network and we also give our proof for connectivity and load balancing. Section IV shows our simulation results. In Section V, concluding remarks are stated.

II. GRAPH REPRESENTATION AND PROBLEM FORMULATION

We consider a dynamic multi-agent network with n agents described by a weighted undirected graph $G = (V, E, A)$, where a finite set of *vertices* $V = \{v_1, v_2, ..., v_n | v \in S\}$, a set of ordered *edges* $E \subseteq V \times V$, and a adjacency matrix $A = [a_{ij}]$ whose entries $a_{ij} = 1$ if $(v_i, v_j) \in E$ and $a_{ij} = 0$ otherwise. Since we do not allow self-loops, for each i we define $a_{ii} = 0$. The ith agent is assigned

Ren Zhong is with Graduate School of Natural Science and Technology, Kanazawa University, Kakuma-machi, Kanazawa, Ishikawa, 920-1192 JAPAN renzhong@moccos.ec.t.kanazawa-u.ac.jp

Shigeru Yamamoto is with Faculty of Electrical and Computer Engineering, Kanazawa University, Kakuma-machi, Kanazawa, Ishikawa, 920-1192 JAPAN shigeru@t.kanazawa-u.ac.jp

978-1-4244-8736-3/11/$26.00

to node v_i. The edges $e_{ij} = (v_i, v_j)$ among each pair of agents is assumed to be dependent on the communication links. The communication capabilities give agents a potential communication bound, which is denoted by a circle centered on the agent i and given radius r_i. The r_i depends on communication capability of agent i. In practical application, every agent has a different radius probably. However, it is more complex. In our condition, we make an assumption for the communication capacities.

Assumption 2.1: Each agent has equivalent communication capability as $r_i \equiv R$, ($i \in n$ and $R \in \mathbb{R}^+$ is a positive scalar), while $\mathbb{R}^+$ is a set of positive real numbers.

Let $d_i \in \mathbb{R}$ denote the degree of node v_i. Suppose each node of the graph is a dynamic agent with dynamics

$$d_i(t+1) = f(d_i(t), u_i(t)) \tag{1}$$

for some function f with d_i denoting the degree of agent i. The choice of f is not only guided by particular applications, but also by numerical considerations. Our Algorithms take the place of the input control $u_i(t)$. We refer to $G_d = (G, d)$ with $d = (d_1, d_2, ..., d_n)^T$ as a network or algebraic graph with $d \in \mathbb{R}^n$ and topology G. A dynamic graph is a dynamical system with a state (G, d) in which the value d evolves according to the network dynamics

$$d(t+1) = F(d(t), u(t)). \tag{2}$$

Here, the ith element of $F(d, u)$ is $f(d_i, u_i)$.

The graph G can be also represented using the graph Laplacian matrix:

$$L(G) = D(G) - A(G) \tag{3}$$

where $D = diag(d_1, d_2, ..., d_n)$ is a diagonal matrix with elements d_i. Let $d_i = \sum_j a_{ij}$ denote the *degree* of agent i. Obviously, the network is in spectral properties of complex networks with symmetric weights $A = A^T$. It is known that L is positive semi-definite and symmetric, its eigenvalues are all nonnegative. Let us denote the eigenvalues of L by

$$0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n \tag{4}$$

in a increasing way. The eigenvector corresponding to the first eigenvalue is always $\mathbf{1}$. The second eigenvalue λ_2 is called *algebraic connectivity* of the network, 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. In other words, $\lambda_2 > 0$ if and only if the graph G is connected. Generally speaking, λ_2 is function of the state of the entire system, thus we can write it as $\lambda_2(L(t))$.

In a dynamic network with switching topology the information flow G is a discrete-state of the system that changes in time. The collective dynamics of n agents applying this consensus mechanism is

$$d(t+1) = -Ld(t). \tag{5}$$

The main objective of this paper is to find a self-organizing mechanism to control the connectivity of dynamic network 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:

$$\lim_{t\to\infty} \lambda_2(L(t)) > 0, \text{and } \lim_{t\to\infty} (d_i(t) - d_j(t)) = 0. \tag{6}$$

The average degree of G denoted $\bar{d} = \frac{1}{n}\sum_i d_i$ is a measure of density of graphs. A graph G is called *k-regular* if all of its nodes have degree k. The size of a graph $|E| = \frac{n\bar{d}}{2}$ is directly determined by its average degree and scale n. A graph is called a *complex network* for large n's. In many practical applications, for example, in computer networking, it is necessary to discuss the degree k as load capacity. Technique to distribute workload evenly across two or more computers, network links, or other resources, in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload is called *load balancing*. Theoretically, a equal bandwidth network with optimality load balancing looks like a k-regular graph.

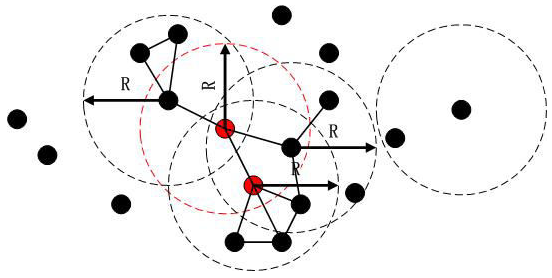


Fig. 1. Neighbor-subgraph

III. A SELF-ORGANIZING MECHANISM ON DYNAMIC NETWORK

In this section, we present our algorithm for each agent. We ultimately obtain an stable state that will characterize connectivity of the network in terms of relative positions and broadcast ranges. We define d_{ij} as the pairwise distance between v_i and v_j to express multi-agents equipped with sensors whose resolution is decaying exponentially with the distance to the object to observe. If the distance is at most equal to the fixed connection radius R, then the agents are said to be neighbors. The set of *neighbors* of the agent i is denoted by

$$N_i = \{v_j \in V | 0 \leq d_{ij} \leq R\}. \tag{7}$$

We give our definition of subgraph associated with neighbor agents, named *neighbor-subgraph*.

Definition 3.1 (Neighbor-Subgraph): A neighor-subgraph with agent i of graph G is a graph $G_i^s = (V_i^s, E_i^s, A_i^s)$ such that $V_i^s = \{v_i\} \cup N_i$, $E_i^s = \{(v_j, v_k) | v_j \in V_i^s, v_k \in N_j \cap V_i^s\}$, and $A_i^s = [a_{ij}]_i \in \mathbb{R}^{s\times s}$, where $a_{ij} = 1$ if $i \in N_j$ and $a_{ij} = 0$ otherwise.

Fig.1 is a simple example of a neighbor-subgraph. In this diagram, the subgragh of ith agent is denoted in a dashed

circle with agent i as the centre of this circle. The radius of this circle is determined by communication capability of the agent i which is limited by Assumption 2.1. Let $\mathfrak{C}_n$ be the set of all connected graphs include n agents. The graph, which is connected at time instant t, is expressed as $\hat{G} \in \mathfrak{C}_n$. Then, we have the second restricted condition as:

Assumption 3.1: The graph $G(0)$ and $G_i^s(0)$ for all $i \in n$ is initially connected, i.e., $G(0) = (V(0), E(0), A(0)) \in \mathfrak{C}_n$, $G_i^s(0) = (V_i^s(0), E_i^s(0), A_i^s(0)) \in \mathfrak{C}_n$.

A. Initial Synchronization and Collision Avoidance

The presence of time delays in the network, as well as the network topology that imposes multihop communication patterns between agents, can result in the agents reaching a decision on an asynchronously. This imply that network processing could result in mixing of information between two agents and preventing all agents from reaching a common outcome. Hence, our mechanism based on distributed control framework relies on some notion of synchronization and asynchronization of all agents. For the active agent, faster agents are forced to wait for their slower neighbors, which guarantees synchronization of all agents to the same time interval. Furthermore, if there are two agents of mutually neighborhood create connection or destroy connection at the same time, it must exist a collision. Avoiding this collision, agent i and its neighbors should execute Algorithm 3.1 asynchronously. Here, we give our assumption for this as:

Assumption 3.2 (Synchronization): All agents are initial asynchronized. And an agent always asynchronize with its neighbors.

Motivated by graph coloring in graph theory, we use same color to display synchronized nodes. By graph coloring, the collision avoidance problem is simplified as a vertex coloring problem such that no two vertices sharing the same edge have the same color. The Welch-Powell Algorithm [19] for a coloring of a graph G is presented. It doesn't always yield a minimal coloring of G. The main idea of Welch-Powell Algorithm is to order the vertices of G according to decreasing degrees and assign one color to each vertex which is not adjacent to a previous vertex which was assigned.

Considering under our situation, we do not need order the whole vertices cause of just knowing information between neighbors. Let different color be denoted by different RGB color number. We give Algorithm 3.1 involved in the inital synchoronced set s_i for each agent i. This algorithm is a distributed algorithm. Firstly, we define a vector C_i with n binary numbers as estimated queue using 0 or 1. We denote by 0 that the agent i is asynchronized by agent j, where $j := \arg\min\{k|C_i(k) = 1\}$. On the other hand, we denote by 1 that the agent i is synchronised with the agent j. Let agent i's neighbor set N_i also translate into this 0,1 Beanlen calculating form set N_i^*, i.e., $n = 4, i = 1, N_i = \{2, 3\}$ so $N_i^* = [0, 1, 1, 0]$. Then, we present our algorithm follow the function

$$C_i = \overline{N_i^*}. \tag{8}$$

Then, we put agent i itself into set s_i. When it contains 1 in C_i, we get $j := \arg\min\{k|C_i(k) = 1\}$ as a synchronous agent for agent i. The major view and methods of the algorithm design adopted in constructing synchronism sequence s_i throngh an alternative set C_i. Furthermore, i and j should be synchronized as:

$$C_i = C_i \cap C_j, C_j = C_i. \tag{9}$$

Algorithm 3.1: The flag set s_i for collision avoidance.

Input:
Neighbor-Subgraph $G_i^s = (V_i^s, E_i^s, A_i^s)$ for all $i \in \{1, 2, \ldots, n\}$.
Output:
A flag set s_i for all $i \in \{1, 2, \ldots, n\}$.

First, we should give initial $s_i = \phi$ for all agent i.
1. $s_i = \{i\}$, $C_i = \overline{N_i^*}$.
2. $C_i(s_i) = 0$.
3. if $\bigcup C_i \neq 0$
4. $\quad j := \arg\min\{k|C_i(k) = 1\}$.
5. end if;
6. if $C_j \supseteq s_i$.
7. $\quad C_i = C_i \cap C_j, C_j = C_i$.
8. end if;
9. $s_i \leftarrow j$, $s_j \leftarrow i$

Theorem 3.1 (Brook's Theorem[18]): For any connected undirected graph G with maximum degree Δ, the chromatic number of G is at most Δ unless G is a clique or an odd cycle, in which case the chromatic number is $\Delta + 1$.

Bounds on the chromatic number is mentioned by Theorem.3.1 as above. It is proofed for graph theory and tell us our algorithm is certainly convergent.

B. An Algorithm by Using Neighbor Estimation

Under the assumptions, we present an algorithm to derive the multi-agent network topology indicated by a graph $G = (V, E, A)$ using neighbor estimation in the time t. Actually, this algorithm is a local distributed algorithm and each agent i run it independently with its subgraph expressed by $G_i^s = (V_i^s, E_i^s, A_i^s)$ and its state flag set s_i which is used to sign synchronous working group of agent i. In the algorithm, we start with

$$V_i^s = \{i, N_i\}, E_i^s = \phi, A_i^s = \phi. \tag{10}$$

According to maximum connected agents, we give E_i^s an initial value

$$E_i^s = \{(v_j, v_k)|v_j \in N_{v_k}\} \tag{11}$$

and A_i^s as

$$A_i^s = [a_{ij}]_i^s = \begin{cases} a_{ij} = 1 & \text{if } i \in N_j \\ a_{ij} = 0 & \text{otherwise} \end{cases}. \tag{12}$$

Let the maximum and minimum degrees are $\Delta_{\max}$ and $\Delta_{\min}$. The difference $\Delta_g = \Delta_{\max} - \Delta_{\min} \geq 0$ will be called the degree gap. The symbol $|.|$ is denoted to count to the number of inside elements. When the flag $s_i \neq \phi$, we compute the average degree of agents as

$$\Delta_d = \frac{1}{|N_i|} \sum_{k=i,N_i} d_k. \tag{13}$$

Algorithm 3.2: The Distributed Control for Each Agent i.

Input:
Neighbor-Subgraph $G_i^s = (V_i^s, E_i^s, A_i^s)$ of agent i and status set s_i of agent i at time t.
Output:
Real connected neighbor-subgraph $G_i^{s'} = (V_i^{s'}, E_i^{s'}, A_i^{s'})$ of agent i.

```
for time → ∞ then
V_i^{s'} = V_i^s
if s_i > 0 then
    Δ_d = (1/|N_i|) Σ_{k=i,N_i} d_k
    if Δ_d < d_i then
        ϑ(v_i^s) = {v_i^s ∈ N_i | λ_2(L_i^s(E_i^s/(v_i^s, v_j^s))) > 0}
    if |ϑ(v_i^s)| > 0 do
        v_max = arg max |N_{i∈ϑ(v_i^s)}|
        E_i^{s'} = E_i^s/(v_i^s, v_max)
        E_{v_max}^{s'} = E_{v_max}^s/(v_i^s, v_max)
        %Remove agent v_max from network.
    end if;
    if Δ_d > d_i and |E_{N_i}| − |E_i| ≠ φ
        v_min = arg min |N_i|
        E_i^{s'} = E_i^s ∪ (v_i^s, v_min)
        E_{v_min}^{s'} = E_{v_min}^s ∪ (v_i^s, v_min)
        %Add agent v_min into network.
    end if;
    N_i = {v_j^s ∈ V | 0 ≤ d_ij ≤ R}
end if;
```

Then, we compare the degree d_i of the agent i with the average degree Δ_d. When $d_i > \Delta$, we obtain the eligible neighbors firstly by

$$\vartheta(v_i^s) = \{v_i^s \in N_i | \lambda_2(L_i^s(E_i^s/(v_i^s, v_j^s)) > 0\}. \tag{14}$$

Then, we use this $\vartheta(v_i^s)$ to limit the connected edges for the agent i. Oppositely, when $\Delta_d > d_i$, we extend a new edge when the connected edges are not maximum, where $|E_{N_i}| - |E_i| \neq \phi$. We designed Algorithm 3.2 of distributed control for each agent in every time. We guarantee the connectivity all the time and approach the degree of agents closer and closer. We give an algorithm for the agent i to establish the connected graph $G_i^{s'} = (V_i^{s'}, E_i^{s'}, A_i^{s'})$ as Algorithm 3.2.

C. Connectivity and Consensus Analysis

Lemma 3.1 (Algebraic Connectivity[17]): Let $\lambda_1(G) \leq \lambda_2(G) \leq ... \leq \lambda_n(G)$ be the ordered eigenvalues of the Laplacian matrix $L(G)$. Then, $\lambda_1(G) = 0$, with corresponding eigenvector $\mathbf{1}$. Furthermore, $\lambda_2(G) > 0$ if and only if graph G is connected and hence $\lambda_2(G)$ is called the *algebraic connectivity* of G.

Proposition 3.1: Under the assumptions, $G(0)$ and $G_i^s(0)$ is connected at the initial time. Algorithm 3.2 ensures that $G_i^s(t)$ is connected for all agent $i \in \{1, 2, \ldots, n\}$ and $t \geq 0$.

Proof: According to Lemma 3.1, $G_i^s(0)$ and $G_i^s(t)$ are connected for all agent i and $t \geq 0$, if and only if $\lambda_2(G_i^s(t)) > 0$ for all $t \geq 0$. Moreover, $\lambda_2(G(0)) > 0$. We define the union graph $G_{ij}^s(V_{ij}^s, E_{ij}^s) = G_i^s(V_i^s, E_i^s) \cup G_j^s(V_j^s, E_j^s)$ by including all edges and the vertices in G_i^s and G_j^s, as $V_{ij}^s = V_i^s \cup V_j^s$, $E_{ij}^s = E_i^s \cup E_j^s$. Accoding to matrix theory and graph theory, we calculate the adjacency matrix A_{ij}^s as:

$$A_{ij}^s = P_{ij} A_i^s P_{ij}^T + P_{ji} A_j^s P_{ji}^T, \tag{15}$$

where P is denoted by unit matrix $I(A_i^s)$ and some zero row vector. We rearrange the A_i^s and A_j^s. The depended row of A_i^s is component of $I(A_i^s)$ and the other row dependent on A_j^s use zero to built. i.e.,

$$A_i^s = \begin{matrix} a_1.- \\ a_2.- \\ a_3.- \end{matrix} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, A_j^s = \begin{matrix} a_2.- \\ a_3.- \\ a_4.- \end{matrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

$$P_{ij} = \begin{matrix} a_1.- \\ a_2.- \\ a_3.- \\ a_4.- \end{matrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, P_{ji} = \begin{matrix} a_1.- \\ a_2.- \\ a_3.- \\ a_4.- \end{matrix} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

From (15), we can derive that

$$L_{ij}^s = P_{ij} L_i^s P_{ij}^T + P_{ji} L_j^s P_{ji}^T. \tag{16}$$

Moreover, $\lambda_2(L_i^s) > 0$ and $\lambda_2(L_j^s) > 0$ have been given. And then, $\lambda_2(L_{ij}^s) > 0$ if and only if $|L_{ij}^s| < |L_i^s| + |L_j^s|$. It is also said that $\exists k \in G_i^s$ and one of its neighbors $l \in G_j^s$, where $l \in N_k$.

(i) If $G_i^s(t) = G(t)$, obviously $\lambda_2(G_i^s(t)) > 0$ iff $\lambda_2(G(t)) > 0$.

(ii) If $G_i^s(t) \subset G(t)$, we union the whole neighbor-subgraph set $G^s = \{G_1^s, G_2^s, \ldots, G_n^s\}$ as:

$$G = G_1^s \cup G_2^s \cup \cdots \cup G_n^s. \tag{17}$$

We divide this union into two parts S_1 and $S2$, which is denoted by

$$G = (G_1^s \cup G_2^s \cup \cdots \cup G_i^s) \cup (G_j^s \cup \cdots \cup G_n^s) \tag{18}$$

while

$$S_1 = G_1^s \cup G_2^s \cup \cdots \cup G_i^s, S_2 = G_j^s \cup \cdots \cup G_n^s. \tag{19}$$

Obviously, for $G(0) = S_1(0) \cup S_2(0)$, $\exists g \in S_1$ and one of its neighbors $l \in S_2$, $l \in N_g$. Assume that $G(t)$ is not connected and then there is $G(t) = S_1(t) \cup S_2(t), \forall k \in S_1$ and its neighbors are not in S_2. It is contradictory such that exists an agent g in graph S_1 has a neighbor in graph S_2 as well as hasn't any neighbor in graph S_2. The proof is then easily completed.

■

Networks with hubs having very large degrees that are commonly known as scale-free networks are fragile to time-delays. In contrast, random graphs and small-world networks are fairly robust to time-delays since they do not have hubs. In addition, construction of engineering networks with nodes that have high degrees is not a good idea for reaching a consensus.

Proposition 3.2: By Algorithm 3.2, the degree of every agents converges to the same value.

Proof: We set that

$$x(0) = \begin{bmatrix} \frac{\mathbf{1}^T A_1 \mathbf{1}}{e_1^T A_1 \mathbf{1}} \\ \frac{\mathbf{1}^T A_2 \mathbf{1}}{e_2^T A_2 \mathbf{1}} \\ \cdots \\ \frac{\mathbf{1}^T A_n \mathbf{1}}{e_n^T A_n \mathbf{1}} \end{bmatrix}, x(1) = \begin{bmatrix} \frac{e_1^T A_1 x_1(0)}{e_1^T A_1 \mathbf{1}} \\ \frac{e_2^T A_2 x_2(0)}{e_2^T A_2 \mathbf{1}} \\ \cdots \\ \frac{e_n^T A_n x_n(0)}{e_n^T A_n \mathbf{1}} \end{bmatrix}. \quad (20)$$

The system can be represented as

$$\begin{aligned} x(t+1) &= (I+D)^{-1}(I+A)x(t) \\ &= (I+D)^{-1}((I+D)-L)x(t) \\ &= \{I-(I+D)^{-1}L\}x(t). \end{aligned} \quad (21)$$

By defining $M = I-(I+D)^{-1}L$, the system can be rewritten as

$$x(t+1) = Mx(t). \quad (22)$$

Then, our goal is to show

$$\lim_{t\to\infty} M^t x(0) = x^*. \quad (23)$$

To show (23), the eigenvalue of the matrix M should satisfy

$$\lambda(M)| \leq 1, \text{ or } \lambda(M) = 1, \quad (24)$$

that is, there exists x^* such that $x^* = Mx^*$. Here, we introduce Lyapunov inequality

$$MP^T M - P \leq 0. \quad (25)$$

We can show that $P = I + D > 0$ satisfies (25) as

$$\begin{aligned} &(I-L(I+D)^{-1})(I+D)(I-(I+D)^{-1}L)-(I+D) \\ &\quad = ((I+D)-L)(I-(I+D)^{-1}L)-I-D \\ &\quad = -2L-L(I+D)^{-1}L \leq 0. \end{aligned} \quad (26)$$

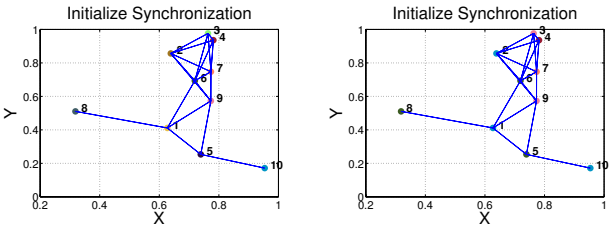
Define $V(x) \triangleq x^T Px$. Then, $V(x) > 0$ for all $x \neq 0$, or $V(x) = 0, x = 0$.

$$\begin{aligned} &V(x(t+1)) \leq V(x(t)) \\ &\Leftrightarrow x(t+1)^T Px(t+1) - x(t)^T Px(t) \leq 0 \\ &\Leftrightarrow x(t)^T M^T PMx(t) - x(t)Px(t) \leq 0. \end{aligned} \quad (27)$$

From (27), we can show that $x(t)$ will converge to a certain value x^*. Furthermore, it is easy to see that the eigenvector x^* satisfies $Lx^* = 0, x(t) = x^*$. Because have $L\mathbf{1} = 0, x^* = \alpha\mathbf{1}$. ■

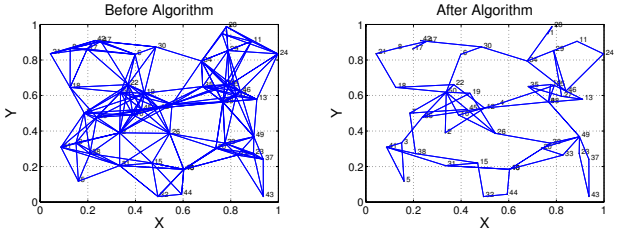
IV. SIMULATION AND ANALYSIS

In this section, we provide some simulation results by our method. We applied Algorithm 3.1 to 10 agents in Fig. 2. The agents are initially asynchronized and marked with the different 10 colors. Fig. 2(a) shows the 10 agents network in a plane with the domain of coordinates axis $X = [0,1]$ and $Y = [0,1]$. By Algorithm 3.1, some of the 10 agents are synchronized which are illustrated with the some color. For example in Fig. 2(b), the agents 1, 2, 10 are synchronized with color blue, the agents 3, 9 are synchronized with color pink, and the agents 5, 6, 8 are synchronized with color green. Meanwhile, Algorithm 3.1 guarantees neighbors asynchronization for all agents.



(a) (b)

Fig. 2. Collision avoidance motivated by vertex coloring



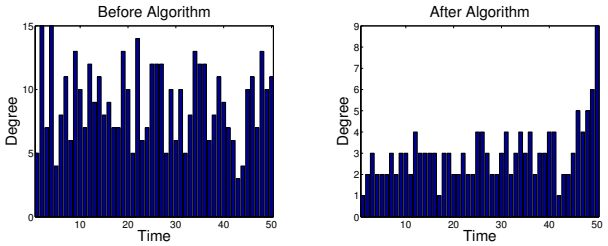
(a) (b)

Fig. 3. Our mechanism simulation.

Fig. 3 shows a network with $n = 50$ agents having difference connectivity. All agents are fixed(i.e., they do not move their position). The lines between two agents are assumed as communication connection. Fig. 3(a) shows an initial network at $t = 0$ before Algotithm 3.2 is applied. Each agent has the maximal connection. In Fig. 3(b), the result of Algorithm 3.2 is represented the network at $t = 100$ with degree balanced topology. Clearly, Fig. 3(b) is strongly connected and more balance.

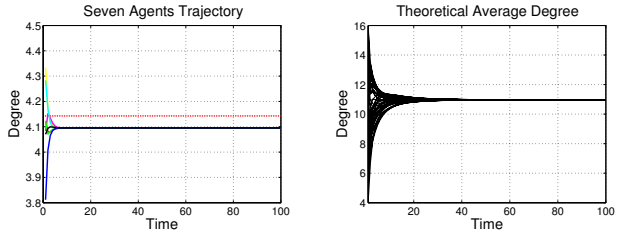
Fig. 4 shows the degree of agents for Fig. 3 on Algorithm 3.2 with 50 agents. The topologies are shown in Fig. 3 and corresponded to every single agent's ID. Fig. 4(a) is related to Fig. 3(a) and Fig. 4(b) is related to Fig. 3(b). In the figures, we know the maximum degree is 15 and the minimum degree is 3 before applying Algorithm 3.2. Then, we get the maximum 9 and minimum 1 through our way. Comparing to in Fig. 4(a), Fig. 4(b) is more balanced.

Fig. 5 shows convergence representing the discrete-states of network with our algorithm. Fig. 5(a) is shown with 7 agents randomly and Fig. 5(b) is shown with 50 agents which



(a) (b)

Fig. 4. The degree of agents



(a) (b)

Fig. 5. Trajectory of the local average degree for all agents

states trajectories of all agents corresponding to the network with topology in Fig. 3. They are both starting with a random initial state and the state trajectories of the system are stable. Furthermore, it is clear that as the number of the edges of the graph decreases, algebraic connectivity (or λ_2) will be always bigger than 0 which is embodied in no 0 degree of Fig. 4, and the balance of agents are tried to struggle to be better.

V. CONCLUSIONS AND FUTURE WORKS

The experimental results are revealed superior process to execute our solution. We propose a distributed control mechanism to drive agents in a constrained dynamic network in this paper. Motivated by k-regular graph and vertex coloring, we explore an approach based on tentative overlay as condition mapping from interrelation and interaction between with each agent, which equipped with local sensing and wireless communication capabilities. It thereby shows that connected balance can be controlled with network's connectivity just using nearest neighbor information. Furthermore, a designed method is presented for helping us avoid collision. For the future, we will be absorbed in moving agents on the multi-agents system.

REFERENCES

[1] L. Yao, G. Yi, and D. Zhaoyang, "Multiagent flocking with formation in a constrained environment," Journal of Control Theory and Applications, (2), pp. 151-159, 2010.

[2] C. W. Reynolds and S. G. Division, "Flocks, herds, and schools: a distributed behavioral model," Computer Graphics (ACM), 21 (4), pp. 25-34, 1987.

[3] T. Vicsek, A. Czirk, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," Physical Review Letters, 75 (6), pp. 1226-1229, 1995.

[4] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," IEEE Transactions on Automatic Control, 52 (5), pp. 863-868, 2007.

[5] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," IEEE Transactions on Automatic Control, 49 (9), pp. 1520-1533, 2004.

[6] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," Proceedings of the American Control Conference, pp. 951-956, 2003.

[7] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," IEEE Transactions on Automatic Control, 48(6), pp. 988-1001, 2003.

[8] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," IEEE Transactions on Automatic Control 55 (4), no. 5404774, pp. 922-938, 2010.

[9] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," Proceedings of the IEEE Conference on Decision and Control, pp. 4997-5002, 2003.

[10] Y. Liang, "Mobile intelligence sharing based on agents in mobile peer-to-peer environment," 3rd International Symposium on Intelligent Information Technology and Security Informatics, IITSI 2010 , art. no. 5453713, pp. 667-670, 2010.

[11] C. Godsil and G. Royle, *Algebraic graph theory*, New York: SpringerVerlag, 2001.

[12] G. Chartrand and L. Lesniak, *Graphs and digraphs*. Chapman and Hall/CRC, 2005.

[13] B. Bela, *Modern graph theory*. New York: SpringerVerlag, 1998

[14] M. Liu and B. Liu, "Some results on the Laplacian spectrum," Computers and Mathematics with Applications 59 (11), pp. 3612-3616, 2010.

[15] N. M. M. DeAbreu, "Old and new results on algebraic connectivity of graphs," Linear Algebra and Its Applications, 423 (1), pp. 53-73, 2007.

[16] M. M. Zavlanos and G. J. Pappas, "Distributed connectivity control of mobile networks," IEEE Transactions on ICE, 24 (6), pp. 1416-1428, 2008.

[17] M. Fiedler, "A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," Czechoslovak Mathematical Journal, Vol.25, pp. 619-633, 1975.

[18] R. L. Brooks, "On colouring the nodes of a network," Proc. Cambridge Philos. Soc., 37, pp. 194-197, 1941.

[19] D. J. A. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems," The Computer Journal 10(1), pp. 85-86, 1967.