DISTRIBUTED CONSENSUS IN NETWORKS

1. INTRODUCTION

In a network of agents, *consensus* means that the states of the agents converge to the same value. Depending on the physical background, an *agent* might represent a robot, a sensor node, or even a computer process. Accordingly, the *state* of an agent might denote the position of a robot, the local time clock of a sensor node, or the workload of a computer process. The value to which all the agents converge is the *consensus value*. A *consensus algorithm* is a protocol or a law that specifies how the agents utilize their collected information to reach a consensus. If a consensus algorithm can be implemented in such a way that each agent only uses the information of its local neighbors in the network, then the consensus algorithm is *distributed*.

The research on consensus can be traced back to 1959, when a consensus model was built to describe how a group of individuals reach a consensus on the probability distribution for an unknown parameter (1). Consensus serves as a fundamental problem in distributed computing, where computer processes need to reach a consensus on certain data value during computation (2,3). Examples of consensus in distributed computing include whether to commit a transaction to a database, agreeing on the identity of a leader, and so on. In the control community, the research focus was initially placed on the convergence analysis of consensus under certain graph assumptions (4–10), and later has been extended from various perspectives, such as agent dynamics (11) and nonlinear control strategies (12–14).

1.1. A Typical Consensus Algorithm

Consider the following multiagent system with the single-integrator dynamics:

$$\dot{x}_i = u_i, \quad i = 1, \dots, n \tag{1}$$

where x_i is the state of agent i, u_i is the control input, n is the number of agents, and $\dot{x}_i$ is the derivative of x_i. The consensus algorithm is typically designed as follows:

$$u_i = \sum_{j=1}^{n} a_{ij}(x_j - x_i) \tag{2}$$

where a_{ij} is the (i, j)th entry of the adjacency matrix of a graph $\mathcal{G}$ that describes the information flows among the agents. That is, $a_{ij} > 0$ if agent i can utilize the information of agent j, and $a_{ij} = 0$ otherwise. Here, if $a_{ij} > 0$, then agent j is a neighbor of agent i. The graph $\mathcal{G}$ is referred to as the *network topology* of the multiagent system. The basic idea behind equation 2 is to drive the state of each agent toward the states of its neighbors such that all the agents' states will converge to a common value. This common value is the consensus value. The algorithm equation 2 is distributed, because each agent only uses the information of its local neighbors. For a multiagent system, a distributed algorithm is preferable to a centralized one due to its scalability. The consensus closed-loop system is then given by

$$\dot{x}_i = \sum_{j=1}^{n} a_{ij}(x_j - x_i) \tag{3}$$

In the discrete-time case, the differential equation (eq. 3) becomes the following difference equation:

$$x_i(k+1) = x_i(k) + \epsilon \sum_{j=1}^{n} a_{ij}[x_j(k) - x_i(k)] \tag{4}$$

where $x_i(k)$ denotes the state of agent i at time k, and $\epsilon > 0$ is the step size. Let $p_{ij} = \epsilon a_{ij}$ and $p_{ii} = 1 - \epsilon \sum_{j=1}^{n} a_{ij}$. Equation 4 can be rewritten as

$$x_i(k+1) = \sum_{j=1}^{n} p_{ij} x_j(k) \tag{5}$$

It can be verified that if ϵ is sufficiently small, then $p_{ii} > 0$. In addition, the row sums $\sum_{j=1}^{n} p_{ij} = 1$, $\forall i = 1, \dots, n$, which implies that $P = [p_{ij}]$ is a row stochastic matrix with positive diagonal entries. The idea behind equation 5 is that each agent updates its next state to be the weighted average of its own and neighbors' current states.

1.2. Consensus Examples in Practical Systems

Example 1 (Multi-Robot Rendezvous). Consider a group of n mobile autonomous robots moving in a two-dimensional plane. Let p_i denote the position of robot i and let S_i denote its sensing region. Each robot is able to sense the positions of all other agents within its sensing region. The objective of rendezvous is to design local control strategies such that all the robots can eventually rendezvous at a single location, that is, reaching a consensus on their positions p_i.

Example 2 (Clock Synchronization in Wireless Sensor Networks). In a wireless sensor network, each sensor node maintains a local time clock

$$c_i = a_i t + b_i$$

where a_i is the skew rate of the clock that determines the clock speed, and b_i is the offset of the clock function that is primarily caused by inaccuracy when first setting the clock. To reach clock synchronization in the sensor network, all the nodes must compensate for their clock parameters a_i and b_i so that all clocks have zero offset error and all tick at the same rate. That is, a consensus can be reached for a_i and b_i, respectively, where the consensus value for the offset b_i is zero.

Example 3 (Workload Balancing for Distributed Computing). Consider a distributed computing system consisting of n computer processes, which collaborate with each other, and a set of tasks (workloads), which have to be executed in the system. Let $q_i(t)$ denote the queue length of the tasks of process i at time t. Denote $p_i(t)$ the productivity of process i, which determines process i's speed of executing tasks. The objective of workload balancing is to balance the fraction $q_i(t)/p_i(t)$ such that the execution time of the tasks can be minimized. The problem can be solved effectively by running a consensus algorithm on $q_i(t)/p_i(t)$.

Example 4 (Target Tracking in Distributed Camera Networks). Consider a network of multiple cameras monitoring an area that contains several moving targets. The state (position) of all targets at time k is represented by a vector $x(k)$. Each camera has a specific detection area, and is able to detect some or none of the targets. The specific targets detected by each camera determines whether $z_i(k)$, the measurement of camera i, consists of multiple or no measurements of the components of $x(k)$. At each time k, camera i uses a consensus-based Kalman filter and the measurement $z_i(t)$ to update its estimate, denoted by $\hat{x}_i(k)$, of $x(k)$, where the consensus algorithm is incorporated with the Kalman filter to keep the estimates at different cameras cohesive.

Example 5 (Altitude Alignment of Unmanned Air Vehicles (UAVs)). Suppose that a group of UAVs is initialized at different altitudes. The center of the UAVs is defined as the average of all UAVs' altitudes, and is not known *a priori*. Each UAV controls its vertical climb rate using the relative altitudes of its neighbors such that all the UAVs will reach a consensus on their altitudes while keeping the center unchanged. It turns out that the above altitude alignment problem can be solved by an average consensus problem.

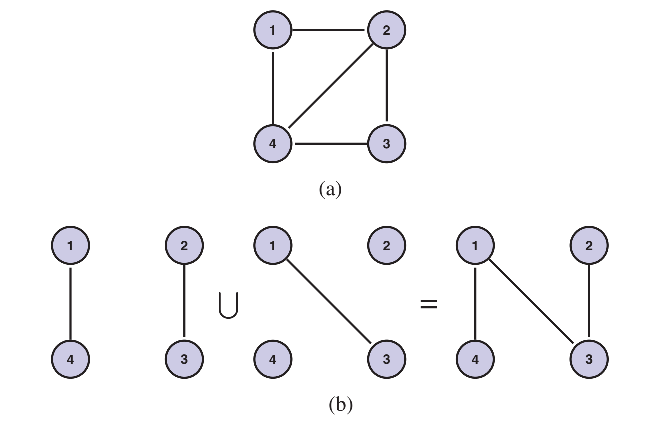

Figure 1. Illustration of "connected" and "jointly connected" for undirected graphs. The graph in (a) is connected. The first two graphs in panel (b) are jointly connected and their union is given by the third graph.

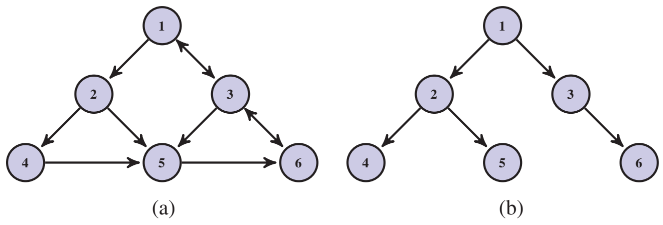

Figure 2. Example of a strongly connected digraph (a). One of its directed spanning trees is shown in (b).

2. NETWORK TOPOLOGIES FOR INFORMATION FLOWS

2.1. Graph Connectedness

In equation 2, the graph $\mathcal{G}$ describes the information flows among the agents, and is referred to as the network topology of the system. It turns out that the connectedness of the graph is a critical condition for the convergence of the closed-loop system equation 3. In the following, we introduce different types of graphs and their connectedness notions.

(i) *Undirected graph & its connectedness:* An *undirected graph* is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of undirected edges. Here, $(u, v) \in \mathcal{E}$ means that nodes u and v can receive information from each other. In an undirected graph, $(u, v) \in \mathcal{E} \iff (v, u) \in \mathcal{E}$ for $u, v \in \mathcal{V}$. The neighbor set of node u is defined by $\mathcal{N}_u = \{v : (v, u) \in \mathcal{E}\}$. A *path* from node v_1 to node v_k is a sequence of nodes $v_1, \ldots, v_k$, such that for each i, $1 \le i \le k-1$, (v_i, v_{i+1}) is an edge. An undirected graph is *connected* if there is a path from any node to any other node in the graph (see Figure 1a). For a collection of undirected graphs $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1), \ldots, \mathcal{G}_m(\mathcal{V}, \mathcal{E}_m)$, each with the same vertex set $\mathcal{V}$, their union is the undirected graph given by $\cup_i \mathcal{G}_i = (\mathcal{V}, \cup_i \mathcal{E}_i)$. The collection of the undirected graphs is *jointly connected* if the union is a connected graph (see Figure 1b).

(ii) *Directed graph & its connectedness:* A *directed graph* (digraph) is also defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of directed edges. Here, $(u, v) \in \mathcal{E}$ means that node v can receive information from node u. For a directed graph, $(u, v) \in \mathcal{E}$ does not necessarily imply $(v, u) \in \mathcal{E}$. The in-neighbor set and out-neighbor set of node u are defined, respectively, by $\mathcal{N}_u^{\text{in}} = \{v : (v, u) \in \mathcal{E}\}$ and $\mathcal{N}_u^{\text{out}} = \{v : (u, v) \in \mathcal{E}\}$. A *directed path* from node v_1 to node v_k is a sequence of nodes $v_1, \ldots, v_k$, such that for each i, $1 \le i \le k-1$, (v_i, v_{i+1}) is a directed edge. A digraph $\mathcal{G}$ is *strongly connected* if there is a directed path from every node to every other node (see Figure 2a). A *directed tree* is a special digraph where each node has exactly one in-neighbor except for the root node that has directed paths to any other node. A *directed spanning tree* of a digraph is a directed tree that includes all the nodes of the digraph (see Figure 2b). A digraph $\mathcal{G}$ *has a directed spanning tree* if there exists a subgraph of $\mathcal{G}$ that is a directed spanning tree. For a collection of digraphs, the definitions of strongly connected and has a directed spanning tree can be extended to *jointly strongly connected* and *have a jointly directed spanning tree* analogously.

2.2. Graph Laplacian

The Laplacian matrix is an important matrix in the study of graphs as it reveals some fundamental properties of graphs, such as the connectedness. In the literature, the

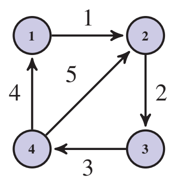

Figure 3. The digraph obtained by assigning an orientation to the edges in the graph of Figure 1a.

Laplacian matrix is also called the admittance matrix or Kirchhoff matrix (15).

For an undirected graph (a digraph) $\mathcal{G}$, its adjacency matrix $A = [a_{ij}]$ is defined as follows: $a_{ij} = 1$ if $(j, i) \in \mathcal{E}$; $a_{ij} = 0$ otherwise. If a number, not necessarily one, is assigned to each edge, then the graph is *weighted*. The *degree* (*in-degree* for a digraph) of node i is defined as $d_i = \sum_{j=1}^{n} a_{ij}$, and the *degree matrix* (*in-degree matrix* for a digraph) is defined as $D = \mathrm{diag}([d_1, \ldots, d_n])$. For a digraph, the out-degree of node i is defined as $d_i^{\mathrm{out}} = \sum_{j=1}^{n} a_{ji}$, and the out-degree matrix is defined as $D^{\mathrm{out}} = \mathrm{diag}([d_1^{\mathrm{out}}, \ldots, d_n^{\mathrm{out}}])$. A digraph is *balanced* if, for each node, the in-degree equals to the out-degree. The *Laplacian matrix* of $\mathcal{G}$ is then given by $L = D - A$. For undirected graphs, there is an alternative definition of the Laplacian matrix. Let E be the incidence matrix of an undirected graph $\mathcal{G}$. By assigning an orientation to each edge of $\mathcal{G}$, the incidence matrix of $\mathcal{G}$ is defined as $E = [e_{ij}] \in \mathbb{R}^{n\times m}$, where $e_{ij} = -1$ if the jth edge leaves the ith node, $e_{ij} = 1$ if it enters the ith node, and $e_{ij} = 0$ otherwise. Here, n is the number of nodes, and m is the number of edges. The Laplacian matrix for the undirected graph $\mathcal{G}$ can then be defined as $L = EE^T$. For a directed graph, an orientation has already been given and the incidence matrix can be defined similarly. However, the relationship $L = EE^T$ does not hold for directed graphs in the general case.

Example 6 (Laplacian Matrix). Let $\mathcal{G}$ be the undirected graph given by Figure 1a. The adjacency matrix and degree matrix of $\mathcal{G}$ are given, respectively, by

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \text{and } D = \mathrm{diag}([2, 3, 2, 3])$$

The Laplacian matrix is then given by

$$L = D - A = \begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

If we give an orientation to each edge in $\mathcal{G}$ and an order to the edges, we obtain the digraph of Figure 3. Thus, the incidence matrix of $\mathcal{G}$ is given by

$$E = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 \end{bmatrix}$$

It can be verified that $L = EE^T$. That is, the two definitions of the Laplacian matrix for undirected graphs are equivalent. Using the Laplacian matrix L, the system equation 3 can be rewritten into a compact form as

$$\dot{x} = -Lx \tag{6}$$

where $x = [x_1^T, \ldots, x_n^T]^T$ is the stack vector of all the agents' states.

To conclude this section, we summarize several important properties of the Laplacian matrix as follows (16,17).

(i) The Laplacian matrix has zero row sums. Therefore, it has a zero eigenvalue with an eigenvector with all ones.

(ii) For an undirected graph, the Laplacian matrix L is symmetric and positive semidefinite. Zero is a simple eigenvalue of L and all other eigenvalues are positive if and only if the graph is connected. In addition, L can be diagonalized as

$$L = U^T \Lambda U \tag{7}$$

where $U \in \mathbb{R}^{n\times n}$ is a unitary matrix satisfying $U^T U = UU^T = I$ with I being the identity matrix, and $\Lambda = \mathrm{diag}([\lambda_1, \ldots, \lambda_n])$ with $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ denoting the eigenvalues of L.

(iii) For a digraph, the Laplacian matrix L might not be symmetric and the real parts of the eigenvalues of L are nonnegative. Zero is a simple eigenvalue of L and all other eigenvalues have positive real parts if and only if the digraph has a directed spanning tree.

3. DEFINITIONS OF CONSENSUS

3.1. Characterizations of Consensus

Let span 1 denote the space spanned by the vector with all ones. A formal definition of consensus is given below.

Definition 1 (Consensus). The system equation 3 is said to reach a consensus if and only if $\lim_{t\to\infty}[x_i(t) - x_j(t)] = 0$ for all $i \neq j$, or equivalently the stack vector x converges to the space span 1.

A vector belonging to span 1 must be an eigenvector of the Laplacian matrix corresponding to the zero eigenvalue. Thus, using the properties of the Laplacian matrix at the end of Section 2, an important result can be stated as follows: if the undirected graph (respectively, digraph) is connected (respectively, has a directed spanning tree), then

$$Lx = 0 \quad \text{iff} \quad x \in \mathrm{span}1 \tag{8}$$

That is, Lx is a measurement of the consensus error and $Lx = 0$ can be used to characterize consensus.

Recall the definition of E as the incidence matrix of the graph $\mathcal{G}$. The term $E^T x$ is a vector that contains the entry $x_i - x_j$ for $(i, j) \in \mathcal{E}$. It thus follows that

$$E^T x = 0 \quad \text{iff} \quad x_i - x_j = 0,\ \forall (i, j) \in \mathcal{E} \tag{9}$$

If the undirected graph (respectively, digraph) is connected (respectively, has a directed spanning tree), then equation 9 can be further written as

$$E^T x = 0 \quad \text{iff} \quad x_i - x_j = 0,\ \forall i \neq j$$

That is, $E^T x$ is another measurement of the consensus error and can be used to characterize consensus.

Recall the definition of U and Λ in equation 7 for an undirected graph. Let $\tilde{U} \in \mathbb{R}^{(n-1)\times n}$ be the matrix constructing from U by removing the first row and let $\tilde{\Lambda} = \text{diag}([\lambda_2, \dots, \lambda_n])$. It can be verified that $L = \tilde{U}^T \tilde{\Lambda} \tilde{U}$ by using $\lambda_1 = 0$. If $\tilde{U}x = 0$, then $Lx = U^T \Lambda U x = \tilde{U}^T \tilde{\Lambda} \tilde{U} x = 0$. If $Lx = 0$, then $U^T \Lambda U x = 0$, which yields that $x^T U^T \Lambda U x = 0$. This is equivalent to $\tilde{U}x = 0$ for an undirected and connected graph. The above statement can be summarized as follows: if the undirected graph $\mathcal{G}$ is connected, then

$$\tilde{U}x = 0 \quad \text{iff} \quad Lx = 0 \tag{10}$$

That is, $\tilde{U}x$ is a measurement of the consensus error for undirected graphs.

3.2. Average Consensus

The definition of consensus requires that all the states of the agents converge to the same value. However, it does not specify the final consensus value, which motivates the following definition of average consensus.

Definition 2 (Average Consensus). The system equation 3 is said to reach an average consensus if and only if $\lim_{t\to\infty} x_i(t) = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$. Depending on the application areas, the initial states $x_i(0)$ can be set to any static quantities that need to be averaged among the agents. Average consensus implies consensus, but the reverse might not be true. Compared with consensus, average consensus imposes a restriction on the consensus value, which equals the average of all the agents' initial states. However, when certain invariance property holds, average consensus is equivalent to consensus as we show below.

Let $\mathcal{G}$ be an undirected and connected graph. The Laplacian matrix L for $\mathcal{G}$ must be symmetric. Since $\mathbf{1}$ is a right eigenvector of L corresponding to the zero eigenvalue ($L\mathbf{1} = 0$), it is also a left eigenvector ($\mathbf{1}^T L = 0$). Under the consensus system equation 3,

$$\mathbf{1}^T \dot{x}(t) = \mathbf{1}^T L x(t) = 0$$

which indicates that

$$\mathbf{1}^T x(t) \equiv \mathbf{1}^T x(0), \quad \forall t \geq 0 \tag{11}$$

Equation 11 says that the sum of the agents' states is invariant under the consensus system equation 3 and the undirected network topology. If the invariance property holds, then consensus would imply average consensus, which can be shown by the following statement: suppose that a consensus is reached at time t (t could be ∞). One has $x_i(t) = x_j(t)$ for all $i \neq j$ and hence $x_i(t) = \frac{1}{n}\sum_{j=1}^{n} x_j(t)$. It thus follows from equation 11 that $x_i(t) = \frac{1}{n}\sum_{j=1}^{n} x_j(0)$.

It is worth mentioning that the invariance property equation 11 is a consequence of both the consensus algorithm and the undirected network topology. The invariance property can be extended for more complicated consensus algorithms (such as nonlinear consensus algorithms) and network topologies (such as balanced digraphs).

3.3. Leader-following Consensus

In the leader-following consensus, the final consensus value is determined by a leader agent, labelled as 0 without loss of generality. The leader agent can be a physical or virtual one. All the other agents are called followers. The state of the leader might be time invariant or time varying, and is available to only a portion of the followers. The objective of leader-following consensus is that the states of the followers will finally track the state of the leader. Let r_0 denote the state of the leader. If r_0 is time-invariant (e.g., the leader is governed by the singler-integrator dynamics $\dot{r}_0 = u_0$ with $u_0 = 0$), then the consensus algorithm equation 2 can be slightly modified to reach leader-following consensus. The leader-following consensus algorithm for equation 1 takes the following form:

$$u_i = \sum_{j=1}^{n} a_{ij}(x_j - x_i) + a_{i0}(r_0 - x_i) \tag{12}$$

where $a_{i0} > 0$ if follower i can access the state information of the leader, and $a_{i0} = 0$ otherwise. The leader-following consensus algorithm equation 12 can be seen as a special case of the normal consensus algorithm equation 2, where the leader is viewed as an agent without neighbors.

Define the leader-following consensus error for each agent as $e_i = x_i - r_0$. Equation 12 can be written in terms of e_i as

$$\dot{e}_i = \sum_{j=1}^{n} a_{ij}(e_j - e_i) + a_{i0}e_i \tag{13}$$

Let $A_0 = \text{diag}([a_{10}, \dots, a_{n0}])$ and $e = [e_1^T, \dots, e_n^T]^T$. We can write equation 13 in a vector form as

$$\dot{e} = -(L + A_0)e \tag{14}$$

The dynamical behavior of equation 14 is totally determined by the eigenvalues of the matrix $L + A_0$. The network topology $\mathcal{G}$ describes the information flows among the followers, but it does not contain the information about the leader agent. In order to include the leader information, we need to define an extended graph $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$, where $\bar{\mathcal{V}} = \mathcal{V} \cup \{0\}$ with node 0 denoting the leader agent and $\bar{\mathcal{E}}$ is the set of the edges that is includes the information flows from the leader to the followers. If the graph $\bar{\mathcal{G}}$ has a directed spanning tree (or equivalently the leader has directed paths to all followers), then all the eigenvalues of $L + A_0$ are negative and the system equation 14 is asymptotically stable, that is, $\lim_{t\to\infty} e = 0$. That is, the leader-following consensus is reached.

3.4. Dynamic Average Consensus

In dynamic average consensus, each agent i is associated with a reference signal r_i. Here, r_i is generally time varying. The objective of dynamic average consensus is for the agents to reach a consensus on the average of the references (see Figure 4). If we see $x_i(0)$ in the average

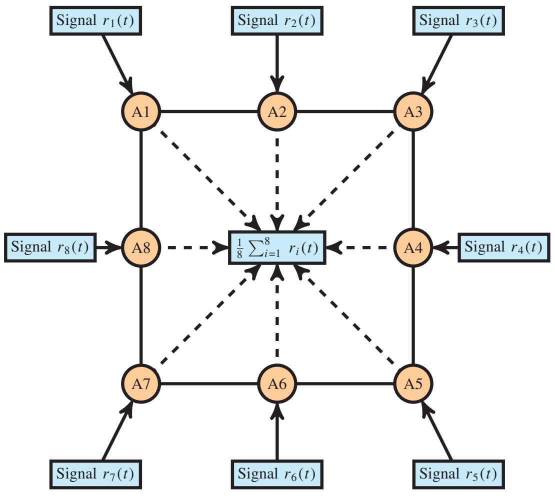

Figure 4. Illustration of dynamic average consensus.

consensus problem as a constant signal, then the average consensus problem can be included in the framework of dynamic average consensus. Thus, the dynamic average consensus problem can be seen as a generalization of the average consensus problem.

Incorporating the derivatives of the references into the consensus algorithm equation 2 yields the following dynamic average consensus algorithm (18) for equation 1

$$u_i = \sum_{j=1}^{n} a_{ij}(x_j - x_i) + \dot{r}_i \tag{15}$$

where $\sum_{j=1}^{n} a_{ij}(x_j - x_i)$ is a proportional term and $\dot{r}_i$ is a derivative term. For equation 1 using equation 15, if the network topology is undirected, then

$$\sum_{i=1}^{n} \dot{x}_i \equiv \sum_{i=1}^{n} \dot{r}_i, \quad \forall t \geq 0$$

Thus, if the following initialization condition holds

$$\sum_{i=1}^{n} x_i(0) = \sum_{i=1}^{n} r_i(0)$$

then

$$\sum_{i=1}^{n} x_i \equiv \sum_{i=1}^{n} r_i \tag{16}$$

Equation 16 indicates that if the consensus problem is solved, that is, $x_i(t) = x_j(t)$ for all $i \neq j$, which implies that $x_i(t) = \frac{1}{n}\sum_{j=1}^{n} x_j(t)$, then the dynamic average consensus problem is solved. It has been shown that the algorithm equation 15 can reach dynamic average consensus if the references have steady states. That is, the references will finally converge to certain constant values.

For more general reference signals, a nonsmooth dynamic average consensus algorithm is given in Reference (19) as follows:

$$u_i(t) = \alpha \sum_{j=1}^{n} a_{ij}\mathrm{sgn}(x_j - x_i) + \dot{r}_i \tag{17}$$

where $\alpha > 0$ is a control gain, $x_i(0) = r_i(0)$, and $\mathrm{sgn}(\cdot)$ is the signum function defined componentwise. It has been shown that for arbitrary references with bounded derivatives if α is sufficiently large and the graph is undirected and connected, then dynamic average consensus is reached by the proposed algorithm equation 17.

4. APPROACHES FOR CONSENSUS ANALYSIS

4.1. The Decoupling Approach

Consensus is a problem that needs the cooperation of multiple agents. There exist couplings among the agents, which becomes a main challenge in consensus analysis. It is natural to come up with the idea of decoupling the system such that the complexity of the system can be reduced to the single agent level.

Consider the classical consensus system in the vector form, that is, equation 6. In addition, suppose that the network topology is undirected. Recall the diagonalization equation in equation 7. Equation 6 can be rewritten as

$$\dot{x}(t) = -U^T \Lambda U x(t) \tag{18}$$

Multiplying both sides of equation 18 with U yields that

$$U\dot{x}(t) = -\Lambda U x(t) \tag{19}$$

where $UU^T = I$ is used. Recall the definition of $\tilde{U}$ as the matrix constructing from U by removing the first row, and define $\tilde{x} = \tilde{U}x$. It thus follows that

$$\dot{\tilde{x}}(t) = -\tilde{\Lambda}\tilde{x}(t) \tag{20}$$

where $\tilde{\Lambda} = \mathrm{diag}([\lambda_2, \ldots, \lambda_n])$ is a diagonal matrix. Using equation 10, it follows that a consensus is reached iff $\tilde{x}(t) \to 0$. Equation 20 can be rewritten as

$$\dot{\tilde{x}}_i(t) = -\lambda_i \tilde{x}_i(t), \quad \forall i = 2, \ldots, n$$

which becomes a stability problem for $n-1$ independent subsystems. If the network topology is connected, then $0 < \lambda_2 \leq \cdots \leq \lambda_n$, which implies that a consensus is reached asymptotically.

The idea of the decoupling approach can also be applied to the directed network topology case. In this case, L might not be symmetric. Thus, the diagonalization form cannot be used. Instead, we can use the Jordan canonical form:

$$L = U^T J U \tag{21}$$

where $J = \mathrm{diag}([J_1, \ldots, J_p])$ is a block diagonal matrix with p is the number of different eigenvalues of L. Each J_i is a square matrix of the form

$$J_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}$$

Under the Jordan form equation 21, the system can be reduced to p independent subsystems, each corresponding to a J_i. Suppose that J_i is of dimension p_i. Due to the special structure of J_i, the analysis of these p_i subsystems can be performed in a sequential way: the last subsystem, the

p_ith one, can be analyzed first and independently, since it has no coupling with the other subsystems. After the stability of the p_ith system is proved, the $(p_i - 1)$th system that is only coupled with the subsystem p_i can be analyzed. The process can be repeated until the stability of the first system in J_i is proved.

4.2. The Approach Based on Infinite Matrix Products

For a multiagent system, the network topology might be time varying. For instance, in Example 1, the sensing range of a robot is subject to change during the movement of the robot. Thus, the neighbors of the robot are possibly changing during its movement. In the time-varying network topology case, the consensus system equation 3 becomes

$$\dot{x}_i = \sum_{j=1}^{n} a_{ij}(t)(x_j - x_i) \tag{22}$$

where $a_{ij}(t)$ is the (i, j)th entry of the time-varying adjacency matrix $A(t)$. Let $L(t)$ denote the Laplacian matrix at time t. In the time-varying network topology case, the decoupling approach presented in the last section does not work in general, because the unitary matrix U and the diagonal matrix Λ defined in equation 18 are also time varying.

In equation 22, let t_i, $i = 1, \ldots, \infty$, be the switching times, that is, the times when the network topology changes from one to another. Here, we consider the case that the number of switchings is infinite. If the graph stops switching after a finite time, the rest of the time can be divided into an infinite number of bounded time intervals. The network topology is fixed during two switching times $[t_i, t_{i+1})$. The lengths of the time intervals $\tau_i = t_{i+1} - t_i$ are called the dwell times, for which the lower bound is assumed to be positive (4). Stack all the agents' states at time t_i as $x(t_i)$. Using the well-known result in linear system theory, it follows that

$$x(t_{i+1}) = \mathrm{e}^{-L(t_i)\tau_i} x(t_i) \tag{23}$$

where the matrix exponential $\mathrm{e}^{-L(t_i)}$ is defined by the following power series $\mathrm{e}^{-L(t_i)\tau_i} = \sum_{k=0}^{\infty} \frac{[-L(t_i)\tau_i]^k}{k}$. When $t \to \infty$, if the state of the system equation 23 converges, then the convergent value must be given by

$$\lim_{t\to\infty} x(t) = \lim_{i\to\infty} \mathrm{e}^{-L(t_i)\tau_i} \cdots e^{-L(0)\tau_0} x(0) \tag{24}$$

If $\lim_{i\to\infty} \mathrm{e}^{-L(t_i)\tau_i} \cdots \mathrm{e}^{-L(0)\tau_0}$ is a matrix having the same row vectors, that is, $\lim_{i\to\infty} \mathrm{e}^{-L(t_i)\tau_i} \cdots \mathrm{e}^{-L(0)\tau_0} = \mathbf{1}v^T$ for certain vector v, then a consensus is reached for any initial state $x(0)$. That is, the convergence of consensus can be drawn from the convergence of an infinite series of matrix products.

The definition of *ergodic matrix* is useful in characterizing the convergence of matrix products. A matrix M is called *ergodic* if M is stochastic and $\lim_{i\to\infty} M^i$ is a matrix of rank 1. A convergence result for a series of the infinite product of ergodic matrices is established below.

Theorem 1 (Wolfowitz): Let $M_1, \ldots, M_p$ be a finite set of ergodic matrices with the property that for each sequence $M_{i_1}, \ldots, M_{i_j}$ of positive length, the matrix product $M_{i_j} \cdots M_{i_1}$ is ergodic. Then for each infinite sequence, there exists a row vector v^T such that $\lim_{j\to\infty} M_{i_j} \cdots M_{i_1} = \mathbf{1}v^T$.

In Reference (4), it has been shown via the Wolfowitz Theorem that if there exists an infinite sequence of contiguous, uniformly bounded time intervals such that across each interval the network topology is jointly connected, then a consensus is reached.

4.3. Lyapunov Function Based Approach

In the literature, the following smooth function $V(x) = x^T Lx$ is employed to study the convergence of the consensus system equation 3, where $x = [x_1, \ldots, x_n]^T$ with $x_i \in \mathbb{R}$ being the state of agent i. When $x_i \in \mathbb{R}^m$, the function $V(x)$ can be applied to each dimension. For an undirected graph, the function $V(x)$ can be rewritten as

$$V(x) = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} a_{ij}(x_i - x_j)^2 \tag{25}$$

If the network topology is connected, then $V(x) = 0$ iff all x_is reach a consensus. Thus, $V(x)$ serves as a Lyapunov function candidate for consensus. The consensus system equation 3 can be written in the following gradient-descent form:

$$\dot{x} = -\nabla V(x)$$

It thus follows that x will converge to the set $\{x : \nabla V(x) = Lx = \mathbf{0}\}$. The consensus convergence can then be concluded according to equation 8.

The set-valued Lyapunov function

$$V_{\text{conv}} = (\text{conv}\{x_1, \ldots, x_n\})^n$$

is used in Reference (7) to analyze the convergence of a nonlinear consensus model, where $\text{conv}\{x_1, \ldots, x_n\}$ denotes the convex hull and $(\cdot)^n$ denotes the Cartesian product. The analysis of the function V_{conv} is performed under a modified Lyapunov stability framework. The set-valued Lyapunov function and the modified Lyapunov framework are motivated from the following observations. First, consensus is different from the classical stability concept, because consensus deals with a continuum of equilibrium points, while the classical stability concept studies individual, typically isolated, equilibria. Although it is sometimes possible to transform a consensus problem to a stability problem, this approach might not work when the network topology is directed or the agents' model is complicated. In this case, it might not be possible to decompose the problem to the local agent level with local information. Second, consensus is also different from the stability of a set of equilibria (20). The set stability does not fully capture some convergence properties of consensus. For example, even though the set stability may be used to assert that the individual agents' states converge to a common value, this common value might eventually grow unbounded, which might not be desired in some real applications. Thus, a distinguished feature of the set-valued Lyapunov function is that it can be used to conclude that the state of the system converges to one bounded equilibrium out of a continuum of equilibria.

In Reference (19), the following nonsmooth Lyapunov function

$$V_{\text{abs}}(x) = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} a_{ij}|x_j - x_i|$$

is employed to prove consensus, where $|\cdot|$ denotes the absolute value. If the network topology is undirected, then $V_{\text{abs}}(x)$ can also be rewritten in terms of the incidence matrix as $V_{\text{abs}}(x) = \|E^T x\|_1$, where E denotes the incidence matrix and $\|\cdot\|_1$ is the one norm. Because the absolute value function is not differentiable, the function V_{abs} is not smooth and its derivative is not well defined. Fortunately, the analysis of the function V_{abs} can be performed under a nonsmooth analysis framework (21), where the set-valued Lie derivative is used to replace the derivative of V_{abs}. The nonsmooth Lypapunov function provides an alternative for the convergence analysis of nonsmooth consensus systems when a smooth one does not exist or meet the requirement. A benefit of the function $V_{\text{abs}}(x)$ is that it can be used to prove finite-time convergence for nonsmooth consensus algorithms that converge in a finite time.

Finally, it is worth mentioning that other types of Lyapunov functions such as the max–min function $V_{\text{max,min}} = \max\{x_1, \dots, x_n\} - \min\{x_1, \dots, x_n\}$ (22) is also employed in the literature.

5. DESIGN ISSUES IN CONSENSUS

The convergence speed is a critical factor in evaluating the performance of a consensus algorithm. In Reference (23), the following discrete-time consensus system is considered:

$$x_i(t+1) = w_{ii}x_i(t) + \sum_{j\in\mathcal{N}_i} w_{ij}x_j(t) \tag{26}$$

where w_{ij} is the weight of the edge (j, i). Let $W = [w_{ij}]$ and rewrite equation 26 in a vector form as

$$x(t+1) = Wx(t) \tag{27}$$

The *asymptotic convergence factor* is proposed as a measurement of the convergence speed, which is defined as

$$r_{\text{asym}}(W) = \sup_{x(0)\neq\bar{x}} \lim_{t\to\infty} \left(\frac{\|x(t) - \bar{x}\|}{\|x(0) - \bar{x}\|}\right)^{1/t}$$

where $x(0)$ is the initial state, $\bar{x}$ is the consensus value, and $\|\cdot\|$ is the Euclidean norm of a vector. The following fast consensus problem is considered in Reference (23):

$$\begin{aligned} &\text{minimize} && r_{\text{asym}}(W) \\ &\text{subject to} && \lim_{t\to\infty} W^t = \mathbf{1}\mathbf{1}^T/n \end{aligned} \tag{28}$$

That is, design the weight matrix W such that the system equation 27 can reach a consensus and the convergence speed is as fast as possible. It is proved that $r_{\text{asym}}(W) = \rho(W - \mathbf{1}\mathbf{1}^T/n)$ and the condition $\lim_{t\to\infty} W^t = \mathbf{1}\mathbf{1}^T/n$ in equation 28 can be replaced with $\mathbf{1}^T W = \mathbf{1}^T$ and $W\mathbf{1} = \mathbf{1}$, where $\rho(\cdot)$ denotes the spectral radius of a matrix. Thus, the problem equation 28 is equivalent to

$$\begin{aligned} &\text{minimize} && \rho(W - \mathbf{1}\mathbf{1}^T/n) \\ &\text{subject to} && \mathbf{1}^T W = \mathbf{1}^T, \quad W\mathbf{1} = \mathbf{1} \end{aligned} \tag{29}$$

When the matrix W is symmetric, the spectral radius of W equals to its spectral norm. Thus, equation 29 can be rewritten as

$$\begin{aligned} &\text{minimize} && \|W - \mathbf{1}\mathbf{1}^T/n\| \\ &\text{subject to} && W = W^T, \quad W\mathbf{1} = \mathbf{1} \end{aligned}$$

which is a convex optimization problem whose globally optimal solution can be found efficiently. Here, $\|\cdot\|$ denotes the spectral norm (i.e., the maximal singular value) of a matrix.

The problem on maximizing the second smallest eigenvalue of the Laplacian matrix of a state-dependent graph $\mathcal{G}$ is considered in Reference (24). The vertex set and edge set of $\mathcal{G}$ are given, respectively, by $\mathcal{V} = \{1, \dots, n\}$ and $\mathcal{E} = \{(i,j) : i = 1, \dots, n-1, j = 2, \dots, n, i < j\}$. The weight function between any two nodes is defined as

$$w_{ij} = f(|x_i - x_j|)$$

where x_i is the state of agent i. In this setting, the weights of the edges depend on the states of the agents. Thus, the graph is called state dependent. The state-dependent Laplacian is then defined as

$$[L_{\mathcal{G}}(x)]_{ij} \triangleq \begin{cases} -w_{ij} & \text{if } i \neq j \\ \sum_{s\neq i} w_{is} & \text{if } i = j \end{cases}$$

Under the above framework, the following optimization problem is considered:

$$\begin{aligned} &\max_{x} && \lambda_2(L_{\mathcal{G}}(x)) \\ &\text{subject to} && |x_i - x_j|^2 \geq \rho_1 \end{aligned}$$

where $\rho_1 > 0$ is a constant that is introduced to prevent the weights from getting arbitrarily close to each other. A greedy algorithm is then proposed to solve the above optimization problem.

6. ADVANCED TOPICS IN CONSENSUS

6.1. Agent Dynamics

Equation 1 describes the relation between the state x_i and the input u_i, and is referred to as the dynamics of agent i. The model equation 1 is called the single-integrator model, which is the simplest agent dynamics considered in the literature. In reality, the agent dynamics could be more complicated. The agent dynamics has an effect on the control input design of consensus as we can see below.

Consider the double-integrator model

$$\ddot{x}_i = u_i \tag{30}$$

where $\ddot{x}_i$ denotes the second-order derivative of x_i. If we use the same control input $u_i = \sum_{j=1}^{n} a_{ij}(x_j - x_i)$, it can be verified that the closed-loop system $\ddot{x}_i = \sum_{j=1}^{n} a_{ij}(x_j - x_i)$ will not reach a consensus. Thus, the agent dynamics is a factor that needs to be taken into account in consensus design. In fact, to solve the consensus problem for double integrators, the state information x_i is not enough and the first-order derivative information is also needed. The control input for equation 30 can take the following form

$$u_i = \alpha \sum_{j=1}^{n} a_{ij}(x_j - x_i) + \beta \sum_{j=1}^{n} a_{ij}(\dot{x}_j - \dot{x}_i)$$

Here, α and β are two constants, and β needs to be sufficiently large for directed graphs having a directed spanning tree. The lower bound for β needs to satisfy certain global condition, which usually is given in terms of the eigenvalues of the Laplacian matrix (25). There is another control input design for equation 30:

$$u_i = \alpha \sum_{j=1}^{n} a_{ij}(x_j - x_i) - \kappa \dot{x}_i$$

where κ is a constant and the term $-\kappa\dot{x}_i$ is used to drive the velocities of the agents to zero (26).

A more general model for linear agent dynamics is given by

$$\dot{x}_i = Ax_i + Bu_i \tag{31}$$

where x_i is the state of agent i, u_i is the control input, and A and B are the constant matrices with compatible sizes. If $A = 0$ and $B = 1$, then equation 31 reduces to the single-integrator model. If $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, then equation 31 reduces to the double-integrator model. It can be verified that for the system equation 31 consensus cannot be reached for arbitrary (A, B). For example, if $B = 0$, then the control input has no effect on the closed-loop system. It is usually assumed that (A, B) is stabilizable, that is, there exists $P > 0$ satisfying

$$AP + PA^T - 2BB^T < 0 \tag{32}$$

A static consensus algorithm is given by

$$u_i = cK \sum_{j=1}^{n} a_{ij}(x_i - x_j) \tag{33}$$

where $c > 0$ is a coupling weight, and $K = -B^T P^{-1}$ is a feedback gain matrix with P satisfying equation 32. It has been shown in Reference (27) that to reach a consensus the coupling weight c should be large enough and satisfies $c \geq \frac{1}{\lambda_2}$, where λ_2 is the second smallest eigenvalue of the Laplacian matrix. However, λ_2 is the global information, which implies that the consensus algorithm equation 33 cannot be implemented in a fully distributed way.

To solve the above issue, an adaptive gain design framework is presented in Reference (28). The algorithm takes the following form:

$$\begin{aligned} u_i &= K \sum_{j=1}^{n} c_{ij} a_{ij}(x_i - x_j) \\ \dot{c}_{ij} &= \kappa_{ij} a_{ij}(x_i - x_j)^T \Gamma (x_i - x_j) \end{aligned} \tag{34}$$

where c_{ij} denotes the time-varying coupling weight between agents i and j with $c_{ij}(0) = c_{ji}(0)$, $\kappa_{ij} = \kappa_{ji}$ are positive constants, and $\Gamma = P^{-1}BB^T P^{-1}$. The basic idea of equation 34 is to use the adaptive gain c_{ij} to replace the static gain c in equation 33. Using equation 34, the requirement on knowing λ_2 is relaxed.

In addition to the above linear models, nonlinear agent dynamics can also be considered in consensus, such as the unicycle model (29), the Euler–Lagrange model (30,31), attitude dynamics for rigid bodies (32,33), and underactuated systems (34). The output information y_i can also be introduced in the above models in the scenario that the state measurement is not available to the agents (35).

6.2. Communication Delays

Time delays arise when information is exchanged between neighboring agents via a communication network. The effect of communication delays are often neglected for the ease of theoretical analysis. However, it is well known that a small communication delay can even destroy the stability of a system.

Assume that the network topology is undirected and connected, and the communication delays in all channels all equal to τ. The consensus system equation 3 becomes

$$\dot{x}_i(t) = \sum_{j=1}^{n} a_{ij}[x_j(t-\tau) - x_i(t-\tau)] \tag{35}$$

It has been proved in Reference (5) that consensus can be reached if the communication delay $\tau \in [0, \frac{\pi}{2\lambda_{\max}(L)})$, where $\lambda_{\max}(L)$ denotes the maximal eigenvalue of the Laplacian matrix L.

Consensus with nonuniform delays and a switching directed network topology is investigated in Reference 36 for the following system:

$$\dot{x}_i = -\sum_{j=1}^{n} a_{ij}^{\sigma(t)} \kappa_{ij}[x_i - \mathbb{T}_{ij}(x_j)] \tag{36}$$

where x_i is the state of agent i and $a_{ij}^{\sigma(t)}$ the (i, j)th entry of the adjacency matrix with $\sigma(t)$ the switching signal of the network topology. The coupling functions $\kappa_{ij}(z)$ are such that $\kappa_{ij}(z) = \overline{\kappa_{ij}}(\|z\|)(z/\|z\|)$ with $\overline{\kappa_{ij}}(\|z\|)$ being nonlinear, continuous gains satisfying $\kappa_{ij}(0) = \overline{\kappa_{ij}}(0) = 0$ and the following sector condition:

$$\bar{\mathcal{K}}z \leq \overline{\kappa_{ij}}(z) \geq \underline{\mathcal{K}}z, \quad \forall z \geq 0 \tag{37}$$

The symbol $\mathbb{T}_{ij}$ is a delay operator, which can represent the three most common delay models: constant delays $\mathbb{T}_{ij}(x_j) = x_j(t - \tau_{ij})$ for some $\tau_{ij} \in [0, \mathcal{T}]$ with $\mathcal{T}$ a positive constant, time-varying delays $\mathbb{T}_{ij}(x_j) = x_j(t - \tau_{ij}(t))$ with $\tau_{ij} : \mathbb{R} \mapsto [0, \mathcal{T}]$, and distributed delays $\mathbb{T}_{ij}(x_j) = \int_0^{\mathcal{T}} \phi_{ij}(\eta) x_j(t - \eta) d\eta$ with the delay kernel ϕ_{ij} satisfying $\phi_{ij}(\eta) \geq 0$ for all $\eta \in [0, \mathcal{T}]$ and $\int_0^{\mathcal{T}} \phi_{ij}(\eta) d\eta = 1$. It is shown that a consensus is reached for arbitrary bounded heterogeneous constant, time-varying, or distributed delays if the network topology has a jointly directed spanning tree.

A nonlinear consensus system with nonuniform delays is also given in Reference (37) as follows:

$$\dot{x}_i(t) = k_i \sum_{j=1}^{n} a_{ij} f_{ij}[x_j(t - \tau_{ij}) - x_i(t)] \tag{38}$$

where τ_{ij} denotes the constant time delay from agent j to agent i, k_i are constants, and f_{ij} are locally passive functions on $[-\sigma_{ij}^{-}, \sigma_{ij}^{+}]$ for some $\sigma_{ij}^{-} > 0$ and $\sigma_{ij}^{+} > 0$. For the time-delayed system equation 38, the state space is infinite-dimensional and is given by the Banach space $\mathcal{C}([-\tau, 0], \mathbb{R}^n)$, where $\tau = \max_{i,j} \tau_{ij}$. Define $\gamma = \min_{i,j=1,\ldots,n}\{\sigma_{ij}^{-}, \sigma_{ij}^{+}\}$. If the initial states of the agents are less than $\frac{\gamma}{2}$, it can be proved that consensus is still reached irrespective of the size of the communication delays (37).

6.3. Quantization

For a real digital network, communication channels always have a finite communication capacity. That is, agents can only transmit a finite amount of information at each time step. This indicates that the agents cannot use the exact state information in their control inputs. The above observation motivates the study of quantization effects in consensus.

In what follows, two kinds of quantizers are presented. The first one is the so-called *uniform quantizer*, which is described as follows. For $L_q \in \mathbb{N}$ where $\mathbb{N}$ denotes the set of all natural numbers, define the uniform set of quantization levels (38)

$$S_{L_q} = \left\{-1 + \frac{2l-1}{L_q} : l \in \{1, \ldots, L_q\}\right\} \cup \{-1\} \cup \{1\} \tag{39}$$

The parameter L_q determines the number of quantization levels m in the following way: $m = L_q + 2$. The uniform quantizer is then defined as

$$\mathrm{unq}_{L_q}(x) = \begin{cases} -1 + \frac{2l-1}{L_q} & \text{if } -1 + \frac{2(l-1)}{L_q} \le x \le -1 + \frac{2l-1}{L_q} \\ 1 & \text{if } x > 1 \\ -1 & \text{if } x < -1 \end{cases}$$

where a larger value of L_q corresponds a more accurate uniform quantizer. Another quantizer that is commonly used is the so-called *logarithmic quantizer*. Given an accuracy parameter $\delta \in (0, 1)$, the logarithmic set of quantization levels is defined by

$$S_\delta = \left\{\left(\frac{1+\delta}{1-\delta}\right)^l\right\}_{l\in\mathbb{Z}} \cup \{0\} \cup \left\{-\left(\frac{1+\delta}{1-\delta}\right)^l\right\}_{l\in\mathbb{Z}} \tag{40}$$

The corresponding logarithmic quantizer (39) is given by

$$\mathrm{lgq}_\delta(x) = \begin{cases} (\frac{1+\delta}{1-\delta})^l & \text{if } \frac{(1+\delta)^{l-1}}{(1-\delta)^l} \le x \le \frac{(1+\delta)^l}{(1-\delta)^{l+1}} \\ 0 & \text{if } x = 0 \\ -\mathrm{lgq}_\delta(-x) & \text{if } x < 0 \end{cases} \tag{41}$$

where a smaller value of the parameter δ corresponds to a more accurate logarithmic quantizer.

In Reference (40), a quantized average consensus problem is considered. The objective is to design distributed algorithms such that the value of each agent will finally converge to an integer approximation of the average of the initial states subject to the following two constraints:

(C1) The state of each node is always an integer.

(C2) The sum of states in the network does not change with time.

Let $x(t) = [x_1(t), \ldots, x_n(t)]$ denote the state of the multi-agent system, where $x_i(t)$ is the state of agent i. A class of quantized gossip algorithms is proposed in Reference (40) as follows. For a pair (i, j), if $|x_i(t) - x_j(t)| = 0$, then the states are left unchanged, namely, $x_k(t+1) = x_k(t)$ for $k = i, j$; if $|x_i(t) - x_j(t)| \ge 1$, it is required that

(P1) $x_i(t+1) + x_j(t+1) = x_i(t) + x_j(t)$,

(P2) if $|x_i(t) - x_j(t)| > 1$, then $|x_i(t+1) - x_j(t+1)| < |x_i(t) - x_j(t)|$, and

(P3) if $|x_i(t) - x_j(t)| = 1$ and (without loss of generality) $x_i(t) < x_j(t)$, then $x_i(t+1) = x_j(t)$ and $x_j(t+1) = x_i$.

It is proved that algorithms satisfying (P1)–(P3) can solve the quantized average consensus problem.

Note that in the above quantized average consensus problem, the agents converge to an integer approximation of the average of the initial states, instead of the exact average. What if we want to drive the agents to the exact average while considering quantization effects? It turns out that a dynamic coder or decoder scheme can be introduced to solve the average consensus problem under quantized communication. The coder is used to transform the state to the message to be transmitted to neighboring agents, while the decoder is used to transform the message received from neighboring agents to the state (estimated). For the logarithmic quantizer, the following coder/decoder scheme is studied in Reference (41):

$$\begin{aligned} \xi(t+1) &= \xi(t) + \alpha(t) \\ \alpha(t) &= \mathrm{lgq}_\delta(x(t) - \xi(t)) \\ \hat{x}(t) &= \xi(t) + \alpha(t) \end{aligned} \tag{42}$$

where $\xi(t)$ is the coder/decoder state, $\alpha(t)$ is the coder, and $\hat{x}(t)$ is the decoder. It is shown that if the logarithmic quantizer is accurate enough, that is, δ is small enough, then average consensus can be reached under quantized communication.

6.4. Event-Triggered Mechanism

As an alternative of time-triggered control, event-triggered control has the distinct feature that the control input is updated only when certain event occurs. It is shown that the event-triggered mechanism outperforms the traditional time-triggered mechanism in many cases (42). Whether an event should take place is usually specified via a triggering condition. The times at which an event occurs are called the triggering times. An important requirement of event-triggered control is to avoid the Zeno behavior. That is, the interevent intervals should have a lower bound.

An event-triggered consensus algorithm for equation 1 is given in Reference (43) as

$$u_i(t) = \sum_{j=1}^{n} a_{ij}[x_j(t_{k'(t)}^{j}) - x_i(t_k^i)], \text{ for } t \in [t_k^i, t_{k+1}^i) \tag{43}$$

where x_i is the state of agent i, t_k^i are the event times of agent i at which its control input is updated, and $t_{k'(t)}^j$ is the last event time of agent j. The state measurement error for agent i is then defined by

$$e_i = x_i(t) - x_i(t_k^i), \quad t \in [t_k^i, t_{k+1}^i) \tag{44}$$

Using the following triggering condition

$$e_i^2 = \frac{\sigma_i a(1 - a|\mathcal{N}_i|)}{|\mathcal{N}_i|} z_i^2 \tag{45}$$

where $\mathcal{N}_i$ is the neighbor set of agent i, $0 < \sigma_i < 1$, $0 < a < \frac{1}{|\mathcal{N}_i|}$, and $z_i = \sum_{j \in \mathcal{N}_i}(x_i - x_j)$. It can be shown that consensus is reached for equation 43 under the triggering condition equation 45. Moreover, a positive lower bound on interevent intervals can be derived. Note that to implement the control policy equation 43 and equation 45, the triggering condition needs to be checked continuously. Such continuous detection violates the original purpose of introducing event-triggered control as a means for reducing communication requirements. Therefore, the system equation 43 and equation 45 is extended from the event-triggered case to the self-triggered case in Reference (43). The basic idea of self-triggered control is to find a sequence of update times t_k to satisfy the triggering condition equation 45, where the next update time t_{k+1} is determined using only the information of the system at t_k.

A combination of the event-triggered control scheme and the periodic time-triggered control scheme is proposed in Reference (44). The dynamics of each agent is described by equation 1. The event condition for agent i has the following form

$$\|e_i(t_k^i + lh)\|_2^2 \leq \sigma_i \|z_i(t_k^i + lh)\|_2^2 \tag{46}$$

where σ_i is a positive constant, t_k^i is the kth event time of agent i, and is an integer multiple of h, h is the sampling period, the measurement error is defined by

$$e_i(t_k^i + lh) = x_i(t_k^i) - x_i(t_k^i + lh) \tag{47}$$

and

$$z_i(t_k^i + lh) = \sum_{j \in \mathcal{N}_i} [x_i(t_k^i + lh) - x_j(t_k^i + lh)] \tag{48}$$

At each periodic sampling instant, each agent samples the state information from its neighbors. If the condition in equation 47 is violated, agent i will update its own control action and notify its neighbors to update their control actions by using its current state information. It is shown that if $h \in (0, \frac{1}{2\lambda_n}]$ and $\max_i \sigma_i < \frac{1}{\lambda_n^2}$, where λ_n is the largest eigenvalue of the Laplacian matrix associated with an undirected graph, then consensus can be reached.

6.5. Finite-Time Convergence

The consensus system equation 3 converges in an exponential way, that is, convergence takes place as $t \to \infty$. However, in a number of situations, it is desirable that consensus is reached in a finite time, which motivates the study on finite-time consensus algorithms. Benefits of finite-time consensus include better disturbance rejection and robustness against uncertainties.

Define the potential function $\phi = \frac{1}{2}x^T Lx$, where $x = [x_1, \ldots, x_n]^T$ is the vector of all agents' states with $x_i \in \mathbb{R}$ being the state of agent i and L is the Laplacian matrix. The gradient of ϕ is given by $\nabla\phi = Lx$. In Reference (13), two finite-time consensus systems are proposed as

$$\dot{x} = -\frac{\nabla\phi}{\|\nabla\phi\|} \tag{49}$$

$$\dot{x} = -\mathrm{sgn}(\nabla\phi) \tag{50}$$

where sgn is the signum function. In equation 49, the vector field $\frac{\nabla\phi}{\|\nabla\phi\|}$ is not defined when $\nabla\phi = 0$. However, this would not be a problem for consensus, because when $\nabla\phi = 0$ a consensus has been reached. Equivalently, for each agent, the above two consensus systems can be written in the following way

$$\dot{x}_i = \sum_{j=1}^{n} a_{ij} \frac{(x_j - x_i)}{\|Lx\|} \tag{51}$$

$$\dot{x}_i = \mathrm{sgn}\left[\sum_{j=1}^{n} a_{ij}(x_j - x_i)\right] \tag{52}$$

The consensus system equation 52 is distributed, while equation 51 is not distributed because the term $\|Lx\|$ is global information. These two systems are both discontinuous.

A continuous system that ensures finite-time consensus is introduced in Reference (14) as

$$\dot{x}_i = \sum_{j=1}^{n} a_{ij}\mathrm{sgn}(x_j - x_i)|x_j - x_i|^{\alpha_{ij}} \tag{53}$$

where $0 < \alpha_{ij} < 1$ are constants. For equation 53, we have a few interesting observations: if $\alpha_{ij} = 1$, then equation 53 reduces to the linear consensus algorithm equation 3; if $\alpha_{ij} = 0$, then equation 53 reduces to the following discontinuous system:

$$\dot{x}_i = \sum_{j=1}^{n} a_{ij}\mathrm{sgn}(x_j - x_i) \tag{54}$$

Equation 54 is a special case of the dynamic average consensus algorithm equation 17 with all reference signals equal to zero. Thus, equation 53 can be seen as a bridge connecting the continuous linear system equation 3 and the nonsmooth system equation 54.

The finite-time convergence property of the following discrete-time system

$$x_i(k+1) = w_{ii}x_i(k) + \sum_{j=1}^{n} w_{ij}x_j(k)$$

is considered in Reference (45), where w_{ij} is the weight of the edge (i, j). Let $W = [w_{ij}]$ denote the matrix of weights and let $\lambda_1, \ldots, \lambda_m$ be the m distinct eigenvalues. The minimal polynomial of W is given by $q(t) = \Pi_{i=1}^{m}(t - \lambda_i)^{m_i}$, where m_i is the size of the largest Jordan block of W corresponding to the eigenvalue λ_i. Let d_W denote the degree of the minimal polynomial of W. It is shown that each node can immediately calculate the consensus value after at most d_W time steps.

6.6. State Constraints

Suppose that the state of each agent i is constrained in a nonempty closed convex set X_i that is only available to agent i. Assume that $X = \cap_{i=1}^{n} X_i$ is nonempty. The agents work cooperatively to reach a consensus on a certain value $x^* \in X$. The above problem is called the *constrained consensus* problem (46).

This problem can be solved by the following system:

$$x_i(k+1) = P_{X_i}\left(\sum_{j=1}^{n} a_{ij}(k)x_j(k)\right) \tag{55}$$

where $a_{ij}(k)$ is the (i,j)th element of the adjacency matrix at time k, and the projection function P_{X_i} is defined as

$$P_{X_i}(\bar{x}) = \text{argmin}_{x\in X_i} \|x - \bar{x}\|.$$

The analysis in the case of balanced digraphs is given in Reference 46. There is an interesting connection between the consensus system equation 55 and the following multiagent optimization problem for finding a common point in the given closed convex sets $X_1, \ldots, X_n$:

$$\text{minimize} \quad (1/2)\sum_{i=1}^{n} \|x - P_{X_i}(x)\| \tag{56}$$

The algorithm equation 55 can be seen as a gradient descent algorithm to solve the convex optimization problem equation 56.

The results of Reference 46 have been extended to unbalanced digraphs with communication delays in Reference 47. The system equation 55 becomes

$$x_i(k+1) = P_{X_i}\left(\sum_{j=1}^{n} a_{ij}(k)x_j[k - \tau_{ij}(k)]\right) \tag{57}$$

where $\tau_{ij}(k)$ is the delay from agent j to agent i at time k. The delays are assumed to be upper bounded, and the agents are assumed to have no self delays, that is, $\tau_{ii}(k) = 0$. It is shown that constrained consensus is reached if the unbalanced network is jointly strongly connected.

7. CONNECTIONS WITH OTHER COOPERATIVE CONTROL PROBLEMS

7.1. Synchronization of Coupled Oscillators

Synchronization is a fascinating phenomenon that is observed and studied in various disciplines, such as biology, physics, and engineering applications. The research on synchronization can be traced back to Reference (48), and it still receives intensive attention nowadays (49,50). A simple yet fundamental synchronization model of coupled oscillators is given by

$$\dot{\theta}_i = \omega_i + \kappa \sum_{j=1}^{n} a_{ij}\sin(\theta_j - \theta_i) \tag{58}$$

Here, θ_i is the phase of oscillator i, ω_i is the frequency, $\sin(\theta_i - \theta_j)$ is the nonlinear coupling between oscillators i and j, and κ is the coupling strength. If we ignore the couplings in equation 58, then each isolated oscillator is governed by the following dynamics $\dot{\theta}_i = \omega_i$. The rich behavior of the model equation 58 is a result of both oscillator i's own frequency ω_i and the effect of its neighbors $\kappa\sum_{j=1}^{n} a_{ij}\sin(\theta_j - \theta_i)$. When the network topology is an all-to-all graph, equation 58 becomes the well-known *Kuramoto model*.

If all the frequencies are identical, that is, $\omega_i = \omega$ for all $i = 1, \ldots, n$, then under a suitable transformation on θ_i (51), the model equation 58 can be simplified to

$$\dot{\theta}_i = \kappa\sum_{j=1}^{n} a_{ij}\sin(\theta_j - \theta_i) \tag{59}$$

Define the function $\text{sinc}(x) = \sin(x)/x$ for $x \neq 0$. Equation 59 can then be written as

$$\dot{\theta}_i = \sum_{j=1}^{n} b_{ij}(\theta)(\theta_j - \theta_i) \tag{60}$$

where $b_{ij}(\theta) = a_{ij}\text{sinc}(\theta_j - \theta_i)$ with θ the stack vector of all θ_i. The model has a similar form to that of the consensus model equation 3. The only difference is that the new adjacency matrix $B = [b_{ij}]$ depends on the state θ. That is, equation 60 is essentially a consensus model with a state-dependent network topology.

Despite the above similarity between synchronization and consensus, they have some nontrivial differences. The research on synchronization of coupled oscillators are more focused on the roles of the frequency function ω_i and the coupling function $\kappa\sum_{j=1}^{n} a_{ij}\sin(\theta_j - \theta_i)$, and the relationship between these two functions. It does not give much attention to the network topology and does not care much about whether an algorithm is distributed or not. That is why the Kuramoto model has received so much attention, even though it considers all-to-all network connections. On the contrary, the role of network topologies is highlighted in the research of consensus. The necessary and/or sufficient conditions on network topologies, the impact of asymmetric network topologies, the dynamic network topologies are all central topics in consensus.

7.2. Formation Control

The objective of formation control is to drive a group of agents to form and maintain a prespecified geometric pattern. Usually, the geometric pattern is described by a set of constant vectors p_{ij} $((i,j) \in \mathcal{E})$, which specifies the desired relative position (state) between agents i and j. To reach a formation, we need to design control algorithms such that $x_j - x_i - p_{ji} \to 0$, where x_i denotes the position of agent i. There are various approaches to formation control, such as consensus-based approaches (52), behavior-based approaches (53), and virtual structure-based approaches (54). A large number of formation control problems are solved via consensus-based approaches, which suggests a strong connection between formation control and consensus.

Consider the formation control system

$$\dot{x}_i = \sum_{j=1}^{n} a_{ij}(x_j - x_i - p_{ji}) \tag{61}$$

Suppose that the formation is feasible, that is, there exist x_i^*, $i = 1, \dots, n$, satisfying $x_j^* - x_i^* = p_{ji}$ for $(j, i) \in \mathcal{E}$. Define the new state $\tilde{x}_i = x_i - x_i^*$. Equation 61 can then be rewritten as

$$\dot{\tilde{x}}_i = \sum_{j=1}^{n} a_{ij}(\tilde{x}_j - \tilde{x}_i) \tag{62}$$

which is a consensus system for $\tilde{x}_i$. It is worth pointing out some nontrivial facts in the above transformation. First, in the formation control law equation 61, the formation is specified by relative positions. Second, the formation control law only uses relative position information. These two facts play a critical role in the feasibility of transforming a formation control algorithm to a consensus algorithm.

7.3. Containment Control

In containment control, the agents are required to move toward the convex hull formed by a group of leaders (references). A containment control system takes the following form:

$$\dot{x}_i = -\sum_{j \in \mathcal{V}_F} a_{ij}(x_i - x_j) - \sum_{j \in \mathcal{V}_L} a_{ij}(x_i - r_j) \tag{63}$$

where x_i is the state of the ith agent, r_j is the state of the jth leader, $\mathcal{V}_F$ is the set of the followers, and $\mathcal{V}_L$ is the set of the leaders. Define $x = [x_1, \dots, x_{|\mathcal{V}_F|}]^T$ and $r = [r_1, \dots, r_{|\mathcal{V}_L|}]$, where $|\mathcal{V}_F|$ and $|\mathcal{V}_L|$ denote respectively the number of the followers and the number of the leaders. Equation 63 can be rewritten in a matrix form as

$$\dot{x} = -L_1 x - L_2 r \tag{64}$$

Here, $L_1 = L + A_0$, where L is the Laplacian matrix of the followers, and $A_0 = \text{diag}([a_{10}, \dots, a_{|\mathcal{V}_F|0}])$ with $a_{i0} = \sum_{j \in \mathcal{V}_L} a_{ij} > 0$ if agent i can access at least one leader. The matrix L_2 denotes the coupling between the followers and the leaders.

Since containment control and dynamic average consensus all involve multiple leaders (references), one might wonder whether these two problems are equivalent. It turns out that the containment control problem has essential difference with the dynamic average consensus problem. To illustrate, let us see a containment control example given by Figure 5. Here, for comparison, we let the number of leaders equal to the number of followers. But in general the number of leaders can be different from that of followers. In Figure 5, x_i denotes follower i and r_i denotes leader (reference) i. For convenience, we consider constant references, that is r_i is not time varying. In this example, the matrices defined in equation 64 are given by $L_1 = L + I$ with

$$L \triangleq \begin{bmatrix} 2 & -1 & 0 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

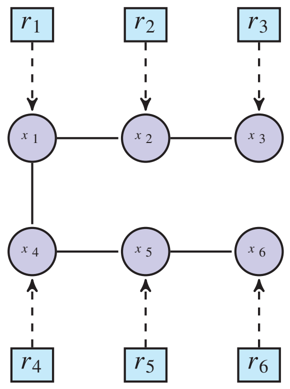

Figure 5. Example of a containment control.

and $L_2 = -I$. In equation 64, x will converge to $-L_1^{-1}L_2 r$, where $-L_1^{-1}L_2$ is a row-stochastic matrix. Suppose that $-L_1^{-1}L_2 = \frac{1}{N}\mathbf{1}\mathbf{1}^T$. Using $L_2 = -I$, it follows that $L_1^{-1} = \frac{1}{N}\mathbf{1}\mathbf{1}^T$, which is not possible because $\frac{1}{N}\mathbf{1}\mathbf{1}^T$ is not invertible. It thus follows that $-L_1^{-1}L_2 \neq \frac{1}{N}\mathbf{1}\mathbf{1}^T$, which indicates that x_i will converge to the convex hull formed by the leaders but not the average of the leaders. Thus, there exists essential difference between containment control and dynamic average consensus.

It turns out that the containment control problem can be transformed to the leader-following consensus problem. In containment control, the tracking error is defined as $e = x + L_1^{-1}L_2 r$. Note that L_1 is invertible. Thus, the derivative of e is given by

$$\begin{aligned} \dot{e} &= -L_1 x - L_2 r \\ &= -L_1(x + L_1^{-1}L_2 r) \\ &= -(L + A_0)e \end{aligned} \tag{65}$$

which has the same form to that of leader-following consensus equation 14. Thus, we can conclude that for constant references the closed-loop system equation 63 for containment control can be reduced to the closed-loop system for leader-following consensus.

7.4. Flocking

Flocking is a phenomenon that can be observed from many natural examples, such as a crowd of birds or a swarm of fish. In flocking, each agent tries to keep an "appropriate" distance from its neighboring agents. This distance should not be too large such that group cohesion can be achieved, and should not be too close such that collision can be avoided. In addition, each agent attempts to match the velocity with its neighboring agents, that is, a consensus on the velocities of the agents should be reached.

Consider a group of the double-integrator agents:

$$\dot{q}_i = p_i$$
$$\dot{p}_i = u_i$$

where q_i is the position of agent i, p_i is its velocity, and u_i is its control input. A flocking algorithm is proposed in Reference 55 as follows:

$$u_i = \underbrace{\sum_{j\in\mathcal{N}_i} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij}}_{\text{gradient-based term}} + \underbrace{\sum_{j\in\mathcal{N}_i} (p_j - p_i)}_{\text{consensus term}} \tag{66}$$

where $\phi_\alpha(z)$ is a smooth pairwise attractive/repulsive potential with a global minimum at $z = d_\alpha$ where $d_\alpha > 0$ is a constant, and $n_{ij} = [(q_j - q_i)/\sqrt{1 + \epsilon\|q_j - q_i\|^2}]$ is a vector along the line connecting q_i to q_j with $\epsilon > 0$ a constant parameter. The σ-norm is defined as $\|z\|_\sigma = \frac{1}{\epsilon}(\sqrt{1 + \epsilon\|z\|^2}) - 1$. Define $V(q) = \frac{1}{2}\sum_i \sum_{j\neq i} \phi_\alpha(\|q_j - q_i\|_\sigma)$. It can be verified that $\sum_{j\in\mathcal{N}_i} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} = -\nabla_{q_i} V(q)$. That is, the first term on the right-hand side of equation 66 is a gradient-based term. The second term on the right-hand side of equation 66 is a consensus term, which is used to reach a consensus on the velocities of the agents.

8. CONCLUSIONS

This article has offered a brief overview on distributed consensus. Several definitions on consensus have been introduced, such as average consensus, leader-following consensus, and dynamic average consensus. Three main techniques for convergence analysis of consensus have been elaborated. The design issues related to consensus have also been discussed. Some advanced topics on consensus, such as complicated agent dynamics, finite-time convergence, and constrained consensus, have also been presented. Finally, the connections of consensus with other cooperative control problems have been investigated.

As a tutorial paper, the primary focus of this article is to present the most basic and fundamental results of consensus, and thus it cannot cover all the topics in this area. Some interesting topics that are missing from this article include consensus under stochastic noises or disturbances (56,57), connectivity maintenance (58,59), consensus under saturated inputs (60), and consensus on general functions (61). There are a number of survey papers on consensus or related topics, see References (62–67) for example. The reader is also referred to the books or research monographs (68–74) for a more detailed and complete treatment of the topic.

Although significant progress has been reported in this area, there remain some unsolved challenges and opportunities.

- *Heterogeneity of agents:* Most of the consensus problems are motivated from certain stability problems for single systems. Therefore, the agents might share some common models or parameters of the single systems. These common models or parameters make the consensus algorithms not fully distributed. It would be more interesting to study multiagent systems with heterogeneous dynamics.
- *Network topology in control:* Currently, the network topology is mainly used to describe the information flows among agents. Its role in control algorithm design has not been fully appreciated. Sometimes, it might be desirable to design or choose different control schemes or parameters for a multiagent system according to its network topology. This leads to two subproblems: one is how to characterize the network topology, and the other is how to define the control law according the chosen characterization of the network. It might be possible to develop a totally new network-dependent control scheme for multiagent systems, which can be a good complement to control theory.
- *Consensus in data mining:* Data mining is a subfield of computer science that aims at learning information from data and transforming it into an appropriate structure for further use. Currently, most data mining algorithms are centralized. It is possible to borrow the idea of consensus to redesign the centralized data mining algorithms such that the newly designed algorithms can be implemented in a distributed manner, while maintaining satisfactory performance. For instance, consensus may be used in aggregation of clustering, whose objective is to find a single clustering that is better than the existing ones.
- *Consensus in biology:* With the advent of genomics era, massive biological data are starting to appear. Tools from classical machine learning are often employed to analyze the biological data. However, due to the huge volume of data, classical machine learning approaches need to be adapted to suit for the "big data," where distributed consensus algorithms might play an important role.

BIBLIOGRAPHY

1. E. Eisenberg and D. Gale. *Ann. Math. Stat.* 1959, pp 165–168.
2. D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc, 1989.
3. J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. *IEEE Trans. Automat. Contr.* **1986**, *31*(9), pp 803–812.
4. A. Jadbabaie, J. Lin, and A. S. Morse. *IEEE Trans. Automat. Contr.* **2003**, *48*(6), pp 998–1001.
5. R. Olfati-Saber and R. M. Murray. *IEEE Trans. Automat. Contr.* **2004**, *49*(9), pp 1520–1533.
6. W. Ren and R. W. Beard. *IEEE Trans. Automat. Contr.* **2005**, *50*(5), pp 655–661.
7. L. Moreau. *IEEE Trans. Automat. Contr.* **2005**, *50*(2), pp 169–182.
8. Z. Lin, M. Broucke, and B. Francis. *IEEE Trans. Automat. Contr.* **2004**, *49*(4), pp 622–629.
9. M. Cao, A. S. Morse, and B. D. Anderson. *SIAM J. Control Optim.* **2008**, *47*(2), pp 575–600.
10. M. Cao, A. S. Morse, and B. D. Anderson. *SIAM J. Control Optim.* **2008**, *47*(2), pp 601–623.
11. W. Ren. *IEEE Trans. Automat. Contr.* **2008**, *53*(6), pp 1503–1509.
12. Z. Lin, B. Francis, and M. Maggiore. *SIAM J. Control Optim.* **2007**, *46*(1), pp 288–307.
13. J. Cortes. *Automatica* **2006**, *42*(11), pp 1993–2000.

14. Q. Hui, W. M. Haddad, and S. P. Bhat. *IEEE Trans. Automat. Contr.* **2008**, *53*(8), pp 1887–1900.
15. D. M. Cvetkovic, M. Doob, and H. Sachs, *Spectra of graphs: Theory and application*, volume 413. Academic Press: New York, 1980.
16. C. Godsil and G. Royle. *Algebraic Graph Theory.* Springer-Verlag: New York, 2001.
17. R. Agaev and P. Chebotarev *Autom. Remote Control* **2000**, *61*(9), pp 1424–1450.
18. D. P. Spanos, R. Olfati-Saber, and R. M. Murray. Dynamic Consensus on Mobile Networks, in *Proc. of The 16th IFAC World Congress*; Prague, Czech, 2005.
19. F. Chen, Y. Cao, and W. Ren. *IEEE Trans. Automat. Contr.* **2012**, *57*(12), pp 3169–3174.
20. V. I. Zubov and L. F. Boron. *Methods of AM Lyapunov and Their Application.* Noordhoff Groningen, 1964.
21. F. H. Clarke. *Optimization and Nonsmooth Analysis.* Wiley & Sons: New York, 1983.
22. L. Moreau. Stability of Continuous-Time Distributed Consensus Algorithms. *IEEE Conference on Decision and Control;* IEEE, pp 3998-4003, 2004.
23. L. Xiao and S. Boyd. *Syst. Control Lett.* **2004**, *53*(1), pp 65–78.
24. Y. Kim and M. Mesbahi. *IEEE Trans. Automat. Contr.* **2006**, *51*(1), pp 116–120.
25. W. Yu, G. Chen, and M. Cao. *Automatica* **2010**, *46*(6), pp 1089–1095.
26. G. Xie and L. Wang. *Int. J. Robust Nonlinear Control* **2007**, *17*(10-11), pp 941–959.
27. Z. Li, Z. Duan, G. Chen, and L. Huang. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2010**, *57*(1), pp 213–224.
28. Z. Li, W. Ren, X. Liu, and M. Fu, *IEEE Trans. Automat. Contr.* **2013**, *58*(7), pp 1786–1791.
29. K. D. Do. *IEEE Trans. Control Syst. Technol.* **2008**, *16*(3), pp 527–538.
30. W. Ren, *Int. J. Control* 2009, *82*(11), pp 2137–2149.
31. E. Nuno, R. Ortega, L. Basanez, and D. Hill. *IEEE Trans. Automat. Contr.* **2011**, *56*(4), pp 935–941.
32. D. V. Dimarogonas, P. Tsiotras, and K. Kyriakopoulos. *Syst. Control Lett.* **2009**, *58*(6), pp 429–435.
33. A. Sarlette, R. Sepulchre, and N. E. Leonard. *Automatica* **2009**, *45*(2), pp 572–577.
34. W. Dong and J. Farrell. *IEEE Trans. Automat. Contr.* **2008**, *53*(6), pp 1434–1448.
35. H. Kim, H. Shim, and J. H. Seo, *IEEE Trans. Automat. Contr.* **2011**, *56*(1), pp 200–206.
36. U. Munz, A. Papachristodoulou, and F. Allgower. *IEEE Trans. Automat. Contr.* **2011**, *56*(12), pp 2976–2982.
37. A. Papachristodoulou, A. Jadbabaie, and U. Munz. *IEEE Trans. Automat. Contr.* **2010**, *55*(6), pp 1471–1477.
38. D. F. Delchamps. *IEEE Trans. Automat. Contr.* **1990**, *35*(8), pp 916–924.
39. N. Elia and S. K. Mitter. *IEEE Trans. Automat. Contr.* **2001**, *46*(9), pp 1384–1400.
40. A. Kashyap, T. Başar, and R. Srikant. *Automatica* **2007**, *43*(7), pp 1192–1203.
41. R. Carli, F. Bullo, and S. Zampieri. *Int. J. Robust Nonlinear Control* **2010**, *20*(2), pp 156–175.
42. X. Meng and T. Chen. *IEEE Trans. Automat. Contr.* **2012**, *57*(12), pp 3252–3259.
43. D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson. *IEEE Trans. Automat. Contr.* **2012**, *57*(5), pp 1291–1297.
44. X. Meng and T. Chen. Automatica **2013**, *49*(7), pp 2125–2132.
45. S. Sundaram and C. N. Hadjicostis. *IEEE Trans. Automat. Contr.* **2011**, *56*(7), pp 1495–1508.
46. A. Nedić, A. Ozdaglar, and P. Parrilo. *IEEE Trans. Automat. Contr.* **2010**, *55*(4), pp 922–938.
47. P. Lin and W. Ren. *IEEE Trans. Automat. Contr.* **2014**, *59*(3), pp 775–781.
48. C. Huygens. *Horologium oscillatorium: 1673.* Dawson, 1966.
49. M. G. Rosenblum, A. S. Pikovsky, and J. Kurths. *Phys. Rev. Lett.* **1996**, *76*(11), p 1804.
50. R. E. Mirollo and S. H. Strogatz, *SIAM J. Appl. Math.* **1990**, *50*(6), pp 1645–1662.
51. F. Dörfler, M. Chertkov, and F. Bullo. *Proc. Natl. Acad. Sci.* **2013**, *110*(6), pp 2005–2010.
52. M. Porfiri, D. G. Roberson, and D. J. Stilwell. *Automatica* **2007**, *43*(8), pp 1318–1328.
53. T. Balch and R. C. Arkin. *IEEE Trans. Robot. Autom.* **1998**, *14*(6), pp 926–939.
54. M. A. Lewis and K. H. Tan. *Autonom. Robot.* **1997**, *4*(4), pp 387–403.
55. R. Olfati-Saber. *IEEE Trans. Automat. Contr.* **2006**, *51*(3), pp 401–420.
56. S. Kar and J. F. Moura. *IEEE Trans. Signal Process.* **2009**, *57*(1), pp 355–369.
57. T. Li and J. F. Zhang. *IEEE Trans. Automat. Contr.* **2010**, *55*(9), pp 2043–2057.
58. M. M. Zavlanos and G. J. Pappas. *IEEE Trans. Robot.* **2008**, *24*(6), pp 1416–1428.
59. T. Gustavi, D. V. Dimarogonas, M. Egerstedt, and X. Hu. *Automatica* **2010**, *46*(1), pp 133–139.
60. H. Su, M. Z. Chen, J. Lam, and Z. Lin, *IEEE Trans. Circuits Syst. I Regul. Pap.* **2013**, *60*(7), pp 1881–1889.
61. J. Cortes. *Automatica* **2008**, *44*(3), pp 726–737.
62. R. Olfati-Saber, J. A. Fax and R. M. Murray. *Proc. IEEE* **2007**, *95*(1), pp 215–233.
63. Y. Cao, W. Yu, W. Ren, and G. Chen. *IEEE Trans. Indus. Inform.* **2013**, *9*(1), pp 427–438.
64. N. E. Leonard, D. Paley, F. Lekien, R. Sepulchre, D. M. Fratantoni, and R. E. Davis. *Proc. IEEE* **2007**, *95*(1), pp 48–74.
65. W. Ren, R. W. Beard, and E. M. Atkins. *IEEE Control Syst. Mag.* **2007**, *2*(27), pp 71–82.
66. G. Antonelli. *IEEE Control Syst. Mag.* **2013**, *33*(1), pp 76–88.
67. F. Garin and L. Schenato. *Networked Control Systems.* Springer-Verlag, 2010, pp 75–107.
68. W. Ren and R. W. Beard. *Distributed Consensus in Multi-Vehicle Cooperative Control.* Springer-Verlag, 2008.
69. F. Bullo, J. Cortés, and S. Martinez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms.* Princeton University Press, 2009.
70. Z. Qu. *Cooperative Control of Dynamical Systems.* Springer-Verlag, 2009.
71. M. Mesbahi and M. Egerstedt. *Graph Theoretic Methods in Multiagent Networks.* Princeton University Press, 2010.

72. H. Bai, M. Arcak, and J. Wen. *Cooperative Control Design: A Systematic, Passivity-Based Approach.* Springer-Verlag, 2011.

73. W. Ren and Y. Cao. *Distributed Coordination of Multi-Agent Networks.* Springer-Verlag, 2011.

74. F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das. *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches.* Springer-Verlag, 2013.

FEI CHEN
Xiamen University, Xiamen, Fujian, China

WEI REN
University of California, Riverside, Riverside, CA, USA