Encyclopedia of Systems and Control
DOI 10.1007/978-1-4471-5102-9_214-1

Averaging Algorithms and Consensus

Wei Ren*
Department of Electrical Engineering, University of California, Riverside, CA, USA

Abstract

In this article, we overview averaging algorithms and consensus in the context of distributed coordination and control of networked systems. The two subjects are closely related but not identical. Distributed consensus means that a team of agents reaches an agreement on certain variables of interest by interacting with their neighbors. Distributed averaging aims at computing the average of certain variables of interest among multiple agents by local communication. Hence averaging can be treated as a special case of consensus – average consensus. For distributed consensus, we introduce distributed algorithms for agents with single-integrator, general linear, and nonlinear dynamics. For distributed averaging, we introduce static and dynamic averaging algorithms. The former is useful for computing the average of initial conditions (or constant signals), while the latter is useful for computing the average of time-varying signals. Future research directions are also discussed.

Keywords Multi-agent systems • Distributed control • Networked systems • Coordination • Cooperative control

Introduction

In the area of control of networked systems, low cost, high adaptivity and scalability, great robustness, and easy maintenance are critical factors. To achieve these factors, distributed coordination and control algorithms that rely on only local interaction between neighboring agents to achieve collective group behavior are more favorable than centralized ones. In this article, we overview averaging algorithms and consensus in the context of distributed coordination and control of networked systems.

Distributed consensus means that a team of agents reaches an agreement on certain variables of interest by interacting with their neighbors. A consensus algorithm is an update law that drives the variables of interest of all agents in the network to converge to a common value (Jadbabaie et al. 2003; Olfati-Saber et al. 2007; Ren and Beard 2008). Examples of the variables of interest include a local representation of the center and shape of a formation, the rendezvous time, the length of a perimeter being monitored, the direction of motion for a multi-vehicle swarm, and the probability that a target has been identified. Consensus algorithms have applications in rendezvous, formation control, flocking, attitude alignment, and sensor networks (Bullo et al. 2009; Qu 2009; Mesbahi and Egerstedt 2010; Ren and Cao 2011; Bai et al. 2011a). Distributed averaging algorithms aim at computing the average of certain variables of interest among multiple agents by local communication. Distributed averaging finds applications in distributed computing, distributed signal processing, and distributed optimization (Tsitsiklis et al. 1986). Hence the variables of interest are dependent on the applications (e.g., a sensor measurement or a network

*E-mail: ren@ee.ucr.edu

Encyclopedia of Systems and Control
DOI 10.1007/978-1-4471-5102-9_214-1

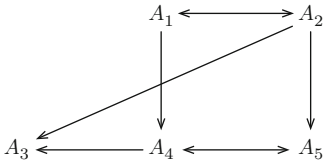

Fig. 1 A directed graph that characterizes the interaction among five agents, where $A_i, i = 1, \ldots, 5$, denotes agent i. An *arrow* from agent j to agent i indicates that agent i receives information from agent j. The directed graph has a directed spanning tree but is not strongly connected. Here both agents 1 and 2 have directed paths to all other agents

quantity). Consensus and averaging algorithms are closely connected and yet nonidentical. When all agents are able to compute the average, they essentially reach a consensus, the so-called average consensus. On the other hand, when the agents reach a consensus, the consensus value might or might not be the average value.

Graph Theory Notations. Suppose that there are n agents in a network. A *network topology* (equivalently, *graph*) $\mathcal{G}$ consisting of a node set $\mathcal{V} = \{1, \ldots, n\}$ and an edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ will be used to model *interaction* (communication or sensing) between the n agents. An *edge* (i, j) in a *directed graph* denotes that agent j can obtain information from agent i, but not necessarily vice versa. In contrast, an edge (i, j) in an *undirected graph* denotes that agents i and j can obtain information from each other. Agent j is a *(in-) neighbor* of agent i if $(j, i) \in \mathcal{E}$. Let $\mathcal{N}_i$ denote the neighbor set of agent i. We assume that $i \in \mathcal{N}_i$. A *directed path* is a sequence of edges in a directed graph of the form $(i_1, i_2), (i_2, i_3), \ldots$, where $i_j \in \mathcal{V}$. An *undirected path* in an undirected graph is defined analogously. A directed graph is *strongly connected* if there is a directed path from every agent to every other agent. An undirected graph is *connected* if there is an undirected path between every pair of distinct agents. A directed graph *has a directed spanning tree* if there exists at least one agent that has directed paths to all other agents. For example, Fig. 1 shows a directed graph that has a directed spanning but is not strongly connected. The *adjacency matrix* $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$ associated with $\mathcal{G}$ is defined such that a_{ij} (the weight of edge (j, i)) is positive if agent j is a neighbor of agent i while $a_{ij} = 0$ otherwise. The (nonsymmetric) *Laplacian matrix* (Agaev and Chebotarev 2005) $\mathcal{L} = [\ell_{ij}] \in \mathbb{R}^{n \times n}$ associated with $\mathcal{A}$ and hence $\mathcal{G}$ is defined as $\ell_{ii} = \sum_{j \neq i} a_{ij}$ and $\ell_{ij} = -a_{ij}$ for all $i \neq j$. For an undirected graph, we assume that $a_{ij} = a_{ji}$. A graph is *balanced* if for every agent the total edge weights of its incoming links is equal to the total edge weights of its outgoing links ($\sum_{j=1}^{n} a_{ij} = \sum_{j=1}^{n} a_{ji}$ for all i).

Consensus

Consensus has a long history in management science, statistical physics, and distributed computing and finds recent interests in distributed control. While in the area of distributed control of networked systems the term *consensus* was initially more or less dominantly referred to the case of a continuous-time version of a distributed linear averaging algorithm, such a term has been broadened to a great extent later on. Related problems to consensus include synchronization, agreement, and rendezvous. The study of consensus can be categorized in various manners. For example, in terms of the final consensus value, the agents could reach a consensus on the average, a weighted average, the maximum value, the minimum value, or a general function of their initial conditions, or even a (changing) state that serves as a reference. A consensus algorithm

DOI 10.1007/978-1-4471-5102-9_214-1

could be linear or nonlinear. Consensus algorithms can be designed for agents with linear or nonlinear dynamics. As the agent dynamics become more complicated, so do the algorithm design and analysis. Numerous issues are also involved in consensus such as network topologies (fixed vs. switching, deterministic vs. random, directed vs. undirected, asynchronous vs. synchronous), time delay, quantization, optimality, sampling effects, and convergence speed. For example, in real applications, due to nonuniform communication/sensing ranges or limited field of view of sensors, the network topology could be directed rather than undirected. Also due to unreliable communication/sensing and limited communication/sensing ranges, the network topology could be switching rather than fixed.

Consensus for Agents with Single-Integrator Dynamics

We start with a fundamental consensus algorithm for agents with single-integrator dynamics. The results in this section follow from Jadbabaie et al. (2003), Olfati-Saber et al. (2007), Ren and Beard (2008), Moreau (2005), and Agaev and Chebotarev (2000). Consider agents with single-integrator dynamics

$$\dot{x}_i(t) = u_i(t), \quad i = 1, \ldots, n, \tag{1}$$

where x_i is the state and u_i is the control input. A common consensus algorithm for (1) is

$$u_i(t) = \sum_{j \in \mathcal{N}_i(t)} a_{ij}(t)[x_j(t) - x_i(t)], \tag{2}$$

where $\mathcal{N}_i(t)$ is the neighbor set of agent i at time t and $a_{ij}(t)$ is the (i, j) entry of the adjacency matrix $\mathcal{A}$ of the graph $\mathcal{G}$ at time t. A consequence of (2) is that the state $x_i(t)$ of agent i is driven toward the states of its neighbors or equivalently toward the weighted average of its neighbors' states. The closed-loop system of (1) using (2) can be written in matrix form as

$$\dot{x}(t) = -\mathcal{L}(t)x(t), \tag{3}$$

where x is a column stack vector of all x_i and $\mathcal{L}$ is the Laplacian matrix. Consensus is *reached* if for all initial states, the agents' states eventually become identical. That is, for all $x_i(0)$, $\|x_i(t) - x_j(t)\|$ approaches zero eventually.

The properties of the Laplacian matrix $\mathcal{L}$ play an important role in the analysis of the closed-loop system (3). When the graph $\mathcal{G}$ (and hence the associated Laplacian matrix $\mathcal{L}$) is fixed, (3) can be analyzed by studying the eigenvalues and eigenvectors of $\mathcal{L}$. Due to its special structure, for any graph $\mathcal{G}$, the associated Laplacian matrix $\mathcal{L}$ has at least one zero eigenvalue with an associated right eigenvector $\mathbf{1}$ (column vector of all ones) and all other eigenvalues have positive real parts. To ensure consensus, it is equivalent to ensure that $\mathcal{L}$ has a simple zero eigenvalue. It can be shown that the following three statements are equivalent: (i) the agents reach a consensus exponentially for arbitrary initial states; (ii) the graph $\mathcal{G}$ has a directed spanning tree; and (iii) the Laplacian matrix $\mathcal{L}$ has a simple zero eigenvalue with an associated right eigenvector $\mathbf{1}$ and all other eigenvalues have positive real parts. When consensus is reached, the final consensus value is a weighted average of the initial states of those agents that have directed paths to all other agents (see Fig. 2 for an illustration).

DOI 10.1007/978-1-4471-5102-9_214-1

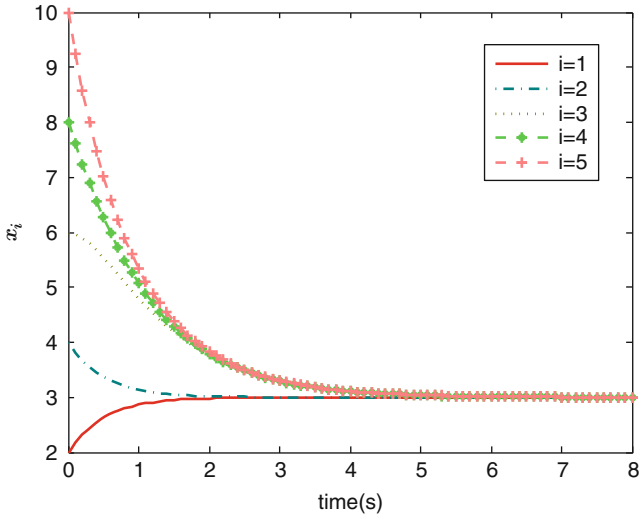

Fig. 2 Consensus for five agents using the algorithm (2) for (1). Here the graph $\mathcal{G}$ is given by Fig. 1. The initial states are chosen as $x_i(0) = 2i$, where $i = 1, \ldots, 5$. Consensus is reached as $\mathcal{G}$ has a directed spanning tree. The final consensus value is a weighted average of the initial states of agents 1 and 2

When the graph $\mathcal{G}(t)$ is switching at time instants $t_0, t_1, \ldots$, the solution to the closed-loop system (3) is given by $x(t) = \Phi(t, 0)x(0)$, where $\Phi(t, 0)$ is the transition matrix corresponding to $-\mathcal{L}(t)$. Consensus is reached if $\Phi(t, 0)$ eventually converges to a matrix with identical rows. Here $\Phi(t, 0) = \Phi(t, t_k)\Phi(t_k, t_{k-1}) \cdots \Phi(t_1, 0)$, where $\Phi(t_k, t_{k-1})$ is the transition matrix corresponding to $\mathcal{L}(t)$ at time interval $[t_{k-1}, t_k]$. It turns out that each transition matrix is a *row-stochastic* matrix with positive diagonal entries. A square matrix is row stochastic if all its entries are nonnegative and all of its row sums are one. The consensus convergence can be analyzed by studying the product of row-stochastic matrices. Another analysis technique is a Lyapunov approach (e.g., $\max x_i - \min x_i$). It can be shown that the agents' states reach a consensus if there exists an infinite sequence of contiguous, uniformly bounded time intervals, with the property that across each such interval, the union of the graphs $\mathcal{G}(t)$ has a directed spanning tree. That is, across each such interval, there exists at least one agent that can directly or indirectly influence all other agents. It is also possible to achieve certain nice features by designing nonlinear consensus algorithms of the form $u_i(t) = \sum_{j \in \mathcal{N}_i(t)} a_{ij}(t)\psi[x_j(t) - x_i(t)]$, where $\psi(\cdot)$ is a nonlinear function satisfying certain properties. One example is a continuous nondecreasing odd function. For example, a saturation type function could be introduced to account for actuator saturation and a signum type function could be introduced to achieve finite-time convergence.

As shown above, for single-integrator dynamics, the consensus convergence is determined entirely by the network topologies. The primary reason is that the single-integrator dynamics are internally stable. However, when more complicated agent dynamics are involved, the consensus algorithm design and analysis become more complicated. On one hand, whether the graph is undirected (respectively, switching) or not has significant influence on the complexity of the consensus analysis. On the other hand, not only the network topology but also the agent dynamics themselves and the parameters in the consensus algorithm play important roles. Next we introduce consensus for agents with general linear and nonlinear dynamics.

Encyclopedia of Systems and Control
DOI 10.1007/978-1-4471-5102-9_214-1

Consensus for Agents with General Linear Dynamics

In some circumstances, it is relevant to deal with agents with general linear dynamics, which can also be regarded as linearized models of certain nonlinear dynamics. The results in this section follow from Li et al. (2010). Consider agents with general linear dynamics

$$\dot{x}_i = Ax_i + Bu_i, \quad y_i = Cx_i, \tag{4}$$

where $x_i \in \mathbb{R}^m$, $u_i \in \mathbb{R}^p$, and $y_i \in \mathbb{R}^q$ are, respectively, the state, the control input, and the output of agent i and A, B, C are constant matrices with compatible dimensions.

When each agent has access to the relative states between itself and its neighbors, a distributed static consensus algorithm is designed for (4) as

$$u_i = cK \sum_{j \in \mathcal{N}_i} a_{ij}(x_i - x_j), \tag{5}$$

where $c > 0$ is a coupling gain, $K \in \mathbb{R}^{p \times m}$ is the feedback gain matrix, and $\mathcal{N}_i$ and a_{ij} are defined as in (2). It can be shown that if the graph $\mathcal{G}$ has a directed spanning tree, consensus is reached using (5) for (4) if and only if all the matrices $A + c\lambda_i(\mathcal{L})BK$, where $\lambda_i(\mathcal{L}) \neq 0$ are Hurwitz. Here $\lambda_i(\mathcal{L})$ denotes the ith eigenvalue of the Laplacian matrix $\mathcal{L}$. A necessary condition for reaching a consensus is that the pair (A, B) is stabilizable. The consensus algorithm (5) can be designed via two steps:

(a) Solve the linear matrix inequality $A^T P + PA - 2BB^T < 0$ to get a positive-definite solution P. Then let the feedback gain matrix $K = -B^T P^{-1}$.
(b) Select the coupling strength c larger than the threshold value $1/\min_{\lambda_i(\mathcal{L}) \neq 0} \mathrm{Re}[\lambda_i(\mathcal{L})]$, where $\mathrm{Re}(\cdot)$ denotes the real part.

Note that here the threshold value depends on the eigenvalues of the Laplacian matrix, which is in some sense global information. To overcome such a limitation, it is possible to introduce adaptive gains in the algorithm design. The gains could be updated dynamically using local information.

When the relative states between each agent and its neighbors are not available, one is motivated to make use of the output information and employ observer-based design to estimate the relative states. An observer-type consensus algorithm is designed for (4) as

$$\begin{aligned} \dot{v}_i &= (A + BF)v_i + cL \sum_{j \in \mathcal{N}_i} a_{ij}[C(v_i - v_j) - (y_i - y_j)], \\ u_i &= Fv_i, \quad i = 1, \cdots, n, \end{aligned} \tag{6}$$

where $v_i \in \mathbb{R}^m$ are the observer states, $F \in \mathbb{R}^{p \times n}$ and $L \in \mathbb{R}^{m \times q}$ are the feedback gain matrices, and $c > 0$ is a coupling gain. Here the algorithm (6) uses not only the relative outputs between each agent and its neighbors but also its own and neighbors' observer states. While relative outputs could be obtained through local measurements, the neighbors' observer states can only be obtained via communication. It can be shown that if the graph $\mathcal{G}$ has a directed spanning tree, consensus is reached using (6) for (4) if the matrices $A + BF$ and $A + c\lambda_i(\mathcal{L})LC$, where $\lambda_i(\mathcal{L}) \neq 0$, are Hurwitz. The observer-type consensus algorithm (6) can be seen as an extension of the single-

DOI 10.1007/978-1-4471-5102-9_214-1

system observer design to multi-agent systems. Here the *separation principle* of the traditional observer design still holds in the multi-agent setting in the sense that the feedback gain matrices F and L can be designed separately.

Consensus for Agents with Nonlinear Dynamics

In multi-agent applications, agents usually represent physical vehicles with special dynamics, especially nonlinear dynamics for the most part. Examples include Lagrangian systems for robotic manipulators and autonomous robots, nonholonomic systems for unicycles, attitude dynamics for rigid bodies, and general nonlinear systems. Similar to the consensus algorithms for linear multi-agent systems, the consensus algorithms used for these nonlinear agents are often designed based on state differences between each agent and its neighbors. But due to the inherent nonlinearity, the problem is more complicated and additional terms might be required in the algorithm design. The main techniques used in the consensus analysis for nonlinear multi-agent systems are often Lyapunov-based techniques (Lyapunov functions, passivity theory, nonlinear contraction analysis, and potential functions).

Early results on consensus for agents with nonlinear dynamics primarily focus on undirected graphs to exploit the symmetry to facilitate the construction of Lyapunov function candidates. Unfortunately, the extension from an undirected graph to a directed one is nontrivial. For example, the directed graph does not preserve the passivity properties in general. Moreover, the directed graph could cause difficulties in the design of (positive-definite) Lyapunov functions. One approach is to integrate the nonnegative left eigenvector of the Laplacian matrix associated with the zero eigenvalue into the Lyapunov function, which is valid for strongly connected graphs and has been applied in some problems. Another approach is based on sliding mode control. The idea is to design a sliding surface for reaching a consensus. Taking multiple Lagrangian systems as an example, the agent dynamics are represented by

$$M_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + g_i(q_i) = \tau_i, \quad i = 1, \cdots, n, \tag{7}$$

where $q_i \in \mathbb{R}^p$ is the vector of generalized coordinates, $M_i(q_i) \in \mathbb{R}^{p\times p}$ is the symmetric positive-definite inertia matrix, $C_i(q_i, \dot{q}_i)\dot{q}_i \in \mathbb{R}^p$ is the vector of Coriolis and centrifugal torques, $g_i(q_i) \in \mathbb{R}^p$ is the vector of gravitational torque, and $\tau_i \in \mathbb{R}^p$ is the vector of control torque on the ith agent. The sliding surface can be designed as

$$s_i = \dot{q}_i - \dot{q}_{ri} = \dot{q}_i + \alpha \sum_{j\in\mathcal{N}_i} a_{ij}(q_i - q_j) \tag{8}$$

where α is a positive scalar. Note that when $s_i = 0$, (8) is actually the closed-loop system of a consensus algorithm for single integrators. Then if the control torque τ_i can be designed using only local information from neighbors to drive s_i to zero, consensus will be reached as s_i can be treated as a vanishing disturbance to a system that reaches consensus exponentially.

It is generally very challenging to deal with general directed or switching graphs for agents with more complicated dynamics other than single-integrator dynamics. In some cases, the challenge could be overcome by introducing and updating additional auxiliary variables (often observer-based algorithms) and exchanging these variables between neighbors (see, e.g., (6)). In the algorithm design, the agents might use not only relative physical states between neighbors but also

DOI 10.1007/978-1-4471-5102-9_214-1

local auxiliary variables from neighbors. While relative physical states could be obtained through sensing, the exchange of auxiliary variables can only be achieved by communication. Hence such generalization is obtained at the price of increased communication between the neighboring agents. Unlike some other algorithms, it is generally impossible to implement the algorithm relying on purely relative sensing between neighbors without the need for communication.

Averaging Algorithms

Existing distributed averaging algorithms are primarily static averaging algorithms based on linear local average iterations or gossip iterations. These algorithms are capable of computing the average of the initial conditions of all agents (or constant signals) in a network. In particular, the linear local-average-iteration algorithms are usually synchronous, where at each iteration each agent repeatedly updates its state to be the average of those of its neighbors. The gossip algorithms are asynchronous, where at each iteration a random pair of agents are selected to exchange their states and update them to be the average of the two. Dynamic averaging algorithms are of significance when there exist time-varying signals. The objective is to compute the average of these time-varying signals in a distributed manner.

Static Averaging

Take a linear local-average-iteration algorithm as an example. The results in this section follow from Tsitsiklis et al. (1986), Jadbabaie et al. (2003), and Olfati-Saber et al. (2007). Let x_i be the information state of agent i. A linear local-average-iteration-type algorithm has the form

$$x_i[k+1] = \sum_{j \in \mathcal{N}_i[k]} a_{ij}[k] x_j[k], \quad i = 1, \ldots, n, \tag{9}$$

where k denotes a communication event, $\mathcal{N}_i[k]$ denotes the neighbor set of agent i, and $a_{ij}[k]$ is the (i, j) entry of the adjacency matrix $\mathcal{A}$ of the graph $\mathcal{G}$ that represents the communication topology at time k, with the additional assumption that $\mathcal{A}$ is row stochastic and $a_{ii}[k] > 0$ for all $i = 1, \ldots, n$. Intuitively, the information state of each agent is updated as the weighted average of its current state and the current states of its neighbors at each iteration. Note that an agent maintains its current state if it does not exchange information with other agents at that event instant. In fact, a discretized version of the closed-loop system of (1) using (2) (with a sufficiently small sampling period) takes in the form of (9). The objective here is for all agents to compute the average of their initial states by communicating with only their neighbors. That is, each $x_i[k]$ approaches $\frac{1}{n}\sum_{j=1}^{n} x_j[0]$ eventually. To compute the average of multiple constant signals c_i, we could simply set $x_i[0] = c_i$. The algorithm (9) can be written in matrix form as $x[k+1] = \mathcal{A}[k]x[k]$, where x is a column stack vector of all x_i and $\mathcal{A}[k] = [a_{ij}[k]]$ is a row-stochastic matrix.

When the graph $\mathcal{G}$ (and hence the matrix $\mathcal{A}$) is fixed, the convergence of the algorithm (9) can be analyzed by studying the eigenvalues and eigenvectors of the row-stochastic matrix $\mathcal{A}$. Because all diagonal entries of $\mathcal{A}$ are positive, Gershgorin's disc theorem implies that all eigenvalues of $\mathcal{A}$ are either within the open unit disk or at one. When the graph $\mathcal{G}$ is strongly connected, the Perron-Frobenius theorem implies that $\mathcal{A}$ has a simple eigenvalue at one with an associated right

eigenvector $\mathbf{1}$ and an associated positive left eigenvector. Hence when $\mathcal{G}$ is strongly connected, it turns out that $\lim_{k\to\infty}\mathcal{A}^k = \mathbf{1}v^T$, where v^T is a positive left eigenvector of $\mathcal{A}$ associated with the eigenvalue one and satisfies $v^T\mathbf{1} = 1$. Note that $x[k] = \mathcal{A}^k x[0]$. Hence, each agent's state $x_i[k]$ approaches $v^T x[0]$ eventually. If it can be further ensured that $v = \frac{1}{n}\mathbf{1}$, then averaging is achieved. It can be shown that the agents' states converge to the average of their initial values if and only if the directed graph $\mathcal{G}$ is both strongly connected and balanced or the undirected graph $\mathcal{G}$ is connected. When the graph is switching, the convergence of the algorithm (9) can be analyzed by studying the product of row-stochastic matrices. Such analysis is closely related to Markov chains. It can be shown that the agents' states converge to the average of their initial values if the directed graph $\mathcal{G}$ is balanced at each communication event and strongly connected in a joint manner or the undirected graph $\mathcal{G}$ is jointly connected.

Dynamic Averaging

In a more general setting, there exist n time-varying signals, $r_i(t)$, $i = 1,\ldots,n$, which could be an external signal or an output from a dynamical system. Here $r_i(t)$ is available to only agent i and each agent can exchange information with only its neighbors. Each agent maintains a local estimate, denoted by $x_i(t)$, of the average of all the signals $\bar{r}(t) \triangleq \frac{1}{n}\sum_{k=1}^n r_k(t)$. The objective is to design a distributed algorithm for agent i based on $r_i(t)$ and $x_j(t)$, $j \in \mathcal{N}_i(t)$, such that all agents will finally track the average that changes over time. That is, $\|x_i(t) - \bar{r}(t)\|$, $i = 1,\ldots,n$, approaches zero eventually. Such a dynamic averaging idea finds applications in distributed sensor fusion with time-varying measurements (Spanos and Murray 2005; Bai et al. 2011b) and distributed estimation and tracking (Yang et al. 2008).

Figure 3 illustrates the dynamic averaging idea. If there exists a central station that can always access the signals of all agents, then it is trivial to compute the average. Unfortunately, in a distributed context, where there does not exist a central station and each agent can only communicate with its local neighbors, it is challenging for each agent to compute the average that changes over time. While each agent could compute the average of its own and local neighbors' signals, this will not be the average of all signals.

When the signal $r_i(t)$ can be arbitrary but its derivative exists and is bounded almost everywhere, a distributed nonlinear nonsmooth algorithm is designed in Chen et al. (2012) as

$$\begin{aligned}\dot{\phi}_i(t) &= \alpha\sum_{j\in\mathcal{N}_i}\mathrm{sgn}[x_j(t) - x_i(t)]\\ x_i(t) &= \phi_i(t) + r_i(t), \quad i = 1,\ldots,n,\end{aligned} \tag{10}$$

where α is a positive scalar, $\mathcal{N}_i$ denotes the neighbor set of agent i, $\mathrm{sgn}(\cdot)$ is the signum function defined componentwise, ϕ_i is the internal state of the estimator with $\phi_i(0) = 0$, and x_i is the estimate of the average $\bar{r}(t)$. Due to the existence of the discontinuous signum function, the solution of (10) is understood in the Filippov sense (Cortes 2008).

The idea behind the algorithm (10) is as follows. First, (10) is designed to ensure that $\sum_{i=1}^n x_i(t) = \sum_{i=1}^n r_i(t)$ holds for all time. Note that $\sum_{i=1}^n x_i(t) = \sum_{i=1}^n \phi_i(t) + \sum_{i=1}^n r_i(t)$. When the graph $\mathcal{G}$ is undirected and $\phi_i(0) = 0$, it follows that $\sum_{i=1}^n \phi_i(t) = \sum_{i=1}^n \phi_i(0) + \alpha\sum_{i=1}^n\sum_{j\in\mathcal{N}_i}\int_0^t \mathrm{sgn}[x_j(\tau) - x_i(\tau)]d\tau = 0$. As a result, $\sum_{i=1}^n x_i(t) = \sum_{i=1}^n r_i(t)$ holds for all time. Second, when $\mathcal{G}$ is connected, if the algorithm (10) guarantees that all estimates x_i approach

DOI 10.1007/978-1-4471-5102-9_214-1

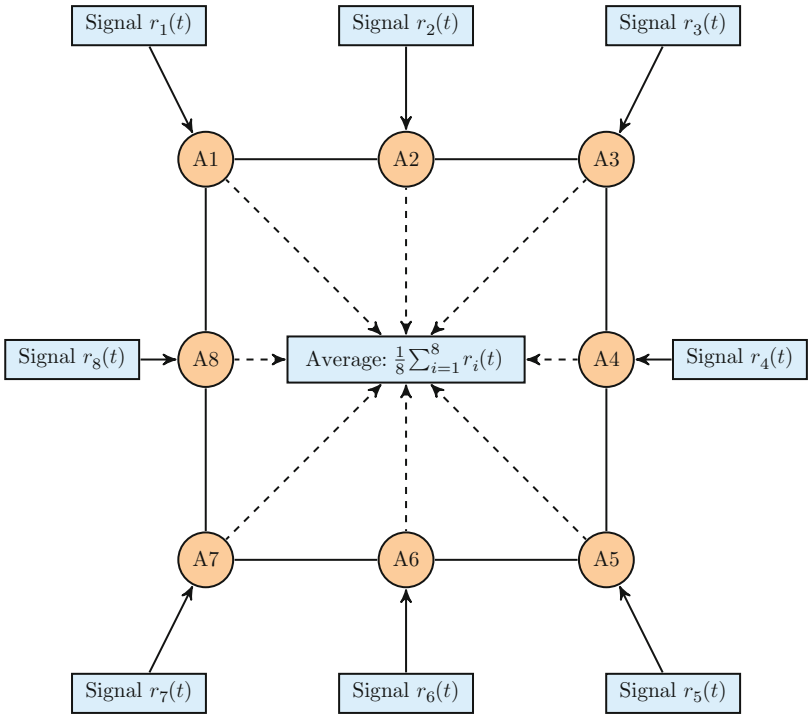

Fig. 3 Illustration of distributed averaging of multiple (time-varying) signals. Here A_i denotes agent i and $r_i(t)$ denotes a (time-varying) signal associated with agent i. Each agent needs to compute the average of all agents' signals but can communicate with only its neighbors

the same value in *finite time*, then it can be guaranteed that each estimate approaches the average of all signals in finite time.

Summary and Future Research Directions

Averaging algorithms and consensus play an important role in distributed control of networked systems. While there is significant progress in this direction, there are still numerous open problems. For example, it is challenging to achieve averaging when the graph is not balanced. It is generally not clear how to deal with a general directed or switching graph for nonlinear agents or nonlinear algorithms when the algorithms are based on only interagent physical state coupling without the need for communicating additional auxiliary variables between neighbors. The study of consensus for multiple underactuated agents remains a challenge. Furthermore, when the agents' dynamics are heterogeneous, it is challenging to design consensus algorithms. In addition, in the existing study, it is often assumed that the agents are cooperative. When there exist faulty or malicious agents, the problem becomes more involved.

Cross-References

▸ Dynamic Graphs, Connectivity
▸ Distributed Optimization
▸ Flocking in Networked Systems

DOI 10.1007/978-1-4471-5102-9_214-1

▸ Graphs for Modeling Networked Interactions
▸ Control of Networked Systems, Overview
▸ Oscillator Synchronization
▸ Vehicular chains

Bibliography

Agaev R, Chebotarev P (2000) The matrix of maximum out forests of a digraph and its applications. Autom Remote Control 61(9):1424–1450

Agaev R, Chebotarev P (2005) On the spectra of nonsymmetric Laplacian matrices. Linear Algebra Appl 399:157–178

Bai H, Arcak M, Wen J (2011a) Cooperative control design: a systematic, passivity-based approach. Springer, New York

Bai H, Freeman RA, Lynch KM (2011b) Distributed Kalman filtering using the internal model average consensus estimator. In: Proceedings of the American control conference, San Francisco, pp 1500–1505

Bullo F, Cortes J, Martinez S (2009) Distributed control of robotic networks. Princeton University Press, Princeton

Chen F, Cao Y, Ren W (2012) Distributed average tracking of multiple time-varying reference signals with bounded derivatives. IEEE Trans Autom Control 57(12):3169–3174

Cortes J (2008) Discontinuous dynamical systems. IEEE Control Syst Mag 28(3):36–73

Jadbabaie A, Lin J, Morse AS (2003) Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans Autom Control 48(6):988–1001

Li Z, Duan Z, Chen G, Huang L (2010) Consensus of multiagent systems and synchronization of complex networks: a unified viewpoint. IEEE Trans Circuits Syst I Regul Pap 57(1):213–224

Mesbahi M, Egerstedt M (2010) Graph theoretic methods for multiagent networks. Princeton University Press, Princeton

Moreau L (2005) Stability of multi-agent systems with time-dependent communication links. IEEE Trans Autom Control 50(2):169–182

Olfati-Saber R, Fax JA, Murray RM (2007) Consensus and cooperation in networked multi-agent systems. Proc IEEE 95(1):215–233

Qu Z (2009) Cooperative control of dynamical systems: applications to autonomous vehicles. Springer, London

Ren W, Beard RW (2008) Distributed consensus in multi-vehicle cooperative control. Springer, London

Ren W, Cao Y (2011) Distributed coordination of multi-agent networks. Springer, London

Spanos DP, Murray RM (2005) Distributed sensor fusion using dynamic consensus. In: Proceedings of the IFAC world congress, Prague

Tsitsiklis JN, Bertsekas DP, Athans M (1986) Distributed asynchronous deterministic and stochastic gradient optimization algorithms. IEEE Trans Autom Control 31(9):803–812

Yang P, Freeman RA, Lynch KM (2008) Multi-agent coordination by decentralized estimation and control. IEEE Trans Autom Control 53(11):2480–2496