Intelligent distributed systems

Daniele Fontanelli

Department of Industrial Engineering
University of Trento
E-mail address: *daniele.fontanelli@unitn.it*

2022/2023

Outline

1. Background
   - Stochastic matrices
2. Linear Consensus
   - Continuous time systems
   - Design of the consensus algorithm
     - Continuous time case
     - Discrete time case
3. Take home message

Outline

Graph properties

Many results in the field of distributed systems can be reformulated with graph properties.

Definition (Graph)

A *graph* is an ordered pair $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ comprising a set $\mathcal{N}$ of *nodes* or *vertices* together with a set $\mathcal{E}$ of *edges*, which are 2-element subsets of $\mathcal{N}$.

Usually the graph is used to describe the *topology* of the systems' connection in a network. Each node represents a system while an edge the communication between the two connected systems.

Graph representation

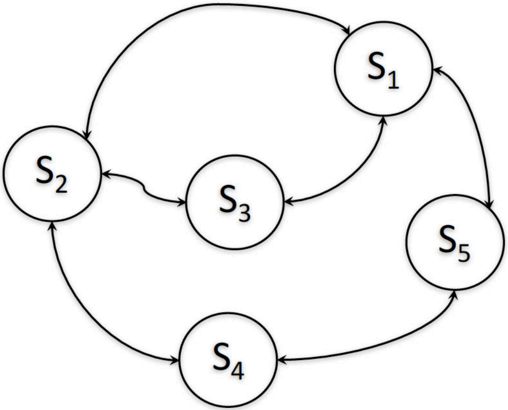

Figure: The graph representation of the distributed systems for a generic network topology.

Graph properties

Digraph and undirected graphs

Definition (Edge)

Let $\mathcal{N} = \{1, 2, \dots, n\}$ be the set of nodes. Hence, the pair $(j, i) \in \mathcal{E}$ implies that node i can *receive information* from node j.

Definition (Digraph)

A *directed graph* or *digraph* is an ordered pair $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ in which the edges, known also as *arcs* in this case, connects an *ordered* pairs of nodes.

Definition (Undirected Graph)

An *undirected graph* is an ordered pair $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ in which if $(j, i) \in \mathcal{E}$ hence $(i, j) \in \mathcal{E}$.

Graph representation

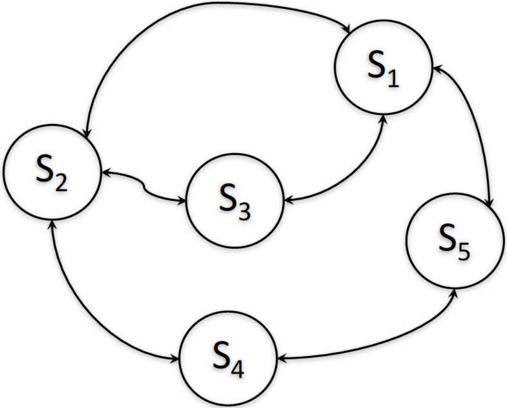

Figure: Undirected graph.

Graph representation

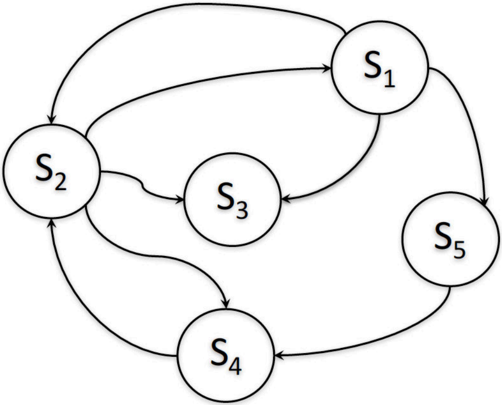

Figure: Directed graph.

Graph properties

Practical implication

- In practice, the edge represents a *communication link* between two nodes.
- In a digraph, the communication obeys a *simplex* modality.
- In an undirected graph, the communication obeys to the *half-duplex* or *full-duplex* modality. The difference between the two is that in the latter the communication is bidirectional and in the same time instant.

Graph properties

Self loops and nodes degree

Definition (Self loops)

A graph includes all the *self-loops* if and only if $(i, i) \in \mathcal{E}$, $\forall i \in \mathcal{N}$.

Definition (Set of sending neighbours and in-degree)

The *set of neighbours that can send* the information to node i is defined by $\mathcal{V}_{in}(i) = \{j | (j, i) \in \mathcal{E}, i \neq j\}$. Hence, the *in-degree* of node i is defined as $d_{in}(i) = |\mathcal{V}_{in}(i)|$, where $| \cdot |$ represents the cardinality of a set.

Definition (Set of receiving neighbours and out-degree)

The *set of neighbours that can receive* the information from node i is defined by $\mathcal{V}_{out}(i) = \{j | (i, j) \in \mathcal{E}, i \neq j\}$. Hence, the *out-degree* of node i is defined as $d_{out}(i) = |\mathcal{V}_{out}(i)|$.

Graph representation

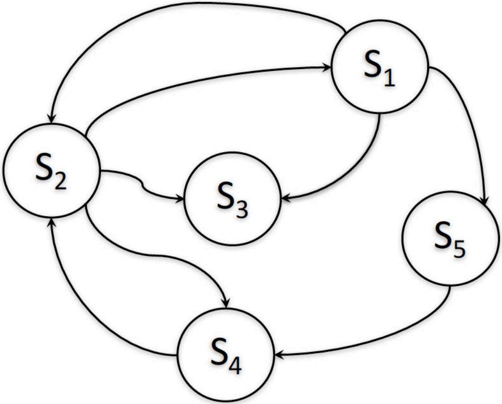

Figure: $\mathcal{V}_{in}(1) = \{2\}$, $\mathcal{V}_{in}(2) = \{1, 4\}$, $\mathcal{V}_{out}(1) = \{2, 3, 5\}$, $\mathcal{V}_{out}(2) = \{1, 3, 4\}$.
Hence, $d_{in}(1) = 1$, $d_{in}(2) = 2$, $d_{out}(1) = 3$, $d_{out}(2) = 3$.

Graph properties

Practical implication

- In practice, *self–loops are always considered* since it is commonplace that each node can have information from its sensors/actuators.
- For an undirected graph, in-neighbours and out-neighbours of a node i coincide and they are simply denoted by the set $\mathcal{V}(i)$, whose degree is $d(i) = |\mathcal{V}(i)|$.

Graph representation

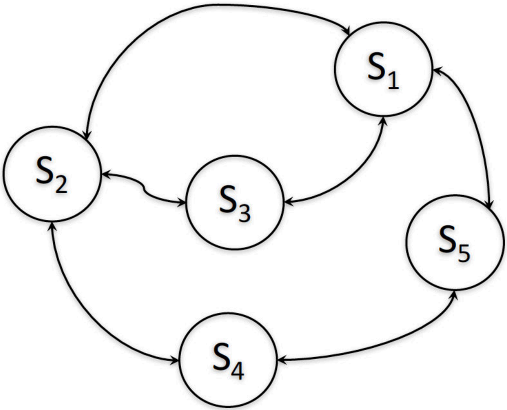

Figure: $\mathcal{V}(1) = \{2, 3, 5\}$, $\mathcal{V}(2) = \{1, 3, 4\}$. Hence, $d(1) = 3$, $d(2) = 3$.

Graph properties

Graph rooted, connected and complete

Definition (Rooted)

A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is *rooted* if there exists a node $k \in \mathcal{N}$ such that for any other node $j \in \mathcal{N}$ there is a unique path from k to j.

Definition (Strongly connected)

A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is *strongly connected* if there exists a path from any node to any other node.

Definition (Complete)

A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is *complete* if $(i, j) \in \mathcal{E}$, $\forall i, j \in \mathcal{N}$.

Graph representation

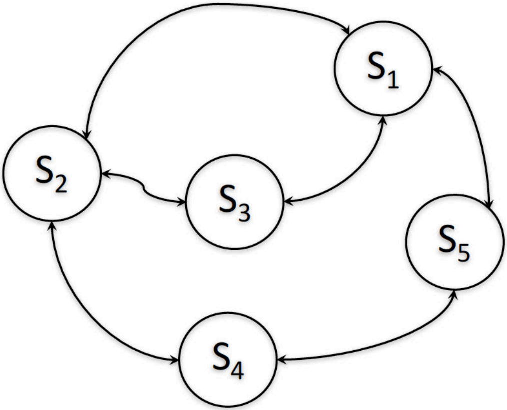

Figure: This graph is *rooted* and *strongly connected*. It is not *complete*, but it becomes complete if $\mathcal{E}^{new} = \mathcal{E} \cup \{(1,4),(2,5),(3,4),(3,5)\}$.

Graph representation

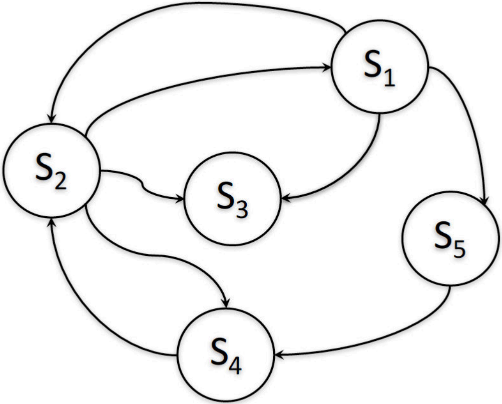

Figure: This graph is *rooted* in all the nodes apart from S_3. It is *not strongly connected*. It is *not complete*.

Graph properties

Diameter

Definition (Diameter)

The *diameter* of a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is defined as the length of the longest among all shortest paths connecting any two nodes in a strongly connected graph.

Graph representation

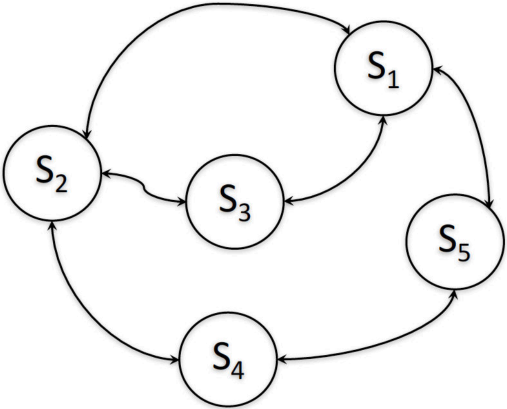

Figure: The *diameter* of this graph is 2.

Graph properties

Matrices

Definition (Adjacency matrix)

The *adjacency matrix* $A \in \mathbb{R}^{n \times n}$ of a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is a matrix having $a_{ij} = 1$ if $(j, i) \in \mathcal{E}$ and $i \neq j$, and $a_{ij} = 0$.

Definition (Degree matrix)

The *degree matrix* D of an *undirected graph* $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is defined as $D = \text{diag}(d(1), d(2), \ldots, d(n))$.

Graph representation

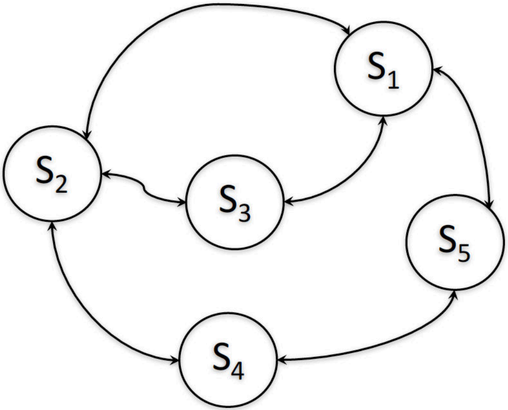

Figure: The *adjacency matrix* of this graph is $A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$.

Graph representation

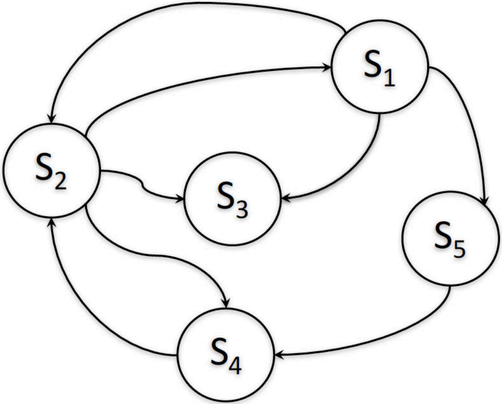

Figure: The *adjacency matrix* of this graph is $A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$.

Graph representation

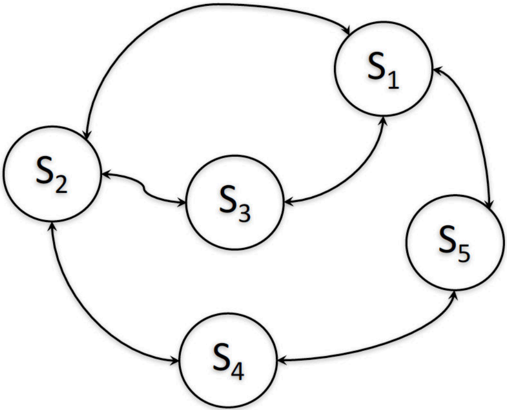

Figure: The *degree matrix* of this graph is $D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$.

Networks and graphs

Models

- Let us suppose to have a the very simple case of n sensors measuring *the same* phenomenon but from different locations.
- Each measured value is a variable $x_i(t)$ and the distributed system *state* can then be represented by $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T \in \mathbb{R}^n$.
- We saw that each sensor *local estimation process* matches the *global estimation process* if an *agreement* is reached, i.e. *all the estimation processes give the same result*.
- A straightforward solution is the mean and the design of the matrix Q: since the *consensus protocol* depends on the *topology*, can we use directly the adjacency matrix to design the protocol?

Networks and graphs

Models

- Assuming a *complete graph*, we can have for example the following *update equation*:

$$x(t+1) = \frac{1}{n}(I_n + A)x(t) = Qx(t).$$

- One important property of the matrix Q is that its rows and its columns, in this case, sums up to one, i.e. a *double stochastic* matrix.
- We will see in the following how the protocol can be inferred from the graph properties even if the matrix Q changes in time, i.e., the communication topology is time varying.

Graph representation

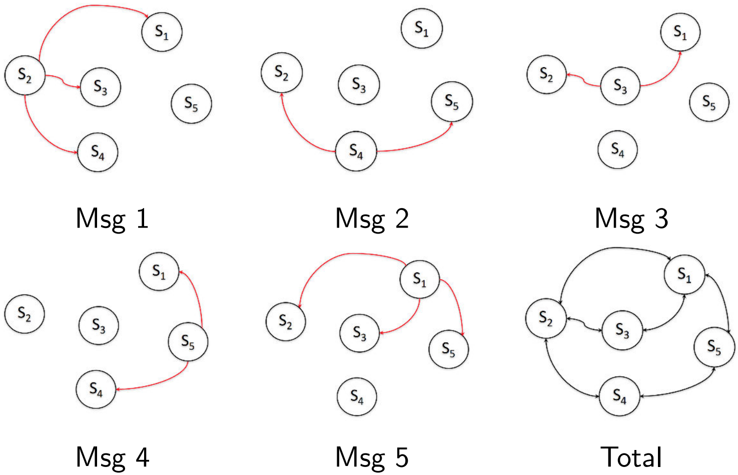

Figure: Time varying combination of graphs.

Graph properties

Matrices

Definition (Laplacian matrix)

The *Laplacian matrix* L of an *undirected graph* $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is defined as $L = D - A$.

Graph representation

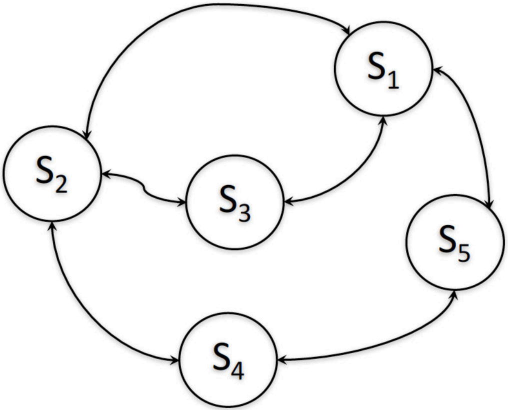

Figure: The *adjacency matrix* of this graph is $A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$.

Graph representation

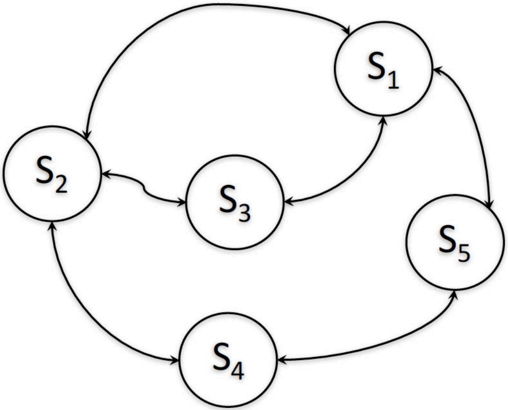

Figure: The *degree matrix* of this graph is $D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$.

Graph representation

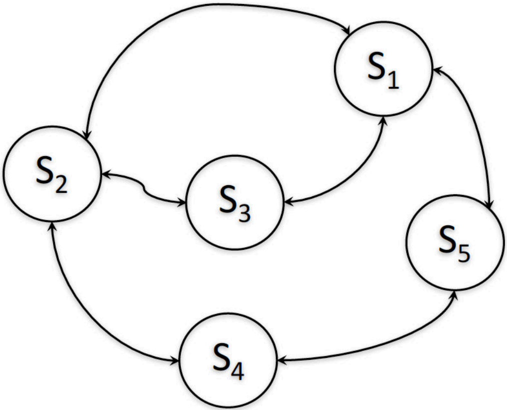

Figure: The *Laplacian matrix* of this graph is $L = \begin{bmatrix} 3 & -1 & -1 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ 0 & -1 & 0 & 2 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}$.

Graph properties

Properties of the Laplacian matrix

- The Laplacian matrix is *positive semidefinite*.
- The Laplacian matrix verifies $L\mathbf{1} = 0$.

Definition (Positive definite)

A matrix $M \in \mathbb{R}^{n\times n}$ is *positive definite* if and only if $x^T M x > 0$, $\forall x \in \mathbb{R}^n$ and $x \neq 0$. Accordingly, is *negative definite*, *positive semidefinite* and *negative semidefinite* if respectively $x^T M x < 0$, $x^T M x \geq 0$ and $x^T M x \leq 0$.

Outline

Stochastic matrices

Definition (Stochastic matrix)

A *stochastic matrix* is a matrix $Q \in \mathbb{R}^{n \times n}$ if and only if $q_{ij} \geq 0$ and $\sum_{j=0}^{n} q_{ij} = 1$, $\forall i$.

Definition (Doubly-stochastic matrix)

A stochastic matrix Q is *doubly-stochastic* if also $\sum_{i=0}^{n} q_{ij} = 1$, $\forall j$.

Clearly, if Q is a stochastic *symmetric* matrix, i.e., $Q = Q^T$, then it is also doubly-stochastic.

Stochastic matrices

Circulant matrices

Definition (Circulant matrix)

A *circulant matrix* is a matrix having the sequence of the rows with elements shifted by one position.

$$C = \begin{bmatrix} c_1 & c_2 & c_3 & \dots & c_n \\ c_n & c_1 & c_2 & \dots & c_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \\ c_2 & c_3 & c_4 & \dots & c_1 \end{bmatrix}$$

Stochastic matrices

Spectral radius

Definition (Spectral radius)

The *spectral radius* of a matrix $M \in \mathbb{R}^{n \times n}$ is defined as $\rho(M) \triangleq \max_i |\lambda_i|$, $i = 1, \ldots, n$ and λ_i is the i–th eigenvalue of M.

Definition (Essential spectral radius)

The *essential spectral radius* $\rho_2(M)$ of a matrix $M \in \mathbb{R}^{n \times n}$ is defined as the *second largest eigenvalue* of M.

Stochastic matrices

Properties

If Q is a stochastic matrix, then:

- If λ_i is an eigenvalue of Q, then $|\lambda_i| \leq 1$, $\forall i$.
- Moreover, $Q\mathbf{1} = \mathbf{1}$. Notice that this condition ensures that any vector with constant entries, i.e. $x(t) = \alpha\mathbf{1}$, is a *fixed point* for the dynamic $x(t+1) = Qx(t)$.
- If the eigenvalues are ordered, it follows that $\rho(Q) = |\lambda_1| = 1$, while $\rho_2(Q) = |\lambda_2| \leq 1$.

Stochastic matrices

Graph relation

- Using the previously introduced definitions, we say that the graph $\mathcal{G}_q = (\mathcal{N}, \mathcal{E}_q)$ is associated to the stochastic matrix $Q \in \mathbb{R}^{n \times n}$ if $\mathcal{N} = \{1, 2, \ldots, n\}$ and $\mathcal{E}_q = \{(j, i) | q_{ij} > 0\}$.

Outline

Linear consensus

- Let us consider the following general *update equation*:

$$x(t+1) = Q(t)x(t), \text{ with } Q(t) \in \mathbb{R}^{n\times n} \text{ and } x(t) \in \mathbb{R}^n.$$

- We will assume that $Q(t)$ is a *stochastic matrix*.
- It is easy to see that the *update equation* for node i is given by:

$$x_i(t+1) = \sum_{j=1}^{n} q_{ij}(t)x_j(t) = x_i(t) + \sum_{j=1}^{n} q_{ij}(t)(x_j(t) - x_i(t)),$$

i.e., it is associated to a graph with all the self-loops.

Linear consensus

- Notice that this formulation

$$x_i(t+1) = x_i(t) + \sum_{j=1}^{n} q_{ij}(t)(x_j(t) - x_i(t)),$$

expresses that this updating rule is *distributed*: each node uses its own information plus the information it can receive. Nonetheless, we will give now some properties that ensures the *global* convergence.

Linear consensus

Linear consensus problem

Definition (Linear consensus problem)

With respect to the previous update equation, the matrix $Q(t)$ solves the *consensus problem* if

$$\lim_{t\to+\infty} x_i(t) = \alpha, \ \ \forall i,$$

or, in matrix form, assuming $x(t) = [x_1(t), x_2(t), \dots, x_n(t)]^T$,

$$\lim_{t\to+\infty} x(t) = \alpha\mathbf{1}.$$

Linear consensus

Linear consensus problem

Notice that since

$$\begin{aligned} x(t+1) &= Q(t)x(t) = Q(t)Q(t-1)x(t-1) = \\ &= Q(t)Q(t-1)\ldots Q(0)x(0) = \Phi(t)x(0), \end{aligned}$$

where $\Phi(t)$ is the *t-step transition matrix*, we have that

$$\lim_{t \to +\infty} x(t) = \lim_{t \to +\infty} \Phi(t)x(0) = \alpha \mathbf{1}.$$

Linear consensus

Linear average consensus problem

Definition (Linear average consensus problem)

With respect to the previous update equation, the matrix $Q(t)$ solves the *average consensus problem* if

$$\lim_{t \to +\infty} x_i(t) = \frac{1}{n} \sum_{i=1}^{n} x_i(0), \ \ \forall i,$$

or, in matrix form

$$\lim_{t \to +\infty} x(t) = \left(\frac{1}{n} \mathbf{1}^T x(0) \right) \mathbf{1} = \frac{1}{n} \mathbf{1} \mathbf{1}^T x(0).$$

Linear consensus

Convergence theorems

The next theorems describe some *sufficient conditions* which guarantee deterministic consensus, i.e., when $Q = Q(t)$, $\forall t$.

Theorem (Deterministic convergence)

If the graph $\mathcal{G}_q = (\mathcal{N}, \mathcal{E}_q)$ contains all the self–loops and it is also rooted, then

$$\lim_{t \to +\infty} Q^t = \mathbf{1}\beta^T,$$

where $\beta \in \mathbb{R}^n$ is the left eigenvector of Q for $\lambda_1 = 1$. Moreover, we have

$$\beta_j \geq 0 \text{ and } \mathbf{1}^T \beta = 1.$$

Notice that being the *left eigenvector* of Q for $\lambda_1 = 1$, means $\beta^T Q = \beta^T$, which implies $\beta^T x(t+1) = \beta^T x(t)$ at each step.

Linear consensus

Convergence theorems (contd..)

Theorem (Average consensus)

If $\mathcal{G}_q = (\mathcal{N}, \mathcal{E}_q)$ contains all the self–loops and it is also strongly connected, *then* $\beta_j > 0$, $\forall j$.

If Q *is doubly-stochastic, then* $\mathcal{G}_q = (\mathcal{N}, \mathcal{E}_q)$ *is* strongly connected *and*

$$\beta = \frac{1}{n}\mathbf{1},$$

i.e., it solves the average consensus problem.

Linear consensus

Convergence theorems (contd..)

In other words, since

$$\lim_{t\to+\infty} x(t) = \lim_{t\to+\infty} \Phi(t)x(0),$$

the *t-step transition matrix* converges to

$$\lim_{t\to+\infty} \Phi(t) = \frac{1}{n}\mathbf{1}\mathbf{1}^T = J_a.$$

Linear consensus

Convergence theorems (contd..)

An alternative formulation of the convergence is given by the following Theorem.

Theorem (**Average consensus bis)**

The previous condition hold if and only if*: a)* $\mathbf{1}^T Q = \mathbf{1}^T$*; b)* $Q\mathbf{1} = \mathbf{1}$*; c)* $\rho\left(Q - J_a\right) < 1$.

(See Lin Xiao, Stephen Boyd, "Fast linear iterations for distributed averaging", In Systems and Control Letters, v. 53, pp. 65–78, 2004)
Notice that if the elements of Q are all non negative, these are the conditions of the previous Theorem formulation, i.e., Q doubly stochastic and the graph containing all the self-loops.

Linear consensus

Corner cases

- Existence of self-loops is *not necessary* to reach consensus. Indeed, taking Q with only one column equal to $\mathbf{1}$ reaches a consensus.
- However, the fact of being rooted *without self-loops* does not guarantee consensus. For example, taking Q with the anti-diagonal equal to $\mathbf{1}$ defines a periodic dynamic.

Linear consensus

Theorem (Rate of convergence)

The rate of convergence of all the cases in the previous theorem is exponential *and its rate is given by* $\rho_2(Q)$.

Outline

Continuous time consensus

Before going into the details of this case, let us consider a special class of matrices.

Definition

A *Metzler matrix* is a matrix whose off-diagonal elements are nonnegative and the row-sum is null.

It then follows that if M is Metzler, $M\mathbf{1} = 0$.
Notice that the *negative* graph Laplacian $-L$ is a *Metzler matrix*.

Continuous time consensus

- Consider a continuous time system

$$\dot{x}(t) = M(t)x(t).$$

- If $M(t)$ is a *Metzler matrix* than the network achieves a consensus under general connectivity properties of the associated graph.
- Indeed, $\forall x(t) = c\mathbf{1}$, we have $\dot{x} = 0$, i.e., an *equilibrium point*.
- Is this equilibrium point asymptotically stable?

Continuous time consensus

Stability proof

- To prove stability, let's compute the *agreement error*:

$$e_i(t) = x_{i+1}(t) - x_1(t), \forall i = 1, \ldots, n-1,$$

and $e = [e_1, e_2, \ldots, e_{n-1}]^T$.

- Therefore, assuming for simplicity $M(t) = -L$, one gets:

$$\dot{e} = -\tilde{L}e,$$

where

$$\tilde{L} = \begin{bmatrix} l_{22} - l_{12} & \cdots & l_{2n} - l_{1n} \\ \vdots & \ddots & \vdots \\ l_{n2} - l_{12} & \cdots & l_{nn} - l_{1n} \end{bmatrix}.$$

Continuous time consensus

Stability proof

- Since the eigenvalues of L are $\lambda_1, \lambda_2, \dots, \lambda_n$, with $\lambda_1 = 0$, it is possible to show that the eigenvalues of $\tilde{L}$ are $\lambda_2, \dots, \lambda_n$.
- Then we can make use of the following theorems on the Laplacian matrix...

Continuous time consensus

Stability proof

Theorem

For any eigenvalue λ_i, $i = 1, \ldots, n$, of the Laplacian matrix L associated to $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ either $\lambda_i = 0$ or Re$(\lambda_i) > 0$.

Indeed, L is a *positive semidefinite* matrix.

Theorem

The (di)graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is rooted if and only if *Rank$(L) = n - 1$, with L being the Laplacian of $\mathcal{G}$.*

Continuous time consensus

Stability proof

- It then follows that the continuous system reaches a consensus solution *asymptotically* *if and only if* the (di)graph $\mathcal{G}$ is rooted.

Continuous time consensus

Extensions

There is a lot of literature of the last ten years that extends this basic idea to more complicated and more realistic scenarios.

- *Multidimensionality*: The consensus for continuous time systems is indeed used for multidimensional systems, even though specific and difficult technical solutions are needed.
- *Delays*: The consensus protocols are ideal, since they don't explicitly consider the presence of the communication delays.
- *Noise*: Optimal consensus protocols have been also designed, which are able to minimise the presence of the noise in the sent data.
- *Nonlinear*: The linear consensus have been also extended to nonlinear systems, like robotic vehicles.

Outline

Design of consensus algorithms

- In the previous sections we have considered the *analysis* of consensus algorithms.
- From an engineering point of view, it is also important to understand how to *design* such algorithms, aka *consensus protocols*.
- The design can be synthesised in what follows: *Given the communication graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ of a network with n nodes, find a matrix $Q(t)$ compatible with $\mathcal{G}$ that achieve (average) consensus*.
- In practice this problem amounts to find the values of the elements of the matrix $q_{ij} > 0$ corresponding to an edge (j, i), i.e., j sends to i.

Design of consensus algorithms

Local vs optimal design

There are basically two approaches to the design of the consensus algorithms:

- *Global optimal design*: This approach tends to find an optimal solution to some *global performance index*. In this case, a *centralised* solution is quite often necessary, which is feasible for networks with a *limited number* of nodes and *fixed topology*.
- *Local design*: This approach designs the consensus algorithm using only *local information* and independently from other nodes. Of course, optimality is *not guaranteed*. This is the approach we adopt in this course.

Design of consensus algorithms

Continuous case

- For the *local* design of the consensus protocol in the continuous time case, it is easy to see that in the case of simple integrator dynamics

$$\dot{x}_i(t) = \alpha_i u_i(t), \alpha_i \in \mathbb{R},$$

where $u_i(t)$ is the input, a simple *consensus protocol* like the following

$$u_i(t) = \frac{\beta}{\alpha_i} \sum_{j=1}^{n} l_{ij}(x_j(t) - x_i(t)), \beta \in \mathbb{R}$$

where l_{ij} are the elements of the Laplacian matrix L, reaches asymptotically a consensus.

Design of consensus algorithms

Discrete case for consensus

- A possible strategy for the *local* design of a consensus protocol for a discrete time system:

$$x_i(t+1) = x_i(t) + \alpha_i u_i(t), \alpha_i \in \mathbb{R},$$

is given by the input

$$u_i(t) = \frac{1}{\alpha_i} \sum_{j=1}^{n} \frac{1}{1+d_{in}(i)} (x_j(t) - x_i(t)), \forall (j,i) \in \mathcal{E}.$$

Design of consensus algorithms

Discrete case for consensus

- Substituting $u_i(t)$, one has

$$
\begin{aligned}
x_i(t+1) = & x_i(t) + \alpha_i u_i(t) = x_i(t) + \sum_{j=1}^{n} \frac{1}{1 + d_{in}(i)} (x_j(t) - x_i(t)) \\
& = x_i(t) + \sum_{j=1}^{n} q_{ij} (x_j(t) - x_i(t)).
\end{aligned}
$$

- In matrix form, defining $Q = (q_{ij})$, i.e., $x(t+1) = Qx(t)$, it is easy to verify how Q is a *stochastic matrix*, i.e. it reaches consensus, *but not* average consensus for general (di)graphs.

Design of consensus algorithms

Discrete case for consensus

Consensus is reached *asymptotically*.

We will now focus on *average consensus approaches*.

What about the *rate of convergence toward the average consensus?*

Design of consensus algorithms

Discrete time consensus with convergence requirements

Let us define the *average consensus equilibrium point* $\overline{x} = \frac{1}{n}\mathbf{1}^T x(0) = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$.
Let us define the *asymptotic convergence factor* as

$$r_a = \sup_{x(0)\neq\overline{x}} \lim_{t\rightarrow+\infty} \left(\frac{\|x(t)-\overline{x}\|_2}{\|x(0)-\overline{x}\|_2}\right)^{\frac{1}{t}},$$

which expresses how *fast* the average consensus is reached.
Similarly, let us define the *per-step convergence factor* as

$$r_s = \sup_{x(t)\neq\overline{x}} \frac{\|x(t+1)-\overline{x}\|_2}{\|x(t)-\overline{x}\|_2},$$

which expresses how *fast* is the contraction *per step* towards the average consensus.

Design of consensus algorithms

Discrete time consensus with convergence requirements

In the *average consensus*, the *t-step transition matrix* converges to

$$\lim_{t \to +\infty} \Phi(t) = \frac{1}{n}\mathbf{1}\mathbf{1}^T = J_a,$$

if and only if: a) $\mathbf{1}^T Q = \mathbf{1}^T$; b) $Q\mathbf{1} = \mathbf{1}$; c) $\rho\left(Q - J_a\right) < 1$, i.e. *Average consensus bis* Theorem.

In such a case, we have that the *asymptotic convergence factor* r_a is given by

$$r_a = \rho\left(Q - J_a\right).$$

Similarly, the *per-step convergence factor* r_s is given by

$$r_s = \|Q - J_a\|_2.$$

Design of consensus algorithms

A closer look to the per-step convergence

Let us understand why the *per-step convergence factor* is relevant.
We first notice that

$$x(t+1) - J_a x(0) = Q(x(t) - J_a x(0)) = (Q - J_a)(x(t) - J_a x(0)).$$

(Indeed, since $QJ_a x(0) = J_a x(0)$ and $J_a x(t) - J_a x(0) = 0$, $\forall t$).
Using the Euclidean norms, we have

$$\begin{aligned}\|x(t+1) - J_a x(0)\|_2 &= \|(Q - J_a)(x(t) - J_a x(0))\|_2 \\ &\leq \|Q - J_a\|_2 \|x(t) - J_a x(0)\|_2.\end{aligned}$$

Therefore, if $r_s = \|Q - J_a\|_2 < 1$, at each step $x(t)$ tends towards the average. In other words, r_s is a measure of the *worst case asymptotic rate of convergence towards the consensus*.

Design of consensus algorithms

Discrete time consensus with convergence requirements

One possible problem that can be tackled is the choice of the Q elements that maximises the *asymptotic* or the *per-step* contraction, i.e. that let the system to converge *as fast as possible*.
It can be shown that if $Q = Q^T$, the solution in both cases is given by

$$\min_{Q} \rho\left(Q - J_a\right) \;\; s.t. \;\; Q = Q^T, Q\mathbf{1} = \mathbf{1}.$$

Design of consensus algorithms

Discrete case for average consensus in undirected graphs

- A possible strategy for the *local* design of a consensus protocol in *rooted undirected graphs* reaching *average consensus* for a discrete time system:

$$x_i(t+1) = x_i(t) + \alpha_i u_i(t), \alpha_i \in \mathbb{R},$$

  is to solve the previously defined optimal problem assuming elements in Q that are all equal.
- The strategy is to set all edge weights to ε and the self-weights to satisfy $Q\mathbf{1} = \mathbf{1}$.

Design of consensus algorithms

Discrete case for average consensus in undirected graphs

- Therefore, one has

$$q_{ij} = \begin{cases} \varepsilon & \text{if } (j,i) \in \mathcal{E} \text{ and } i \neq j, \\ 1 - \varepsilon d(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

- Notice that such an approach corresponds to select

$$Q = I - \varepsilon L,$$

where L is the associated Laplacian matrix.

- Since L is *positive semidefinite*, to ensure that $\rho\left(Q - J_a\right) < 1$, we have $\varepsilon > 0$.

Design of consensus algorithms

Discrete case for average consensus in undirected graphs

- It can be shown (Lin Xiao, Stephen Boyd, "Fast linear iterations for distributed averaging", In Systems and Control Letters, v. 53, pp. 65–78, 2004) that the i–th largest eigenvalue of Q is given by

$$\lambda_i(Q) = 1 - \varepsilon \lambda_{n-i+1}(L),$$

which yields

$$\rho\left(Q - J_a\right) = \max\{\lambda_2(Q), -\lambda_n(Q)\} = \max\{1-\varepsilon\lambda_{n-1}(L), \varepsilon\lambda_1(L)-1\}.$$

- As a consequence, we have that

$$0 < \varepsilon < \frac{2}{\lambda_1(L)}.$$

Design of consensus algorithms

Discrete case for average consensus in undirected graphs

- In order to have positive diagonal terms of Q, usually a limit is given by
$$\varepsilon < \frac{1}{\max_i d(i)},$$
which ensures that Q is a stochastic matrix.
- A common usual choice is to use $\varepsilon = \frac{1}{1+\max_i d(i)}$, aka *max degree weights*.
- *WARNING!*: Notice that to compute the value of ε at least a bound on the degree of each node is needed, hence the algorithm is *partially local*.

Design of consensus algorithms

Discrete case for average consensus in undirected graphs

- With the presented choice of the matrix Q, one has for the input

$$u_i(t) = \frac{1}{\alpha_i} \sum_{j=1}^{n} \varepsilon(x_j(t) - x_i(t)), \forall (j, i) \in \mathcal{E}.$$

Design of consensus algorithms

Discrete case for average consensus in undirected graphs

- Since the graph is *rooted* and *undirected* it will be also *strongly connected*. This implies that: a) Q is a doubly-stochastic matrix and b) the *average consensus* can be reached since the hypotheses of the Theorem on the deterministic convergence are verified.

Design of consensus algorithms

Discrete case for average consensus in undirected graphs

- Since $Q = I - \varepsilon L$, it can be considered as a *discrete version of a continuous time* average consensus solution.
- Indeed, let us recall the dynamic for the continuous case $\dot{x}(t) = Mx(t) = -Lx(t)$.
- The continuous time dynamic can be discretised with *sampling time* ε, yielding to

$$x(t+1) = e^{-\varepsilon L}x(t) = \sum_{i=0}^{+\infty} \frac{(-L)^i \varepsilon^i}{i!} x(t) = (I - \varepsilon L + \mathcal{O}(\varepsilon))x(t),$$

which, by neglecting $\mathcal{O}(\varepsilon)$, it is the adopted discrete time consensus protocol!

Design of consensus algorithms

Discrete case for average consensus in undirected graphs

- A different strategy for the same problem is instead given by

$$q_{ij} = \begin{cases} \frac{1}{\max(d(i),d(j))+1} & \text{if } (j,i) \in \mathcal{E} \text{ and } i \neq j, \\ 1 - \sum_{j=1,i\neq j}^{n} q_{ij} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

- Notice that $\forall i$:

$$q_{ii} = 1 - \sum_{j=1,i\neq j}^{n} q_{ij} \geq 1 - \sum_{j=1,i\neq j}^{n} \frac{1}{d(i)+1} = 1 - \frac{d(i)}{d(i)+1} > 0,$$

which ensures that the matrix Q is again doubly-stochastic.

Design of consensus algorithms

Discrete case for average consensus in undirected graphs

- The solution here proposed adopts the *Metropolis-Hastings* weights, which is a *local solution* and ensures *faster* convergence with respect to the Laplacian based solution.
- The case of digraph is also considered in literature, but not considered in these notes.

Outline

Linear Consensus Theory

The *topology* of the network can be described using *graph theory* tools.
An important role in the definition of the stability of a *linear consensus algorithm* is played by the graph *Laplacian*.
A *stochastic matrix* ensures the convergence on a *consensus equilibrium*, while a *doubly stochastic* matrix ensures convergence towards the *average consensus equilibrium*.
Consensus algorithms can be defined for continuous (i.e., *Metzler matrices*) and discrete dynamics.
It is possible to design an effective *consensus protocol* using optimisation tools.