



UNIVERSITÀ DEGLI STUDI
DI CAGLIARI



Introduction to Open Multi-Agent Systems - Part I

Mauro Franceschelli*

*Department of Electrical and Electronic Engineering, University of Cagliari, Italy

SIDRA Ph.D. Summer School • July 13-15, 2026 • Bertinoro, Italy

2026 SIDRA PhD Summer School: Introduction to Open Multi-Agent Systems (OMAS)

Day 1 – 3h – Morning – Franceschelli

- Introduction to agents and MAS
- Graphs as network models and Consensus in MAS
- Why openness changes everything

Day 1 – 3h – Afternoon – Franceschelli

- Challenges in OMAS
- Open Average Consensus
- Stability in contractive OMAS

Day 2 – 3h – Morning – Deplano

- Operator theory for open systems
- Gradient descent

Day 2 – 3h – Afternoon – Deplano

- Proximal gradient
- Distributed Gradient Descent (DGD)
- DGD in OMAS

Day 3 – 3h – Morning – Deplano

- Distributed Peaceman–Rachford splitting in OMAS
- Open ADMM
- Open FedPRS

Outline

- 1 Introduction to Agents and MAS: Intuition and examples
- 2 Graphs as models of network topology
- 3 Consensus in linear MAS
- 4 Why openness changes everything

Outline

- 1 Introduction to Agents and MAS: Intuition and examples
- 2 Graphs as models of network topology
- 3 Consensus in linear MAS
- 4 Why openness changes everything

What is an agent?

General definition:

Agents are **systems** that inhabit some complex dynamic environment, sense and act **autonomously** in this environment, and by doing so realize a set of **goals** or tasks for which they are designed.

In Computer Science:

A **software agent** is (historically) a computer program that acts on behalf of a user or other program, implying authority to decide which actions are appropriate.

⇒ The term "Agent" comes from Latin "agere" which means "to act/to do"

What is an agent?

General definition:

Agents are **systems** that inhabit some complex dynamic environment, sense and act **autonomously** in this environment, and by doing so realize a set of **goals** or tasks for which they are designed.

In Computer Science:

A **software agent** is (historically) a computer program that acts on behalf of a user or other program, implying authority to decide which actions are appropriate.

⇒ The term "Agent" comes from Latin "agere" which means "to act/to do"

What is an agent?

General definition:

Agents are **systems** that inhabit some complex dynamic environment, sense and act **autonomously** in this environment, and by doing so realize a set of **goals** or tasks for which they are designed.

In Computer Science:

A **software agent** is (historically) a computer program that acts on behalf of a user or other program, implying authority to decide which actions are appropriate.

⇒ The term "Agent" comes from Latin "agere" which means "to act/to do"

What is an agent? Some examples

In **Systems and Control theory**:

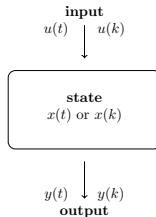
Agents are "Autonomous Systems" which are represented as a **dynamical system** with inputs, a state, and outputs.

The formal mathematical models of dynamical systems, and thus autonomous agents, are embodied by robots, vehicles, energy systems, biological organisms, processing units etc.



Drone, Power generator, Bird

Model of an agent: dynamical system



Formal models:

- Continuous time (CT):

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= h(x(t)), \quad t \in \mathbb{R}. \end{aligned}$$
- Discrete time (DT):

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) \\ y(k) &= h(x(k)), \quad k \in \mathbb{N}. \end{aligned}$$

Large Language Models as Dynamical Systems

Even modern AI systems can be viewed as discrete-time dynamical systems

$$x_{t+1} = f_{\theta}(x_t, u_t, \xi_t)$$

Dynamical System	Large Language Model
State x_t	Context / hidden representation at step t
Update law f_{θ}	Transformer map with parameters θ
Input u_t	User prompt or external instruction
Noise ξ_t	Sampling randomness / decoding choice
y_t	Next token generated at step t
Trajectory	Generated sequence of tokens, represented numerically.

$u_t = \text{"What is a dynamical system?"}, \quad x_t = \text{"A dynamical system is"}, \quad y_t = \text{" a"}$

$$x_{t+1} = \text{concat}(x_t, y_t) = \text{"A dynamical system is a"}$$

Key idea

A trained LLM is a static nonlinear map. When its generated output is fed back as the next input, it becomes a discrete-time, high-dimensional, stochastic dynamical system.

What is a multi-agent system?

General definition:

A multi-agent system consists of a **large** number of **coupled** agents, forming a **network** that work together to achieve a common goal that is **beyond** the individual capabilities or knowledge of each agent.

In systems and control theory:

A multi-agent system is a **network** of coupled **dynamical systems** whose state and output dynamics depend on both the agent's dynamics and on the pattern of couplings encoded in the network.

⇒ Usually, each agent has the same dynamics (it is of the same type), their couplings have the same structure and the network topology is **complex**.

What is a multi-agent system?

General definition:

A multi-agent system consists of a **large** number of **coupled** agents, forming a **network** that work together to achieve a common goal that is **beyond** the individual capabilities or knowledge of each agent.

In systems and control theory:

A multi-agent system is a **network** of coupled **dynamical systems** whose state and output dynamics depend on both the agent's dynamics and on the pattern of couplings encoded in the network.

⇒ Usually, each agent has the same dynamics (it is of the same type), their couplings have the same structure and the network topology is **complex**.

What is a multi-agent system?

General definition:

A multi-agent system consists of a **large** number of **coupled** agents, forming a **network** that work together to achieve a common goal that is **beyond** the individual capabilities or knowledge of each agent.

In systems and control theory:

A multi-agent system is a **network** of coupled **dynamical systems** whose state and output dynamics depend on both the agent's dynamics and on the pattern of couplings encoded in the network.

⇒ Usually, each agent has the same dynamics (it is of the same type), their couplings have the same structure and the network topology is **complex**.

Multi-agent systems in nature



School of fishes



Flock of birds



Human crowd



Multi-robot systems



100 drone swarms, 2015



10000 drone swarm, 2025



Multi-robot warehouse systems



Consensus as emergent behavior

The most fundamental emergent behavior to coordinate dynamical systems is that of "synchronization", "agreement" or "consensus" among state variables of different agents

- Synchronization experiment with coupled metronomes 
- Velocity and direction alignment in flocks
- Synchronization of power generators in the power transmission network in the electric grid

Consensus as theoretical and Engineering problem

Definition: Consensus problem

The consensus problem consists in the design of a local interaction protocols such that the state of each agent is steered toward the same value, i.e.,

$$\lim_{k \rightarrow \infty} \|x_i(k) - x_j(k)\| = 0, \quad i, j = 1, \dots, n, \forall x(0) \in \mathbb{R}^n$$

Additionally, we may ask that such **consensus value** is constant

$$\lim_{k \rightarrow \infty} x_i(k) = c \quad i = 1, \dots, n, \forall x(0) \in \mathbb{R}^n$$

When the constant c is equal to the average of the initial states

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

we call it **consensus on the average**.

Optimization

An optimization problem formalizes the search for the **best feasible solution**:

$$\begin{array}{ll} \min_x & f(x) \\ \text{s.t.} & x \in \mathcal{X}. \end{array}$$

where:

- $x \in \mathbb{R}^p$ is the decision variable;
- $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is the objective function;
- $\mathcal{X} \subseteq \mathbb{R}^p$ is the admissible decision space (constraints).

Objective: find the best feasible solution x^* that minimizes the objective function f within $\mathcal{X}$.

Distributed Consensus based Optimization

In a multi-agent system, each agent holds:

- $x_i \in \mathbb{R}^p$ a local decision variable;
- $f_i : \mathbb{R}^p \rightarrow \mathbb{R}$ a local objective function;
- $\mathcal{X}_i \subseteq \mathbb{R}^p$ a local admissible decision space (constraints);

so that the optimization problem can be solved with **distributed** algorithms:

$$\begin{aligned}
 & \min_{\{x_i\}_{i \in \mathcal{V}}} \sum_{i \in \mathcal{V}} f_i(x_i) \\
 & \text{s.t.} \quad x_i \in \mathcal{X}_i, \quad \forall i \in \mathcal{V} \text{ (local constraints),} \\
 & \quad \quad x_i = x_j, \quad \forall (i, j) \in \mathcal{E} \text{ (consensus constraints).}
 \end{aligned}$$

Distributed estimation

Scenario

A network of sensors observes an unknown quantity

$$\theta \in \mathbb{R}^p.$$

Each agent i has only a local noisy measurement

$$y_i = H_i \theta + v_i,$$

where H_i is local and v_i is measurement noise.

Goal: Estimate θ without sending all data to a fusion center.

Distributed estimation problem

A standard least-squares formulation is

$$\min_{\theta} \sum_{i=1}^N \frac{1}{2} \|y_i - H_i \theta\|^2.$$

Introduce local estimates θ_i :

$$\min_{\theta_1, \dots, \theta_N} \sum_{i=1}^N \frac{1}{2} \|y_i - H_i \theta_i\|^2$$

$$\text{s.t.} \quad \theta_i = \theta_j, \quad (i, j) \in \mathcal{E}.$$

Key idea: local sensing + neighbor communication + consensus constraints produce a network-wide estimate.

Federated learning as distributed optimization

Private local datasets

Each agent owns a private dataset

$$\mathcal{D}_i = \{(\xi_{il}, y_{il})\}_{\ell=1}^{m_i}.$$

The data are not shared with the other agents.

Local empirical risk

For a neural network h_w , agent i defines a Loss function (Objective function)

$$F_i(w) = \frac{1}{m_i} \sum_{\ell=1}^{m_i} \ell(h_w(\xi_{il}), y_{il}).$$

Global learning objective

Train one common model:

$$\min_w \sum_{i=1}^N F_i(w).$$

The variable w contains all neural-network parameters (weights).

Consensus over models (weights)

Each agent keeps a local model w_i :

$$\min_{w_1, \dots, w_N} \sum_{i=1}^N F_i(w_i)$$

$$\text{s.t.} \quad w_i = w_j, \quad (i, j) \in \mathcal{E}.$$

Formation control in multi-agent systems

Collective control objective

A group of mobile agents must move as a coherent formation:

$$p_i \in \mathbb{R}^d, \quad v_i \in \mathbb{R}^d.$$

Typical objectives:

- align velocities;
- maintain desired relative distances;
- avoid collisions;
- preserve group cohesion.

Consensus viewpoint

Simple velocity alignment is a consensus problem:

$$v_i = v_j, \quad \forall i, j.$$

A standard local feedback law is

$$\dot{v}_i = - \sum_{j \in \mathcal{N}_i} a_{ij} (v_i - v_j).$$

Each agent only compares its velocity with its neighbors.

Key idea: formation control combines consensus on velocities with constraints on relative positions.

Formation control as distributed optimization

Example of Network Objective function

Formation control can be framed as minimizing a collective objective function defined over edges in a graph:

$$\min_{\{p_i, v_i, i \in V\}} \underbrace{\sum_{(i,j) \in \mathcal{E}} \alpha \|v_i - v_j\|^2}_{\text{velocity alignment}} + \underbrace{\sum_{(i,j) \in \mathcal{E}} \beta (\|p_i - p_j\| - d_{ij})^2}_{\text{formation keeping}} + \underbrace{\sum_{(i,j) \in \mathcal{E}} \gamma \frac{1}{\|p_i - p_j\|^2 + \eta}}_{\text{collision avoidance}} = \min_{\{p_i, v_i, i \in V\}} \sum_{i \in V} \sum_{j \in \mathcal{N}_i} J_{ij}$$

Local implementation

Agent i uses only relative distance $p_i - p_j$ and relative velocity $v_i - v_j$ information.

<https://eater.net/boids>

Control interpretation

The local feedback control law has the form

$$u_i = -\nabla_{p_i, v_i} \sum_{j \in \mathcal{N}_i} J_{ij}.$$

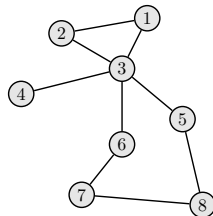
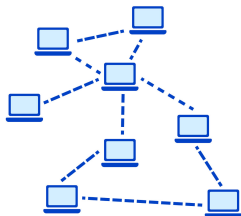
Interesting questions for Multi-Agent Systems

- Can we **explain** the emergent behavior of networks of coupled agents (convergence properties) by knowing the structure of their local interactions but not their particular topology?
- Can we **design** distributed control algorithms based on these ideas to build networks of mobile robots/sensors/energy systems/etc that exhibit the desired behavior emerging from simple local interactions?
- **How many agents should we use? What if this number changes?**

Model of the network: graph theory

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ encodes the patterns of couplings among the dynamical systems

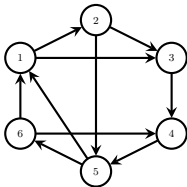
- $\mathcal{V} = \{1, \dots, n\}$ is the set of nodes representing agents;
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges connecting agents, there exists an edge (i, j) between agent i and agent j if they interact.
- Each agent $i \in \mathcal{V}$ has a set of neighbors $\mathcal{N}_i$, i.e., agents that share an edge with it. The number of neighbors is the agents' degree d_i .
- A path from node i to node j in a graph is a sequence of nodes starting from i and ending on j such that each consecutive node shares an edge with its predecessor node.
- $\mathcal{G}$ is called connected if there exists a path between any two nodes in the node set $\mathcal{V}$



Model of the network: graph theory

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is called directed if we add a direction to the information flow

- $(i, j) \in \mathcal{E}$ if there is a directed edge going from node i to node j
- Each agent $i \in \mathcal{V}$ has a set of incoming edges from in-neighbors $\mathcal{N}_{in,i}$, i.e., agents that send information to it.
- Each agent $i \in \mathcal{V}$ has a set of out-going edges from out-neighbors $\mathcal{N}_{out,i}$, i.e., agents that send information to it.



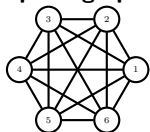
⇒ Directed graphs model directed sensing and monodirectional communications among agents.

Facts about the adjacency matrix II

- A directed path p_{ij} in a graph is a sequence of consecutive directed edges that start in node i and end in j .
- If there exist at least two nodes in the graph which do not have a directed path between each other, but all nodes are reachable from any other if the direction of the edges is not considered, then the graph is called weakly connected.
- A node i is **globally reachable** if there exist a directed path from every node $j \in V$ to node i .
- If all nodes of $\mathcal{G}$ are globally reachable then the graph is called **strongly connected**.

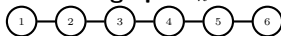
Canonical graph families: complete, line, ring, star

Complete graph K_n



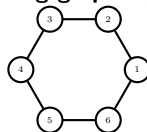
$$\{i, j\} \in \mathcal{E} \quad \forall i \neq j$$

Line graph P_n



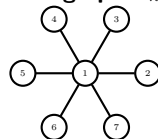
$$\mathcal{E} = \{\{i, i+1\}\}$$

Ring graph C_n



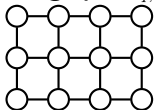
$$\mathcal{E} = \{\{i, i+1\}\} \cup \{\{n, 1\}\} \quad \mathcal{E} = \{\{1, i\} : i = 2, \dots, n\}$$

Star graph S_n



Grid, k -regular, and proximity graphs

Grid graph $G_{q,r}$

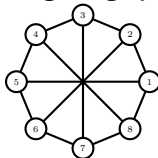


$$n = qr$$

Cartesian product:

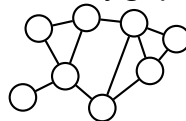
$$G_{q,r} = P_q \times P_r.$$

k -regular graph



$$d_i = k, \quad \forall i \in V.$$

Proximity graph

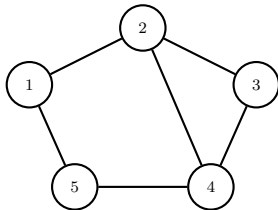


$$\{i, j\} \in \mathcal{E} \iff \|p_i - p_j\| \leq R.$$

Edges depend on geometry and the communication or sensing radius R .

The adjacency matrix of an undirected graph

The binary adjacency matrix encodes an **undirected graph** which represents the **network topology**.



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$a_{ij} = \begin{cases} 1, & \text{if } \{i, j\} \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases}$$

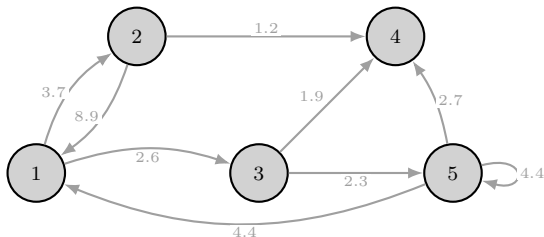
$$A = A^T, \quad a_{ii} = 0.$$

The adjacency matrix links **graph theory** with **non-negative matrix theory**.

⇒ For undirected graphs, adjacency matrices are symmetric and binary.

The adjacency matrix of a weighted directed graph

The adjacency matrix encodes a **weighted directed graph** which represents the **network topology**.



$$A = \begin{bmatrix} 0 & 3.7 & 2.6 & 0 & 0 \\ 8.9 & 0 & 0 & 1.2 & 0 \\ 0 & 0 & 0 & 1.9 & 2.3 \\ 0 & 0 & 0 & 0 & 0 \\ 4.4 & 0 & 0 & 2.7 & 4.4 \end{bmatrix}.$$

The adjacency links **graph theory** with **non-negative matrix theory**.

⇒ Some hard questions in non-negative matrix theory are simple in graph theory, and vice versa.

Facts about the adjacency matrix I

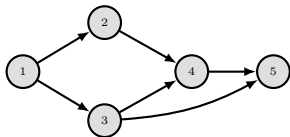
- If $\mathcal{G}$ has only positive edge weights, then A is a non-negative matrix. (Products of non-negative matrices are non-negative)
- If the out-degree $d_{i,out}$ of each node (sum of the weights of outgoing edges incident on it) is equal to one, then A is a **row-stochastic matrix** $A\mathbf{1}_n = \mathbf{1}_n$
- If the in-degree $d_{i,in}$ of each node (sum of the weights of incoming edges incident on it) is equal to one, then A is a **column-stochastic matrix** $A^T\mathbf{1}_n = \mathbf{1}_n$
- If $\mathcal{G}$ is **weight-balanced** (sum of the weights of incoming edges equal to sum of outgoing edges for every node) then $A\mathbf{1}_n = A^T\mathbf{1}_n$, if they also add up to 1 then A is said **doubly stochastic**.

Convention: In most network systems the entry a_{ij} is the weight assigned assigned to the edge connecting i and j with information flow $j \rightarrow i$. In literature, the information flow may represent "sending information" or "sensing quantities". Thus the direction of edge may be different depending on the context. In algebraic graph theory an adjacency is stochastic if all the out-degrees are equal to 1. In systems and control theory we find also the opposite convention.

Facts about the adjacency matrix II

- The element a_{ij} of A^k is positive if and only if there exists a directed path of length k from node i to node j .
- If $\mathcal{G}$ has self-loops at each node (coefficients $a_{ii} > 0$) and it is strongly connected, then matrix A^k for all $k \geq n - 1$ is a **positive matrix** (all its elements are positive)
- A non-negative matrix for which there exists $k^* > 0$ such that for all $k \geq k^*$ A^k is positive, is called Primitive.

Powers of the adjacency matrix count paths



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A^2 = \begin{bmatrix} 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A^3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Key property

$$[A^k]_{ij} = \#\{\text{directed paths from } i \text{ to } j \text{ of length } k\}.$$

$$[A^2]_{14} = 2: \quad 1 \rightarrow 2 \rightarrow 4, \quad 1 \rightarrow 3 \rightarrow 4.$$

$$[A^3]_{15} = 2: \quad 1 \rightarrow 2 \rightarrow 4 \rightarrow 5, \quad 1 \rightarrow 3 \rightarrow 4 \rightarrow 5.$$

Facts about the adjacency matrix III

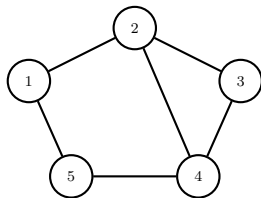
From Perron–Frobenius Theorem: If a non-negative matrix A is Primitive:

- Matrix A has a largest eigenvalue strictly positive with unitary algebraic multiplicity and which satisfies $\lambda > |\mu|$ for all other eigenvalues $|\mu|$.
- The right and left eigenvectors v and w of λ are unique and positive, up to rescaling.
- if A is row-stochastic, its largest eigenvalue is $\lambda = 1$

The Laplacian matrix of graph

The Laplacian matrix is obtained from the **degree matrix** and the **adjacency matrix** (out-degree matrix and the **adjacency matrix** for weighted directed graphs):

$$L = D - A.$$



$$\mathcal{V} = \{1, \dots, 5\}$$

$$\mathcal{E} = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 1\}, \{2, 4\}\}.$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix},$$

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}.$$

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

Main properties of the Laplacian matrix

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an **undirected graph** and let $L = D - A$

Structural properties

- L is symmetric:

$$L = L^\top.$$

- The diagonal entries are node degrees:

$$L_{ii} = d_i.$$

- The off-diagonal entries encode edges:

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

Algebraic properties

- Rows and columns sum to zero:

$$L\mathbf{1} = 0, \quad \mathbf{1}^\top L = 0.$$

- For undirected graphs:

$$x^\top Lx = \sum_{(i,j) \in \mathcal{E}} (x_i - x_j)^2.$$

- L is positive semidefinite:

$$x^\top Lx \geq 0, \quad \forall x \in \mathbb{R}^n.$$

Note: The quadratic form $x^\top Lx$ measures "disagreement" across the edges of the graph.

Gershgorin Disk Theorem

Let $A = [a_{ij}] \in \mathbb{C}^{n \times n}$.

Theorem

Every eigenvalue $\lambda \in \sigma(A)$ lies in at least one disk

$$\mathcal{D}_i = \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \right\}.$$

Each disk $\mathcal{D}_i$ has

$$\text{center in } a_{ii}, \quad \text{radius } r_i = \sum_{j \neq i} |a_{ij}|.$$

Proof of Gershgorin Disk Theorem

Consider the eigenvalue equation $Ax = \lambda x$, with $x \neq 0$. Choose $i \in \{1, \dots, n\}$ such that

$$|x_i| = \max_{j \in \{1, \dots, n\}} |x_j| > 0.$$

The i -th component gives

$$\lambda x_i = \sum_{j=1}^n a_{ij} x_j, \quad \lambda - a_{ii} = \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} \frac{x_j}{x_i}.$$

Taking absolute values,

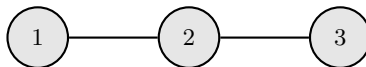
$$|\lambda - a_{ii}| = \left| \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} \frac{x_j}{x_i} \right| \leq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| \frac{|x_j|}{|x_i|} \leq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|.$$

Therefore,

$$\lambda \in \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \right\}.$$

Thus every eigenvalue of A lies in at least one Gershgorin disk.

Example: path graph with three nodes



Its Laplacian matrix is

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

For a Laplacian matrix,

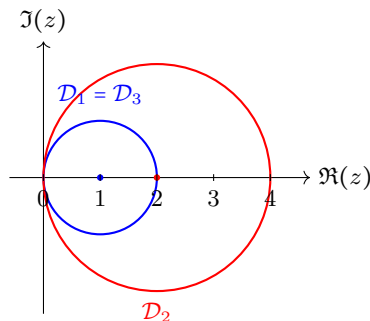
$$L_{ii} = d_i, \quad \sum_{j \neq i} |L_{ij}| = d_i.$$

Therefore each Gershgorin disk is centered at d_i and has radius d_i .

Gershgorin disks for the Laplacian of the path graph

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

the disks are $\mathcal{D}_1 : |z - 1| \leq 1$, $\mathcal{D}_2 : |z - 2| \leq 2$, $\mathcal{D}_3 : |z - 1| \leq 1$.



Consequence for Laplacian eigenvalues

The union of the Gershgorin disks is contained in the interval

$$[0, 4].$$

Thus,

$$\sigma(L) \subseteq [0, 4].$$

Indeed, for the path graph with three nodes,

$$\sigma(L) = \{0, 1, 3\}.$$

General bound

For any undirected graph Laplacian L ,

$$0 \leq \lambda_i(L) \leq 2D_{\max},$$

where $D_{\max}$ is the maximum node degree.

Spectrum of the Laplacian matrix

Since $L = L^\top$, all eigenvalues are real and can be ordered as $0 = \lambda_1(L) \leq \lambda_2(L) \leq \dots \leq \lambda_n(L) \leq 2d_{max}$.

Connectivity information

- $\lambda_1(L) = 0$, with eigenvector $\mathbf{1}$:

$$L\mathbf{1} = 0.$$

- The multiplicity of the zero eigenvalue equals the number of connected components of G .
- The largest eigenvalue is less than the maximum degree $\lambda_n(L) \leq 2d_{max}$
- The graph Diameter (length of longest shortest path) satisfies: $Diameter \geq \frac{4}{n\lambda_2(L)}$.

Algebraic connectivity

The second-smallest eigenvalue $\lambda_2(L)$ is called the **algebraic connectivity** of the graph.

The graph is connected if and only if $\lambda_2(L) > 0$.

It measures how well connected the graph is:

$\lambda_2(L)$ small $\Rightarrow$ G badly connected

$\lambda_2(L)$ large $\Rightarrow$ G well connected.

Note: Algebraic graph theory transforms a graph theoretical property into an algebraic property and vice-versa.

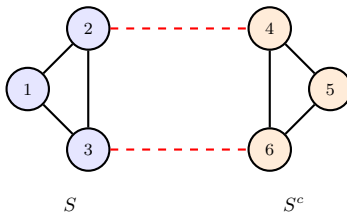
Connectivity as a graph cut problem

A graph is **well connected** if it is hard to split it into two disconnected communities.

$$\mathcal{V} = S \cup S^c, \quad S \cap S^c = \emptyset, S \neq \emptyset, S^c \neq \emptyset.$$

The **cut** induced by S is

$$\text{cut}(S, S^c) = |\{\{i, j\} \in \mathcal{E} : i \in S, j \in S^c\}|.$$



$$\text{cut}(S, S^c) = 2.$$

Minimum cut and balanced partitions

The minimum cut problem asks for the smallest number of edges that must be removed to disconnect the graph:

$$\min_{\emptyset \neq S \subsetneq \mathcal{V}} \text{cut}(S, S^c).$$

Note: Good connectivity means that to separate the two partitions we need to cut many edges.

Minimum cut and balanced partitions

The minimum cut problem asks for the smallest number of edges that must be removed to disconnect the graph:

$$\min_{\emptyset \neq S \subsetneq \mathcal{V}} \text{cut}(S, S^c).$$

Note: Good connectivity means that to separate the two partitions we need to cut many edges.

From graph cuts to the Laplacian quadratic form

Associate a binary vector $z \in \{-1, +1\}^n$ with a partition:

$$z_i = \begin{cases} +1, & i \in S, \\ -1, & i \in S^c. \end{cases}$$

For an undirected graph,

$$z^\top L z = \sum_{\{i,j\} \in \mathcal{E}} (z_i - z_j)^2.$$

If (i, j) is inside the same community, then $z_i - z_j = 0$. If (i, j) crosses the cut, then $z_i - z_j = \pm 2$, hence

$$(z_i - z_j)^2 = 4.$$

Therefore,

$$z^\top L z = 4 \text{ cut}(S, S^c).$$

Cut problem as a discrete optimization problem

$$\min_{z \in \{-1, +1\}^n \setminus \{-\mathbf{1}_n, \mathbf{1}_n\}} z^\top L z = 4 \min_{\emptyset \neq S \subsetneq \mathcal{V}} \text{cut}(S, S^c).$$



Relaxation: algebraic connectivity

The discrete cut problem is hard because $z_i \in \{-1, +1\}$.

We can find a lower bound to the optimal objective value by relaxing to $x_i \in [-1, +1]$:

$$\min_{x \in \mathbb{R}^n} x^\top Lx \quad \text{s.t.} \quad x^\top \mathbf{1} = 0, \quad \|x\|_\infty \leq 1.$$

As final step, we consider $\|x\|_2 = 1$ instead of $\|x\|_\infty = 1$ since $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$. Thus, this heuristic method yields by the Rayleigh–Ritz theorem, (since $L\mathbf{1} = \mathbf{0} \Rightarrow \lambda_1(L) = 0$):

$$\lambda_2(L) = \min_{\substack{x \neq 0 \\ x^\top \mathbf{1} = 0, \\ \|x\|_2 = 1}} x^\top Lx$$

Interpretation

$\lambda_2(L)$ is a heuristic continuous relaxation of the minimum cut problem.

Fiedler vector: detecting communities

The eigenvector v_2 associated with $\lambda_2(L)$ is called the **Fiedler vector**:

$$Lv_2 = \lambda_2(L)v_2, \quad v_2^\top \mathbf{1} = 0.$$

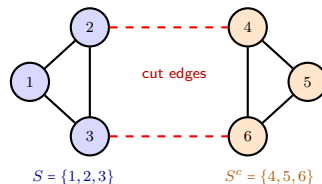
Spectral partitioning rule

Use the sign of the Fiedler vector:

$$S = \{i : (v_2)_i \geq 0\}, \quad S^c = \{i : (v_2)_i < 0\}.$$

Example:

$$v_2 \approx \begin{bmatrix} 0.42 \\ 0.38 \\ 0.31 \\ -0.29 \\ -0.36 \\ -0.44 \end{bmatrix} \implies \{1, 2, 3\} \mid \{4, 5, 6\}.$$



$$\lambda_2(L) = 1.$$



Laplacian spectra of canonical graph families

Let $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ be the eigenvalues of L .

Complete graph K_n

$$\text{spec}(L) = \{0, \underbrace{n, \dots, n}_{n-1}\}.$$

$$\lambda_2(K_n) = n.$$

Line graph P_n

$$\lambda_j = 2 - 2 \cos\left(\frac{(j-1)\pi}{n}\right), \quad j = 1, \dots, n.$$

$$\lambda_2(P_n) = 2 - 2 \cos\left(\frac{\pi}{n}\right) \sim \frac{\pi^2}{n^2}.$$

Ring graph C_n

$$\lambda_j = 2 - 2 \cos\left(\frac{2\pi(j-1)}{n}\right), \quad j = 1, \dots, n.$$

$$\lambda_2(C_n) = 2 - 2 \cos\left(\frac{2\pi}{n}\right) \sim \frac{4\pi^2}{n^2}.$$

Star graph S_n

$$\text{spec}(L) = \{0, \underbrace{1, \dots, 1}_{n-2}, n\}.$$

$$\lambda_2(S_n) = 1.$$

Spectral properties: grid, k -regular, proximity graphs

Grid graph $G_{q,r} = P_q \times P_r$

The Laplacian spectrum is the sum of path spectra:

$$\lambda_{a,b} = \left[2 - 2 \cos\left(\frac{a\pi}{q}\right) \right] + \left[2 - 2 \cos\left(\frac{b\pi}{r}\right) \right],$$

$$a = 0, \dots, q-1, \quad b = 0, \dots, r-1.$$

If $q = r = m$, then $n = m^2$ and

$$\lambda_2 = 2 - 2 \cos\left(\frac{\pi}{m}\right) \sim \frac{\pi^2}{m^2} = \frac{\pi^2}{n}.$$

k -regular graph

$L = kI - A$. If A has eigenvalues

$$\mu_1 = k \geq \mu_2 \geq \dots \geq \mu_n,$$

then

$$\lambda_i(L) = k - \mu_i(A).$$

In particular,

$$\lambda_1 = 0, \quad \lambda_2 = k - \mu_2(A).$$

Spectral properties: random and regular graphs

Erdős–Rényi random graph $G(n, p)$

Each edge is present independently with probability p .

$$\mathbb{E}[d_i] = (n-1)p.$$

For connected dense random graphs, the nonzero Laplacian eigenvalues concentrate around np :

$$\lambda_i(L) \approx np, \quad i = 2, \dots, n.$$

$$\lambda_2(L) \approx np \quad \text{for sufficiently large } np.$$

Connectivity appears around $p \sim \frac{\log n}{n}$.

Random k -regular graph

Every node has degree exactly k :

$$D = kI, \quad L = kI - A.$$

If $\mu_i(A)$ are the adjacency eigenvalues, then

$$\lambda_i(L) = k - \mu_i(A).$$

For large random k -regular graphs,

$$\mu_2(A) \approx 2\sqrt{k-1},$$

hence

$$\lambda_2(L) \approx k - 2\sqrt{k-1}.$$

Modeling the couplings: Local interaction rules

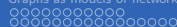
Consider a simple agent dynamics: At each time step move the state closer to that of the neighbors.
The single agent's dynamics in continuous time is:

$$\dot{x}_i(t) = u_i(t)$$

The single agent's dynamics in discrete time (Euler discretization) is:

$$x_i(k+1) = x_i(k) + \varepsilon u_i(k)$$

⇒ This system is called "single integrator" and is useful to approximate the position of a mobile robot and many other network systems.



Modeling the couplings: Local interaction rules

The coupling or **local interaction** between two generic agents can be specified by a (possibly nonlinear) function $f_{ij}(x_i(k) - x_j(k))$.

Based on the network topology specified by the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ an example of linear feedback control input due to inter-agent couplings is

$$u_i(t) = \sum_{j \in \mathcal{N}_i} l_{ij}(x_j(t) - x_i(t)).$$

By substituting u_i in the agent dynamics, we get in CT:

$$\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} l_{ij}(x_j(t) - x_i(t)), \quad i, j \in \mathcal{V}, t \in \mathbb{R}.$$

in DT:

$$x_i(k+1) = x_i(k) + \varepsilon \sum_{j \in \mathcal{N}_i} l_{ij}(x_j(k) - x_i(k)), \quad i, j \in \mathcal{V}, k \in \mathbb{N}.$$

An example of linear multi-agent system modeling

The linear multi-agent system, with vector state $x(k) = [x_1(k), \dots, x_n(k)]^T$ evolves as an autonomous linear dynamical system and the Laplacian matrix emerges from the structure of the network

$$x(k+1) = x(k) - \varepsilon Lx(k), \quad x(k) \in \mathbb{R}^n, \quad x(0) = x_0.$$

Let I be the identity matrix and $P = I - \varepsilon L$:

$$x(k+1) = Px(k), \quad x(k) \in \mathbb{R}^n, \quad x(0) = x_0.$$

If ε is less than the largest node degree of G ($\varepsilon < \frac{1}{D_{max}}$) matrix P , inherits several properties from the properties of matrix L :

If G is undirected and connected: $P = P^T$, $P\mathbf{1} = \mathbf{1}$, $\mathbf{1}^T P = \mathbf{1}^T$, P is a doubly-stochastic matrix, and

$$\sigma(P) = \{1, 1 - \varepsilon\lambda_2(L), 1 - \varepsilon\lambda_3(L), \dots, 1 - \varepsilon\lambda_n(L)\}$$

Spectral radius, induced norm and essential spectral radius

Let $A \in \mathbb{R}^{n \times n}$.

- The **spectral radius** of A is: $\rho(A) = \max\{|\lambda| : \lambda \in \text{spec}(A)\}$.
- The p -induced norm of A is

$$\|A\|_p = \max_{\|x\|_p=1} \|Ax\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}.$$

- For every induced norm, $\rho(A) \leq \|A\|_p$.
- In particular, if $A = A^\top$, then

$$\|A\|_2 = \rho(A).$$

Essential spectral radius

If A is row-stochastic, then $A\mathbf{1} = \mathbf{1}$, hence $1 \in \text{spec}(A)$. The essential spectral radius is

$$\rho_{\text{ess}}(A) = \begin{cases} 0, & \text{spec}(A) = \{1, \dots, 1\}, \\ \max\{|\lambda| : \lambda \in \text{spec}(A) \setminus \{1\}\}, & \text{otherwise.} \end{cases}$$

Spectral radius and essential spectral radius

- Since $P = I - \varepsilon L$ is stochastic its spectral radius is $\rho(P) = 1$,
- The essential spectral radius is $\rho_{ess}(P) = \max(|1 - \varepsilon \lambda_2(L)|, |1 - \varepsilon \lambda_n(L)|)$. If $\varepsilon < \frac{1}{2D_{max}}$ simply $\rho_{ess}(P) = 1 - \varepsilon \lambda_2(L)$.

Consensus problem

Definition: Consensus problem

The consensus problem consists of designing local interaction protocols so that the state of each agent is steered toward the same value.

$$\lim_{k \rightarrow \infty} \|x_i(k) - x_j(k)\| = 0, \quad \forall (i, j) \in E, \forall x(0) \in \mathbb{R}^n$$

Additionally, we may ask that such **consensus value** is constant

$$\lim_{k \rightarrow \infty} x_i(k) = c \quad \forall i \in \mathcal{V}, \forall x(0) \in \mathbb{R}^n$$

When the constant c is equal to the average of the initial states

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

we call it **consensus on the average**.

Consensus for linear MAS with row-stochastic state transition matrices

Theorem: Consensus for linear MAS

Consider a linear multi-agent system with row-stochastic state transition matrix A :

$$x_i(k+1) = a_{ii}x_i(k) + \sum_{j \in \mathcal{N}_i(k)} a_{ij}x_j(k), \quad x(k+1) = Ax(k)$$

Let $\mathcal{G}$ be its associated directed graph. If $\mathcal{G}$ has self-loops and is strongly connected, then the evolution of the vector state of the MAS satisfies

$$\lim_{k \rightarrow \infty} x(k) = w^T x(0) \mathbf{1}_n$$

where $w \in \mathbb{R}^n$ is a positive vector and it is the left eigenvector of A corresponding to its unique eigenvalue equal to one and $\mathbf{1}^T w = 1$.

If, additionally, A is doubly-stochastic, i.e., $\mathcal{G}$ is weight-balanced, then $w = \frac{1}{n} \mathbf{1}_n$, thus

$$\lim_{k \rightarrow \infty} x(k) = \frac{\mathbf{1}_n^T x(0)}{n} \mathbf{1}_n$$

Proof: row-stochastic consensus

Since A is row-stochastic $\Rightarrow A\mathbf{1} = \mathbf{1}$

$\mathcal{G}$ strongly connected with self-loops $\Rightarrow A$ is primitive.

A is row-stochastic + A is primitive + Perron–Frobenius Theorem $\Rightarrow$ the largest eigenvalue is 1 and it is simple (unitary algebraic multiplicity) and all others have magnitude strictly less than 1.

$$\lim_{k \rightarrow \infty} A^k = \mathbf{1}w^\top,$$

where

$$w^\top A = w^\top, \quad w_i > 0, \quad \mathbf{1}^\top w = 1.$$

Thus, for

$$x(k+1) = Ax(k), \quad x(k) = A^k x(0).$$

Taking the limit,

$$\lim_{k \rightarrow \infty} x(k) = \left(\lim_{k \rightarrow \infty} A^k \right) x(0) = \mathbf{1}w^\top x(0).$$

Hence

$$\lim_{k \rightarrow \infty} x_i(k) = w^\top x(0), \quad i = 1, \dots, n.$$

Proof: average consensus for doubly-stochastic matrices

Assume now that A is also column-stochastic:

$$\mathbf{1}^\top A = \mathbf{1}^\top.$$

Then, the sum of the states is time-invariant:

$$\mathbf{1}^\top x(k+1) = \mathbf{1}^\top Ax(k) = \mathbf{1}^\top x(k).$$

Therefore,

$$\mathbf{1}^\top x(k) = \mathbf{1}^\top x(0), \quad \forall k.$$

Since the sum is preserved, the consensus value must satisfy

$$\mathbf{1}^\top \mathbf{1} w^\top x(0) = \mathbf{1}^\top x(0).$$

Thus

$$w^\top x(0) = \frac{1}{n} \mathbf{1}^\top x(0), \quad w = \frac{1}{n} \mathbf{1}.$$

Therefore,

$$\lim_{k \rightarrow \infty} x(k) = \frac{\mathbf{1}^\top x(0)}{n} \mathbf{1}.$$

Consensus in linear MAS in time-varying network topologies

Theorem: Consensus for Laplacian-based MAS with time-varying network topologies

Consider a linear MAS that evolves according to a stochastic weighted adjacency matrix $W(k)$ with at least a positive diagonal element.

If its associated graph $\mathcal{G}(k)$ is undirected and connected at each time step k , then the evolution of the vector state of the MAS satisfies

$$\lim_{k \rightarrow \infty} x(k) = \frac{\mathbf{1}_n^T x(0)}{n} \mathbf{1}_n$$

The local interaction rule:

$$x_i(k+1) = x_i(k) + \varepsilon \sum_{j \in \mathcal{N}_i} l_{ij}(x_j(k) - x_i(k)), \quad i, j \in \mathcal{V}, k \in \mathbb{N}.$$

yields stochastic state transition matrix $P(k) = I - \varepsilon L(k)$ if $0 < \varepsilon < \frac{1}{\max_k d_{\max}(k)}$.

Consensus over switching connected graphs: proof

- $x(k+1) = W(k)x(k)$, since $W(k)$ is stochastic for every k $\rho(W(k)) = 1$
- By assumption $\mathcal{G}$ is undirected and connected at all times, thus $W(k)$ is symmetric and $\rho_{ess} < 1$.
- $W(k)\mathbf{1} = \mathbf{1}$, also $\mathbf{1}^\top W(k) = \mathbf{1}^\top$.
- Therefore the average $\bar{x}(k)$ of $x(k)$ is time-invariant:

$$\bar{x}(k+1) = \frac{\sum_{i \in V} x_i(k+1)}{n} = \frac{1}{n} \mathbf{1}^\top x(k+1) = \frac{1}{n} \mathbf{1}^\top W(k)x(k) = \frac{1}{n} \mathbf{1}^\top x(k) = \frac{\sum_{i \in V} x_i(0)}{n} = \bar{x}(0).$$

- Define the disagreement vector

$$\delta(k) = x(k) - \bar{x}(k)\mathbf{1} = x(k) - \bar{x}(0)\mathbf{1}.$$

- Then

$$\delta(k+1) = W(k)\delta(k), \quad \mathbf{1}^\top \delta(k) = 0.$$

Consensus over switching connected graphs: proof

- Use the common Lyapunov function $V(x) = \|x(k) - \frac{1}{n}\mathbf{1}\mathbf{1}^T x(k)\|_2^2 = \|\delta(k)\|_2^2$.

$$V(k+1) = \|\delta(k+1)\|_2^2 = \|W(k)\delta(k)\|_2^2 \leq \rho_{ess}^2(k)\|\delta(k)\|_2^2 = \rho_{ess}^2(k)V(k)$$

- If $W(k) = P(k)$ then

$$\max_{\|x\|_2=1, x^T \mathbf{1}=0} \|P(k)x\|_2 = \rho_{ess}(P(k)) = \rho_{ess}(I - \varepsilon L(k)) = \max(|1 - \varepsilon\lambda_2(L(k))|, |1 - \varepsilon\lambda_n(L(k))|).$$

- The number of graphs with n nodes is finite ($2^{n(n-1)/2}$), so the switching of the linear system occurs over a finite set of matrices. This implies that each element of $W(k)$ is lower bounded by a constant, if greater than 0. Thus, since $\mathcal{G}(k)$ is connected at all times $\rho_{ess}(k) \leq c < 1$,
- It follows $V(k) \leq c^{2k}V(0) \rightarrow$ Converges exponentially fast.
- $V(k) = 0$ implies $\delta(k) = \mathbf{0}$, and $x(k) = \bar{x}(0)\mathbf{1}_n$, which means

$$\lim_{k \rightarrow \infty} x(k) = \frac{\mathbf{1}_n^T x(0)}{n} \mathbf{1}_n$$

Numerical Simulation

Linear consensus protocol:

$$x_i(k+1) = x_i(k) - \varepsilon \sum_{j \in \mathcal{N}_i(k)} (x_i(k) - x_j(k))$$

- Agents are mobile robots in a plane, thus $x_i(k)$ is a two-element vector representing two Cartesian coordinates.
- $\varepsilon < \frac{1}{n}$.
- $n = 30$
- $G(k)$ is a proximity graph with sensing radius between agents equal to half of box size
- Execute Matlab sim "RendezvousExample.m"

Advantages and disadvantages of multi-agent control

A MAS executing a consensus algorithm is **robust** to

- Communication failures between the agents, as long as the network stays connected;
- Failure of one or more agents which stop operating;
- Time-varying network topology with fast changes;
- Large-scale systems with extremely large number of agents.

A MAS executing a consensus algorithm is **vulnerable** to

- An agent is unable to execute correctly its control protocol but continues to communicate and interact with the others;
- An agent is hijacked and adopts **adversarial behavior** as a consequence of a cyberattack. It continues to interact with the others while executing a different control protocol.

Advantages and disadvantages of multi-agent control

A MAS executing a consensus algorithm is **robust** to

- Communication failures between the agents, as long as the network stays connected;
- Failure of one or more agents which stop operating;
- Time-varying network topology with fast changes;
- Large-scale systems with extremely large number of agents.

A MAS executing a consensus algorithm is **vulnerable** to

- An agent is unable to execute correctly its control protocol but continues to communicate and interact with the others;
- An agent is hijacked and adopts **adversarial behavior** as a consequence of a cyberattack. It continues to interact with the others while executing a different control protocol.

Advantages and disadvantages of multi-agent control

A MAS executing a consensus algorithm is **robust** to

- Communication failures between the agents, as long as the network stays connected;
- Failure of one or more agents which stop operating;
- Time-varying network topology with fast changes;
- Large-scale systems with extremely large number of agents.

A MAS executing a consensus algorithm is **vulnerable** to

- An agent is unable to execute correctly its control protocol but continues to communicate and interact with the others;
- An agent is hijacked and adopts **adversarial behavior** as a consequence of a cyberattack. It continues to interact with the others while executing a different control protocol.

Leader-following and Containment control

If a **single agent** exchanges state information with neighbors but imparts a different input than the consensus protocol, for instance if it is remotely piloted, then the multi-agent system starts following its trajectory in an attempt to rendezvous with it.

⇒ This behavior is actually useful in swarm robotics: one pilot can control large numbers of mobile robots by controlling just one "leader" of them. This behavior is called **leader following**

Leader-following and Containment control

If a **single agent** exchanges state information with neighbors but imparts a different input than the consensus protocol, for instance if it is remotely piloted, then the multi-agent system starts following its trajectory in an attempt to rendezvous with it.

⇒ This behavior is actually useful in swarm robotics: one pilot can control large numbers of mobile robots by controlling just one "leader" of them. This behavior is called **leader following**

Leader-following and Containment control

If **two or more leader agents** exchange state information with neighbors but impart different inputs, then the agents in the multi-agent system converge inside the convex hull, or area, spanned by the position of the leader agents, as long as the interaction graph stays connected

⇒ Also this behavior is actually useful in swarm robotics: one or more pilots can control large numbers of mobile robots so that they are **contained** inside the area defined by the positions of the leader agents as vertices of a polygon or polyhedron. This behavior is called **Containment control**.

Leader-following and Containment control

If **two or more leader agents** exchange state information with neighbors but impart different inputs, then the agents in the multi-agent system converge inside the convex hull, or area, spanned by the position of the leader agents, as long as the interaction graph stays connected

⇒ Also this behavior is actually useful in swarm robotics: one or more pilots can control large numbers of mobile robots so that they are **contained** inside the area defined by the positions of the leader agents as vertices of a polygon or polyhedron. This behavior is called **Containment control**.

Why do we need Open Multi-Agent Systems?

Classical MAS viewpoint

Most models assume a fixed set of agents:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \quad |\mathcal{V}| = N.$$

The state has fixed dimension:

$$x(k) = \begin{bmatrix} x_1(k) & \cdots & x_N(k) \end{bmatrix}^T \in \mathbb{R}^{pN}.$$

This is mathematically convenient: equilibria, convergence, Lyapunov functions, and consensus subspaces are defined in a fixed state space.

But real networks are open

In many applications, agents may:

- join the network;
- leave the network;
- fail, disconnect, or be replaced;
- activate only when resources are available.

$$\mathcal{V} \longrightarrow \mathcal{V}(k), \quad N \longrightarrow N(k) = |\mathcal{V}(k)|.$$

The state dimension itself changes over time:

$$x(k) \in \mathbb{R}^{pN(k)}.$$

Key message: openness is not just a switching topology. It changes the set of agents, the state dimension, and often the control or optimization objective.

Open Multi-Agent Systems: evolving agent sets

Open network model

At each time k , only a subset of agents is active:

$$\mathcal{G}_k = (\mathcal{V}_k, \mathcal{E}_k), \quad N(k) = |\mathcal{V}_k|.$$

The state dimension changes with the network:

$$x(k) \in \mathbb{R}^{pN(k)}.$$

Time-varying agent-set

$$\mathcal{R}_k = \mathcal{V}_k \cap \mathcal{V}_{k-1}$$

$$\mathcal{A}_k = \mathcal{V}_k \setminus \mathcal{V}_{k-1}$$

$$\mathcal{D}_k = \mathcal{V}_{k-1} \setminus \mathcal{V}_k$$

$\mathcal{R}_k$: remaining agents,

$\mathcal{A}_k$: arriving agents,

$\mathcal{D}_k$: departing agents.

Note: an open MAS is not only a time-varying graph. The set of state variables itself changes over time.

Why openness changes the analysis, control and optimization problem

Moving objective

The quantity of interest depends on active agents:

$$\bar{x}(k) = \frac{1}{|\mathcal{V}_k|} \sum_{i \in \mathcal{V}_k} x_i(k).$$

When agents join or leave,

$$\mathcal{V}_k \neq \mathcal{V}_{k-1} \implies \bar{x}(k) \neq \bar{x}(k-1).$$

New theoretical issues

- The state space changes dimension.
- The consensus value may jump.
- Old information may become obsolete.
- Lost information may need to be preserved.
- Convergence to a fixed equilibrium may be impossible.

Need to go from classical stability to "open" stability

The objective is to keep the trajectory close to a moving, dimension-varying point of interest:

$$x(k) \approx x_k^*, \quad x_k^* \in \mathbb{R}^{p|\mathcal{V}_k|}.$$

open stability $\iff$ bounded deviation from a changing target.

References, sources and suggested readings

- F. Bullo, Lectures on Network Systems, 2024, Kindle Direct Publishing, ISBN 978-1986425643,
- Mesbahi, Mehran, and Magnus Egerstedt. Graph Theoretic Methods in Multiagent Networks. Princeton University Press, 2010.
- Godsil, Chris and Royle, Gordon F, Algebraic graph theory, 2013, Springer Science & Business Media
- R. Olfati-Saber, J. A. Fax and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," in Proceedings of the IEEE, vol. 95, no. 1, pp. 215-233, Jan. 2007.
- A. Nedić and A. Olshevsky, "Distributed Optimization Over Time-Varying Directed Graphs," in IEEE Transactions on Automatic Control, vol. 60, no. 3, pp. 601-615, March 2015.
- A. Nedic, "Distributed Gradient Methods for Convex Machine Learning Problems in Networks: Distributed Optimization," in IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 92-101, May 2020.
- Kwang-Kyo Oh, Myoung-Chul Park, Hyo-Sung Ahn, A survey of multi-agent formation control, Automatica, Volume 53, 2015, Pages 424-440.



UNIVERSITÀ DEGLI STUDI
DI **CAGLIARI**



Introduction to Open Multi-Agent Systems - Part I

Thank you for your attention!

Mauro Franceschelli

Email: mauro.franceschelli@unica.it

Webpage: https://web.unica.it/unica/page/en/mauro_franceschelli