Networked Robotics Systems, Cooperative Control and Swarming (ROB-GY 6333 Section A)

- **Week 1:**
    - Syllabus
    - Calendar and Office Hours
    - Introduction to Swarm Robotics
    - Graph Theory
- Join the course Slack channel. Go to **NYU Brightspace** > **Content** > **Welcome to Swarm Robotics: Getting Started**

Photography Courtesy: Tom Fayle

Plague of Locusts,
By Jan Luyken and
Pieter Mortier

Collective Behavior

- How does each animal know what to do/where to go?
- What does it mean to *know*?
- Global vs. local
  - **Consensus:** agree together
  - **Coordination:** act together


Abner Garza for City of Othello/AMAZING ANIMALS+/TMX

Collective Behavior

- How does each animal know what to do/where to go?
- What does it mean to *know*?
- Global vs. local
  - **Consensus:** agree together
  - **Coordination:** act together
- **As roboticists, what can we learn? How can collective behavior/multi-agent systems be helpful to us?**

Why network robots at all?

Cooperative Benefits

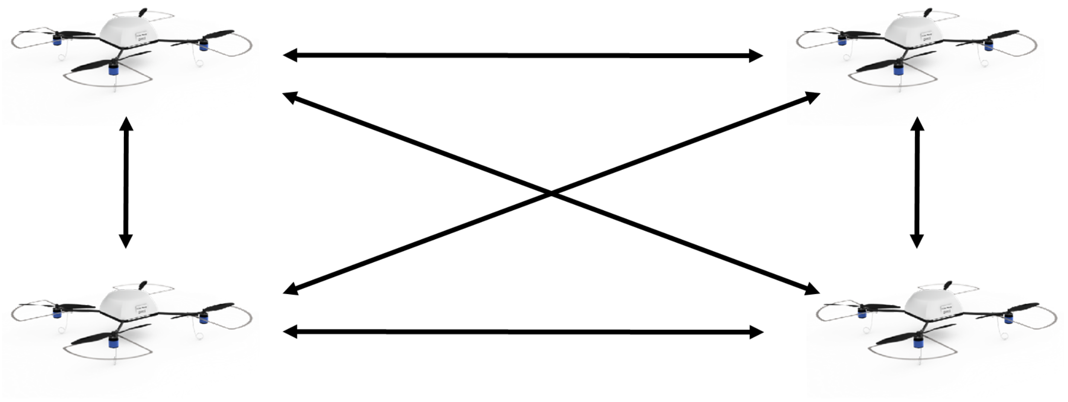
- Cost:
  - Single expensive robot with precise, complete functionality vs. many cheaper robots
- Scale
  - Coverage
- Robustness
  - Adaptable
  - Redundant
  - Efficient (Parallel)

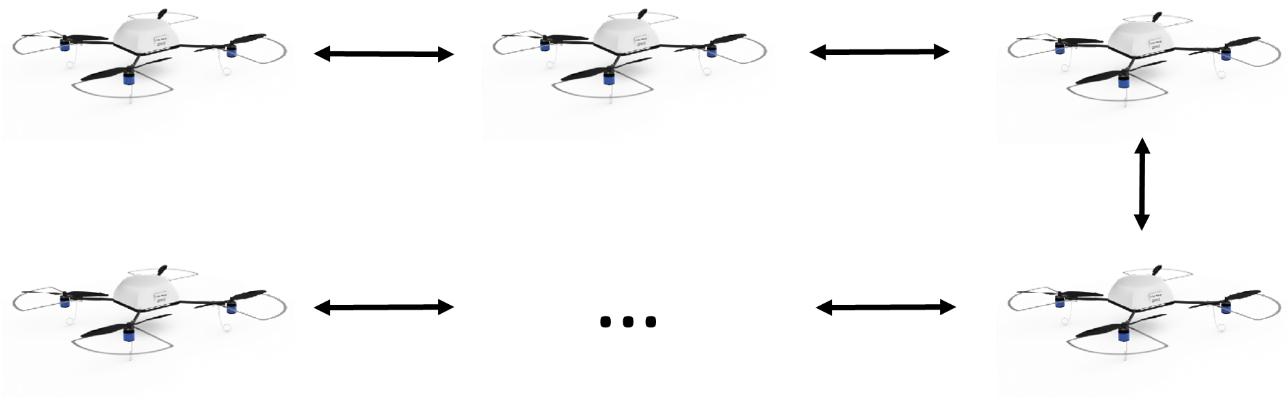
Challenges and Constraints

- **Consensus:** agree together
- **Coordination:** act together
- Communication and Control:
  - Centralization vs. Complexity
  - Composition: Heterogeneous vs. Homogeneous
  - Communication: Synchronous vs. Asynchronous

All-to-All or Neighbors-Only

- What minimum connectivity is necessary?
- How to design “local” control law





Common Challenges and Constraints

- No access to a centralized computer
- Locality in communication (limited by environment/hardware)
- Locality in sensing (limited by environment/hardware)
- Each agent has limited power and computational resources
- Design should scale to any number of agents (>1000!)

References

- **Kilobots: A Thousand-Robot Swarm**
  - https://wyss.harvard.edu/media-post/kilobots-a-thousand-robot-swarm/
- **SMORES-EP**
  - https://www.grasp.upenn.edu/projects/smores-ep/
- **Crazyswarm ()**
  - https://www.youtube.com/watch?v=D0CrjoYDt9w&t=0s
- **Multi-Robot Manipulation without Communication**
  - https://www.youtube.com/watch?v=emZVxcl3Zg4
  - https://web.stanford.edu/~schwager/MyPapers/WangSchwagerDARS14Manipulation.pdf

HARVARD
UNIVERSITY

VE RI TAS

An End-to-End System for Accomplishing Tasks with Modular Robots

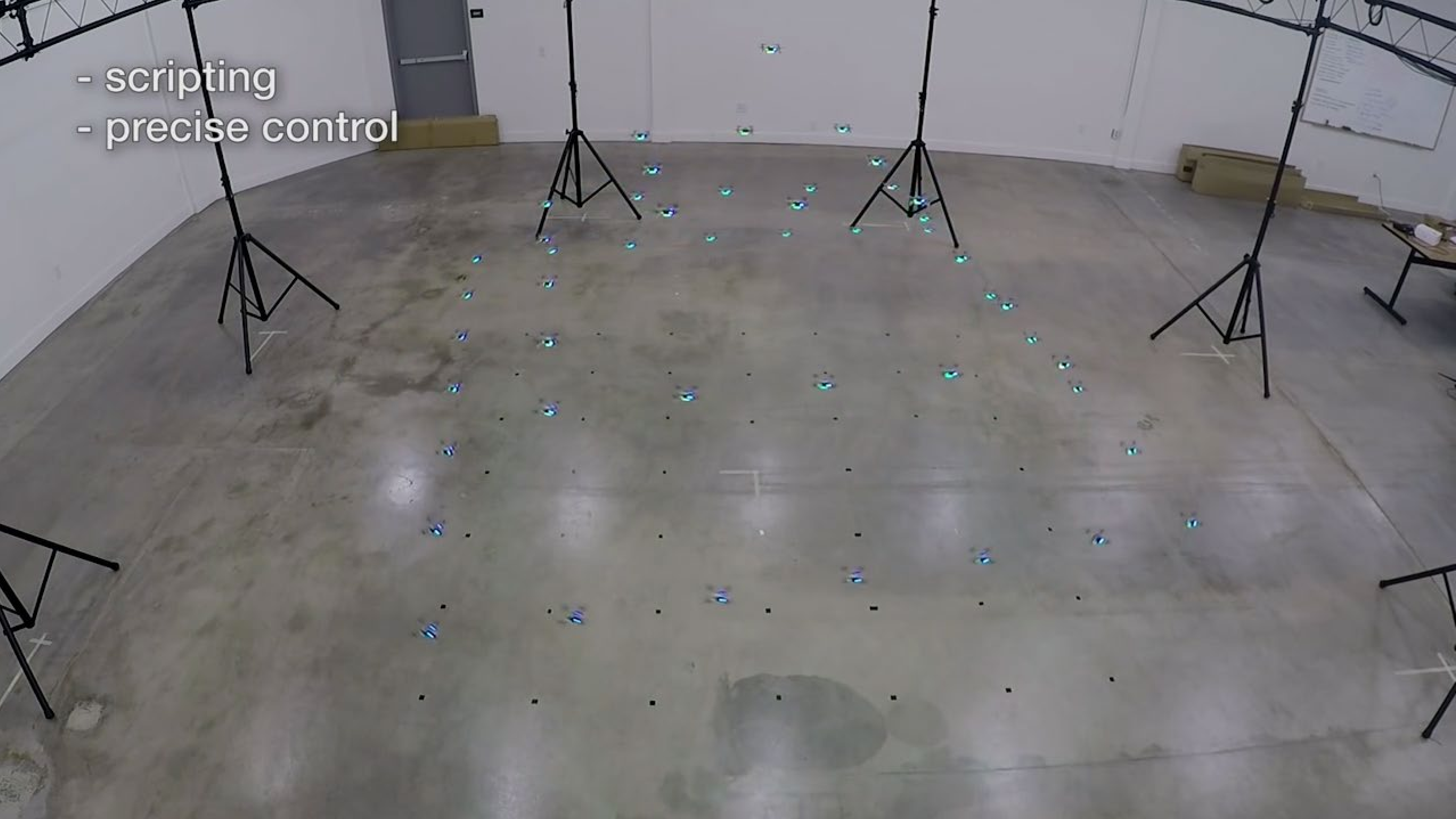
Gangyuan Jing, Tarik Tosun,
Mark Yim and Hadas Kress-Gazit

RSS 2016

CORNELL UNIVERSITY FOUNDED A.D. 1865

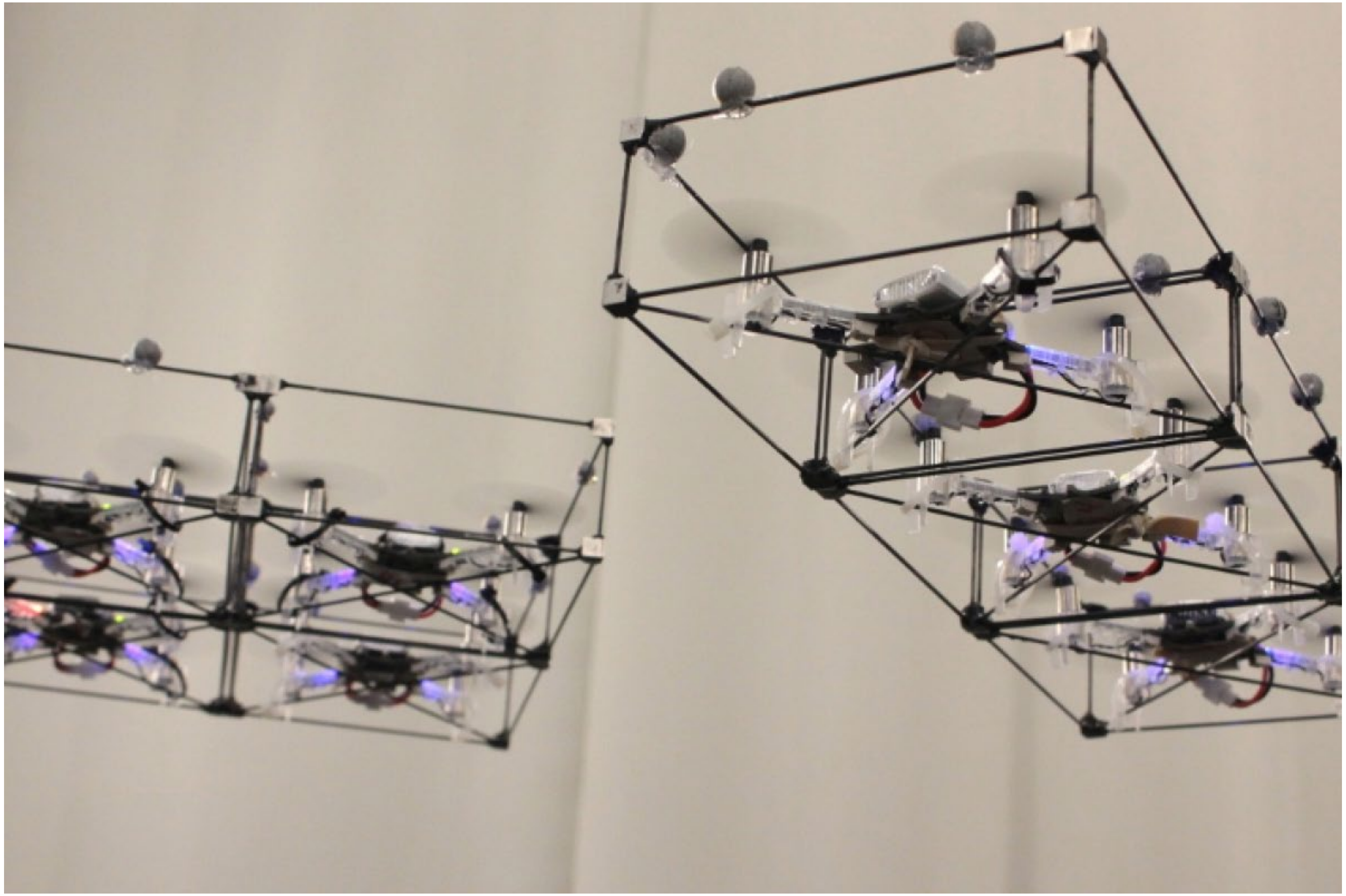
VERIFIABLE ROBOTICS RESEARCHGROUP

Penn Engineering | GRASP Laboratory
General Robotics, Automation, Sensing & Perception Lab



- scripting
- precise control

ModQuad: Assembling Structures in Midair



UPenn GRASP Lab

Simulation 2

1000 robots

Target: A Steinway K-52 Piano

Weight: 273kg, Length: 1.54m, Width: 0.67m, Height: 1.32m

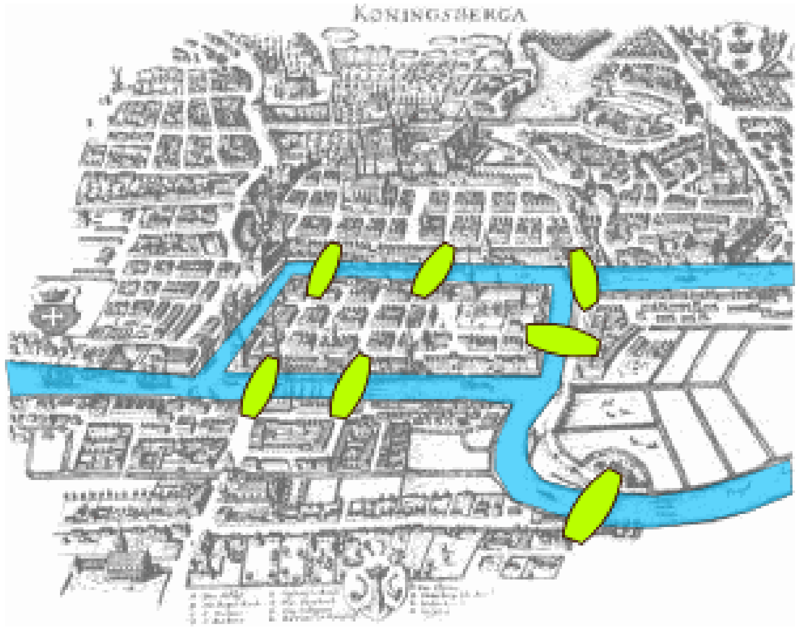
We only draw 40 robots for visualization considerations

Multi-Robot System



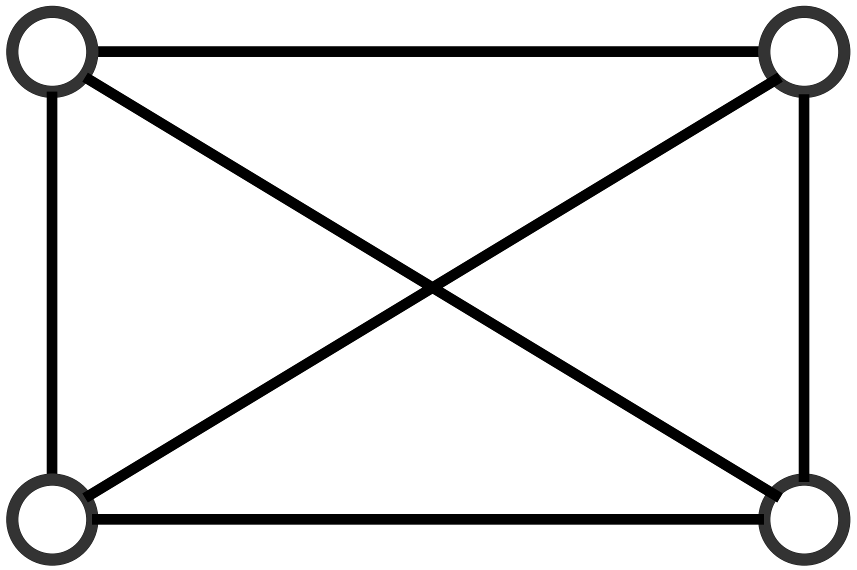

Euler and the Seven Bridges of Königsberg

- Königsberg, Prussia (modern day Kaliningrad)



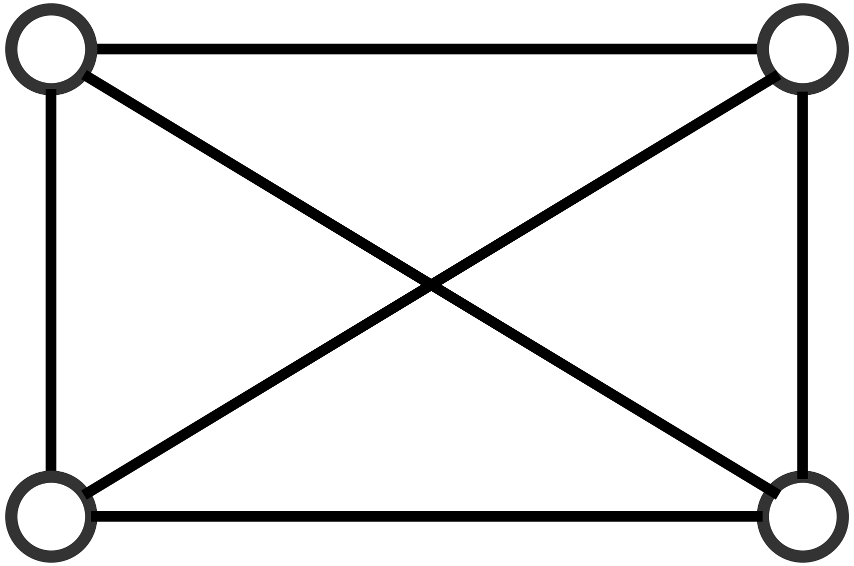
- Can you walk across each bridge only once?
- Foundational problem of graph theory (and to a lesser degree, topology)

Graph



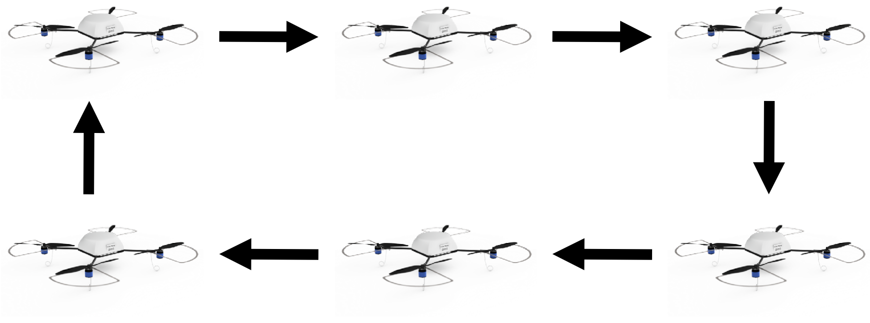
Graph

- Each **vertex** or **node** is a robot
- Each **edge** models the interaction between two robots
- The **graph** describes the connectivity properties of the system
- Many properties about the dynamics of the system can be inferred from the graph (without looking at differential equation)

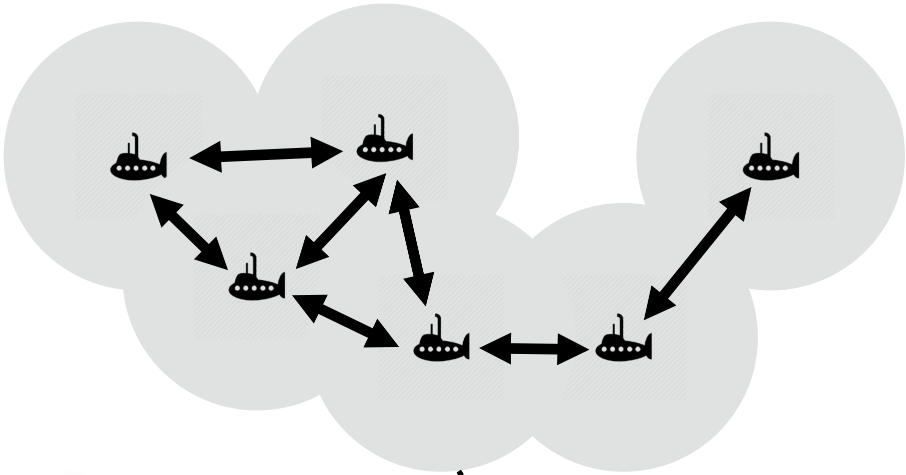


Meaning of Connectivity

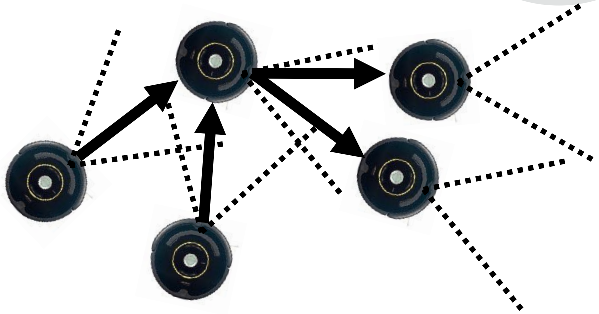
- Control Graph (Follow the leader)



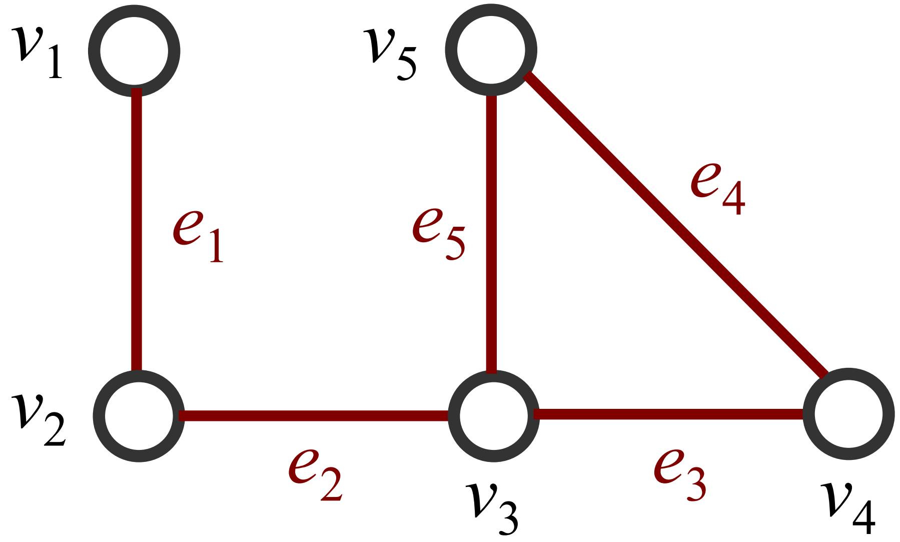
- Communication Graph (Nearest neighbors)



- Sensing Graph



Undirected Graph



- An **undirected graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is composed of:

1. A finite set of **vertices**

   $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$

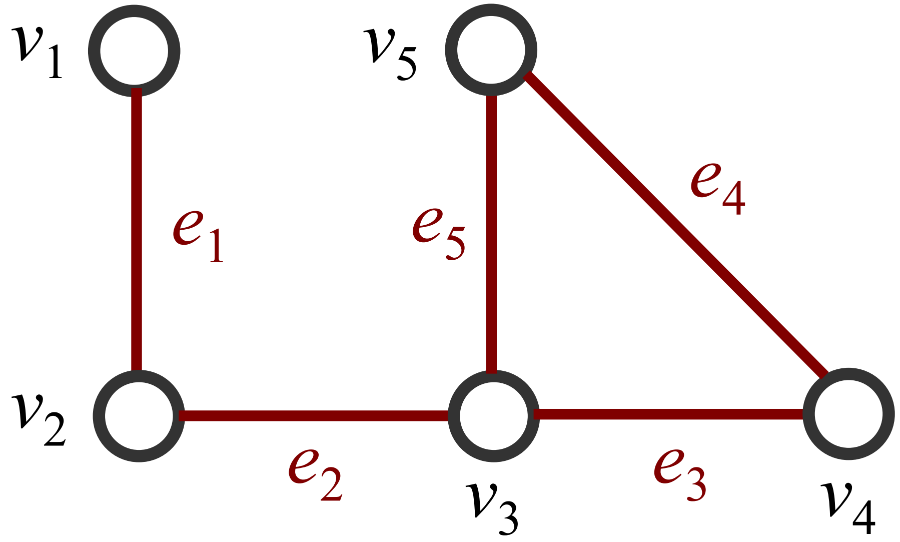
2. A finite set of **edges**

   $\mathcal{E} \subset \mathcal{V} \times \mathcal{V} = \{(v_i, v_j)\}, i = 1, \ldots, n, j = 1, \ldots, n, i \neq j$

   $(v_i, v_j) \in \mathcal{E} \Longrightarrow (v_j, v_i) \in \mathcal{E}$

subset of unordered pairs of vertices $\mathcal{V} \times \mathcal{V}$

Undirected Graph



- An **undirected graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is composed of:

1. A finite set of **vertices**

   $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$

2. A finite set of **edges**

   $\mathcal{E} \subset \mathcal{V} \times \mathcal{V} = \{(v_i, v_j)\}, i = 1, \dots, n, j = 1, \dots, n, i \neq j$

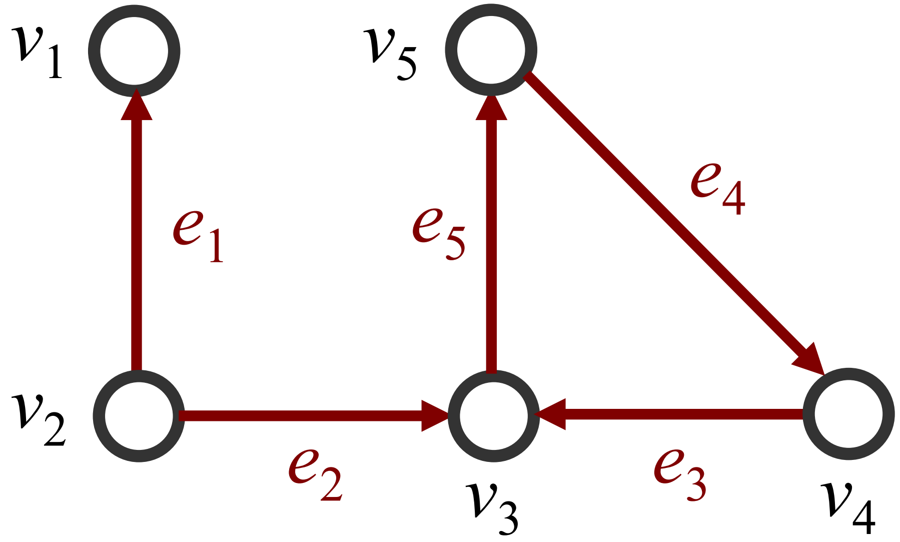
   $(v_i, v_j) \in \mathcal{E} \Longrightarrow (v_j, v_i) \in \mathcal{E}$

subset of unordered pairs of vertices $\mathcal{V} \times \mathcal{V}$

For our graph: $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$

$$\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$$
$$= \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_3, v_5)\}$$

Directed Graph



- An **directed graph** $\mathcal{D} = (\mathcal{V}, \mathcal{E})$ is composed of:

1. A finite set of **vertices**

   $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$

2. A finite set of **edges**

   $\mathcal{E} \subset \mathcal{V} \times \mathcal{V} = \{(v_i, v_j)\}, i = 1, \ldots, n, j = 1, \ldots, n, i \neq j$

   subset of ordered pairs of vertices $\mathcal{V} \times \mathcal{V}$

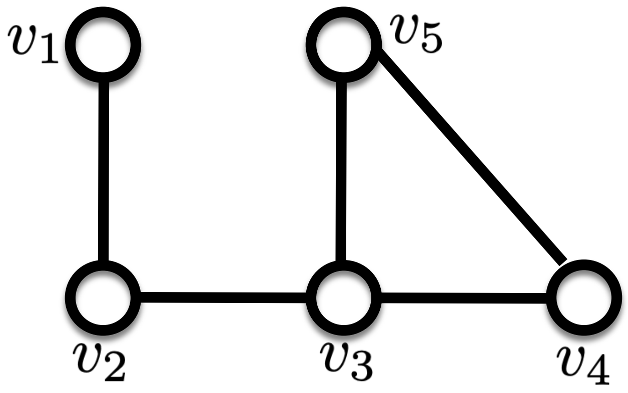
   $(v_i, v_j) \in \mathcal{E} \nRightarrow (v_j, v_i) \in \mathcal{E}$

For our graph: $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$

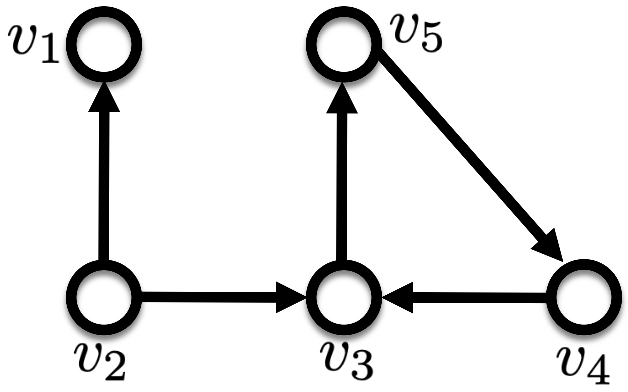
$$\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$$
$$= \{(v_2, v_1), (v_2, v_3), (v_4, v_3), (v_5, v_4), (v_3, v_5)\}$$

Definitions: Neighbor

- A node v_i is a **neighbor** of (or **adjacent** to) v_j if $(v_i, v_j) \in \mathcal{E}$
- The **neighborhood** $\mathcal{N}_i$ is the set of all neighbors of v_i



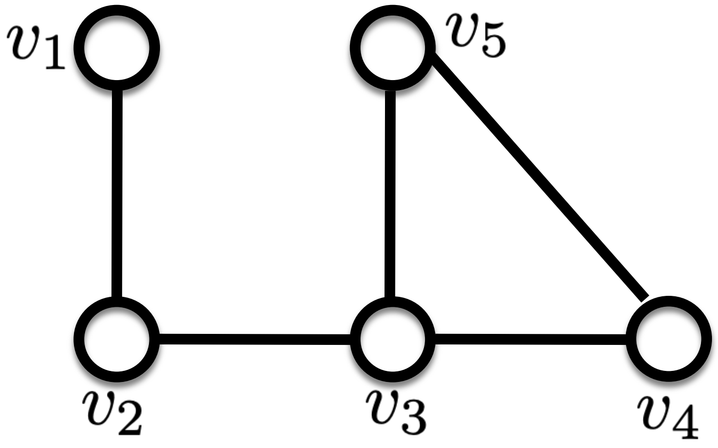
v_2 is adjacent to v_1 and v_3.
v_3 is a neighbor of v_2, v_4 and v_5
the neighborhood of v_2 is $\{v_1, v_3\}$



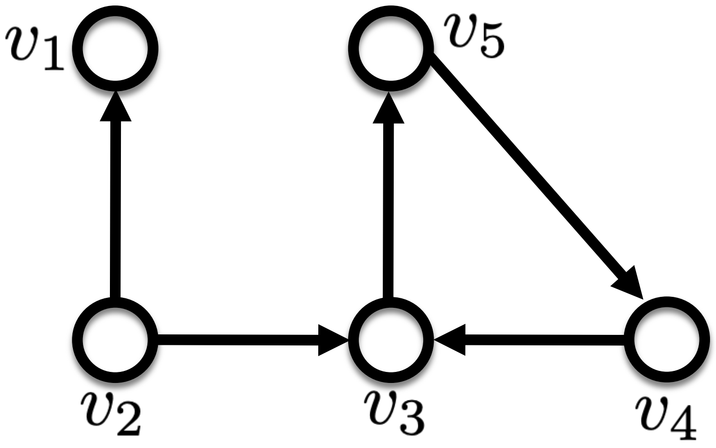
v_2 is adjacent to v_1 and v_3.
v_3 is a neighbor of v_5
but is NOT adjacent to v_4 and v_2

Definitions: Degree

- **Undirected graphs:**
  - The **degree** d_i of a node counts the neighbors $|\mathcal{N}_i|$ of the node
- **Directed graphs:**
  - The **in-degree** d_i^{in} of a node counts its neighbors $|\mathcal{N}_i|$ (number of edges coming *in* the node)
  - The **out-degree** d_i^{out} counts the number of edges coming *out* of the node



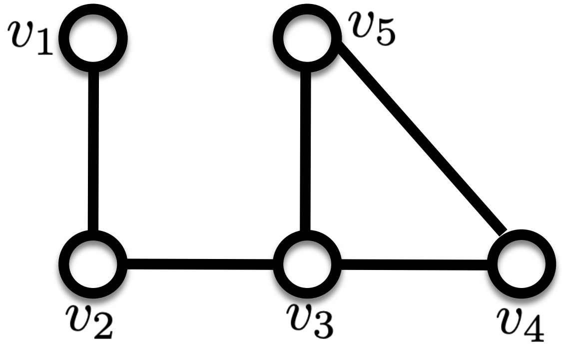
v_1 has degree 1 and v_3 has degree 3



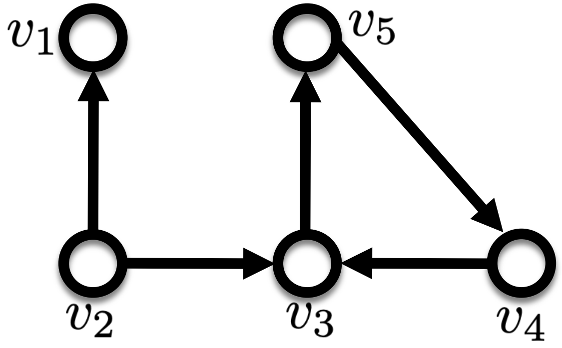
v_1 has in-degree $d_1^{in} = 1$ and out-degree $d_1^{out} = 0$,
v_2 has $d_2^{in} = 0$ and $d_2^{out} = 2$ and
v_3 has $d_3^{in} = 2$ and $d_3^{out} = 1$

Definitions: Path, Cycle

- A **path*** is a sequence of distinct vertices $v_{i_0} v_{i_1} \ldots v_{i_m}$ such that the vertex v_{i_k} is adjacent to $v_{i_{k+1}}$
- A **cycle**** is a sequence of vertices, such that ***only*** the first and last vertex are equal.



$v_1 v_2 v_3 v_4$ is a path
$v_3 v_5 v_4 v_3$ is a cycle



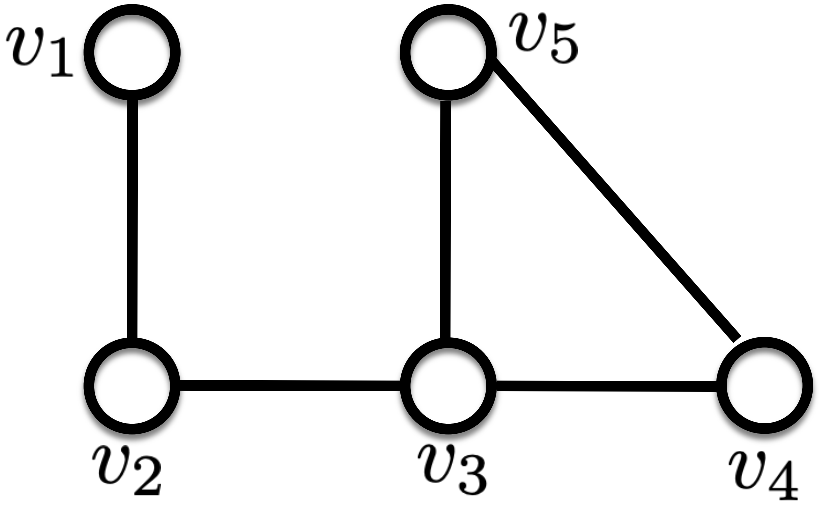
$v_2 v_3 v_5 v_4$ is a path
$v_3 v_5 v_4 v_3$ is a cycle

*Walk, Trail, vs. Path: Walks can repeat both vertices and edges. Trails can repeat vertices but not edges.

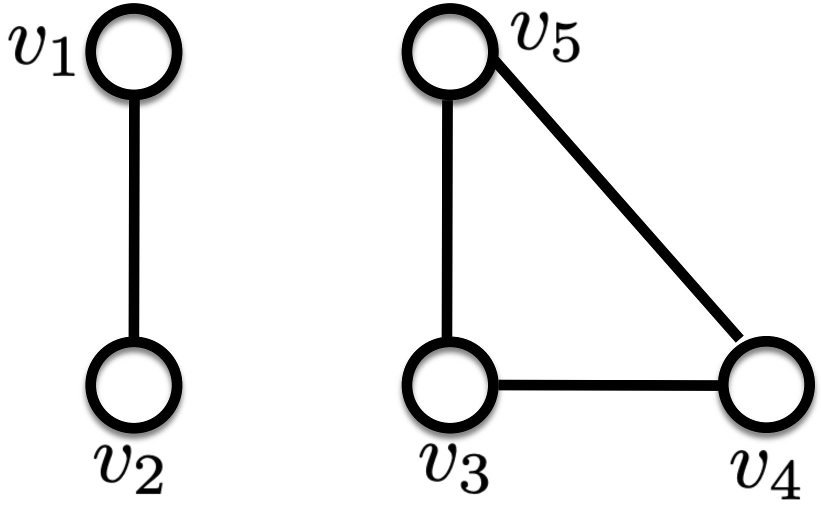
**Circuit vs Cycle. A circuit is any walk where the first and last vertex are equal. Note that a cycle is a trail, but not a path!

Definition: Connectedness

- An undirected graph is **connected** if there exists a path from any vertex to any other vertex (i.e., all vertices can be reached from any vertex)



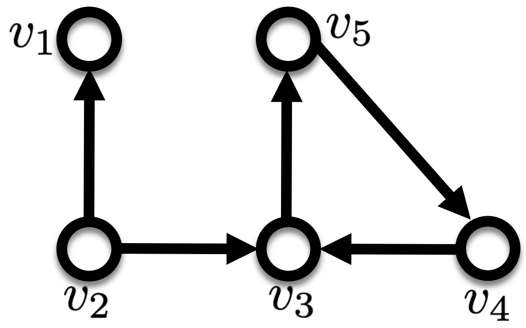
Connected



Disconnected

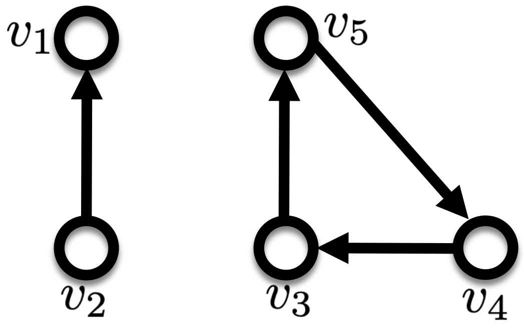
Definition: Connectedness

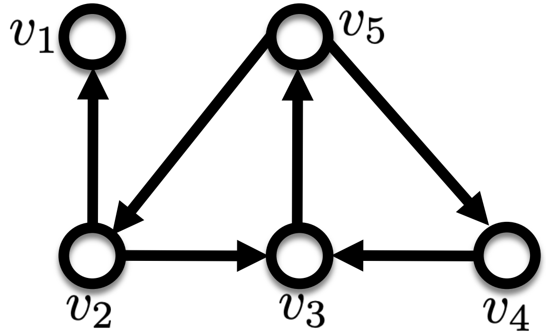
- A directed graph is **strongly connected** if there exists a **directed path** from any vertex to any other vertex
- A directed graph is **weakly connected** if there exists a path from any vertex to any other vertex when the graph is viewed as undirected



Weakly Connected

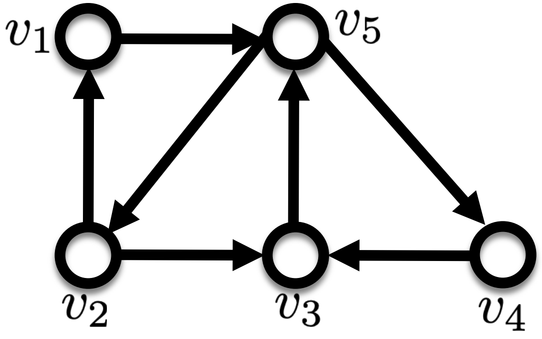
Disconnected





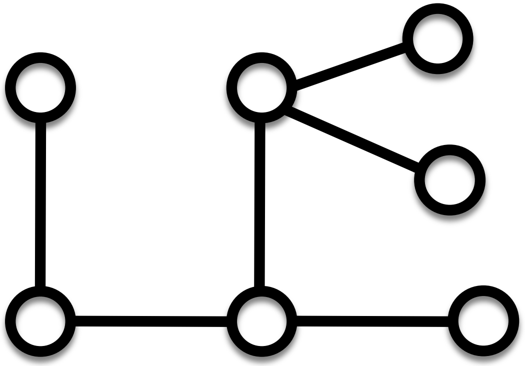
Weakly Connected

Strongly Connected

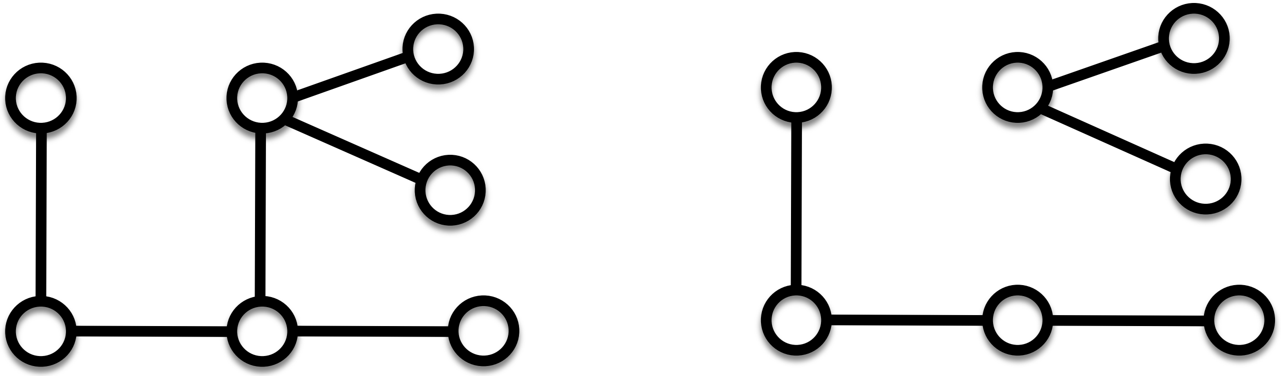


Special graphs

- A **tree** is a connected graph without any cycles

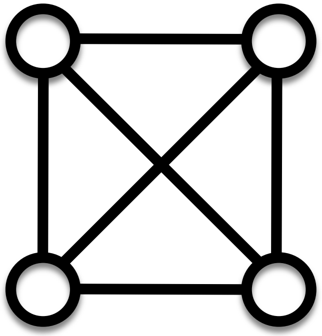


- A **forest** is a graph without any cycles
    - Not necessarily connected!



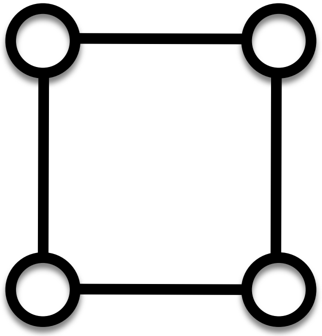
Special graphs

- The **complete graph** over n vertices K_n is the graph where every vertex is adjacent to every other vertex

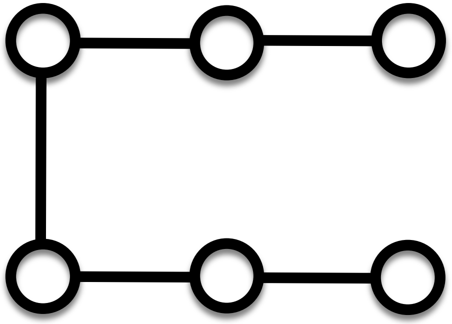


K_4

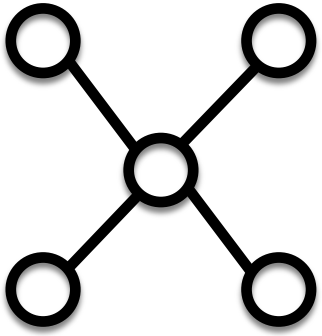
- The **cycle graph** C_4



- The **path graph** P_6



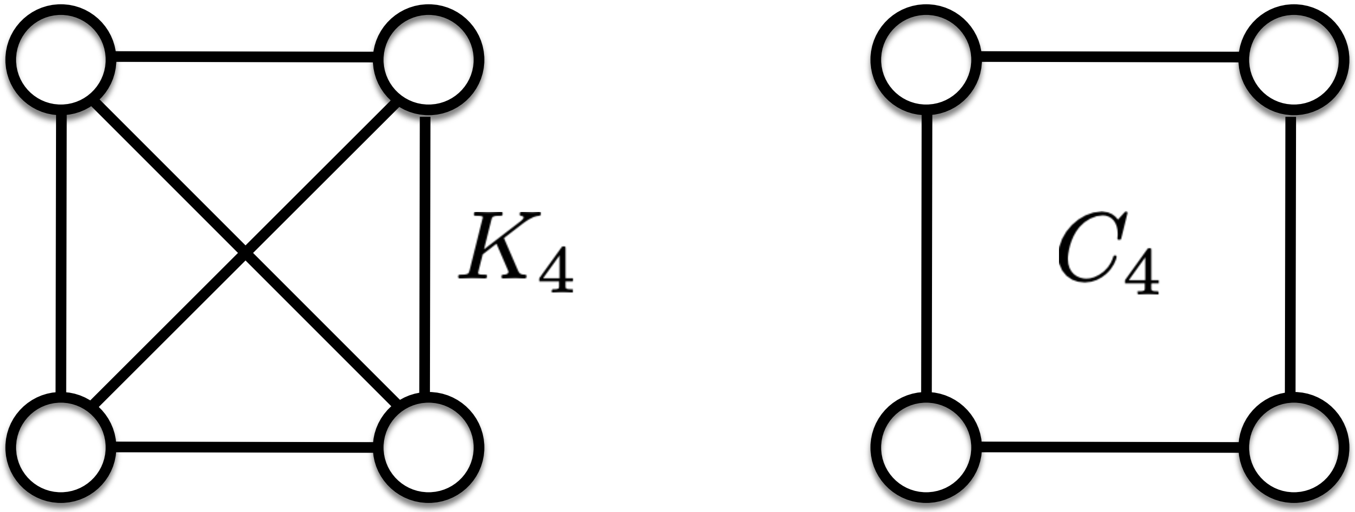
- The **star graph** S_5



Special graphs

- A ***k*-regular graph** is a graph where each vertex has degree k

K_n is (n-1)-regular and C_n is 2-regular



Algebraic Graph Theory

- Describe graphs with matrices
    - ***Matrix* properties** are associated to ***graph* properties**

Adjacency matrix A

$$A \in \mathbb{R}^{n \times n} = A_{ij} \begin{cases} 1 \text{ if } (v_j, v_i) \in \mathcal{E} \\ 0 \text{ if} (v_j, v_i) \notin \mathcal{E} \end{cases}$$

- **Properties:**
  - $A_{ii} = 0$
  - For undirected graphs, A is symmetric : $A_{ij} = A_{ji}$, $A = A^T$
  - For directed graphs, in general: $A \neq A^T$

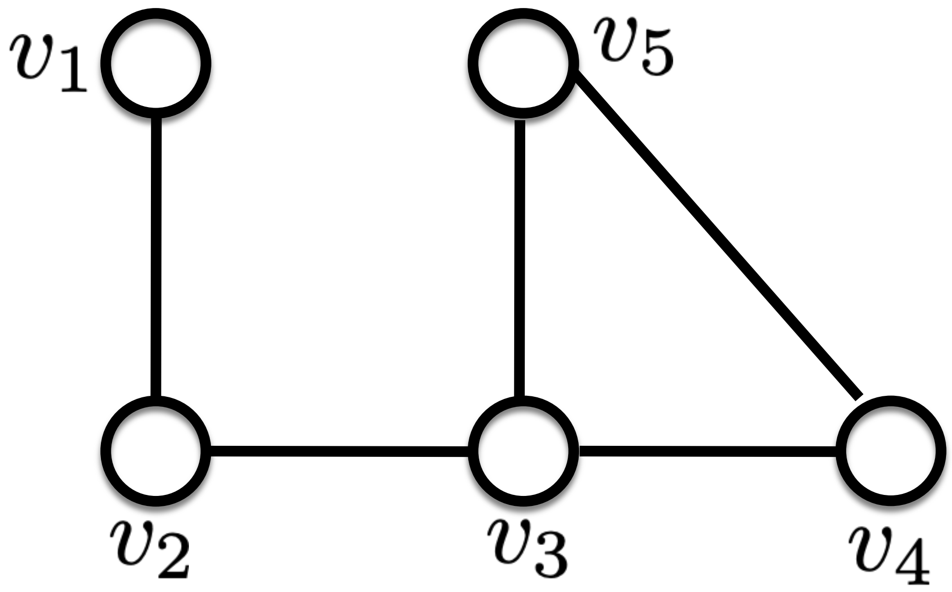
Adjacency matrix A

$$A \in \mathbb{R}^{n\times n} = A_{ij} \begin{cases} 1 \text{ if } (v_j, v_i) \in \mathcal{E} \\ 0 \text{ if} (v_j, v_i) \notin \mathcal{E} \end{cases}$$

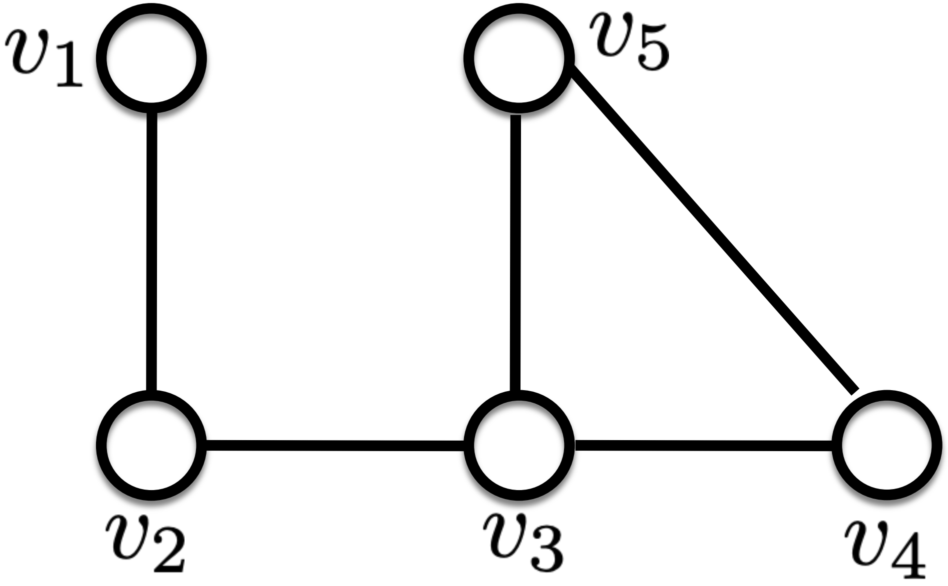
- **Properties:**
  - $A_{ii} = 0$
  - For undirected graphs, A is symmetric : $A_{ij} = A_{ji}$, $A = A^T$
  - For directed graphs, in general: $A \neq A^T$

Note: we use the convention described in the book (Mesbahi and Egerstedt "Graph theoretic methods in multi-agent systems") to define the adjacency matrix. Be aware that some books use other conventions. The chosen convention will be useful to define control laws later

Adjacency Matrix: Example



Adjacency Matrix: Example



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

Degree Matrix

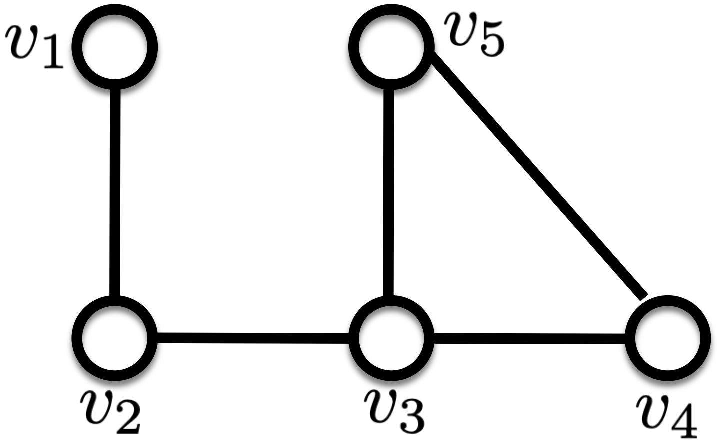
- The **degree matrix** $\Delta \in \mathbb{R}^{n \times n}$ is a diagonal matrix with the node degrees d_i as diagonal elements

$$\Delta = \mathrm{diag}(d_i) = \mathrm{diag}\left(\sum_{j=1}^{N} A_{ij}\right)$$

Degree Matrix

- The **degree matrix** $\Delta \in \mathbb{R}^{n \times n}$ is a diagonal matrix with the node degrees d_i as diagonal elements

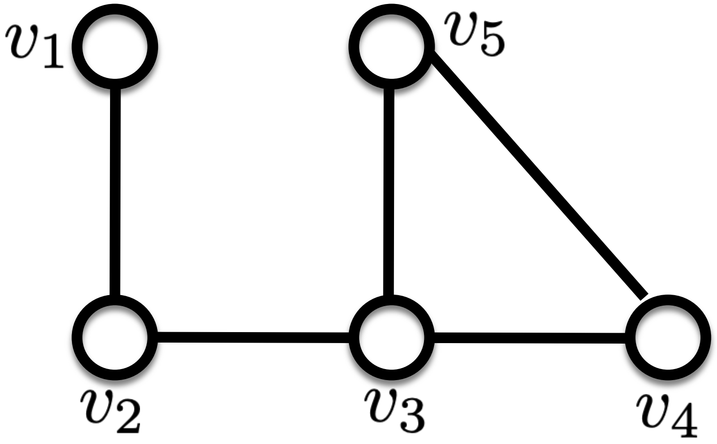
$$\Delta = \mathrm{diag}(d_i) = \mathrm{diag}\left(\sum_{j=1}^{N} A_{ij}\right)$$



Degree Matrix

- The **degree matrix** $\Delta \in \mathbb{R}^{n \times n}$ is a diagonal matrix with the node degrees d_i as diagonal elements

$$\Delta = \mathrm{diag}(d_i) = \mathrm{diag}\left(\sum_{j=1}^{N} A_{ij}\right)$$

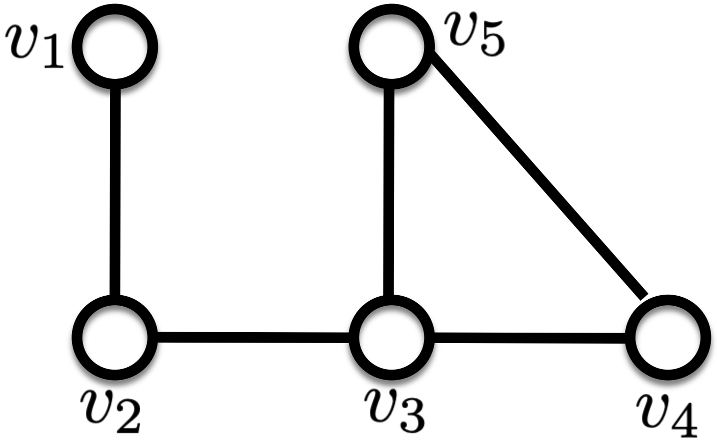


$$\Delta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Degree Matrix

- The **degree matrix** $\Delta \in \mathbb{R}^{n \times n}$ is a diagonal matrix with the node degrees d_i as diagonal elements

$$\Delta = \mathrm{diag}(d_i) = \mathrm{diag}\left(\sum_{j=1}^{N} A_{ij}\right)$$



$$\Delta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

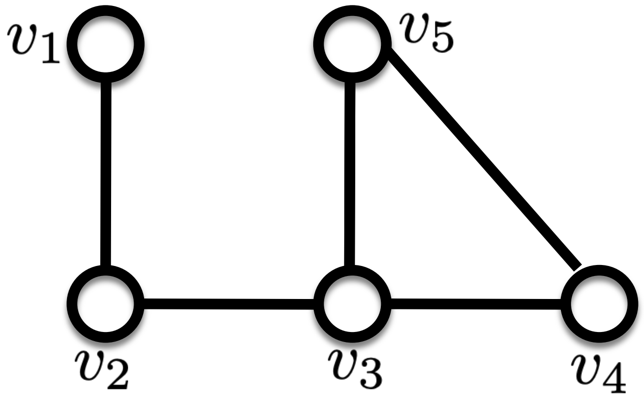
- For directed graphs, we will use the in-degree for the diagonal elements

Incidence matrix

- The **incidence matrix** $E \in \mathbb{R}^{N \times |\mathcal{E}|}$ encodes the incidence relationship between edges and vertices

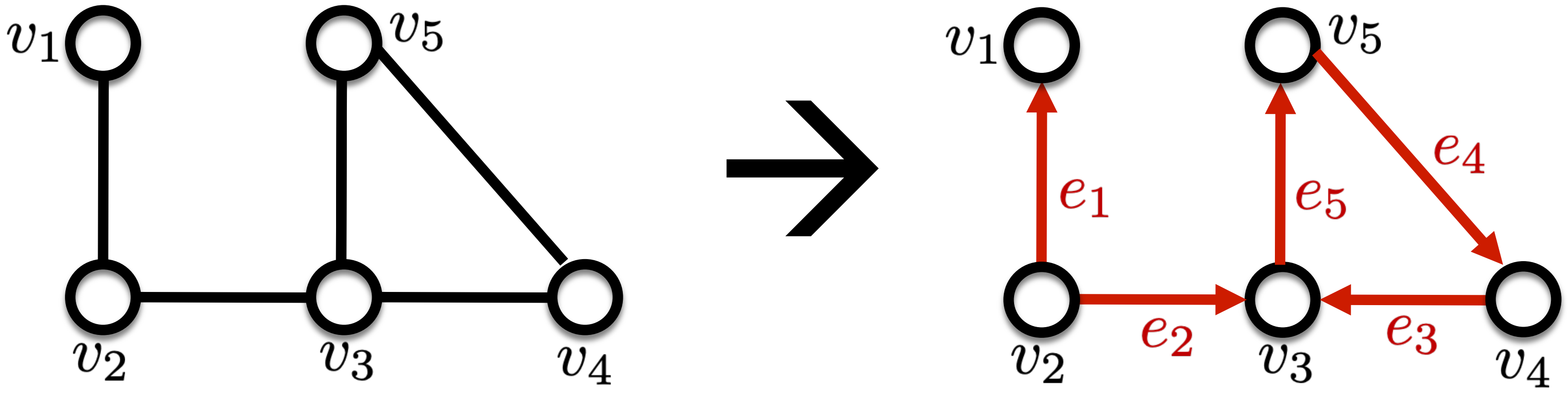
Incidence matrix

- The **incidence matrix** $E \in \mathbb{R}^{N \times |\mathcal{E}|}$ encodes the incidence relationship between edges and vertices



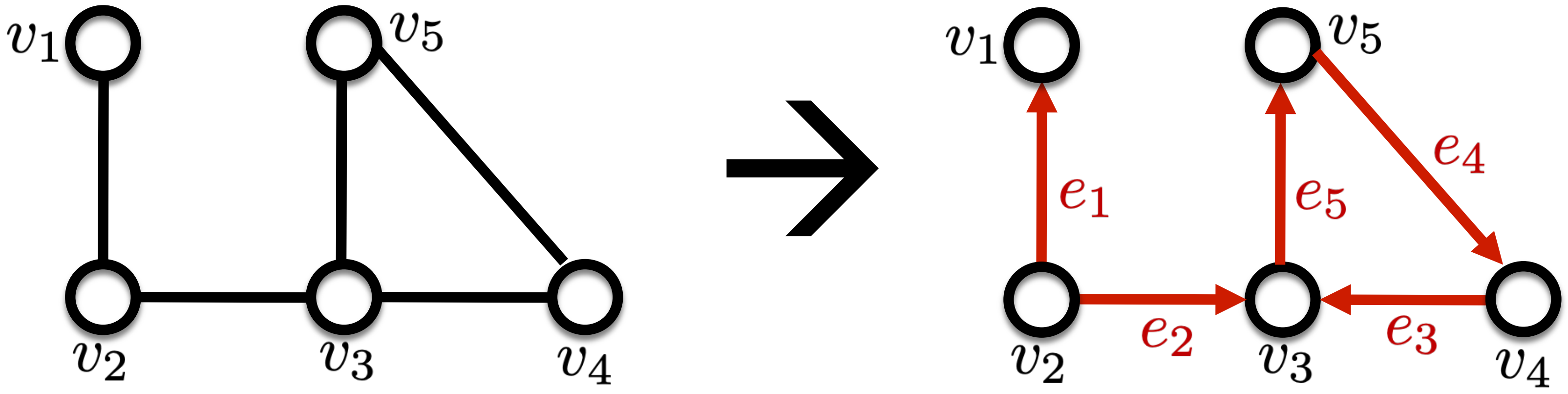
Incidence matrix

- The **incidence matrix** $E \in \mathbb{R}^{N \times |\mathcal{E}|}$ encodes the incidence relationship between edges and vertices
- Assign arbitrary orientation (direction) to all edges and an arbitrary labeling



Incidence matrix

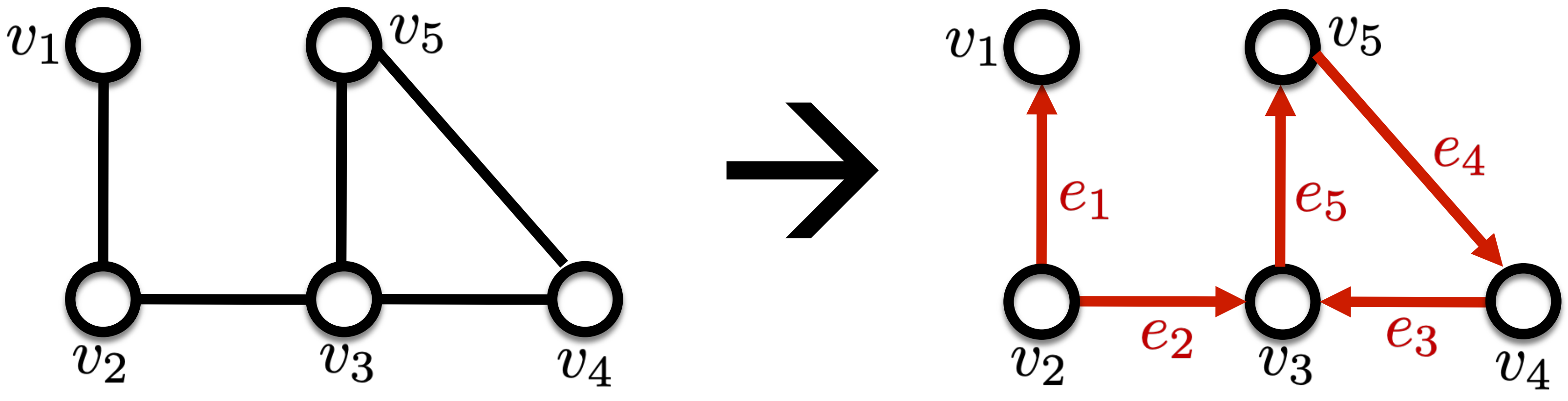
- The **incidence matrix** $E \in \mathbb{R}^{N\times|\mathcal{E}|}$ encodes the incidence relationship between edges and vertices
- Assign arbitrary orientation (direction) to all edges and an arbitrary labeling



$$E_{ij} \begin{cases} -1 \text{ if vertex } v_i \text{ is the tail of edge } e_j \\ 1 \text{ if vertex } v_i \text{ is the head of edge } e_j \\ 0 \text{ otherwise} \end{cases}$$

Incidence matrix

- The **incidence matrix** $E \in \mathbb{R}^{N \times |\mathcal{E}|}$ encodes the incidence relationship between edges and vertices
- Assign arbitrary orientation (direction) to all edges and an arbitrary labeling



$$E_{ij} \begin{cases} -1 \text{ if vertex } v_i \text{ is the tail of edge } e_j \\ 1 \text{ if vertex } v_i \text{ is the head of edge } e_j \\ 0 \text{ otherwise} \end{cases}$$

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

Incidence matrix

- We will only use this for undirected graphs!

Incidence matrix

- We will only use this for undirected graphs!
- **Why are we doing this?**

Laplacian matrix

- The **Laplacian matrix** L is the difference between the **degree matrix** and the **adjacency matrix**
  - **For undirected graphs:**

$$L \in \mathbb{R}^{n \times n} = \Delta - A = EE^T$$

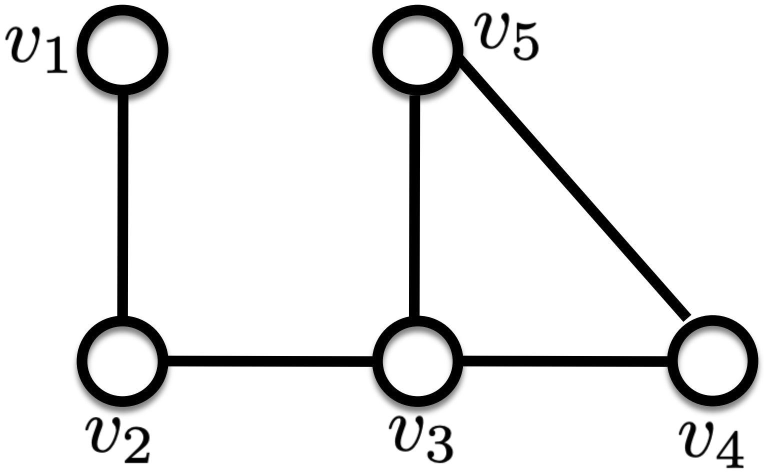
  - **For directed graphs:**

$$L \in \mathbb{R}^{n \times n} = \Delta - A$$

The incidence matrix is not used for directed graphs and the equality would not hold!

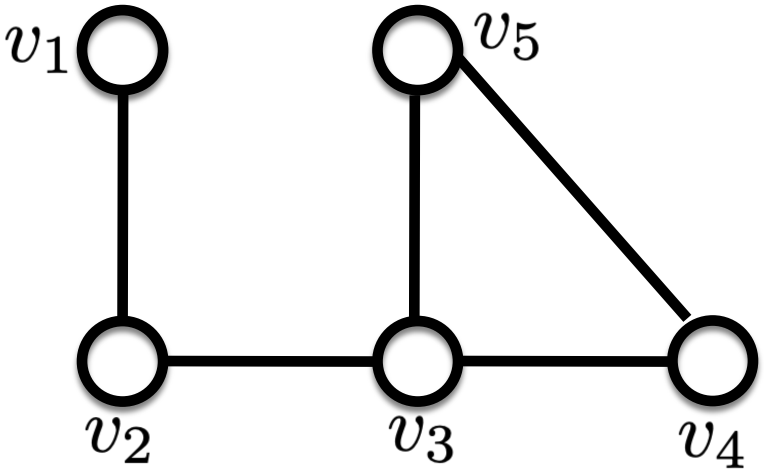
Laplacian matrix: Example

$$L \in \mathbb{R}^{n \times n} = \Delta - A = EE^T$$



Laplacian matrix: Example

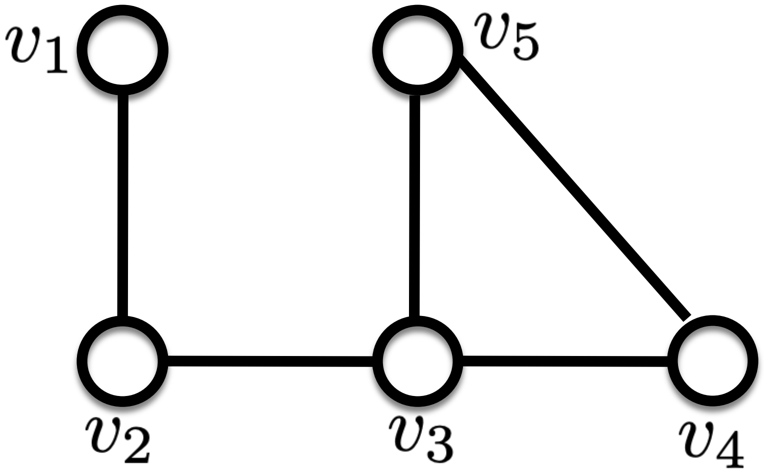
$$L \in \mathbb{R}^{n\times n} = \Delta - A = EE^T$$



$$\Delta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Laplacian matrix: Example

$$L \in \mathbb{R}^{n \times n} = \Delta - A = EE^T$$



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

Properties of Laplacian matrix (Undirected Graphs)

- The Laplacian matrix L is **symmetric** and **positive semi-definite**
  - all the eigenvalues of L are **real** and **non-negative**
- The sum of the columns and rows of L are 0
  - The vector of ones $\mathbf{1} = [1\ 1\ \dots\ 1]^T$ is an eigenvector of L and its associated eigenvalue is $\lambda = 0$.

$$L\mathbf{1} = \mathbf{0} \quad \text{(sum of columns is zero)}$$

  - Similarly, $\mathbf{1}^T$ is a left eigenvector of L
    - $\mathbf{1}^T L = \mathbf{0}^T$ (sum of rows is zero)

Properties of Laplacian matrix (Undirected Graphs)

- Number of 0 eigenvalues of L = number of connected components of the graph
  - Graph is connected if and only if it has only one 0 eigenvalue
- For connected graphs:
  - $\text{rank}(L) = n - 1$
  - Null space is spanned by the vector of ones $\mathbf{1}$

Properties of Laplacian matrix (Directed Graphs)

- The Laplacian matrix L is ***NEITHER*** **necessarily symmetric** ***NOR*** **positive semi-definite**
  - eigenvalues of L can be **complex** and/or have **non-negative real parts**
- The sum of the ***columns*** of L are 0 (but not the sum of the ***rows***)
  - The vector of ones $\mathbf{1} = [1\ 1\ \dots\ 1]^T$ is an eigenvector of L and its associated eigenvalue is $\lambda = 0$.

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

Why?