Lecture 4 - The algebra of graphs: The Incidence Matrix

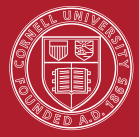
Graph-Based Data Science For Networked Systems
ECE 5260 – ORIE 5735

Anna Scaglione

Cornell Tech

February 4, 2026

Content

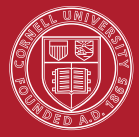


Cornell University

Outline

Cornell University

Outline



Cornell University

Important subsets

- In the last lecture we defined the following subsets of a graph, that are created connecting two nodes
  - **Walks** = arbitrary sequences of nodes and edges
  - **Trails** = walks that do not have repeated edges, and **circuits**= trails which start and end in the same node
  - **Paths** = Trails that do not repeat nodes and **cycles** = Trails that do not repeat nodes except for the initial one that is equal to the last one

Why do we care?

- **Random walks** are featured in many graph dynamics models (browsing history, epidemic models in biology and social science) and appear also algorithms to rank nodes
- **Paths and cycles** are prominent in graph analysis and in algorithms (shortest path, for routing through transportation or computer networks) and in the calculation of **spanning trees**

Why do we care?

- **Random walks** are featured in many graph dynamics models (browsing history, epidemic models in biology and social science) and appear also algorithms to rank nodes
- **Paths and cycles** are prominent in graph analysis and in algorithms (shortest path, for routing through transportation or computer networks) and in the calculation of **spanning trees**

Algebra of networks: adjacency matrix

- We also introduced the adjacency matrix
- The weighted adjacency matrix has elements:

$$A_{uv} = \begin{cases} w_{uv} & \text{if there is an edge from } v \text{ to } u \text{ with weight } w_{uv} \\ 0 & \text{else.} \end{cases}.$$

  The diagonal terms are zero if the network has no self-loops.
- For undirected we need to add $A_{uv} = A_{vu}$, i.e. $\boldsymbol{A} = \boldsymbol{A}^\top$.
- Recall that networkx `A= nx.adjacency_matrix(G)` is the transpose of my definition.

Using A to gain insights

- The main motivation is to have a matrix descriptions is that algebraic operations can reveal properties of the graph more easily than using the representation through a list
- For instance $[\boldsymbol{A}^k]_{uv}$ is the number of walks of length k connecting node u and v...
- At this regard also the Laplacian matrix $\boldsymbol{L} = \text{diag}(\boldsymbol{d}) - \boldsymbol{A}$ is perhaps even more useful (we will present more properties in the next lecture)

Using A to gain insights

- The main motivation is to have a matrix descriptions is that algebraic operations can reveal properties of the graph more easily than using the representation through a list
- For instance $[\boldsymbol{A}^k]_{uv}$ is the number of walks of length k connecting node u and v...
- At this regard also the Laplacian matrix $\boldsymbol{L} = \text{diag}(\boldsymbol{d}) - \boldsymbol{A}$ is perhaps even more useful (we will present more properties in the next lecture)

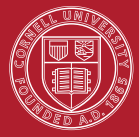
Using A to gain insights

- The main motivation is to have a matrix descriptions is that algebraic operations can reveal properties of the graph more easily than using the representation through a list
- For instance $[\boldsymbol{A}^k]_{uv}$ is the number of walks of length k connecting node u and v...
- At this regard also the Laplacian matrix $\boldsymbol{L} = \text{diag}(\boldsymbol{d}) - \boldsymbol{A}$ is perhaps even more useful (we will present more properties in the next lecture)

Outline

Cornell University

Outline



Cornell University

Graphs isomorphism

- These properties of $\boldsymbol{A}$ are apparent only if you order appropriately its nodes. If not, the corresponding permutation matrix $\boldsymbol{P}$ [1] will give you $\boldsymbol{A}_\pi = \boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^\top$ with the desired structure.
- **Graph Matching**: In general, two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic if and only if there exist a permulation matrix $\boldsymbol{P}$ such that:

$$\boldsymbol{A}_2 = \boldsymbol{P}\boldsymbol{A}_1\boldsymbol{P}^\top$$

- The solution may not be unique as the graphs could contains symmetric structures.

Many **necessary** conditions (not sufficient) are easy to check $|\mathcal{V}_1| = |\mathcal{V}_2|$, $|\mathcal{E}_1| = |\mathcal{E}_2|$, or the fact that $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ have the same eigenvalues (are isospectral), etc.

[1] A permutation matrix $\boldsymbol{P}$ has columns with a single entry equal to 1 and the rest zero. The matrix is also orthogonal $\boldsymbol{P}\boldsymbol{P}^\top = \boldsymbol{P}^\top\boldsymbol{P} = \boldsymbol{I}$.

Graphs isomorphism

- These properties of $\boldsymbol{A}$ are apparent only if you order appropriately its nodes. If not, the corresponding permutation matrix $\boldsymbol{P}$ [1] will give you $\boldsymbol{A}_\pi = \boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^\top$ with the desired structure.
- **Graph Matching**: In general, two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic if and only if there exist a permulation matrix $\boldsymbol{P}$ such that:

$$\boldsymbol{A}_2 = \boldsymbol{P}\boldsymbol{A}_1\boldsymbol{P}^\top$$

- The solution may not be unique as the graphs could contains symmetric structures.

Many **necessary** conditions (not sufficient) are easy to check $|\mathcal{V}_1| = |\mathcal{V}_2|$, $|\mathcal{E}_1| = |\mathcal{E}_2|$, or the fact that $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ have the same eigenvalues (are isospectral), etc.

[1] A permutation matrix $\boldsymbol{P}$ has columns with a single entry equal to 1 and the rest zero. The matrix is also orthogonal $\boldsymbol{P}\boldsymbol{P}^\top = \boldsymbol{P}^\top\boldsymbol{P} = \boldsymbol{I}$.

Graphs isomorphism

- These properties of $\boldsymbol{A}$ are apparent only if you order appropriately its nodes. If not, the corresponding permutation matrix $\boldsymbol{P}$ [1] will give you $\boldsymbol{A}_\pi = \boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^\top$ with the desired structure.
- **Graph Matching**: In general, two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic if and only if there exist a permulation matrix $\boldsymbol{P}$ such that:

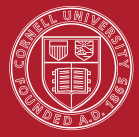
$$\boldsymbol{A}_2 = \boldsymbol{P}\boldsymbol{A}_1\boldsymbol{P}^\top$$

- The solution may not be unique as the graphs could contains symmetric structures.

Many **necessary** conditions (not sufficient) are easy to check $|\mathcal{V}_1| = |\mathcal{V}_2|$, $|\mathcal{E}_1| = |\mathcal{E}_2|$, or the fact that $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ have the same eigenvalues (are isospectral), etc.

[1] A permutation matrix $\boldsymbol{P}$ has columns with a single entry equal to 1 and the rest zero. The matrix is also orthogonal $\boldsymbol{P}\boldsymbol{P}^\top = \boldsymbol{P}^\top\boldsymbol{P} = \boldsymbol{I}$.

Graphs isomorphism



- As we mentioned, the set of permutation matrices $\mathcal{P}_n$ is a subset of the set of orthogonal matrices $\mathcal{O}_n$ and the right $\boldsymbol{P}$ is one the possible solution of the *relaxed* problem:

$$\min_{\boldsymbol{P}\in\mathcal{O}_n} \|\boldsymbol{A}_2 - \boldsymbol{P}\boldsymbol{A}_1\boldsymbol{P}^\top\|^2$$

- There is a theorem that shows that all solutions in $\mathcal{O}_n$ are:

$$\boldsymbol{P}^{\text{opt}} = \boldsymbol{V}_1\boldsymbol{S}\boldsymbol{V}_2^\top$$

  where $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ are the eigenvectors of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ respectively and $\boldsymbol{S} = \text{diag}(\pm 1, \ldots, \pm 1)$

- There are 2^N possible $\boldsymbol{S}$, so in general this fact does not make it easier to find $\boldsymbol{P}^{\text{opt}}$ that is a permutation than brute force search.

- But it does for the so called friendly graphs.

Spectral graph matching

- It can recover the right matching if the graphs are isomorphic and if the adjacency matrices $\boldsymbol{A}_1, \boldsymbol{A}_2$ have distinct eigenvalues.
- **Spectral matching**: Denote by $\overline{\boldsymbol{V}}^{(i)}$ the matrix containing the absolute value of the entries of $\boldsymbol{V}^{(i)}$, i.e. $[\overline{\boldsymbol{V}}^{(i)}]_{lk} = |v_{lk}|, \forall l, k$. The permutation matrix is:

$$\boldsymbol{P}^{\star\star} = \arg\max_{\boldsymbol{P} \in \mathcal{P}_n} \text{trace}(\boldsymbol{P}^T \overline{\boldsymbol{V}}^{(1)} (\overline{\boldsymbol{V}^{(2)}})^T).$$

- The beauty? Since $\overline{\boldsymbol{V}}^{(1)} (\overline{\boldsymbol{V}}^{(2)})^T \geq 0$ the maximization is a **linear assignment problem**, solvable with the Hungarian method by off-the-shelf solvers :).

Graphs with community structure

- Slightly weaker form of matching is subgraph isomorphism, that requires the existence of an isomorphism between one of the graphs and a subgraph of the other
- Application areas in which graph matching techniques have been successfully employed are:
  1. 2D and 3D image analysis and processing;
  2. Document processing;
  3. Biometric identification;
  4. Image databases;
  5. Video analysis;
  6. Biological and biomedical applications.

Graphs with community structure

- Slightly weaker form of matching is subgraph isomorphism, that requires the existence of an isomorphism between one of the graphs and a subgraph of the other
- Application areas in which graph matching techniques have been successfully employed are:
  1. 2D and 3D image analysis and processing;
  2. Document processing;
  3. Biometric identification;
  4. Image databases;
  5. Video analysis;
  6. Biological and biomedical applications.

Graphs with community structure

- Slightly weaker form of matching is subgraph isomorphism, that requires the existence of an isomorphism between one of the graphs and a subgraph of the other
- Application areas in which graph matching techniques have been successfully employed are:
  1. 2D and 3D image analysis and processing;
  2. Document processing;
  3. Biometric identification;
  4. Image databases;
  5. Video analysis;
  6. Biological and biomedical applications.

Graphs with community structure

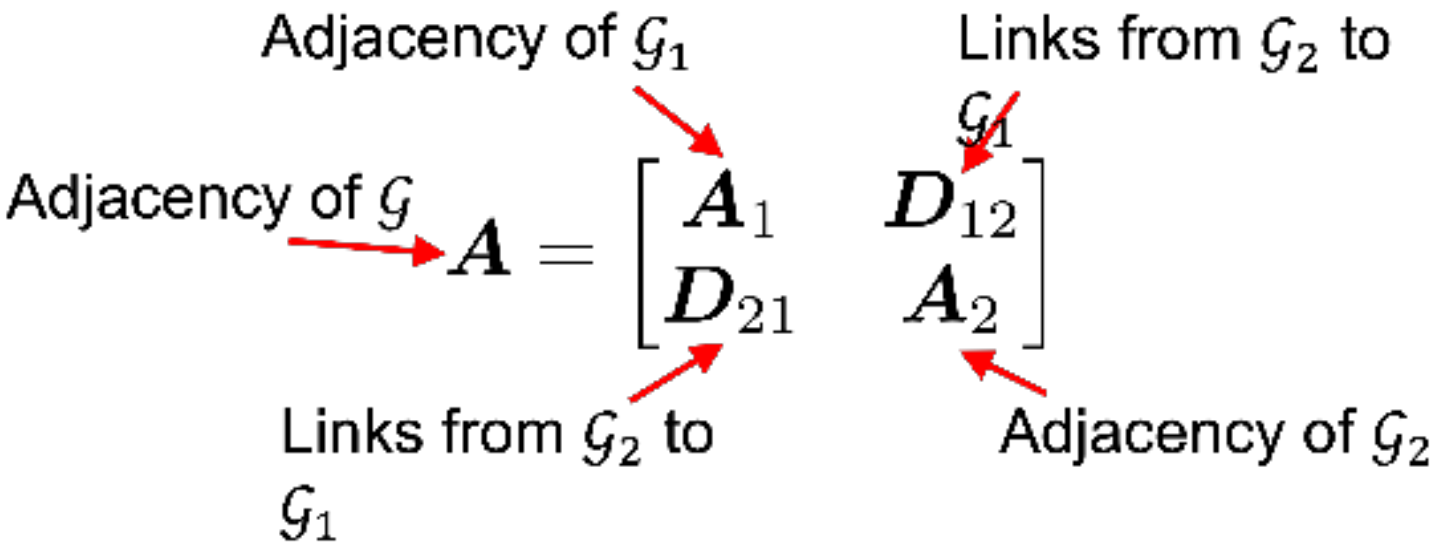
Cornell University

- In general one can unite graphs by adding some nodes that connect them:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}): \quad \mathcal{V} = \mathcal{V}_\infty \cup \mathcal{V}\in, \quad \mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_{12} \cup \mathcal{E}_{21}$$

  where for undirected graphs $\mathcal{E}_{12} = \mathcal{E}_{21}$. This can be done combining more than two graphs of course.

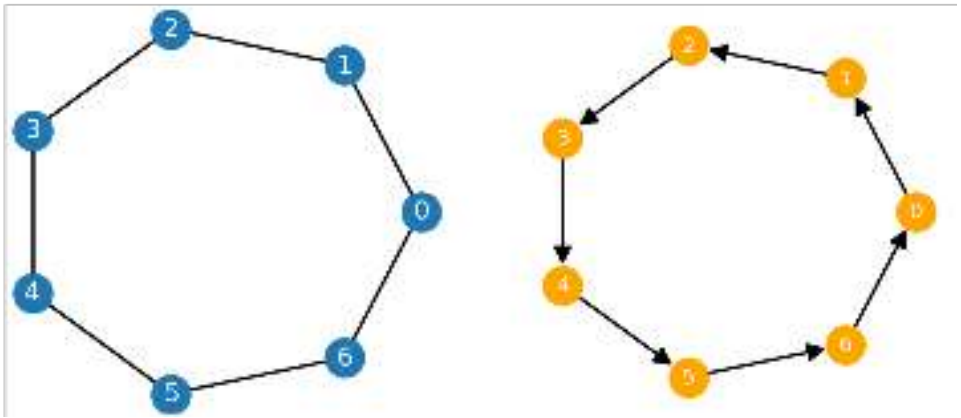
- We can look at the adjacency matrix in a block form:



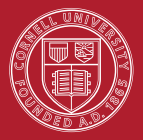
Adjacency matrix for cycle graphs

- The adjacency matrix for cycle graphs, directed and undirected, are Toeplitz matrices (constant across diagonals) and circulant.
- In general graphs that have a Toeplitz circulant structure (with more off diagonals non-zero) have circular symmetry,

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

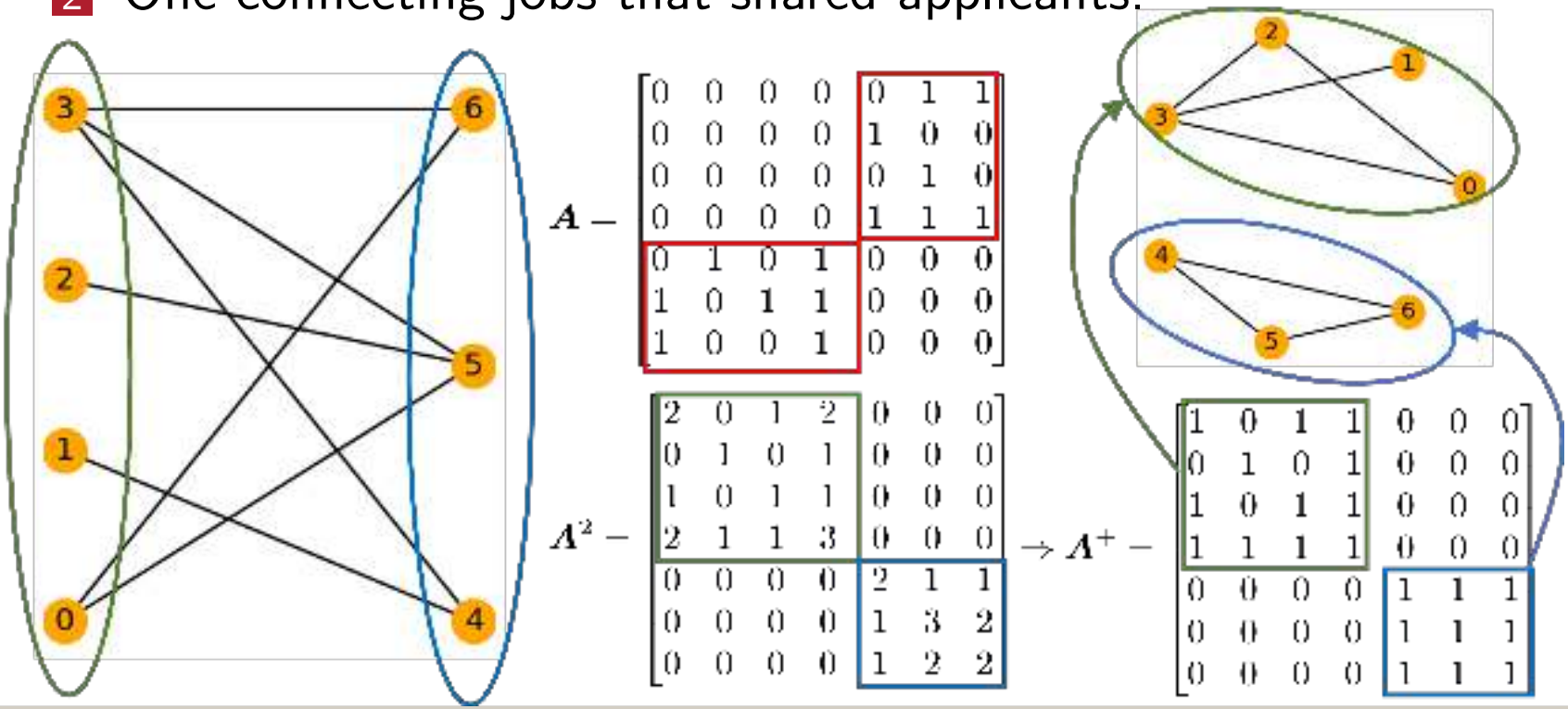


One-mode projection of bipartite graphs



Cornell University

- Consider the use of bipartite graphs to represent group membership (person - job application)
- We can use the adjacency matrix to perform a network one-mode projection to form two derived graphs:
  1. One connecting people that have applied for the same job.
  2. One connecting jobs that shared applicants.



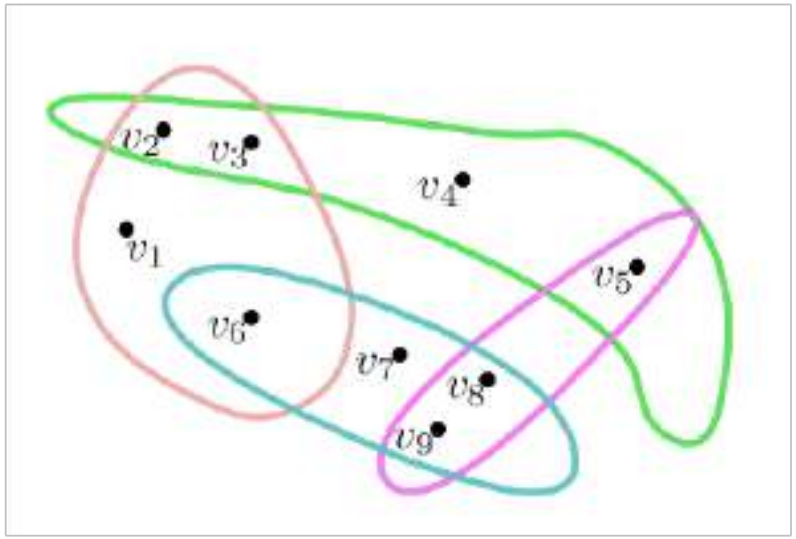
Hypergraphs?

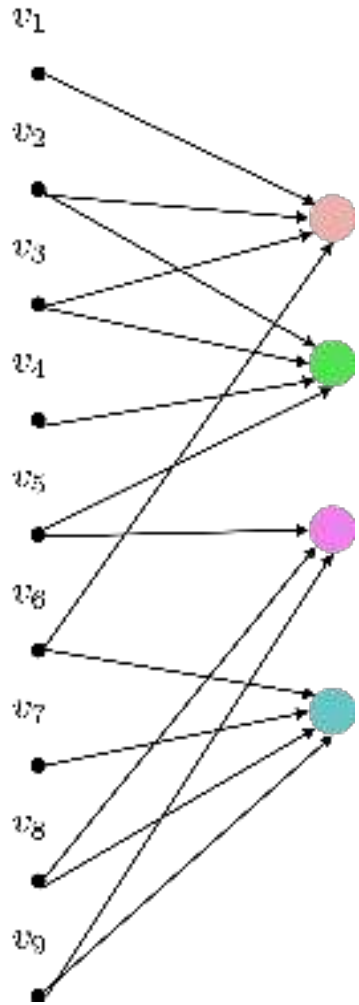
- Hyper-graphs are $\mathcal{H}(\mathcal{V}, \{\mathcal{E}_1, \ldots, \mathcal{E}_n\})$ where each $\mathcal{E}_e \subseteq \mathcal{V}$. An adjacency matrix for $\mathcal{A}$ could indicate if $\mathcal{E}_e \cap \mathcal{E}_{e'} \neq \emptyset$. but is not used, as it hides what is the specific intersection.
- Nonetheless, one defines the neighborhood of a certain node v $\mathcal{N}_v$ and the node degree as follows:

$$\mathcal{N}_v = \{\mathcal{E}_e | \mathcal{E}_e \text{ is in } \mathcal{H}, v \in \mathcal{E}_e\}, \quad d_v = \text{number of hyperedges in } \mathcal{N}_v.$$

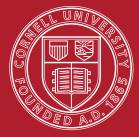
Hypergraphs as Bipartite Graphs

- Hypergraphs can describe relationships like group membership.
- However, to process the data one can map the hypergraph into a bipartite graph.





Outline



Cornell University

Product Graphs

Product Graph

The general definition of graph product of two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ is that of a function that takes two graphs $\mathcal{H} = H(\mathcal{G}_1, \mathcal{G}_2)$ such that $\mathcal{H} = (\mathcal{V}, \mathcal{E})$:

1. $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$, i.e. is the Cartesian product of the node sets
2. A specific condition has to be met for the existence of an edge between two pairs (v_1, v_2) and (u_1, u_2) in $\mathcal{V}_1 \times \mathcal{V}_2$.

Some options for product graphs

- Different options are, for instance:
  - Cartesian or Box product: (v_1, v_2) and (u_1, u_2) have a node in common and the other two nodes have an edge. Either $v_1 = u_1$, and $(u_2, v_2) \in \mathcal{E}_2$ or $v_2 = u_2$ and $(u_1, v_1) \in \mathcal{E}_1$. Then: $|\mathcal{E}| = |\mathcal{V}_1||\mathcal{E}_1| + |\mathcal{V}_2||\mathcal{E}_2|$
  - Tensor or Kronecker product: the node pairs (v_1, v_2) and (u_1, u_2) are adjacent in the tensor product of graphs iff: 1) $(v_1, u_1) \in \mathcal{E}_1$ and 2) $(v_2, u_2) \in \mathcal{E}_2$. Here, $|\mathcal{E}| = 2|\mathcal{E}_1||\mathcal{E}_2|$

  and others....
- A modeling trick: replace time-varying graphs with a single static graph in a higher-dimensional space.

Product graphs adjacency matrix

- Let $\boldsymbol{A} \otimes \boldsymbol{B}$ denote the Kronecker product of two matrices, i.e.:

$$\boldsymbol{A} \otimes \boldsymbol{B} := \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{n1}\boldsymbol{B} & \dots & a_{nn}\boldsymbol{B} \end{bmatrix}$$

- The adjacency matrix of the **box product** $\mathcal{G}_1 \boxdot \mathcal{G}_2$ is:

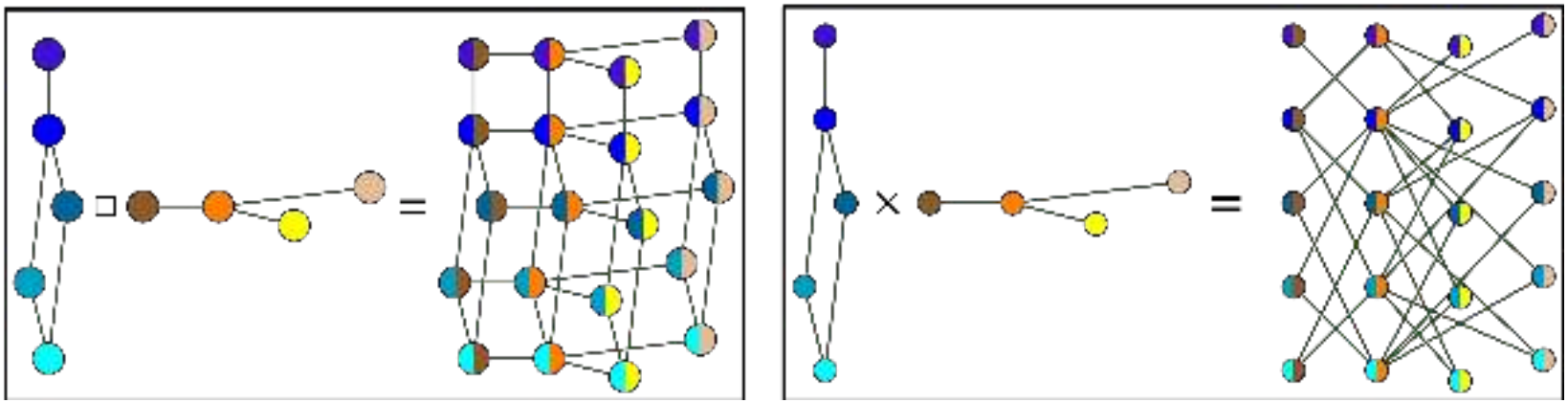
$$\boldsymbol{A}_{1\boxdot 2} = \boldsymbol{A}_1 \otimes \boldsymbol{I}_{|\mathcal{V}_2|} + \boldsymbol{I}_{|\mathcal{V}_1|} \otimes \boldsymbol{A}_2$$

  where $\boldsymbol{I}_N$ is the identity matrix of size N.

- The adjacency matrix of the **tensor product** $\mathcal{G}_1 \times \mathcal{G}_2$ is:

$$\boldsymbol{A}_{1\times 2} = \boldsymbol{A}_1 \otimes \boldsymbol{A}_2$$

Visualization of product graphs

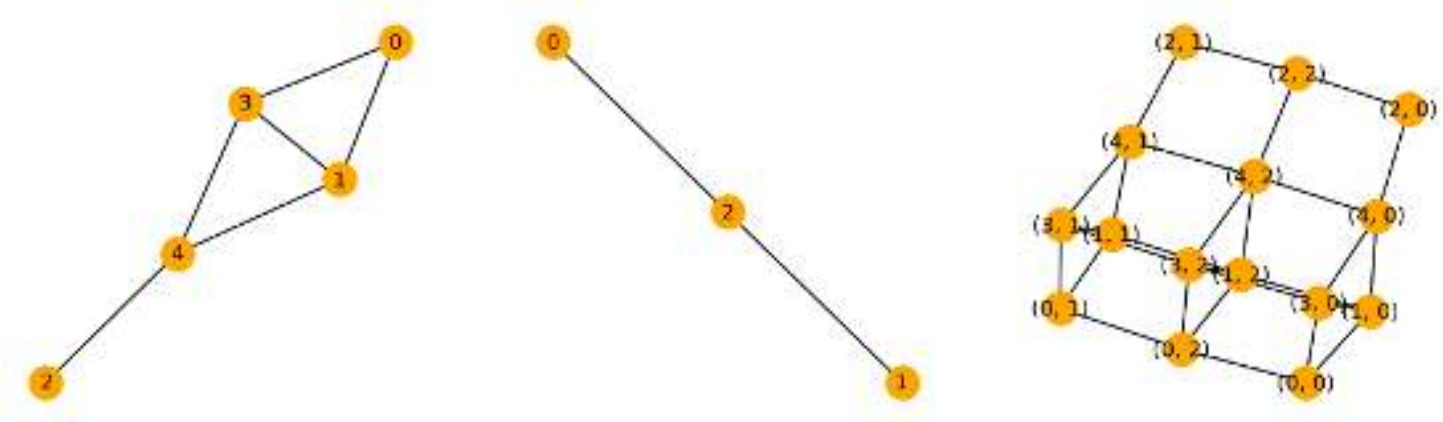


- A special box product is with the graph that represents time, which is like a line (an open cycle), because it can represent operations that are on neighborhoods and over time considering a single graph.
- The cartesian product of graphs is used in lieu of a multigraph to model graph-temporal processes with the same graph at each layer.

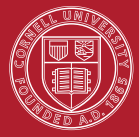
Product graphs adjacency matrix

Cornell University

- Let us denote a graph which is a line with k nodes by $\mathcal{L}_k$.
- $\mathcal{L}_k$ is the graph support of a time series of length k.
- The data points of a graph signal $(\boldsymbol{x}(0), \ldots, \boldsymbol{x}(k-1))$ with support $\mathcal{G}$ are graph signal with support $\mathcal{G} \boxdot \mathcal{L}_k$
- In fact, the graphical support for a time series is the line graph, and the network phenomenon is a signal unfolding on the product graph.



Outline

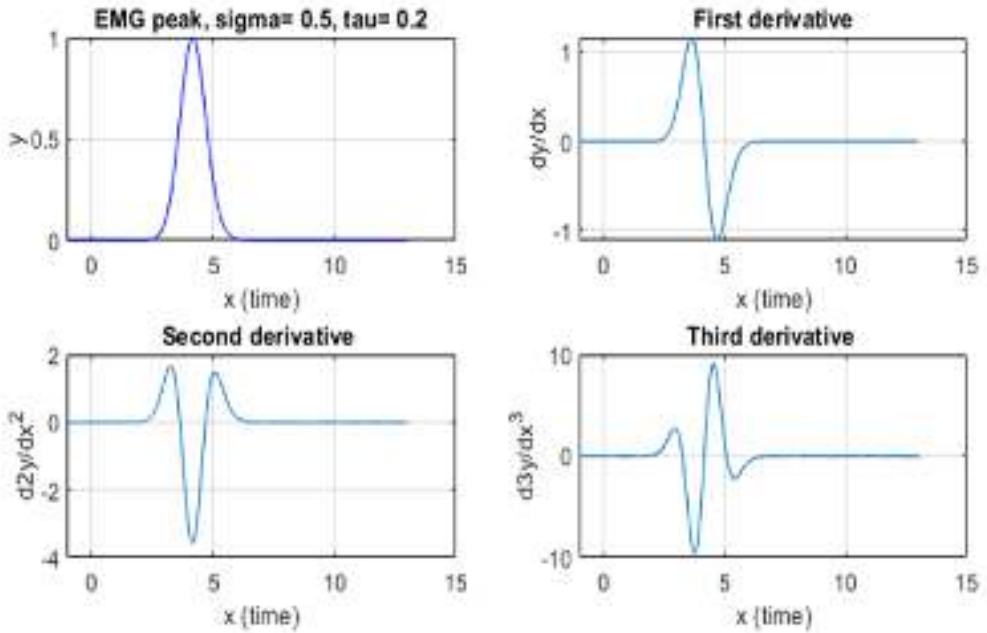


Cornell University

Generalizing the derivative of a signal for graphs

- One very useful notion in analyzing time-series is to have an estimate of the local *variation* of the signal.
- By measuring how locally different the signal is, we can infer how *variable it is*
- In continuous time we have the derivative:

$$\dot{x}(t) = \lim_{dt \to 0} \frac{x(t) - x(t - dt)}{dt}$$

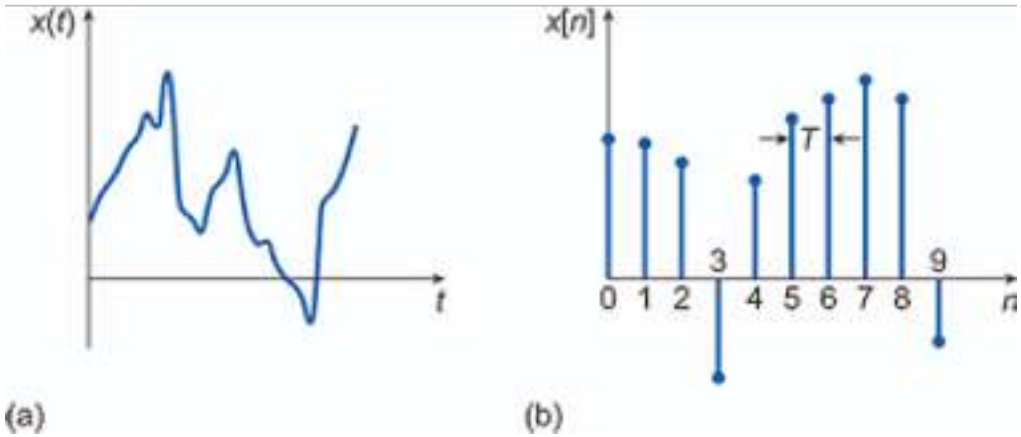


Discrete derivative

- In discrete time, we can use the finite difference signal:

$$dx(t) = x(t) - x(t-1), \quad \text{or} \quad dx(t) = (x(t) - x(t-1))w$$

if the spacing between samples is $T = 1/w$.



- You can view $\boldsymbol{x} = (x(0), \ldots, x(N-1))$ as the signal on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{0, \ldots, N-1\}$ and $\mathcal{E} = \{(0,1), (1,2), \ldots, (N-2, N-1)\}$.

Derivative operation in matrix form

Cornell University

- We can write the derivative operation on a finite sequence as the product of the vector $\boldsymbol{x} = [x(1), \ldots]$ as the product with the following matrix:

$$[x(1), x(2), \ldots, x(6), x(7)] \cdot \overbrace{\begin{bmatrix} 1. & 0. & 0. & 0. & 0. & 0. & 0. \\ -1. & 1. & 0. & 0. & 0. & 0. & 0. \\ 0. & -1. & 1. & 0. & 0. & 0. & 0. \\ 0. & 0. & -1. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & -1. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & -1. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & -1. & 1. \end{bmatrix}}^{\boldsymbol{B}}$$

$$= [x(1) - x(2), x(2) - x(3), \ldots, x(6) - x(7), x(7)]$$

- We can generalize $\boldsymbol{B}$ for an arbitrary graph: it is called the **incidence matrix**

Incidence matrix

- Let $e = 1, \ldots, |\mathcal{E}|$ be the index of an edge $(u, v) \in \mathcal{E}$.
- The incidence matrix $\boldsymbol{B}$ has $|\mathcal{V}|$ rows and $|\mathcal{E}|$ columns and, for a weighted directed graph, is defined as:

$$B_{ue} = \begin{cases} w_e & \text{if } e = (u, v) \in \mathcal{E}, \text{ i.e. enters node } u \\ -w_e & \text{if } e = (k, u) \in \mathcal{E}, \text{ i.e. leaves node } u \\ 0 & \text{else.} \end{cases}$$

- For an unweighted graph $w_e = 1, \forall e \in \mathcal{E}$.
- Each column of B has only two non-zero elements, one the complement of the other.
- If there are no isolated node $\mathbf{1}^\top B = \mathbf{0}$
- Multiplying by -1 a column of B reverses the edge direction.

Incidence matrix

- Let $e = 1, \ldots, |\mathcal{E}|$ be the index of an edge $(u, v) \in \mathcal{E}$.
- The incidence matrix $\boldsymbol{B}$ has $|\mathcal{V}|$ rows and $|\mathcal{E}|$ columns and, for a weighted directed graph, is defined as:

$$B_{ue} = \begin{cases} w_e & \text{if } e = (u, v) \in \mathcal{E}, \text{ i.e. enters node } u \\ -w_e & \text{if } e = (k, u) \in \mathcal{E}, \text{ i.e. leaves node } u \\ 0 & \text{else.} \end{cases}$$

- For an unweighted graph $w_e = 1, \forall e \in \mathcal{E}$.
- Each column of $\boldsymbol{B}$ has only two non-zero elements, one the complement of the other.
- If there are no isolated node $\mathbf{1}^\top \boldsymbol{B} = \mathbf{0}$
- Multiplying by -1 a column of $\boldsymbol{B}$ reverses the edge direction.

Incidence matrix

- Let $e = 1, \ldots, |\mathcal{E}|$ be the index of an edge $(u, v) \in \mathcal{E}$.
- The incidence matrix $\boldsymbol{B}$ has $|\mathcal{V}|$ rows and $|\mathcal{E}|$ columns and, for a weighted directed graph, is defined as:

$$B_{ue} = \begin{cases} w_e & \text{if } e = (u, v) \in \mathcal{E}, \text{ i.e. enters node } u \\ -w_e & \text{if } e = (k, u) \in \mathcal{E}, \text{ i.e. leaves node } u \\ 0 & \text{else.} \end{cases}$$

- For an unweighted graph $w_e = 1, \forall e \in \mathcal{E}$.
- Each column of $\boldsymbol{B}$ has only two non-zero elements, one the complement of the other.
- If there are no isolated node $\mathbf{1}^\top \boldsymbol{B} = \mathbf{0}$
- Multiplying by -1 a column of $\boldsymbol{B}$ reverses the edge direction.

Incidence matrix

- Let $e = 1, \ldots, |\mathcal{E}|$ be the index of an edge $(u, v) \in \mathcal{E}$.
- The incidence matrix $\boldsymbol{B}$ has $|\mathcal{V}|$ rows and $|\mathcal{E}|$ columns and, for a weighted directed graph, is defined as:

$$B_{ue} = \begin{cases} w_e & \text{if } e = (u, v) \in \mathcal{E}, \text{ i.e. enters node } u \\ -w_e & \text{if } e = (k, u) \in \mathcal{E}, \text{ i.e. leaves node } u \\ 0 & \text{else.} \end{cases}$$

- For an unweighted graph $w_e = 1, \forall e \in \mathcal{E}$.
- Each column of $\boldsymbol{B}$ has only two non-zero elements, one the complement of the other.
- If there are no isolated node $\mathbf{1}^\top \boldsymbol{B} = \mathbf{0}$
- Multiplying by -1 a column of $\boldsymbol{B}$ reverses the edge direction.

Incidence matrix - oriented vs. unoriented

- We can use the same definition for an undirected graph, assigning an arbitrary direction. It is called the *oriented* incidence matrix.
- There is also an *unoriented* version, which instead of reversing the sign uses the same value.

$$B_{ue} = \begin{cases} 1 & \text{if } e = (u, v) \in \mathcal{E}, \text{ i.e. enters node } u \\ 0 & \text{else.} \end{cases}$$

In the following I will focus on the **oriented** incidence matrix definition, which is generally more useful.

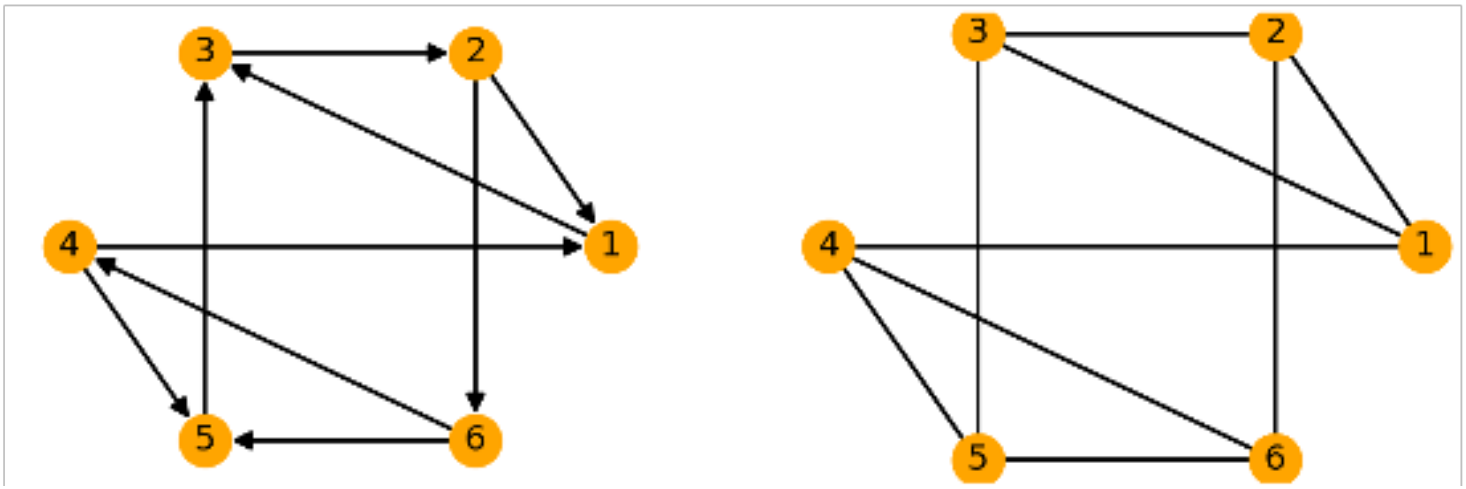
Incidence matrix - Visualization

Oriented

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

Unoriented

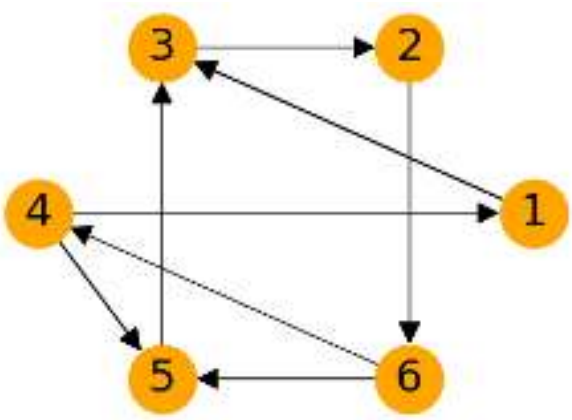
$$\begin{bmatrix} 1. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1. & 0. & 1. & 1. & 0. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. & 1. & 1. & 0. & 0. & 0. \\ 0. & 0. & 1. & 0. & 0. & 0. & 1. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. & 1. & 0. & 1. \\ 0. & 0. & 0. & 1. & 0. & 0. & 0. & 1. & 1. \end{bmatrix}$$



There are $|\mathcal{E}| = 9$ edges and $|\mathcal{V}| = 6$ nodes.

Incidence matrix - incorporating the weights

- Let $\boldsymbol{w}$ is the vector of all edge weights.
- Let diag$(\boldsymbol{w})$ the diagonal matrix with diagonal elements equal to $\boldsymbol{w}$ entries.
- It is often convenient to define $\boldsymbol{B}$ based on the topology, since the weighted version is simply $\boldsymbol{B}' = \boldsymbol{B}\text{diag}(\boldsymbol{w})$.



$$\boldsymbol{B} = \begin{bmatrix} -1.9 & 0. & 0. & 1.1 & 0. & 0. & 0. & 0. \\ 0. & -1.5 & 1.25 & 0. & 0. & 0. & 0. & 0. \\ 1.9 & 0. & -1.25 & 0. & 0. & 1.3 & 0. & 0. \\ 0. & 0. & 0. & -1.1 & -1.75 & 0. & 1.2 & 0. \\ 0. & 0. & 0. & 0. & 1.75 & -1.3 & 0. & 1.25 \\ 0. & 1.5 & 0. & 0. & 0. & 0. & -1.2 & -1.25 \end{bmatrix}$$

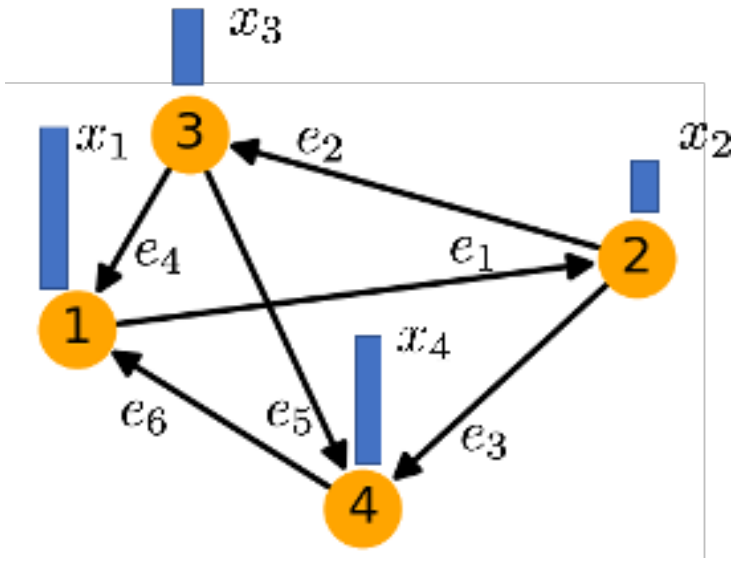
Interpretation

- Let $\boldsymbol{B}$ be the oriented weighted incidence matrix of $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and let $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{V}|}$ be a graph signal, i.e. a vector whose index set is $\mathcal{V}$.
- We can define the edge signal indexed by the set $\mathcal{E}$:

$$\boldsymbol{e} = \boldsymbol{B}^\top \boldsymbol{x}.$$

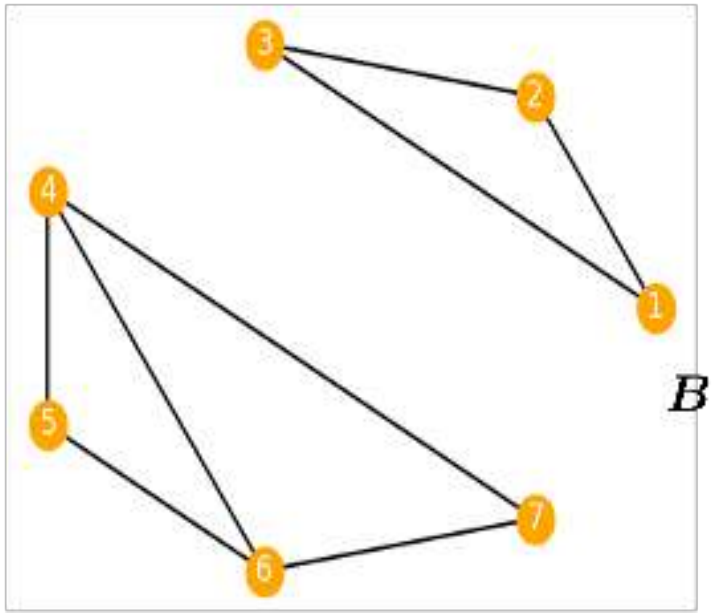
Taking "derivatives" on a graph?

The vector $\boldsymbol{e} = \boldsymbol{B}^\top \boldsymbol{x}$ contains the differences of all nodes values $(x_v - x_u)$ $\forall (u, v) \in \mathcal{E}$.



Incidence matrix - disconnected graphs

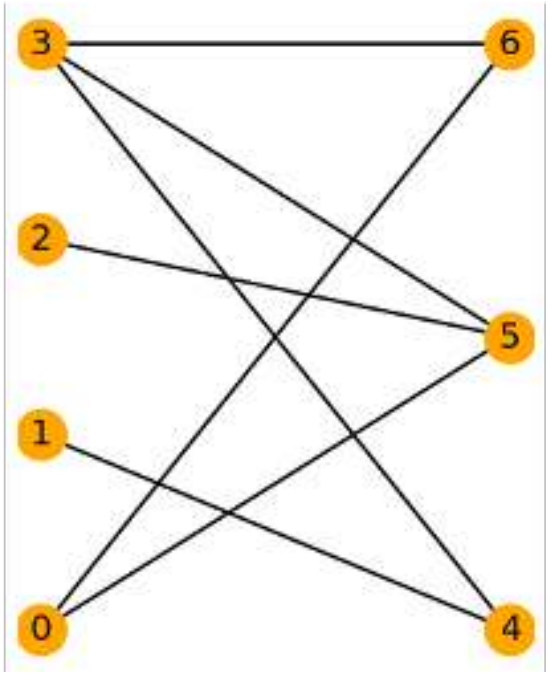
- For disconnected graphs we again find the same block diagonal structure found in the adjacency matrix for the incidence matrix. Those correspond to the two graph components.



$$\boldsymbol{B} = \begin{bmatrix} -1. & -1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 1. & 0. & -1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1. & 1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & -1. & -1. & -1. & 0. & 0. \\ 0. & 0. & 0. & 1. & 0. & 0. & -1. & 0. \\ 0. & 0. & 0. & 0. & 1. & 0. & 1. & -1. \\ 0. & 0. & 0. & 0. & 0. & 1. & 0. & 1. \end{bmatrix}$$

Incidence matrix - bipartite graphs

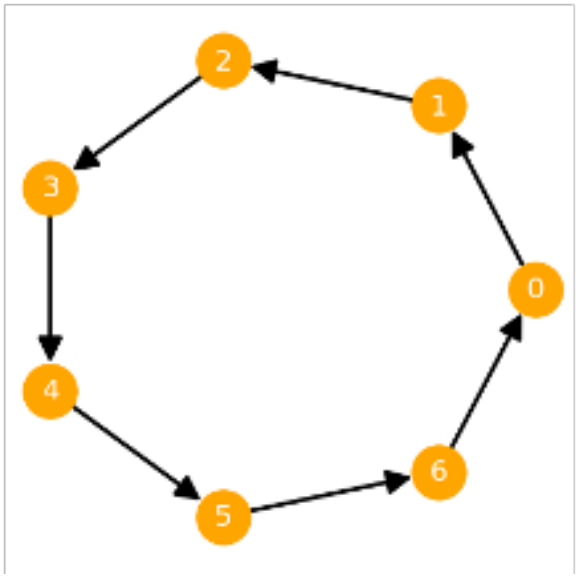
- In this case you can see that the nodes in the first group correspond to a negative sign while those in the second group have all positive signs
- That means that the left nodes are all incident (enter in) the nodes to the right.



$$\boldsymbol{B} = \begin{bmatrix} -1. & -1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & -1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & -1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & -1. & -1. & -1 \\ 0. & 0. & 1. & 0. & 1. & 0. & 0. \\ 1. & 0. & 0. & 1. & 0. & 1. & 0. \\ 0. & 1. & 0. & 0. & 0. & 0. & 1. \end{bmatrix}$$

Incidence matrix - cycle graphs

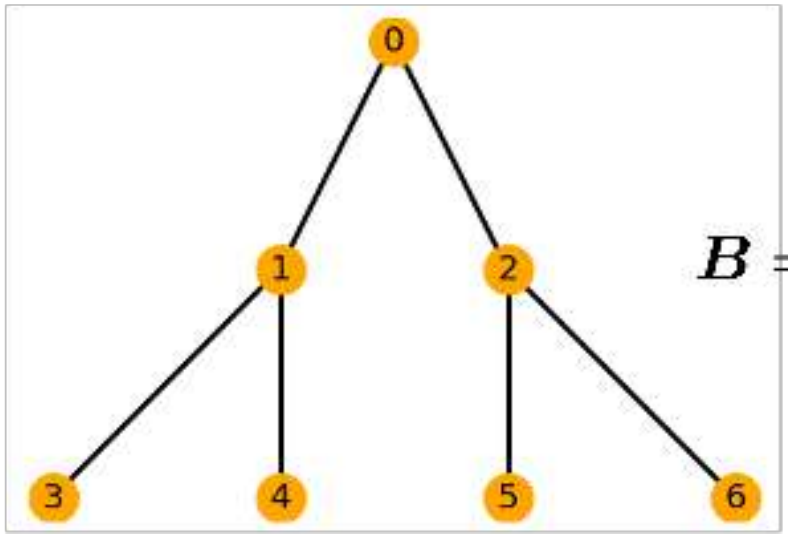
- We can see it has a Toepltiz circulant structure as well



$$\boldsymbol{B} = \begin{bmatrix} -1. & 0. & 0. & 0. & 0. & 0. & 1. \\ 1. & -1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1. & -1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 1. & -1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. & -1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 1. & -1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. & -1. \end{bmatrix}$$

Incidence matrix - trees

- We can see that each sub-tree has the same type of structure



$$\boldsymbol{B} = \begin{bmatrix} -1. & -1. & 0. & 0. & 0. & 0. \\ 1. & 0. & -1. & -1. & 0. & 0. \\ 0. & 1. & 0. & 0. & -1. & -1. \\ 0. & 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. \end{bmatrix}$$

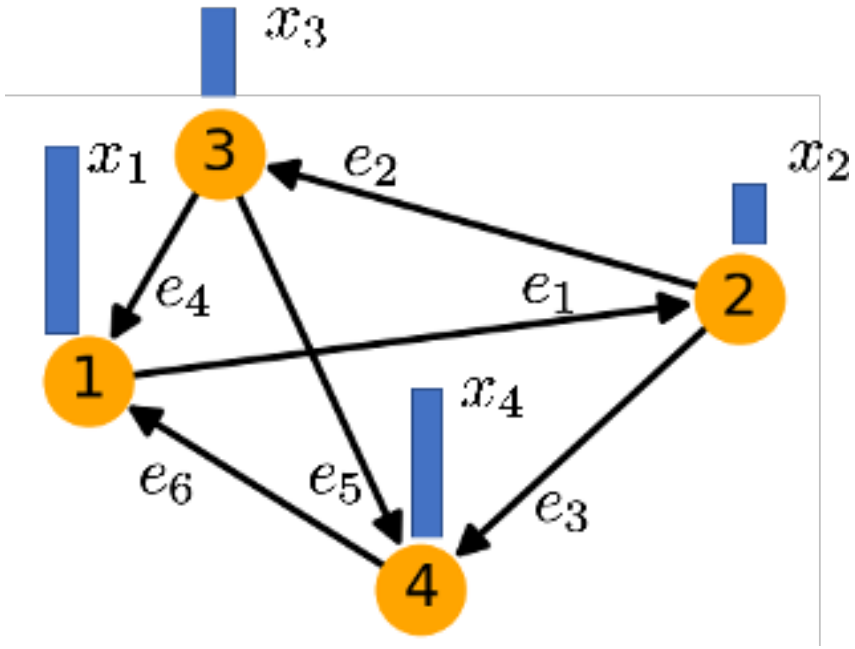
Outline

Cornell University

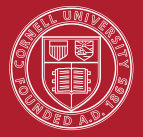
Flows and node signals

- Let $\boldsymbol{B}$ be the oriented weighted incidence matrix of $\mathcal{G}(\mathcal{V}, \mathcal{E})$.
- Let $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{V}|}$ be a graph signal, i.e. a vector whose index set is $\mathcal{V}$.
- We can define the edge signal indexed by the set $\mathcal{E}$:

$$\boldsymbol{e} = \boldsymbol{B}^\top \boldsymbol{x}.$$



The difference of node signals



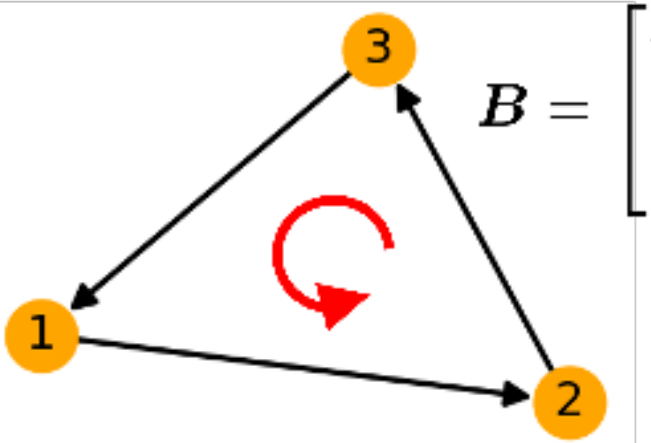
- The entries of $\boldsymbol{e}$ are the (weighted) difference between the values of $\boldsymbol{x}$ connected by the incident edge corresponding to the row of $\boldsymbol{B}^\top$ (or column of $\boldsymbol{B}$).
- Using the notation $a_k := [\boldsymbol{a}]_k$, specifically, for edge $k = (u, v)$:

$$e_k = [\boldsymbol{B}^\top \boldsymbol{x}]_k = w_{uv}(x_u - x_v) \quad k = (u, v) \in \mathcal{E}$$

- Note that, if $\boldsymbol{x} = \mathbf{1}$, then $\boldsymbol{e} = \boldsymbol{B}^\top \boldsymbol{x} = \mathbf{0}$. But, as long as the graph is connected, the null space of $\boldsymbol{B}^\top$ contains only vectors $\propto \mathbf{1}$.
- Can we say anything about the right null-space of $\boldsymbol{B}$ (i.e. $\boldsymbol{f}: \ \boldsymbol{B}\boldsymbol{f} = \mathbf{0}$)?
- Note that the rank of $\boldsymbol{B}$ is equal to $\mathcal{E} - \mathcal{V} + c$ where c is the number of connectec components of the graph.

Cycles - Kichoff's voltage law

- We can check what happens when the graph has cycles looking at the smallest possible example:



$$\boldsymbol{B} = \begin{bmatrix} -1. & 0. & 1. \\ 1. & -1. & 0. \\ 0. & 1. & -1. \end{bmatrix} \Rightarrow \boldsymbol{e} = \boldsymbol{B}^T \boldsymbol{x} = \begin{bmatrix} x_2 - x_1 \\ x_3 - x_2 \\ x_1 - x_3 \end{bmatrix}$$

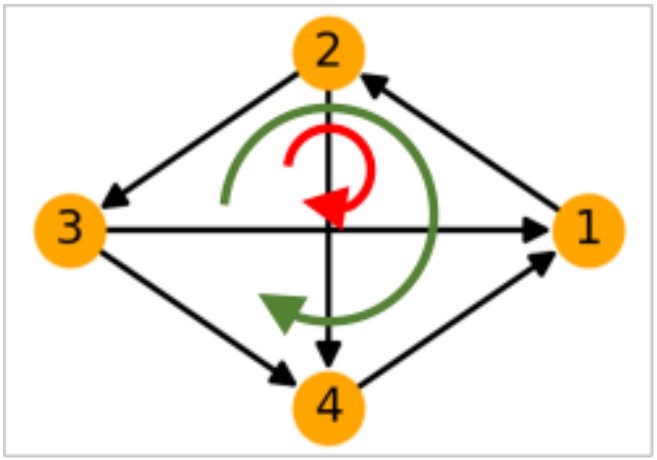
$$\Rightarrow \; e_1 + e_2 + e_3 = 0$$

Kichoff's voltage law

For graph signals $\boldsymbol{x}$, differences of nodal values x_i (a.k.a. potentials) around a cycle add up to zero.

Larger graphs and cycles

- Let us check a case with multiple cycles, longer than 3 hops.
- We can observe from the following example that, again, the set of columns in B corresponding to edges of a loop add up to zero.



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

Flows

- Let us now consider a signal $\boldsymbol{f}$ that is indexed by the edge set $\mathcal{E}$.

$$y_u = [\boldsymbol{B}\boldsymbol{f}]_u = \overbrace{\sum_{e \in \mathcal{N}_u^+} f_e}^{\text{incoming flow } f_u^{\text{in}}} - \overbrace{\sum_{e \in \mathcal{N}_u^-} f_e}^{\text{outgoing flow } f_u^{\text{out}}}$$

Kirchoff's current law

In an electrical network flows add up to zero at every node:

$$\boldsymbol{y} = \boldsymbol{B}\boldsymbol{f} = \boldsymbol{0} \quad \text{(conservation of charge)}$$

Network flows - KVL and KCL + Ohm's law

Network flows

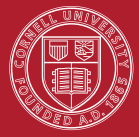
If the graph is connected and $\boldsymbol{x}$ is such that $\boldsymbol{f} = \text{diag}(\boldsymbol{w})\boldsymbol{B}^\top \boldsymbol{x} \neq 0$ (Ohm's law) then $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{f} = \boldsymbol{B}\text{diag}(\boldsymbol{w})\boldsymbol{B}^\top \boldsymbol{x} \neq \boldsymbol{0}$.

- $\boldsymbol{B}\boldsymbol{f} = \boldsymbol{0}$ is true because there is no *external* injection or withdrawal source at the nodes (i.e. think of a battery).

Flows

- Incidence matrices naturally appear in describing flows not only of electric currents information, transportation and pipelines, which are instances of flow networks.
- Kirchoff's law holds also for streets intersections or pipelines junctions that do not leak (nor nor are pumping or drawing flows).
- If vehicles stop in a street or packets are held at a router in a buffer then Kirchoff's law does not hold
- In general, in a connected network, unbalanced potential ($e = B^T x \neq \mathbf{0}$) is compatible with unbalanced flows ($f_u^{\text{in}} \neq f_u^{\text{out}}$).

Flows

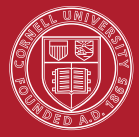


- Incidence matrices naturally appear in describing flows not only of electric currents information, transportation and pipelines, which are instances of flow networks.
- Kirchoff's law holds also for streets intersections or pipelines junctions that do not leak (nor nor are pumping or drawing flows).
- If vehicles stop in a street or packets are held at a router in a buffer then Kirchoff's law does not hold
- In general, in a connected network, unbalanced potential ($e = B^T x \neq 0$) is compatible with unbalanced flows ($f_u^{\text{in}} \neq f_u^{\text{out}}$).

Flows

- Incidence matrices naturally appear in describing flows not only of electric currents information, transportation and pipelines, which are instances of flow networks.
- Kirchoff's law holds also for streets intersections or pipelines junctions that do not leak (nor nor are pumping or drawing flows).
- If vehicles stop in a street or packets are held at a router in a buffer then Kirchoff's law does not hold
- In general, in a connected network, unbalanced potential $(e = B^T x \neq \mathbf{0})$ is compatible with unbalanced flows $(f_u^{\text{in}} \neq f_u^{\text{out}})$.

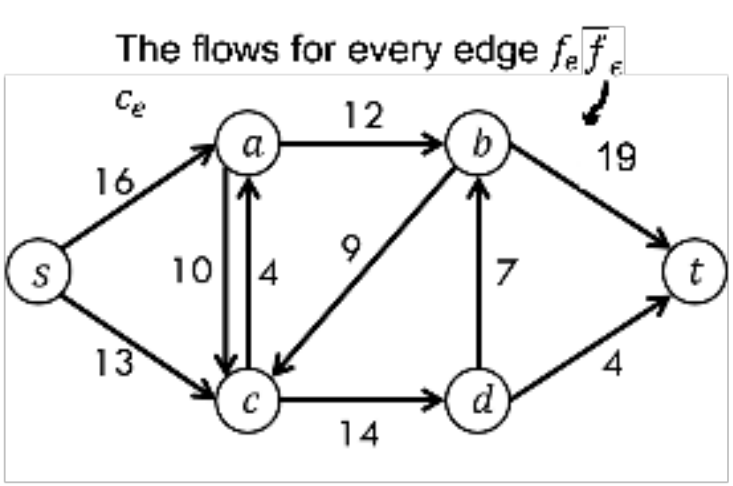
Flows

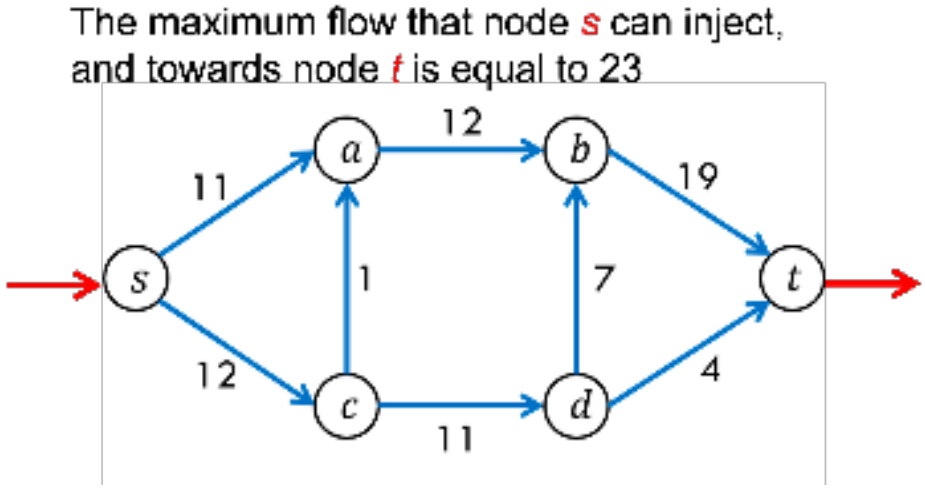


- Incidence matrices naturally appear in describing flows not only of electric currents information, transportation and pipelines, which are instances of flow networks.
- Kirchoff's law holds also for streets intersections or pipelines junctions that do not leak (nor nor are pumping or drawing flows).
- If vehicles stop in a street or packets are held at a router in a buffer then Kirchoff's law does not hold
- In general, in a connected network, unbalanced potential ($\boldsymbol{e} = \boldsymbol{B}^T \boldsymbol{x} \neq \mathbf{0}$) is compatible with unbalanced flows ($f_u^{\text{in}} \neq f_u^{\text{out}}$).

Network flows problems

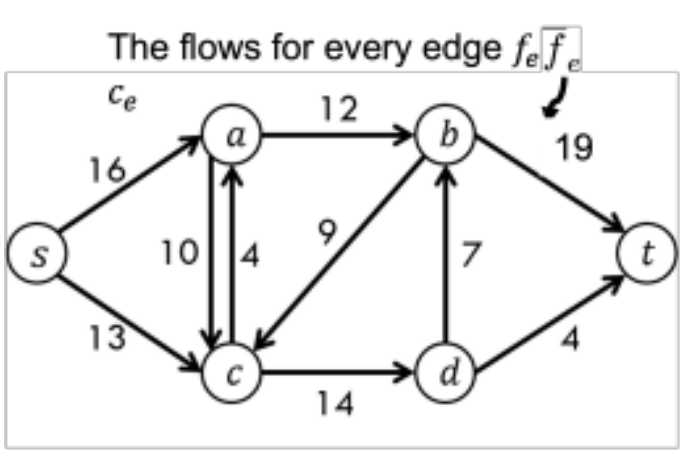
- This is the case in network flow problems, where a source node s is transferring a "mass" y_s to a destination t.
- In such problems there are link capacity limits, $\overline{f}_e$ meaning that: $f_e \leq \overline{f}_e$
- Intermediate nodes obey Kirchoff's law $f_u^{\text{in}} = f_u^{\text{out}} \rightarrow y_u = 0$ while $y_s = f_s^{\text{out}} - f_s^{\text{in}} > 0$ while $y_t = f_t^{\text{out}} - f_t^{\text{in}} = -y_s < 0$

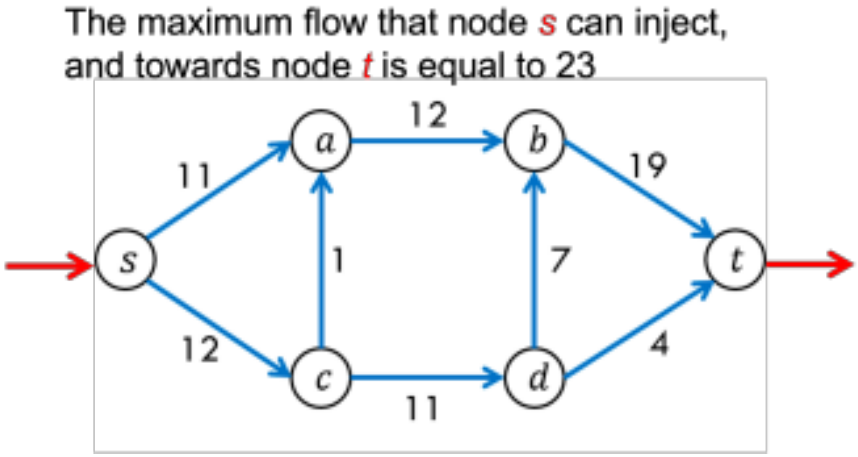




Network flows

- Note that in the example $f_u^{\text{in}} = f_u^{\text{out}}$ for all $u \in \{a, b, c, d\}$





- However, $y_s = f_s^{\text{out}} - \overbrace{f_s^{\text{in}}}^{=0} = 23$ and $y_t = \overbrace{f_t^{\text{out}}}^{=0} - f_t^{\text{in}} = -23$.
- Since both y_s and y_t are non zero, this implies that for this example: $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{f} \neq 0$.

Max-flow problem

- Maximize the total amount of flow subject to flow constraints, i.e.: y_s from s to t to the flow constraints.

$$\begin{aligned} \max \ y_s \ \text{ subject to} \\ & [\boldsymbol{B}\boldsymbol{f}]_u = 0, u \in \mathcal{V}/\{s,t\}, \\ & [\boldsymbol{B}\boldsymbol{f}]_s = -[\boldsymbol{B}\boldsymbol{f}]_t = y_s, \\ & 0 \leq \boldsymbol{f} \leq \overline{\boldsymbol{f}}, \quad , y_s \geq 0 \end{aligned}$$

- This is a linear program (LP), one can use in principle a simplex method for the answer.

Max-flow and min-cut

- An equivalent formulation that is more computationally efficient is based on identifying the bottleneck for the network:
  1. Set as a cost for removing an edge the capacity $\overline{f}_e$
  2. Find the minimum edge cut, such that $\mathcal{E}_{\text{cut}} \subset \mathcal{E}$ disconnects s and t with minimum cost.
- Flows that on a path from s to t will include an edge in $\mathcal{E}_{\text{min-cut}}$.
- Across all edges in the path the flow must be $\leq \overline{f}_e \in \mathcal{E}_{\text{min-cut}}$.

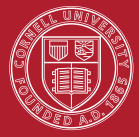
Ford-Fulkerson algorithm

- The algorithm based on a proven property of the solution. Its complexity is $O(|\mathcal{V}| + |\mathcal{E}|)$.
  - A flow $\boldsymbol{f}$ can always be decomposed in two components $\boldsymbol{f} = \boldsymbol{f}_p + \boldsymbol{f}_c$, where:
    1. $\boldsymbol{f}_p$ is a sum of flows that run along paths connecting s and d
    2. $\boldsymbol{f}_c$ is a sum *circulations*, i.e. $\boldsymbol{B}\boldsymbol{f}_c = \boldsymbol{0}$
  - The optimum solution has always $\boldsymbol{f}_c = \boldsymbol{0}$.
  - The flow components of $\boldsymbol{f}_p$ cannot be larger than the capacity of the edge $e \in \mathcal{E}$_min-cut that they cross.
  - So all is needed is to find all paths[2], identify $\mathcal{E}_{\text{min-cut}}$, and

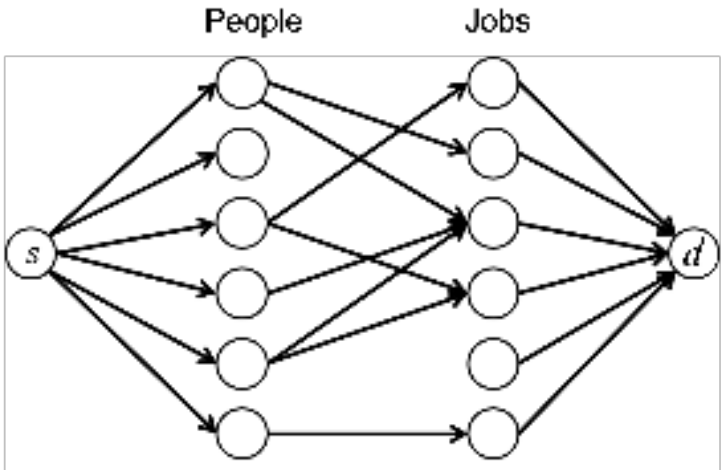
$$\max y_s = \sum_{e \in \mathcal{E}_{\text{min-cut}}} \overline{f}_e$$

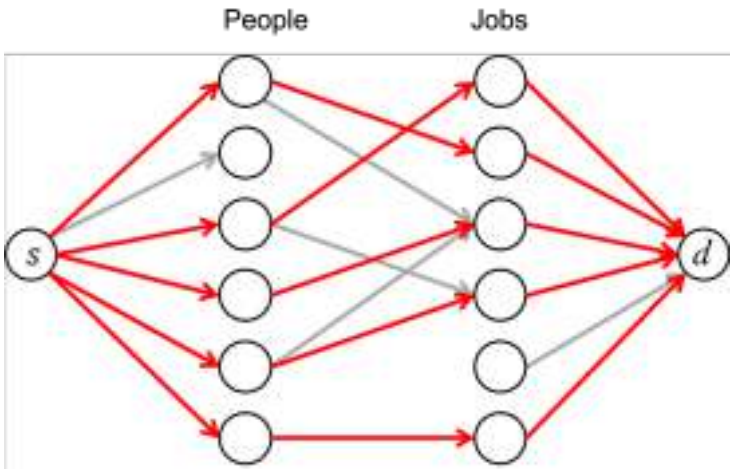
[2] We can use the Depth First Search (DFS) algorithm for that

Maximum Bipartite matching



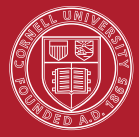
- The same algorithm can be used to solve another problem called *maximum bipartite matching*[3] where the bipartite graph sets are $\mathcal{A}$ and $\mathcal{B}$, say $\mathcal{A}$ are people and $\mathcal{B}$ jobs.
- One adds a source and a destination and capacities (e.g. the number of jobs available for a position).
- The active intermediate links in the optimum flow solution are going to be the maximum number of matches.





[3] See https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/matching.pdf

Outline



Cornell University

Concluding remarks

Cornell University

- We explored the properties of the incidence matrix and studied how it is useful to model network flow problems.
- In the next lecture we will wrap up this part with another important operator: the Laplacian matrix