Basics of Graph Theory

AU4606: Network Optimization

AI4702: Network Intelligence and Optimization

Xiaoming Duan
Department of Automation
Shanghai Jiao Tong University

September 14, 2023

Last time

- Seven Bridges of Königsberg
  - Euler's circuit theorem and its proof
- Problem we study in this course: minimum cost flow problems
  - Shortest path problem
  - Maximum flow problem
  - Assignment problem

Today

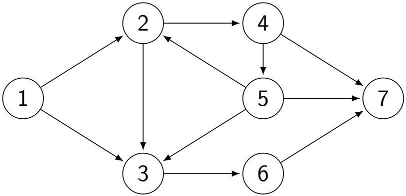
1. Basics of graph theory
   - Graphs
   - Paths, cycles, walks
   - Degrees
   - Subgraphs
   - Connectivity
   - Acyclic graphs
   - Trees
   - Bipartite graphs
2. Graph representations
   - Adjacency matrix
   - Incidence matrix
   - Adjacency list
3. Network transformation

Today

1. Basics of graph theory
   - Graphs
   - Paths, cycles, walks
   - Degrees
   - Subgraphs
   - Connectivity
   - Acyclic graphs
   - Trees
   - Bipartite graphs
2. Graph representations
   - Adjacency matrix
   - Incidence matrix
   - Adjacency list
3. Network transformation

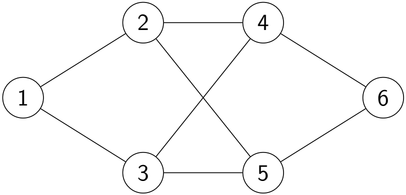
Notation and terminology: graphs

- A graph/network is a pair $G = (N, A)$
  - N: a set of nodes/vertices
  - A: a set of arcs/edges



A directed graph

$$N = \{1, 2, 3, 4, 5, 6, 7\}$$
$$A = \{(1,2), (1,3), (2,3), (2,4), (3,6), (4,7), (5,2), (5,3), (5,7), (6,7)\}$$

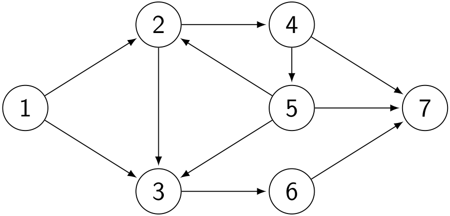


An undirected graph

$$N = \{1, 2, 3, 4, 5, 6\}$$
$$A = \{\{1,2\}, \{1,3\}, \{2,4\}, \{2,5\}, \{3,4\}, \{3,5\}, \{4,6\}, \{5,6\}\}$$

Notation and terminology: paths

In a directed graph $G = (N, A)$



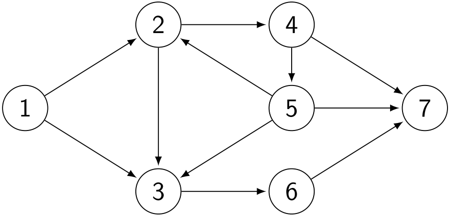
- A path is a sequence of nodes $(i_1, i_2, \ldots, i_r)$ such that
  1. $i_k \neq i_{k'}$ for all $k, k' \in \{1, \ldots, r\}$, i.e., no node repetition
  2. $(i_k, i_{k+1}) \in A$ or $(i_{k+1}, i_k) \in A$, i.e., directions are ignored

  examples: $(1, 2, 3, 5)$, $(7, 5, 4, 2)$, $\cancel{(1, 2, 3, 5, 2)}$
- A directed path is a sequence of nodes $(i_1, i_2, \ldots, i_r)$ such that
  1. $i_k \neq i_{k'}$ for all $k, k' \in \{1, \ldots, r\}$, i.e., no node repetition
  2. $(i_k, i_{k+1}) \in A$, i.e., directions are respected

  examples: $(1, 2, 3, 6, 7)$, $(4, 5, 7)$, $\cancel{(2, 5, 3, 1)}$

Notation and terminology: cycles

In a directed graph $G = (N, A)$



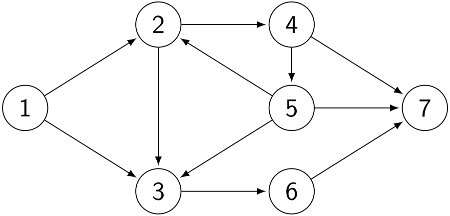
- A cycle is a path $(i_1, i_2, \ldots, i_r)$ such that $i_1 = i_r$
  examples: $(1, 2, 3, 1)$, $(2, 4, 5, 2)$
- A directed cycle is a directed path $(i_1, i_2, \ldots, i_r)$ such that $i_1 = i_r$
  examples: $(2, 4, 5, 2)$

In undirected graphs

- path = directed path
- cycle = directed cycle

Notation and terminology: walks

In a directed graph $G = (N, A)$



- A walk is a sequence of nodes $(i_1, i_2, \ldots, i_r)$ such that
  1. $(i_k, i_{k+1}) \in A$ or $(i_{k+1}, i_k) \in A$, i.e., directions are ignored

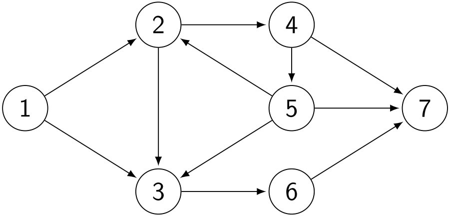
  examples: $(1, 2, 3, 1, 2, 5)$
- A directed walk is a sequence of nodes $(i_1, i_2, \ldots, i_r)$ such that
  1. $(i_k, i_{k+1}) \in A$, i.e., directions are respected

  examples: $(1, 2, 4, 5, 2, 3, 6)$
- A walk $(i_1, i_2, \ldots, i_r)$ is closed if $i_1 = i_r$

Walks allow node repetition, paths do not

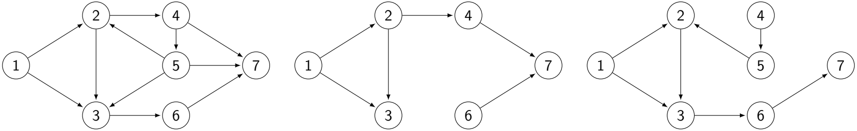
Notation and terminology: degrees

In a directed graph $G = (N, A)$



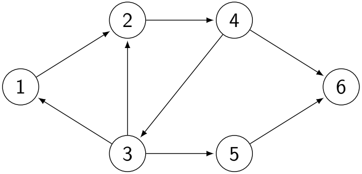
- The in-degree of a node i is the number of incoming arcs to i, e.g.,
  - in-degree $d_{\text{in}}(1) = 0$
  - in-degree $d_{\text{in}}(2) = 2$
  - in-degree $d_{\text{in}}(7) = 3$
- The out-degree of node i is the number of outgoing arcs from i, e.g.,
  - out-degree $d_{\text{out}}(3) = 1$
  - out-degree $d_{\text{out}}(5) = 3$
  - out-degree $d_{\text{out}}(6) = 1$
- The degree of a node i is the sum of its in-degree and out-degree

Notation and terminology: subgraphs

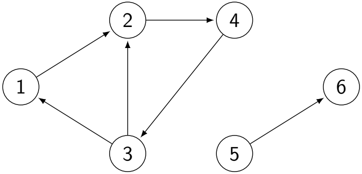


(a) Directed graph G (b) A subgraph of G (c) A spanning subgraph

- A graph $G' = (N', A')$ is a subgraph of $G = (N, A)$ if
  1. $N' \subset N$, and
  2. $A' \subset A$
- A graph $G' = (N', A')$ is a spanning subgraph of $G = (N, A)$ if
  1. $N' = N$, and
  2. $A' \subset A$



(a) Connected, but not strongly



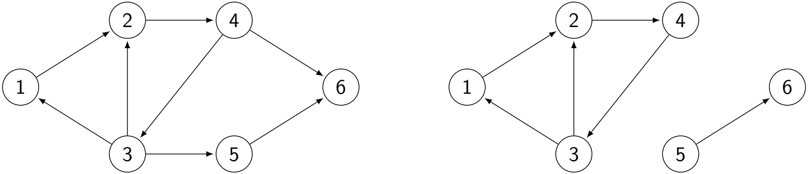
(b) Disconnected

Connectivity

- Nodes i and j are connected if there is at least a path (i.e., ignore directions) from i to j
- A graph is connected if every pair of nodes is connected, otherwise, disconnected

Strong connectivity

- A directed graph is strongly connected if there is at least a directed path from every node to every other node



(a) Connected, but not strongly (b) Disconnected

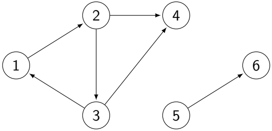
Connectivity

- Nodes i and j are connected if there is at least a path (i.e., ignore directions) from i to j
- A graph is connected if every pair of nodes is connected, otherwise, disconnected

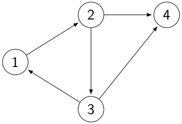
Strong connectivity

- A directed graph is strongly connected if there is at least a directed path from every node to every other node

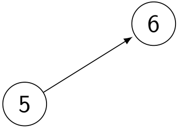
Connectivity vs strong connectivity: whether ignoring arc directions



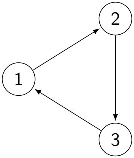
(a) Graph G



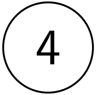
(b) Component 1



(c) Component 2

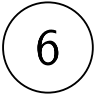


(d) SCC 1



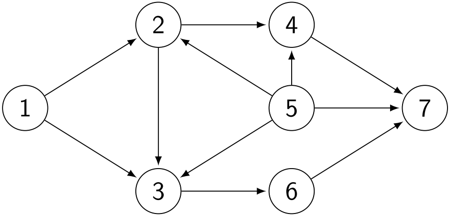
(e) SCC 2

(f) SCC 3

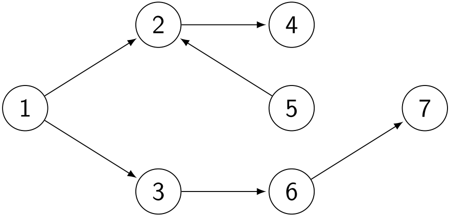


(g) SCC 4

- A connected component of a disconnected graph G is a maximal connected subgraph of G
- A strongly connected component (SCC) of a directed graph G is a maximal strongly connected subgraph of G

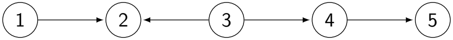


- A graph is acyclic if it contains no directed cycle

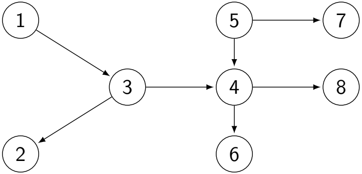


- A tree is a connected graph that contains no cycle

Trees $\Longrightarrow$ acyclic graphs **acyclic graphs $\not\Longrightarrow$ trees**



(a) A tree

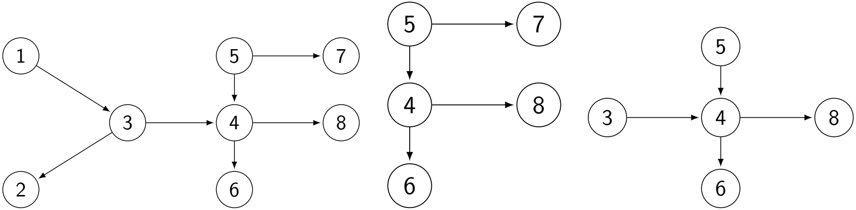


(b) Another tree

- A tree is a connected graph that contains no cycle

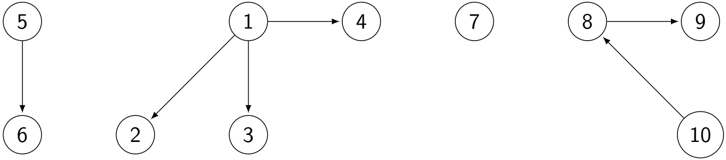
Properties of trees

1. A tree on n nodes contains exactly $n-1$ arcs;
2. A tree has at least two leaf nodes (i.e., nodes with degree 1);
3. Every two nodes of a tree are connected by a unique path.



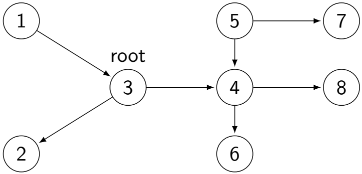
(a) A tree (b) A subtree (c) Another subtree

- A connected subgraph of a tree is a subtree
- A graph that contains no cycle is a forest
  1. not necessarily connected (compared with trees)
  2. cannot contain ANY cycle (compared with acyclic graphs)



A forest

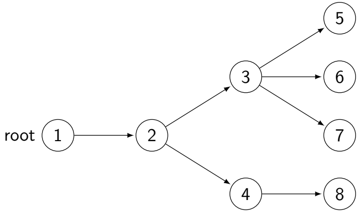
Notation and terminology: rooted trees



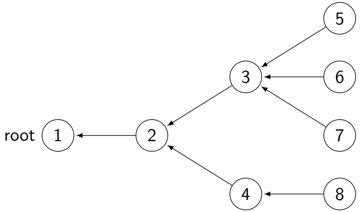
- A rooted tree is a tree with a special designated node, called its root
- Predecessor successor relationships
  1. Each node i (except the root node) has a unique predecessor: next node on the unique path i to the root
     Examples: Pred(1) $= 3$, Pred(7)=5, Pred(6)=4
  2. If node j is the predecrssor of node i, then i is a successor of j
     Examples: Succ(3) $= \{1, 2, 4\}$, Succ(4) $= \{5, 6, 8\}$
  3. The descendants of a node i consist of itself, its successors, successors of its successors, and so on

A unique path from i to root: $(i, \textbf{Pred}(i), \textbf{Pred}(\textbf{Pred}(i)), \ldots)$

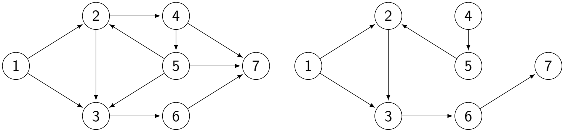
Notation and terminology: special rooted trees



- A tree is a directed out-tree rooted at node i if the unique path in the tree from node i to every other node is a directed path

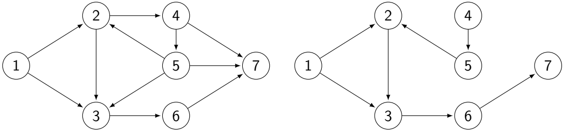


- A tree is a directed in-tree rooted at node i if the unique path in the tree from any node to node i is a directed path



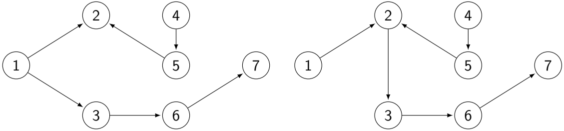
(a) Directed graph G (b) A spanning subgraph

- Recall: $G' = (N', A')$ is a spanning subgraph of $G = (N, A)$ if
  1. $N' = N$, and
  2. $A' \subset A$



(a) Directed graph G (b) A spanning subgraph

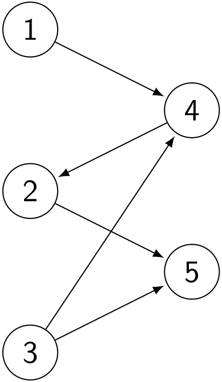
- Recall: $G' = (N', A')$ is a spanning subgraph of $G = (N, A)$ if
  1. $N' = N$, and
  2. $A' \subset A$



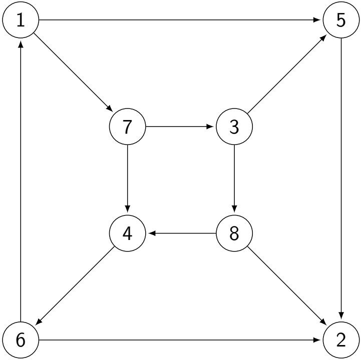
(a) A spanning tree (b) Another spanning tree

- A tree T is a spanning tree of G if T is a spanning subgraph of G

Notation and terminology: bipartite graphs

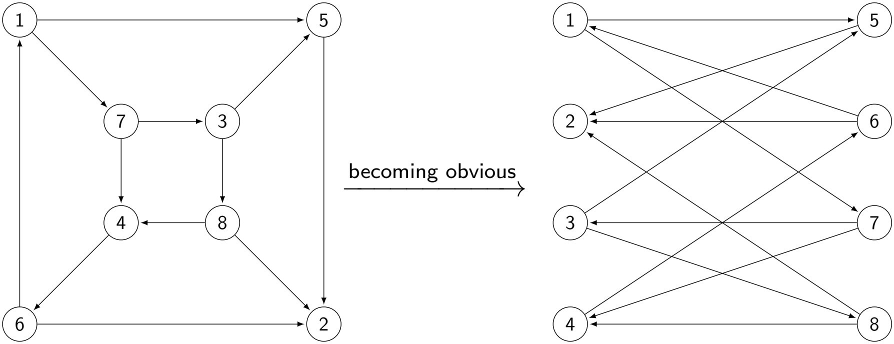


(a) A bipartite graph



(b) A (not obvious) bipartite graph

- A graph $G = (N, A)$ is a bipartite graph if we can partition its node set into two subsets N_1 and N_2 so that for each arc $(i, j) \in A$, either
  1. $i \in N_1$ and $j \in N_2$
  2. or $i \in N_2$ and $j \in N_1$



(a) A (not obvious) bipartite graph

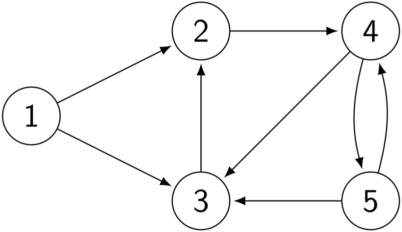
(b) An obvious bipartite graph

Property of bipartite graphs

A connected graph G is a bipartite graph if and only if every cycle (i.e., directions ignored) in G contains an even number of arcs.

Today

1. Basics of graph theory
   - Graphs
   - Paths, cycles, walks
   - Degrees
   - Subgraphs
   - Connectivity
   - Acyclic graphs
   - Trees
   - Bipartite graphs
2. Graph representations
   - Adjacency matrix
   - Incidence matrix
   - Adjacency list
3. Network transformation

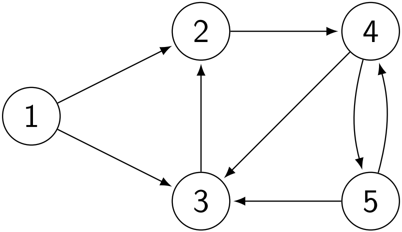


$$\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}
\end{array}$$

- Adjacency matrix $\mathcal{H} = \{h_{ij}\}$ of graph $G = (N, A)$ with n nodes:
  1. $\mathcal{H} \in \{0, 1\}^{n \times n}$
  2. Rows and columns correspond to nodes
  3. $h_{ij} = 1$ if $(i, j) \in A$
  4. $h_{ij} = 0$ if $(i, j) \notin A$

- It has n^2 elements, but only m of them are nonzero
- Space efficient only if the network is sufficiently dense

Network representations: adjacency matrix



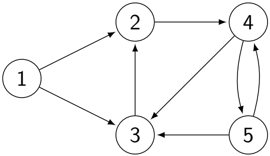
$$\begin{array}{c|c} & \begin{array}{ccccc}1 & 2 & 3 & 4 & 5\end{array} \\ \begin{array}{c}1\\2\\3\\4\\5\end{array} & \begin{bmatrix}0 & 1 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 1\\ 0 & 0 & 1 & 1 & 0\end{bmatrix}\end{array}$$

- $(\mathcal{H}\mathbb{1}_n)_i = d_{\text{out}}(i)$
- $(\mathbb{1}_n^\top \mathcal{H})_i = d_{\text{in}}(i)$

Number of walks and powers of adjacency matrix

Let $\mathcal{H}$ be an adjacency matrix of a directed graph G. Then $(\mathcal{H}^k)_{ij}$ equals the number of directed walks of length k from node i to node j.

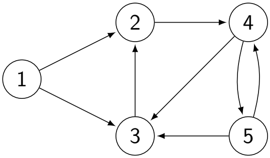
Network representations: incidence matrix



$$
\begin{array}{c}
\\ 1 \\ 2 \\ 3 \\ 4 \\ 5
\end{array}
\begin{array}{c}
\begin{array}{cccccccc}
(1,2) & (1,3) & (2,4) & (3,2) & (4,3) & (4,5) & (5,3) & (5,4)
\end{array} \\
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & -1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & 1 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 1
\end{bmatrix}
\end{array}
$$

- Incidence matrix $B = \{b_{ij}\}$ of $G = (N, A)$ with n nodes and m arcs
  1. $B \in \mathbb{R}^{n \times m}$
  2. Each row corresponds to a node, each column corresponds to an arc
  3. $b_{ij} = 1$ if node i is the head of arc j (arc j has node i as head)
  4. $b_{ij} = -1$ if node i is the tail of arc j (arc j has node i as tail)
  5. Exactly one 1 and one -1 in each column

Network representations: incidence matrix

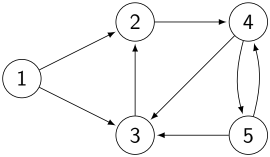


$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccccc} (1,2) & (1,3) & (2,4) & (3,2) & (4,3) & (4,5) & (5,3) & (5,4) \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix} \end{array}$$

Rank of incidence matrix

Let $B \in \mathbb{R}^{n \times m}$ be an incidence matrix of a connected directed graph G. Then $\text{rank}(B) = n - 1$.

Corollary: If $B \in \mathbb{R}^{n \times (n-1)}$ is the incidence matrix of a tree, then B has full column rank.



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccccc} (1,2) & (1,3) & (2,4) & (3,2) & (4,3) & (4,5) & (5,3) & (5,4) \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix} \end{array}$$

- Let $i_1 - e_1 - \cdots - i_k - e_k - i_1$ be a cycle (ignore directions)
- Let $x \in \mathbb{R}^m$ such that for $\ell \in \{1, \ldots, k\}$

$$x_{e_\ell} = \begin{cases} 1, & \text{if } e_\ell \in A \\ -1 & \text{if } e_\ell \notin A \end{cases}$$

  and other elements of x are zero
- Then $Bx = 0$

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in A} c_{ij} f_{ij} \\
\text{subject to} \quad & \sum_{(i,j)\in A} f_{ij} - \sum_{(j,i)\in A} f_{ji} = b(i), \quad \text{for all } i \in N \\
& l_{ij} \le f_{ij} \le u_{ij}
\end{aligned}
$$

- Let $f \in \mathbb{R}^m$ be the vector of flows
- Then the flow balance equation can be written as

$$Bf = b$$

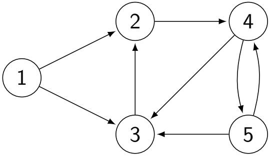
where $b = \begin{bmatrix} b(1) & b(2) & \dots & b(n) \end{bmatrix}^\top$

The properties of the solution to the LP are affected by that of B

Network representations: adjacency list

- The node adjacency list $A(i)$ of a node i is the set of nodes j such that $(i, j) \in A$, i.e.,

$$A(i) = \{j \in N \mid (i, j) \in A\}$$



- $A(1) = \{2, 3\}$
- $A(2) = \{4\}$
- $A(3) = \{2\}$
- $A(4) = \{3, 5\}$
- $A(5) = \{3, 4\}$

Today

1. Basics of graph theory
   - Graphs
   - Paths, cycles, walks
   - Degrees
   - Subgraphs
   - Connectivity
   - Acyclic graphs
   - Trees
   - Bipartite graphs
2. Graph representations
   - Adjacency matrix
   - Incidence matrix
   - Adjacency list
3. Network transformation

Network transformation: removing nonzero lower bounds I

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in A} c_{ij} f_{ij} \\
\text{subject to} \quad & \sum_{(i,j)\in A} f_{ij} - \sum_{(j,i)\in A} f_{ji} = b(i), \quad \text{for all } i \in N \\
& l_{ij} \le f_{ij} \le u_{ij}
\end{aligned}
$$

- Introduce $f'_{ij} = f_{ij} - l_{ij}$, then
  1. Cost: $c_{ij}(f'_{ij} + l_{ij}) = c_{ij} f'_{ij} + c_{ij} l_{ij}$, constant term can be ignored
  2. Bounds: $0 \le f'_{ij} \le u_{ij} - l_{ij}$
  3. Flow balance for each node i

$$
\sum_{(i,j)\in A} (f'_{ij} + l_{ij}) - \sum_{(j,i)\in A} (f'_{ji} + l_{ji}) = b(i)
$$

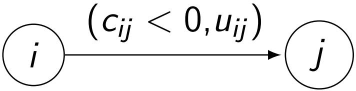
$$
\Longrightarrow \sum_{(i,j)\in A} f'_{ij} - \sum_{(j,i)\in A} f'_{ji} = b(i) - \sum_{(i,j)\in A} l_{ij} + \sum_{(j,i)\in A} l_{ji}
$$

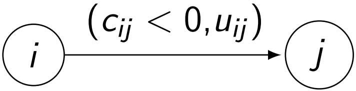
  4. Redefine $b'(i) = b(i) - \sum_{(i,j)\in A} l_{ij} + \sum_{(j,i)\in A} l_{ji}$

Mental picture

(a) With lower bound (b) Without lower bound

- The lower bound is equivalent to having a "preflow" on the arc
- Reduce the supply at i by l_{ij}
- Increase the supply at j by l_{ij}


$(c_{ij} < 0, u_{ij})$
i
j



- The cost can be written as

$$c_{ij} f_{ij} = (-c_{ij})(-f_{ij}) = (-c_{ij})(u_{ij} - f_{ij}) + c_{ij} u_{ij}$$

Network transformation: negative cost I

$$i \xrightarrow{(c_{ij} < 0, u_{ij})} j$$

- The cost can be written as

$$c_{ij} f_{ij} = (-c_{ij})(-f_{ij}) = (-c_{ij})(u_{ij} - f_{ij}) + c_{ij} u_{ij}$$

- Introduce $f_{ji} = u_{ij} - f_{ij}$
  1. Cost: $(-c_{ij}) f_{ji} + c_{ij} u_{ij}$
  2. Bounds: $0 \leq f_{ji} \leq u_{ij}$
  3. Flow balance for node i

$$\sum_{(i,k)\in A} f_{ik} - \sum_{(k,i)\in A} f_{ki} = b(i)$$
$$\implies \sum_{(i,k)\in A, k\neq j} f_{ik} - \sum_{(k,i)\in A} f_{ki} - f_{ji} = b(i) - u_{ij}$$

  4. Redefine $b'(i) = b(i) - u_{ij}$

Network transformation: negative cost II

$$ \textcircled{i} \xrightarrow{(c_{ij} < 0, u_{ij})} \textcircled{j} $$

- Introduce $f_{ji} = u_{ij} - f_{ij}$
  1. Cost: $(-c_{ij})f_{ji} + c_{ij}u_{ij}$
  2. Bounds: $0 \leq f_{ji} \leq u_{ij}$
  3. Flow balance for node i

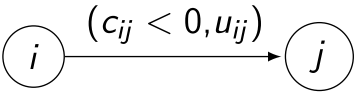
$$ \sum_{(i,k)\in A} f_{ik} - \sum_{(k,i)\in A} f_{ki} = b(i) $$

$$ \Longrightarrow \sum_{(i,k)\in A, k\neq j} f_{ik} - \sum_{(k,i)\in A} f_{ki} - f_{ji} = b(i) - u_{ij} $$

  4. Flow balance for node j

$$ \sum_{(j,k)\in A} f_{jk} - \sum_{(k,j)\in A} f_{kj} = b(j) $$

$$ \Longrightarrow \sum_{(j,k)\in A} f_{jk} + f_{ji} - \sum_{(k,j)\in A, k\neq i} f_{kj} = b(i) + u_{ij} $$

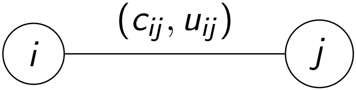


- Introduce $f_{ji} = u_{ij} - f_{ij}$
  1. Cost: $(-c_{ij})f_{ji} + c_{ij}u_{ij}$
  2. Bounds: $0 \leq f_{ji} \leq u_{ij}$
  3. Redefine $b'(i) = b(i) - u_{ij}$, $b'(j) = b(j) + u_{ij}$

Mental picture

(a) Negative cost

(b) Positive cost



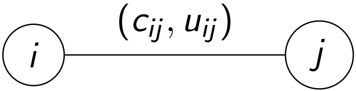
Setup:

- The flow can go either direction with cost c_{ij} per unit
- The total flow has an upper bound u_{ij}

"Naive" formulation:

- Define two variables $f_{ij} \geq 0$ and $f_{ji} \geq 0$
- The cost becomes $c_{ij} f_{ij} + c_{ij} f_{ji}$
- Add a constraint $f_{ij} + f_{ji} \leq u_{ij}$

However, this is not in the standard form of minimum cost flow problem



Property of the optimal solution to the naive formulation

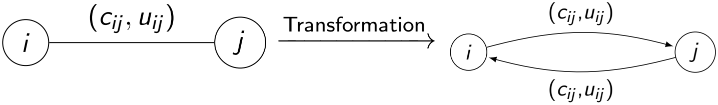
If f^* is an optimal solution to the naive formulation, then we have

1. either $f^*_{ij} > 0$ and $f^*_{ji} = 0$,
2. or $f^*_{ij} = 0$ and $f^*_{ji} > 0$,
3. or $f^*_{ij} = 0$ and $f^*_{ji} = 0$.

Thus, we can replace the constraint $f_{ij} + f_{ji} \le u_{ij}$ by $f_{ij} \le u_{ij}$ and $f_{ji} \le u_{ij}$

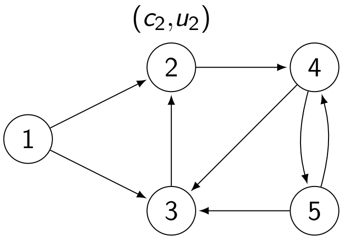
- Larger constraint set, but at the optimal, automatically $f_{ij} + f_{ji} \le u_{ij}$

Mental picture

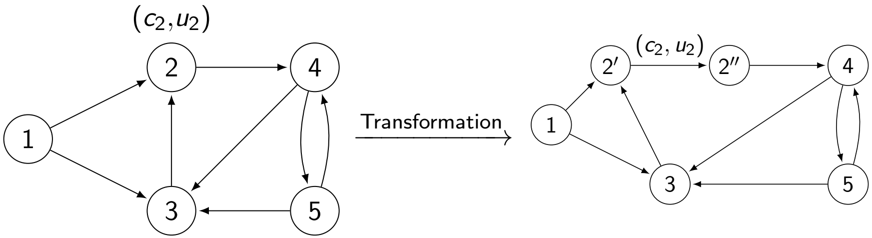


(a) Undirected edge (b) Two directed edges

What if there are capacity constraints and costs on nodes?

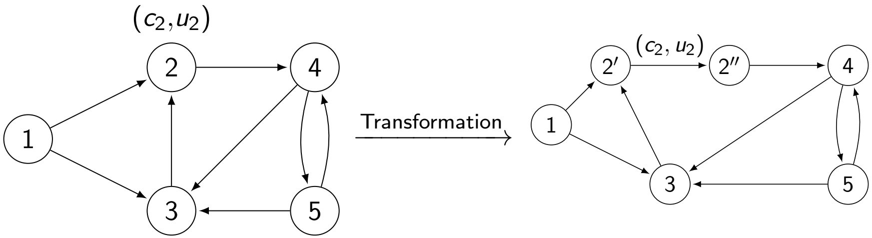


What if there are capacity constraints and costs on nodes?

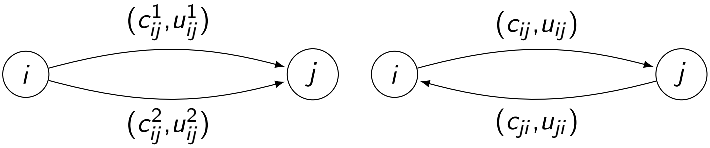


Network transformation: capacities and costs on nodes

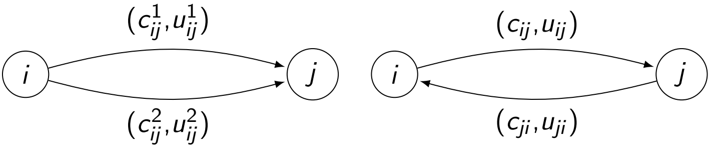
What if there are capacity constraints and costs on nodes?



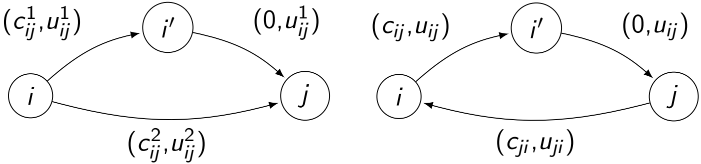
This is called node splitting



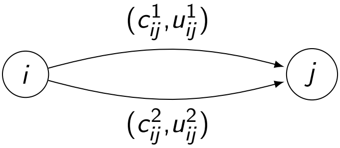
- Combine two edges?

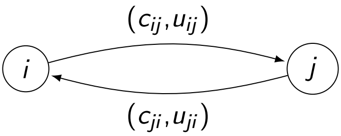


- Combine two edges?
- Introduce an additional node!

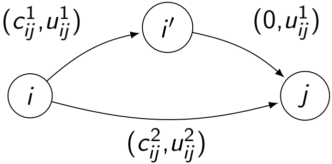


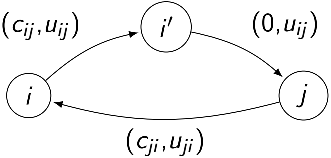
Network transformation: parallel edges





- Combine two edges?
- Introduce an additional node!





By above discussions, we can assume single directed edge with nonnegative costs between nodes and zero capacities lower bound

Upcoming

Week 1-8 (AU4606 & AI4702):

- Introduction (1 lecture)
- Preparations (3 lectures)
  - basics of graph theory (this lecture)
  - algorithm complexity and data structure (next lecture)
  - graph search algorithm
- Shortest path problems (3 lectures)
- Maximum flow problems (5 lectures)
- Minimum cost flow problems (3 lectures)
- Introduction to multi-agent systems (1 lecture)
- Introduction to cloud networks (1 lecture)

Week 9-16 (AU4606):

- Simplex and network simplex methods (2 lectures)
- Global minimum cut problems (3 lectures)
- Minimum spanning tree problems (3 lectures)