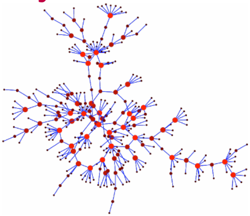
01RMHNG–03RMHPF–01RMING
Network Dynamics
Week 3
Connectivity and Network Flows



Giacomo Como and Fabio Fagnani
DISMA, Politecnico di Torino
{giacomo.como,fabio.fagnani}@polito.it

Torino, October 7–14, 2024

Prologue: multigraphs

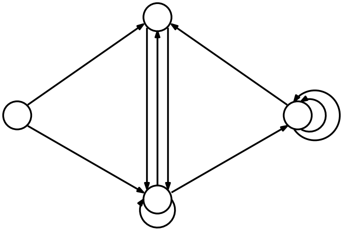
- A (weighted, directed) multigraph is a triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, h)$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of links, $h \in \mathbb{R}^{\mathcal{E}}$, $h > 0$, is the weight vector, equipped with two maps $\theta, \kappa : \mathcal{E} \to \mathcal{V}$ such that

$$\theta(e) = \text{ tail node of link } e \qquad \kappa(e) = \text{ head node of link } e$$

$$h_e > 0 \text{ is the weight of link } e$$

- when $h = \mathbb{1}$, we simply denote multi-graph by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- multi-graphs allow for parallel links: $e_1, e_2 \in \mathcal{E}$ such that

$$\theta(e_1) = \theta(e_2) \qquad \kappa(e_1) = \kappa(e_2)$$



Prologue: multigraphs

- A (weighted, directed) multigraph is a triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, h)$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of links, $h \in \mathbb{R}^{\mathcal{E}}$, $h > 0$, is the weight vector, equipped with two maps $\theta, \kappa : \mathcal{E} \to \mathcal{V}$ such that

$$\theta(e) = \text{ tail node of link } e \qquad \kappa(e) = \text{ head node of link } e$$

$$h_e > 0 \text{ is the weight of link } e$$

- to every graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ we can associate multigraph (without parallel links) $\overline{\mathcal{G}} = (\mathcal{V}, \mathcal{E}, h)$ such that $\forall\, e = (i, j) \in \mathcal{E}$

$$\theta(e) = i \qquad \kappa(e) = j \qquad h_e = W_{ij}$$

- length-l walk $\gamma = (e_1, e_2, \ldots, e_l)$: $\theta(e_k) = \kappa(e_{k-1})$ $\forall 1 \leq k \leq l$
- path $=$ walk $\gamma = (e_1, \ldots, e_l)$ s.t. $\theta(e_h) \neq \theta(e_k)$ $\forall 1 \leq h < k \leq l$

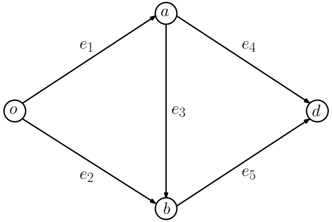
Prologue: node-link incidence matrix

- For (multi-)graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, h)$, node-link incidence matrix

$$B \in \{-1, 0, +1\}^{\mathcal{V} \times \mathcal{E}}$$

$$B_{ie} = \begin{cases} +1 & \text{if} \quad \theta(e) = i \neq \kappa(e) \\ -1 & \text{if} \quad \theta(e) \neq i = \kappa(e) \\ 0 & \text{if} \quad \theta(e) \neq i \neq \kappa(e) \text{ or } \theta(e) = i = \kappa(e) \end{cases}$$

- Example:



$$B = \begin{array}{c} \\ \\ \end{array}\begin{array}{cl} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} & \\ \begin{bmatrix} +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & +1 & +1 & 0 \\ 0 & -1 & -1 & 0 & +1 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix} & \begin{array}{c} o \\ a \\ b \\ d \end{array} \end{array}$$

- Note: unweighted (multi-)graphs without self-loops are completely characterized by their node-link incidence matrix

Prologue: node-link incidence matrix

- For (multi-)graph $\mathcal{G}$, node-link incidence matrix

$$B \in \{-1, 0, +1\}^{\mathcal{V}\times\mathcal{E}}$$

$$B_{ie} = \begin{cases} +1 & \text{if} \quad \theta(e) = i \neq \kappa(e) \\ -1 & \text{if} \quad \theta(e) \neq i = \kappa(e) \\ 0 & \text{if} \quad \theta(e) \neq i \neq \kappa(e) \text{ or } \theta(e) = i = \kappa(e) \end{cases}$$

- Proposition: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ unweighted:

$$BB' = \mathrm{diag}(w) - W + \mathrm{diag}(w^-) - W'$$

• $L = \mathrm{diag}(w) - W$ Laplacian of the graph
• $L^* = \mathrm{diag}(w^-) - W'$ Laplacian of the graph $\mathcal{G}^* = (\mathcal{V}, \mathcal{E}^*, W')$ obtained from $\mathcal{G}$ by reversing the direction of all its links

- $\mathcal{G}$ simple (undirected+unweighted+ no self-loops) $\Longrightarrow BB' = 2L$

From Reachability and Connectedness to Connectivity

In a (multi-)graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- Reachability: node $d \in \mathcal{V}$ is reachable from node $o \in \mathcal{V}$ if there exists at least an o-d path

$$\gamma = (e_1, e_2, \ldots, e_l)$$

- Connectedness: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ (strongly) connected if every $d \in \mathcal{V}$ is reachable from every $o \in \mathcal{V}$, i.e., if

- qualitative properties: connected or not, reachable or not

From Reachability and Connectedness to Connectivity

In a (multi-)graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- Reachability: node $d \in \mathcal{V}$ is reachable from node $o \in \mathcal{V}$ if there exists at least an o-d path

$$\gamma = (e_1, e_2, \ldots, e_l)$$

- Connectedness: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ (strongly) connected if every $d \in \mathcal{V}$ is reachable from every $o \in \mathcal{V}$, i.e., if

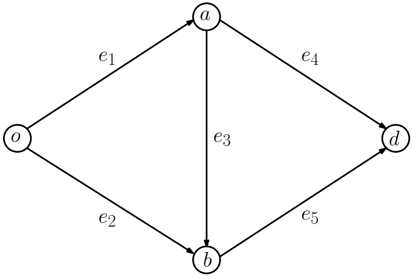
- qualitative properties: connected or not, reachable or not

- Connectivity: There might be several o-d paths

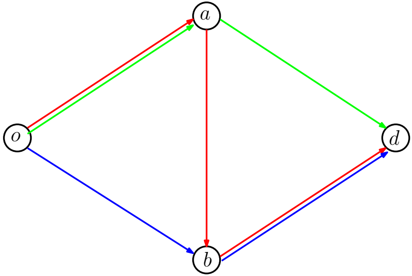
Maximum number of "independent" o-d paths

- quantitative property: how well connected a graph is

Example



Directed graph
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$
$\mathcal{V} = \{o, a, b, d\}$
$\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$



Three distinct *o*-*d* paths:
$\gamma_1 = (e_1, e_4)$
$\gamma_2 = (e_1, e_3, e_5)$
$\gamma_3 = (e_2, e_5)$

Node-connectivity and link-connectivity

Different *o-d* paths may share intermediate nodes or links

$$\gamma_1 = (e_1, e_2, \ldots, e_l) \qquad \tilde{\gamma}_2 = (\tilde{e}_1, \tilde{e}_2, \ldots, \tilde{e}_{\tilde{l}})$$

- γ, $\tilde{\gamma}$ node-independent if they share no *intermediate* node

$$\kappa(e_h) \neq \kappa(\tilde{e}_k) \text{ for all } 1 \leq h < l \text{ and } 1 \leq k < \tilde{l}$$

- γ, $\tilde{\gamma}$ link-independent if they share no link

$$e_h \neq \tilde{e}_k \text{ for all } 1 \leq h \leq l \text{ and } 1 \leq k \leq \tilde{l}$$

Node-connectivity and link-connectivity

Different o-d paths may share intermediate nodes or links

$$\gamma_1 = (e_1, e_2, \dots, e_l) \qquad \tilde{\gamma}_2 = (\tilde{e}_1, \tilde{e}_2, \dots, \tilde{e}_{\tilde{l}})$$

- γ, $\tilde{\gamma}$ node-independent if they share no *intermediate* node

$$\kappa(e_h) \neq \kappa(\tilde{e}_k) \text{ for all } 1 \leq h < l \text{ and } 1 \leq k < \tilde{l}$$

- γ, $\tilde{\gamma}$ link-independent if they share no link

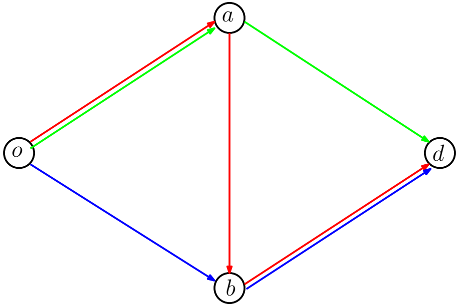
$$e_h \neq \tilde{e}_k \text{ for all } 1 \leq h \leq l \text{ and } 1 \leq k \leq \tilde{l}$$

Node-connectivity: $c_{\text{node}}(o, d)$: # node-independent o-d paths

Link-connectivity: $c_{\text{link}}(o, d)$ # link-independent o-d paths

$$c_{\text{node}}(\mathcal{G}) = \min_{o \neq d \in \mathcal{V}} c_{\text{node}}(o, d), \quad c_{\text{link}}(\mathcal{G}) = \min_{o \neq d \in \mathcal{V}} c_{\text{link}}(o, d)$$

Example: node-connectivity and link-connectivity



γ_1 and γ_3 are both node- and link-independent

γ_2 is neither node- nor link-independent from either γ_1 or γ_3

$c_{\mathsf{node}}(o,d) = c_{\mathsf{link}}(o,d) = 2$

$c_{\mathsf{node}}(\mathcal{G}) = c_{\mathsf{link}}(\mathcal{G}) = 0$ because $\mathcal{G}$ is not connected

Menger's Theorem

How many nodes and links must we remove from a (multi-)graph to disconnect two nodes?

Theorem (Menger) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $i \neq j \in \mathcal{G}$. Then,

- min $\#\{$nodes to remove for j not to be reachable from $i\} = c_{\text{node}}(i, j)$
- min $\#\{$links to remove for j not to be reachable from $i\} = c_{\text{link}}(i, j)$
- min $\#\{$nodes to remove for $\mathcal{G}$ not to be connected$\} = c_{\text{node}}(\mathcal{G})$
- min $\#\{$links to remove for $\mathcal{G}$ not to be connected$\} = c_{\text{link}}(\mathcal{G})$

Proof: $\geq$ can be seen directly.

$\leq$ is special case of more general result: max-flow min-cut theorem.

Link-path incidence matrices

In a (multi-)graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, fix $o \neq d \in \mathcal{V}$

- Γ_{od}: set of all o-d paths in $\mathcal{G}$
- Link-path incidence matrix $A \in \{0,1\}^{\mathcal{E} \times \Gamma_{od}}$:

$$A_{e\gamma} = \begin{cases} 1 & \text{if} \quad \text{link } e \text{ is along path } \gamma \\ 0 & \text{if} \quad \text{link } e \text{ is not along path } \gamma \,. \end{cases}$$

$$A \in \{0,1\}^{\mathcal{E} \times \Gamma_{od}}$$

Link-path incidence matrices

In a (multi-)graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, fix $o \neq d \in \mathcal{V}$

- Γ_{od}: set of all o-d paths in $\mathcal{G}$
- Link-path incidence matrix $A \in \{0,1\}^{\mathcal{E} \times \Gamma_{od}}$:

$$A_{e\gamma} = \begin{cases} 1 & \text{if} \quad \text{link } e \text{ is along path } \gamma \\ 0 & \text{if} \quad \text{link } e \text{ is not along path } \gamma \,. \end{cases}$$

$$A \in \{0,1\}^{\mathcal{E} \times \Gamma_{od}}$$

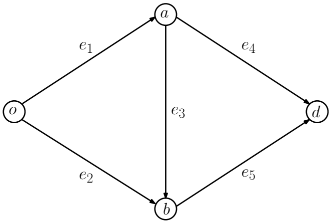
- Observe that, for every $i \in \mathcal{V}$ and $\gamma = (e_1, \ldots, e_l) \in \Gamma_{od}$,

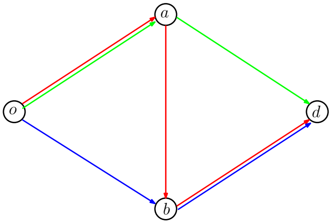
$$(BA)_{i\gamma} = \sum_{e \in \mathcal{E}} B_{ie} A_{e\gamma} = \sum_{h=1}^{l} B_{ie_h} = \begin{cases} +1 & \text{if} \quad i = o \\ -1 & \text{if} \quad i = d \\ 0 & \text{if} \quad i \neq o, d \end{cases}$$

so that

$$BA = \delta^{(o)} \mathbb{1}' - \delta^{(d)} \mathbb{1}'$$

Example





$$A = \begin{array}{c} \\ \end{array}\begin{array}{ccc} \color{green}{\gamma_1} & \color{red}{\gamma_2} & \color{blue}{\gamma_3} \end{array}$$

$$A = \left[\begin{array}{ccc} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{array}\right]\begin{array}{c} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{array}$$

$$\begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array}$$

$$B = \left[\begin{array}{ccccc} +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & +1 & +1 & 0 \\ 0 & -1 & -1 & 0 & +1 \\ 0 & 0 & 0 & -1 & -1 \end{array}\right]\begin{array}{c} o \\ a \\ b \\ d \end{array}$$

$$\begin{array}{ccc} \color{green}{\gamma_1} & \color{red}{\gamma_2} & \color{blue}{\gamma_3} \end{array}$$

$$BA = \left[\begin{array}{ccc} +1 & +1 & +1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{array}\right]\begin{array}{c} o \\ a \\ b \\ d \end{array}$$

Network flows - inflows and outflows

(Multi-)graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

$\nu \in \mathbb{R}^{\mathcal{V}}$ exogenous net flows in the nodes

$\nu_i^+ = \max\{\nu_i, 0\}$ ex. inflow, $\nu_i^- = \max\{-\nu_i, 0\}$ ex. outflow for i

- Mass conservation

$$\sum_{i \in \mathcal{V}} \nu_i = 0$$

total exogenous inflow $\sum_{i \in \mathcal{V}} \nu_i^+ =$ total external outflow $\sum_{i \in \mathcal{V}} \nu_i^-$

$$\text{throughput} \qquad \tau = \sum_{i \in \mathcal{V}} \nu_i^+ = \sum_{i \in \mathcal{V}} \nu_i^- = \frac{1}{2} ||\nu||_1$$

Nodes i such that $\nu_i = \nu_i^+ > 0$: *sources*, *origins*, *generators*

Nodes i such that $\nu_i = -\nu_i^- < 0$: *sinks*, *destinations*, *loads*

Network flows - flow vectors

Flow vectors: $f \in \mathbb{R}_+^{\mathcal{E}}$, satisfying balance constraints

$$\nu_i + \sum_{e:\kappa(e)=i} f_e = \sum_{e:\theta(e)=i} f_e \,, \qquad i \in \mathcal{V}$$

- f_e = flow on link $e \in \mathcal{E}$
- Compact form using node-link incidence matrix:

$$Bf = \nu$$

- Flows from a single origin o to a single destination d: o-d flows

$$Bf = \tau(\delta^{(o)} - \delta^{(d)})$$

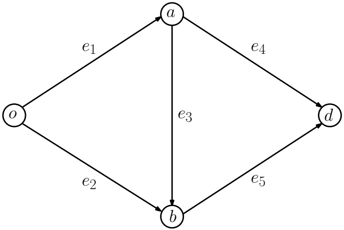
- Unitary o-d flows

$$Bf = \delta^{(o)} - \delta^{(d)}$$

$\delta^{(i)}$ = vector with a 1 entry in the i-th position and 0 everywhere else

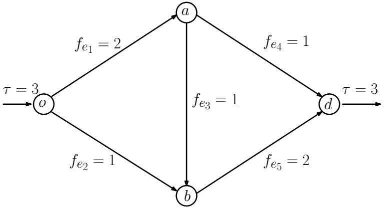
Example: network flows

o-d flow: nonnegative vector $f = (f_1, f_2, f_3, f_4, f_5)$
satisfying the flow balance:

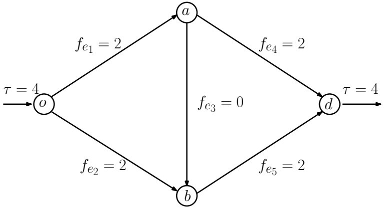


$$\tau = f_1 + f_2$$
$$f_1 = f_3 + f_4$$
$$f_2 + f_3 = f_5$$
$$f_4 + f_5 = \tau$$

Two of the possible o-d flows:



lower throughput $\tau = 3$



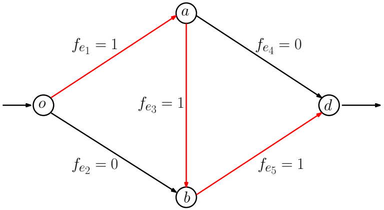
higher throughput $\tau = 4$

Unitary o-d flows

$\forall$ o-d paths $\gamma \in \Gamma_{od}$, $A\delta^{(\gamma)} \in \mathbb{R}^{\mathcal{E}}$:

- γ-th column of the link-path incidence matrix
- has entries 1 for links along the path γ and 0 otherwise
- is a unitary o-d flow

$$BA\delta^{(\gamma)} = \delta^{(o)} - \delta^{(d)}$$



γ_2-th column of the link-path incidence matrix A for the graph is a unitary o-d flow

Network Flow Assignment (and Decomposition)

- $z \in \mathbb{R}_+^{\Gamma_{od}}$, where z_γ aggregate flow on *o*-*d* path γ.

$$f = \sum_{\gamma \in \Gamma_{od}} z_\gamma A \delta^{(\gamma)} = Az$$

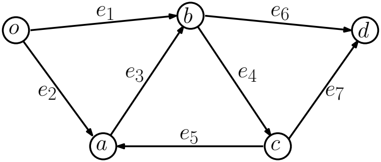
is an *o*-*d* flow of throughput $\tau = \sum_\gamma z_\gamma$

$$f \geq 0 \,, \qquad Bf = BAz = \tau(\delta^{(o)} - \delta^{(d)}) \,, \qquad \tau = \sum_\gamma z_\gamma$$

- Assign flows z to *o*-*d* paths (and cycles) in the graph
$\rightarrow$ unique *o*-*d* flow $f = Az$ on the links
(useful to construct feasible flows)

- Given *o*-*d* flow $f \in \mathbb{R}_+^{\mathcal{E}}$, there is a possibly (and typically) non-unique assignment of flows to both *o*-*d* paths and directed cycles in the graph that induces f (Flow Decomposition Theorem)

Example

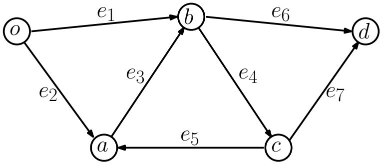


- $\Gamma_{od} = \{(\gamma_1, \gamma_2, \gamma_3, \gamma_4)\}$ where $\gamma_1 = (e_1, e_6)$, $\gamma_2 = (e_2, e_3, e_6)$, $\gamma_3 = (e_1, e_4, e_7)$, and $\gamma_4 = (e_2 e_3, e_4, e_7)$, so that

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}'$$

- directed cycle $\overline{\gamma} = (e_3, e_4, e_5)$, $C = (0\ 0\ 1\ 1\ 1\ 0\ 0)'$

Example



- $\Gamma_{od} = \{(\gamma_1, \gamma_2, \gamma_3, \gamma_4)\}$ where $\gamma_1 = (e_1, e_6)$, $\gamma_2 = (e_2, e_3, e_6)$, $\gamma_3 = (e_1, e_4, e_7)$, and $\gamma_4 = (e_2 e_3, e_4, e_7)$, so that

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}'$$

- directed cycle $\overline{\gamma} = (e_3, e_4, e_5)$, $C = (0\ 0\ 1\ 1\ 1\ 0\ 0)'$
- $z \in \mathbb{R}_+^{\Gamma_{od}}$ and $w \in \mathbb{R}_+ \Rightarrow f = Az + Cw$ is an o-d flow. E.g.,

$$z = (1, 2, 3, 1) \quad w = 1 \qquad \Rightarrow f = (4, 3, 4, 5, 1, 3, 4)$$

- $f \in \mathbb{R}_+^{\mathcal{E}}$ o-d flow $\Rightarrow f = Az + Cw$ where

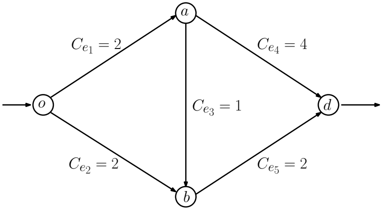
$$z^{(\alpha)} = (\alpha, f_6 - \alpha, f_1 - \alpha, f_2 + \alpha - f_6) \qquad w = f_5$$

for every $0 \le \alpha \le \min\{f_1, f_6\}$.

Capacity

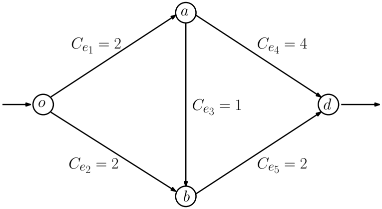
- Link $e \in \mathcal{E}$ has capacity $c_e > 0$:
maximum flow allowable through the link.

- Vector of all link capacities: $c \in \mathbb{R}^{\mathcal{E}}$, $c > 0$.



Maximum throughput τ from node o to node d that can be achieved by a flow f without violating the link capacity constraints?

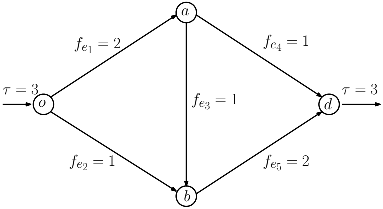
Example: maximum throughput with capacity constraints

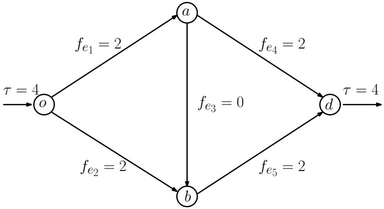


Maximize τ over link flows f_1, f_2, f_3, f_4, f_5 and throughput τ s.t.

$$\tau = f_1 + f_2, \quad f_1 = f_3 + f_4, \quad f_2 + f_3 = f_5, \quad f_4 + f_5 = \tau,$$

$$\tau \geq 0\,, \qquad 0 \leq f \leq c$$





Max flow problem

(Multi-)graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

- Link capacity vector $c \in \mathbb{R}^{\mathcal{E}}$, $c > 0$
- Link flows vector $f \in \mathbb{R}_+^{\mathcal{E}}$
- Throughput $\tau \geq 0$, total flow through the network from node o to node d, associated with f
- Consider two distinct nodes o and d. Maximum flow problem:

$$\begin{array}{rcll} \tau^*_{od} & = & \max \tau & \\ \text{s.t.} & & \tau \geq 0 & \text{throughput nonnegativity} \\ & & 0 \leq f \leq c & \text{nonnegativity and capacity constraints} \\ & & Bf = \tau(\delta^{(o)} - \delta^{(d)}) & \text{mass conservation} \end{array}$$

Linear program: objective function and constraints are linear functions of the variables

- Flow satisfying the constraints: feasible flow. Set of feasible flows nonempty: always contains flow $f = 0$ with throughput $\tau = 0$

Min cut capacity

- *o*-*d* cut: partition of the node set $\mathcal{V}$ in two subsets, $\mathcal{U}$ and $\mathcal{V} \setminus \mathcal{U}$, with $o \in \mathcal{U}$ and $d \in \mathcal{V} \setminus \mathcal{U}$
- (out-)boundary of $\mathcal{U}$ is set of links from $\mathcal{U}$ to $\mathcal{V} \setminus \mathcal{U}$:

$$\partial_{\mathcal{U}} = \{e \in \mathcal{E} : \theta(e) \in \mathcal{U}, \kappa(e) \in \mathcal{V} \setminus \mathcal{U}\}$$

- Capacity of an *o*-*d* cut $\mathcal{U}$ is the aggregate capacity of its boundary:

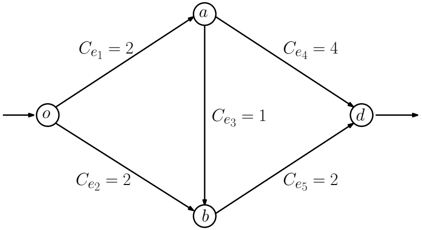
$$c_{\mathcal{U}} := \sum_{e \in \partial_{\mathcal{U}}} c_e$$

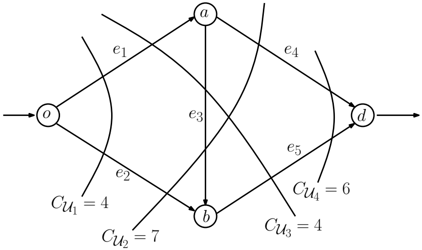
- Min-cut capacity: minimum capacity among all *o*-*d* cuts

$$c^*_{od} = \min_{\substack{\mathcal{U} \subseteq \mathcal{V} \\ o \in \mathcal{U},\, d \notin \mathcal{U}}} c_{\mathcal{U}}$$

- Minimal (capacity) cut: cut $\mathcal{U}$ with $c_{\mathcal{U}} = c^*_{od}$

Example: cut capacity





Four o-d cuts
$\mathcal{U}_1 = \{o\}$,
$\mathcal{U}_2 = \{o, a\}$,
$\mathcal{U}_3 = \{o, b\}$,
$\mathcal{U}_4 = \{o, a, b\}$

with capacities
$c_{\mathcal{U}_1} = c_{e_1} + c_{e_2} = 4$,
$c_{\mathcal{U}_2} = c_{e_2} + c_{e_3} + c_{e_4} = 7$,
$c_{\mathcal{U}_3} = c_{e_1} + c_{e_5} = 4$,
$c_{\mathcal{U}_4} = c_{e_4} + c_{e_5} = 6$

Min-cut capacity: $c^*_{od} = 4$
Minimal capacity cuts: $\mathcal{U}_1$ and $\mathcal{U}_3$

Max-flow min-cut theorem

- How do we guarantee that a flow vector achieves the maximum throughput τ^*_{od} from an origin node o to a destination node d?
- Relate τ^*_{od} to geometrical properties of the graph $\mathcal{G}$
- Max-Flow Min-Cut theorem: maximum throughput τ^*_{od} from o to d (solution of the linear program) coincides with the minimum cut capacity c^*_{od} among all o-d cuts:

- Theorem: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$ (capacited) multigraph. For $o \neq d \in \mathcal{V}$,

$$\tau^*_{od} = c^*_{od}$$

Capacities all integer-valued $\Rightarrow$ integer-valued max throughput flow

Max-flow min-cut theorem (cont'd)

- Network resilience interpretation of max-flow min-cut: minimum total capacity to be removed from the network to make d not reachable from o coincides with the min-cut capacity c^*_{od}

- if $c_e \in \{0, 1\}$ (keep or remove links) $\forall e \in \mathcal{E}$, then integer-valued feasible flows satisfy $f_e \in \{0, 1\}$. Set $\{e \in \mathcal{E} : f_e = 1\}$ is union of link-disjoint o-d paths. Hence max-flow min-cut reduces to Menger

- Proof of Max-flow min-cut Theorem: $\tau^*_{od} = c^*_{od}$. Two steps:
(i) $\tau^*_{od} \leq c^*_{od}$: no feasible flow can have throughput larger than the min-cut capacity (easier)
(ii) $\tau^*_{od} \geq c^*_{od}$: $\exists$ feasible flow with throughput equal to min-cut capacity (harder)

Max-flow min-cut theorem: proof $\tau^*_{od} \leq c^*_{od}$ (1)

- let $\partial^-_{\mathcal{A}} = \{e : \theta(e) \notin \mathcal{A}, \kappa(e) \in \mathcal{A}\}$ be the in-boundary of $\mathcal{A} \subseteq \mathcal{V}$
- Summing for all $i \in \mathcal{U}$ node-wise mass conservation

$$\nu_i + \sum_{e:\kappa(e)=i} f_e = \sum_{e:\theta(e)=i} f_e$$

we get

$$\tau = \sum_{i\in\mathcal{U}} \nu_i = \sum_{i\in\mathcal{U}} \left(\sum_{e\in\partial_i} f_e - \sum_{e\in\partial^-_i} f_e \right) = \sum_{e\in\partial_{\mathcal{U}}} f_e - \sum_{e\in\partial^-_{\mathcal{U}}} f_e$$

Since $0 \leq f_e \leq c_e$ for the flow on every link e,

$$\sum_{e\in\partial_{\mathcal{U}}} c_e \geq \sum_{e\in\partial_{\mathcal{U}}} f_e = \tau + \sum_{e\in\partial^-_{\mathcal{U}}} f_e \geq \tau$$

If we choose minimal capacity cut $\mathcal{U}$

$$c^*_{od} = \sum_{e\in\partial_{\mathcal{U}}} c_e \geq \tau^*_{od}$$

Max-flow min-cut theorem: proof $\tau^*_{od} \geq c^*_{od}$

- need to construct a feasible flow f from o to d with throughput τ equal to the min-cut capacity c^*_{od}.

- iterative algorithm due to Ford and Fulkerson does this in a finite number of steps, by starting with a trivial flow $f^{(0)} = 0$ with throughput $\tau^{(0)} = 0$ and capacity vector $c^{(0)} = c$ and then constructing a feasible flow for which $\tau^*_{od} = c^*_{od}$.

Ford and Fulkerson's algorithm

Start with $f^{(0)} = 0$. Then, for $t \geq 0$

- residual capacity: $c^{(t)} = c - f^{(t)}$
- residual graph: $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$

$$e \in \mathcal{E}_t \quad \Leftrightarrow \quad e \in \mathcal{E} \text{ and } c_e^{(t)} > 0 \quad \text{or} \quad \overline{e} \in \mathcal{E} \text{ and } f_{\overline{e}}^{(t)} > 0$$

where $\overline{e}$ reverse link with $\theta(\overline{e}) = \kappa(e)$ and $\kappa(\overline{e}) = \theta(e)$

- reachable set: $\mathcal{U}_t = \{i \in \mathcal{V} \,:\, i \text{ reachable from } o \text{ in } \mathcal{G}_t\}$
- $d \notin \mathcal{U}_t \quad \Longrightarrow \quad$ algorithm halts
- $d \in \mathcal{U}_t \quad \Longrightarrow \quad$ choose one o-d path in $\mathcal{G}_t$

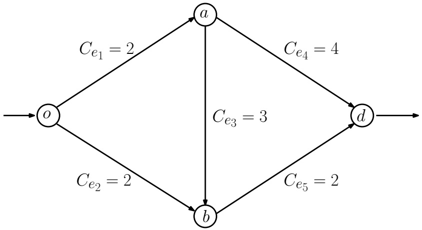
$$\gamma^{(t)} = (e_1, e_2, \ldots, e_l) \qquad\qquad \varepsilon_t := \min_{1 \leq h \leq l} \max \left\{ c_{e_h}^{(t)}, f_{\overline{e}_h}^{(t)} \right\}$$

$$f^{(t+1)} = f^{(t)} + \varepsilon_t \sum_{1 \leq h \leq l} \chi^{(h)}, \qquad \chi^{(h)} = \begin{cases} \delta^{(e_h)} & \text{if} \quad \varepsilon_t > f_{\overline{e}_h}^{(t)} \\ -\delta^{(\overline{e}_h)} & \text{if} \quad \varepsilon_t \leq f_{\overline{e}_h}^{(t)} \end{cases}$$

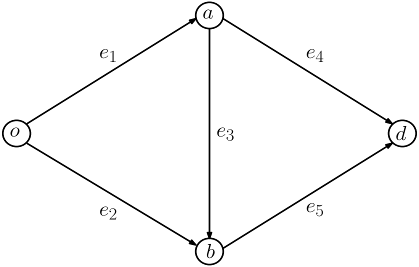
Ford and Fulkerson's algorithm (cont'd)

- if algorithm halts with o-d flow $f^{(t^*)}$, then throughput equal to capacity of some cut
- if link capacities are all positive integers, then algorithm halts in at most c^*_{od} steps and constructed flow vector has integer entries
- for rational capacities $c_e = \frac{n_e}{m}$, it halts in at most $c^*_{od} \cdot m$ steps
- approximating irrational capacities by rational ones $\to$ proof
- Attention: naïve implementation of the Ford-Fulkerson algorithm can fail to converge for irrational capacities
- From computational viewpoint, choice of "augmenting path" in residual graph $\mathcal{G}_t$ is crucial. Effective choice is to select the shortest (i.e., minimal length) o-d path in $\mathcal{G}_t$: Edmonds-Karp algorithm with strongly polynomial complexity $O(|\mathcal{V}||\mathcal{E}|^2)$.
- With further refinements, complexity can be reduced to $O(|\mathcal{V}|^2|\mathcal{E}|)$ (Dinic algorithm): using dynamic trees, the complexity of the Dinic algorithm can be further reduced to $O(|\mathcal{V}||\mathcal{E}|\log|\mathcal{V}|)$.

Ford and Fulkerson's algorithm: example



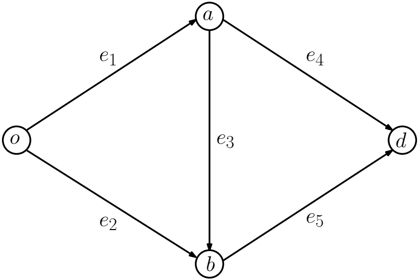
Ford and Fulkerson's algorithm: example



$$f^{(0)} = (0, 0, 0, 0, 0) \qquad c^{(0)} = (2, 2, 3, 4, 2)$$

$$\mathcal{E}_0 = \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\} \qquad \mathcal{U}_0 = \mathcal{V} = \{o, a, b, d\}$$

Ford and Fulkerson's algorithm: example



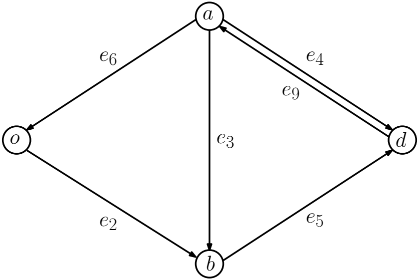
$$f^{(0)} = (0,0,0,0,0) \qquad c^{(0)} = (2,2,3,4,2)$$

$$\mathcal{E}_0 = \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\} \qquad \mathcal{U}_0 = \mathcal{V}$$

- choose $\gamma^{(0)} = (e_1, e_4)$, then

$$\varepsilon_0 = \min\{c_{e_1}, c_{e_4}\} = 2 \qquad f^{(1)} = f^{(0)} + 2(\delta^{(e_1)} + \delta^{(e_4)}) = (2,0,0,2,0)$$

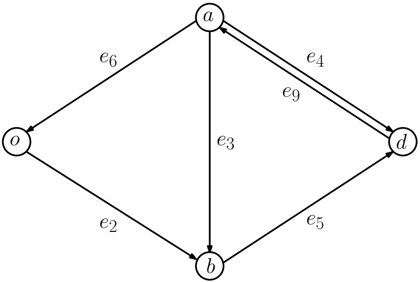
Ford and Fulkerson's algorithm: example



$$f^{(1)} = (2, 0, 0, 2, 0) \qquad c^{(1)} = (0, 2, 3, 2, 2)$$

$$\mathcal{E}_1 = \{e_2, e_3, e_4, e_5, e_6, e_9\} \qquad \mathcal{U}_1 = \mathcal{V}$$

Ford and Fulkerson's algorithm: example



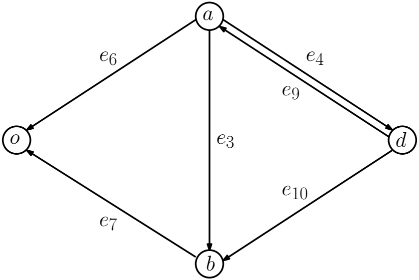
$$f^{(1)} = (2, 0, 0, 2, 0) \qquad c^{(1)} = (0, 2, 3, 2, 2)$$

$$\mathcal{E}_1 = \{e_2, e_3, e_4, e_5, e_6, e_9\} \qquad \mathcal{U}_1 = \mathcal{V}$$

- choose $\gamma_1 = (e_2, e_5)$, then

$$\varepsilon_1 = \min\{c^{(1)}_{e_2}, c^{(1)}_{e_5}\} = 2 \qquad f^{(2)} = f^{(1)} + 2(\delta^{(e_2)} + \delta^{(e_5)}) = (2, 2, 0, 2, 2)$$

Ford and Fulkerson's algorithm: example

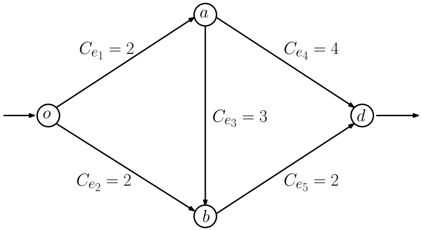


$$f^{(2)} = (2, 2, 0, 2, 2) \qquad\qquad c^{(1)} = (0, 0, 3, 2, 0)$$

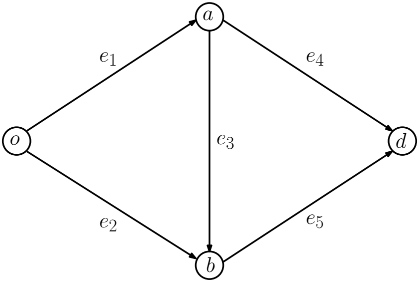
$$\mathcal{E}_2 = \{e_3, e_4, e_6, e_7, e_9, e_{10}\}, \qquad \mathcal{U}_2 = \{o\}$$

- halt: found (o, d)-flow $f^{(2)}$ of throughput 4

Ford and Fulkerson's algorithm: example



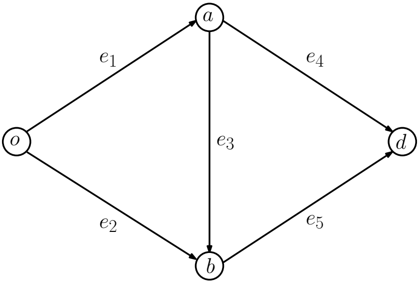
Ford and Fulkerson's algorithm: example



$$f^{(0)} = (0, 0, 0, 0, 0) \qquad c^{(0)} = (2, 2, 3, 4, 2)$$

$$\mathcal{E}_0 = \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\} \qquad \mathcal{U}_0 = \mathcal{V}$$

Ford and Fulkerson's algorithm: example



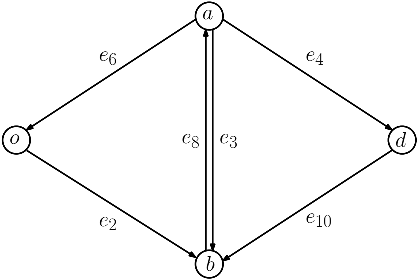
$$f^{(0)} = (0,0,0,0,0) \qquad c^{(0)} = (2,2,3,4,2)$$

$$\mathcal{E}_0 = \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\} \qquad \mathcal{U}_0 = \mathcal{V} = \{o, a, b, d\}$$

- choose $\gamma^{(0)} = (e_1, e_3, e_5)$, then

$$\varepsilon_0 = \min\{c_{e_1}, c_{e_3}, c_{e_5}\} = 2 \qquad f^{(1)} = (2,0,2,0,2)$$

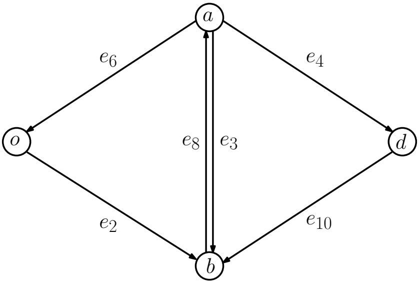
Ford and Fulkerson's algorithm: example



$$f^{(1)} = (2, 0, 2, 0, 2) \qquad c^{(1)} = (0, 2, 1, 4, 0)$$

$$\mathcal{E}_1 = \{e_2, e_3, e_4, e_6, e_8, e_{10}\}, \qquad \mathcal{U}_1 = \mathcal{V}$$

Ford and Fulkerson's algorithm: example



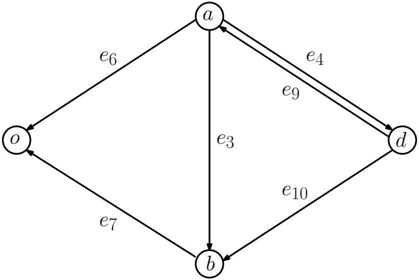
$$f^{(1)} = (2,0,2,0,2) \qquad c^{(1)} = (0,2,1,4,0)$$

$$\mathcal{E}_1 = \{e_2, e_3, e_4, e_6, e_8, e_{10}\}\,, \qquad \mathcal{U}_1 = \mathcal{V}$$

- choose $\gamma_1 = (e_2, e_8, e_4)$, then

$$\varepsilon_1 = \min\{c^{(1)}_{e_2}, f^{(1)}_{e_3}, c^{(1)}_{e_4}\} = 2 \qquad f^{(2)} = (2,2,0,2,2)$$

Ford and Fulkerson's algorithm: example



$$f^{(2)} = (2, 2, 0, 2, 2) \qquad c^{(1)} = (0, 0, 3, 2, 0)$$

$$\mathcal{E}_2 = \{e_3, e_4, e_6, e_7, e_9, e_{10}\}, \qquad \mathcal{U}_2 = \{o\}$$

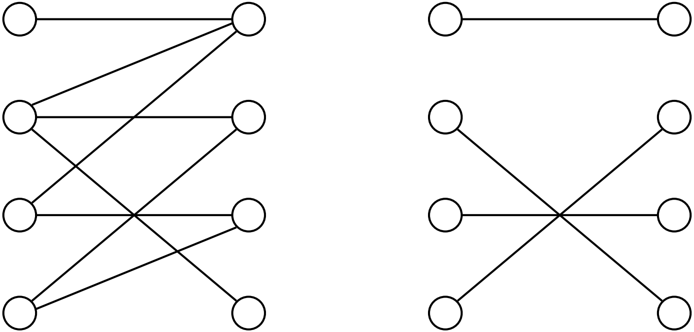
- halt: found (o, d)-flow $f^{(2)}$ of throughput 4

Multiple origin-destination (still single commodity)

- Corollary: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$ be a capacitated multigraph and let ν in $\mathbb{R}^{\mathcal{V}}$ be such that $\mathbb{1}'\nu = 0$. Then, a feasible flow vector f with exogenous net-flow vector ν exists if and only if

$$\sum_{i \in \mathcal{U}} \nu_i \leq c_{\mathcal{U}}\,, \qquad \forall \mathcal{U} \subseteq \mathcal{V}$$

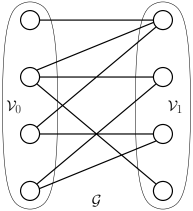
Matchings



In an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, $\overline{\mathcal{E}}$ = set of undirected links

- matching: $\mathcal{M} \subseteq \overline{\mathcal{E}}$ s.t. no self-loops and no adjacent links
- maximum matching: matching of maximal cardinality
- maximum weight matching: matching of maximal weight
- perfect matching: $|\mathcal{M}| = n/2$

Hall's Theorem



Bipartite undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, $\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1$

- complete matching from $\mathcal{V}_0$ to $\mathcal{V}_1$:

$$\text{matching } \mathcal{M} \text{ of cardinality } |\mathcal{V}_0|$$

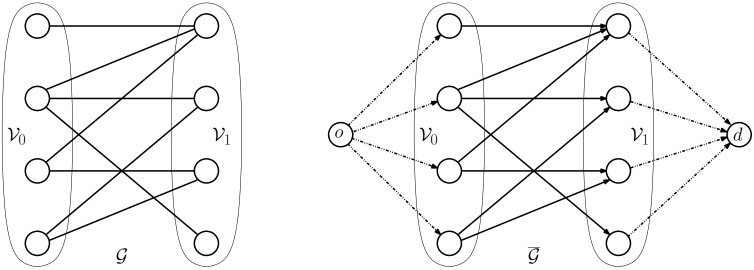
- Theorem: $\exists$ complete matching from $\mathcal{V}_0$ to $\mathcal{V}_1$ if and only if

$$|\mathcal{N}_{\mathcal{S}}| \geq |\mathcal{S}| \qquad \forall \mathcal{S} \subseteq \mathcal{V}_0$$

- Proof: Necessity is clear

For sufficiency, use max-flow min-cut

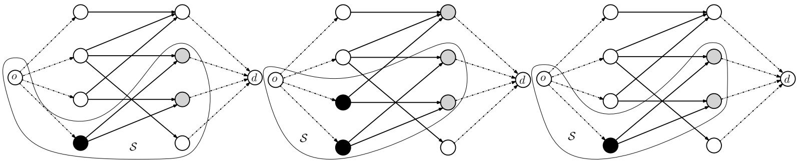
Hall's Theorem: Proof Idea



- from undirected bipartite $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ build directed graph $\mathcal{G}'$ with node set $\mathcal{V} \cup \{s, d\}$, all existing links directed from $\mathcal{V}_1$ to $\mathcal{V}_2$ with capacity $n + 1$, and with new capacity-1 links (s, i) for every $i \in \mathcal{V}_1$ and (j, d) for every $j \in \mathcal{V}_d$.

- prove that min-cut capacity between s and d is equal to n under Hall's Theorem conditions

Hall's Theorem: Proof Idea



- minimal o-d cuts $\mathcal{S}$ are necessarily in the form

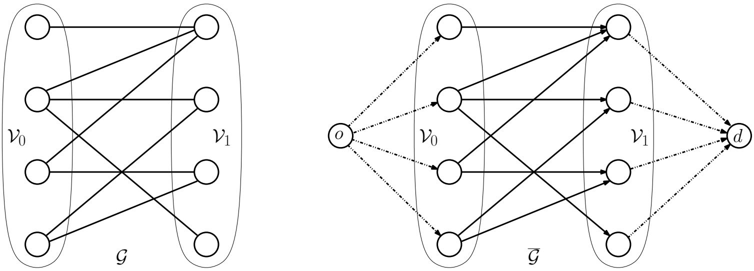
$$\mathcal{S} = \{o\} \cup \mathcal{U} \cup \mathcal{N}^+_{\mathcal{U}} \qquad \text{for some} \qquad \mathcal{U} \subseteq \mathcal{V}_0$$

- then

$$c^*_{od} = \min_{\substack{\mathcal{S} \\ o-d \text{ cut}}} c_{\mathcal{S}} = \min_{\mathcal{U} \subseteq \mathcal{V}_0} \{|\mathcal{V}_0| - |\mathcal{U}| + |\mathcal{N}^+_{\mathcal{U}}|\} = |\mathcal{V}_0| + \min_{\mathcal{U} \subseteq \mathcal{V}_0} \{|\mathcal{N}^+_{\mathcal{U}}| - |\mathcal{U}|\}$$

- $c^*_{od} \leq |\mathcal{V}_0|$ with equality $\iff$ Hall's conditions met
- matchings in $\mathcal{G}$ are in one-to-one correspondence with feasible integer o-d flows in the capacitated multigraph $\overline{\mathcal{G}}$

Hall's Theorem: Proof Idea



- it follows that Ford-Fullkerson (FF) algorithm compute complete matching from $\mathcal{V}_0$ to $\mathcal{V}_1$ when this exists
- in general FF algorithm computes maximal (weight) matching