Algorithm Theory

Chapter 6
Graph Algorithms

Part IV:
Simple Maximum Flow Applications

Fabian Kuhn

Maximum Flow Applications

- Maximum flow has many applications
- Reducing a problem to a max flow problem can even be seen as an important algorithmic technique
- Examples:
  - related network flow problems
  - computation of small cuts
  - computation of matchings
  - computing disjoint paths
  - scheduling problems
  - assignment problems with some side constraints
  - …

Undirected Edges and Vertex Capacities

Undirected Edges:

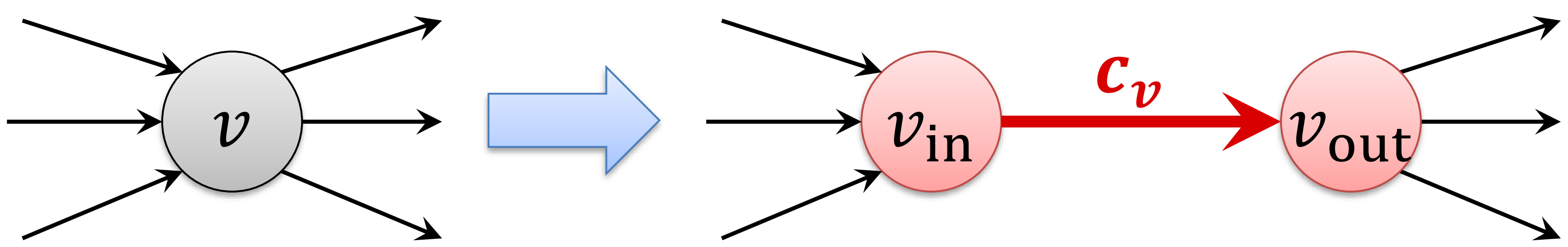
- Undirected edge $\{u, v\}$: add edges (u, v) and (v, u) to network

Vertex Capacities:

- Not only edges, but also (or only) nodes have capacities
- Capacity c_v of node $v \notin \{s, t\}$:

$$f^{\text{in}}(v) = f^{\text{out}}(v) \leq c_v$$

- Replace node v by edge $e_v = \{v_{\text{in}}, v_{\text{out}}\}$:

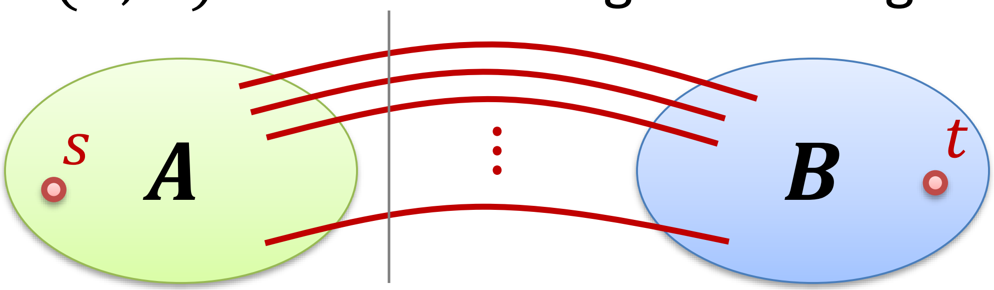


Minimum s-t Cut

Given: undirected graph $G = (V, E)$, nodes $s, t \in V$

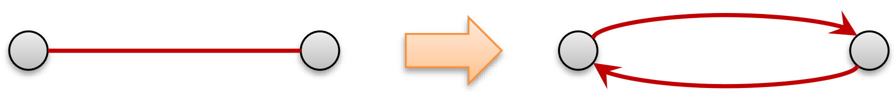
s-t cut: Partition (A, B) of V such that $s \in A$, $t \in B$

Size of cut (A, B): number of edges crossing the cut



size of cut $=$ #edges crossing the cut

Objective: find s-t cut of minimum size

- Create flow network:
  - make edges directed: 
  - edge capacities $= 1$

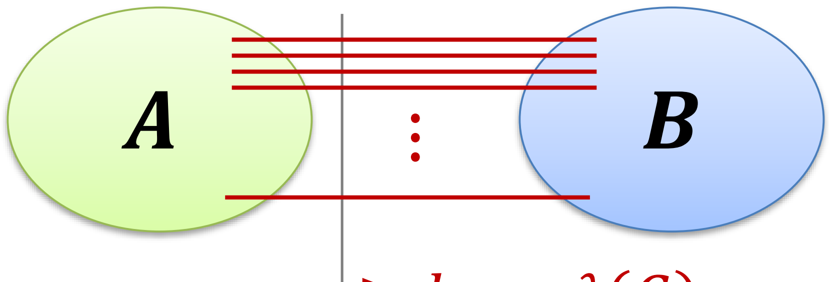
- Size of cut in G $=$ capacity of cut in flow network

Edge Connectivity

Definition: A graph $G = (V, E)$ is k-edge connected for an integer $k \geq 1$ if the graph $G_X = (V, E \setminus X)$ is connected for every edge set

$$X \subseteq E, |X| \leq k - 1.$$

Need to remove $\geq k$ edges to disconnect G



$\geq k =: \lambda(G)$

Edge Connectivity $\lambda(G)$

max k such that G is k-edge connected.

Goal: Compute *edge connectivity $\lambda(G)$* of G
(and edge set X of size $\lambda(G)$ that divides G into ≥ 2 parts)

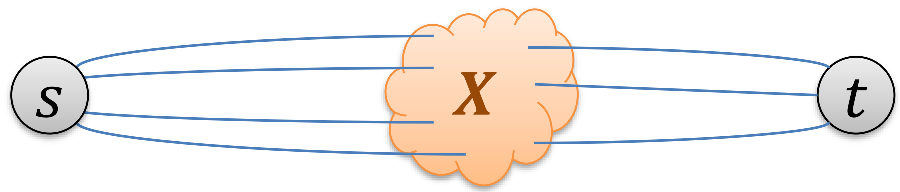
- minimum set X is a minimum s-t cut for some $s, t \in V$
  - Actually for all s, t in different components of $G_X = (V, E \setminus X)$
- Fix s, find min s-t cut for all $t \neq s \Longrightarrow$ running time $O(mn^2)$

Minimum s-t Vertex-Cut

Given: undirected graph $G = (V, E)$, nodes $s, t \in V$

***s*-*t* vertex cut:** Set $X \subset V$ such that $s, t \notin X$ and s and t are in different components of the sub-graph $G[V \setminus X]$ induced by $V \setminus X$

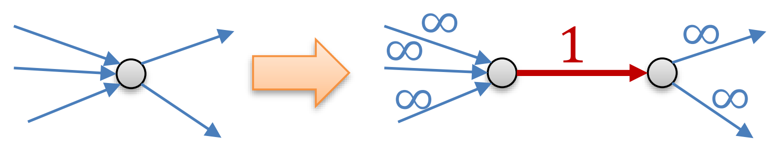
Size of vertex cut: $|X|$



Objective: find s-t vertex-cut of minimum size

- Replace undirected edges $\{u, v\}$ by (u, v) and (v, u)
- Compute max s-t flow for edge capacities ∞ and node capacities

$$c_v = 1 \text{ for } v \neq s, t$$



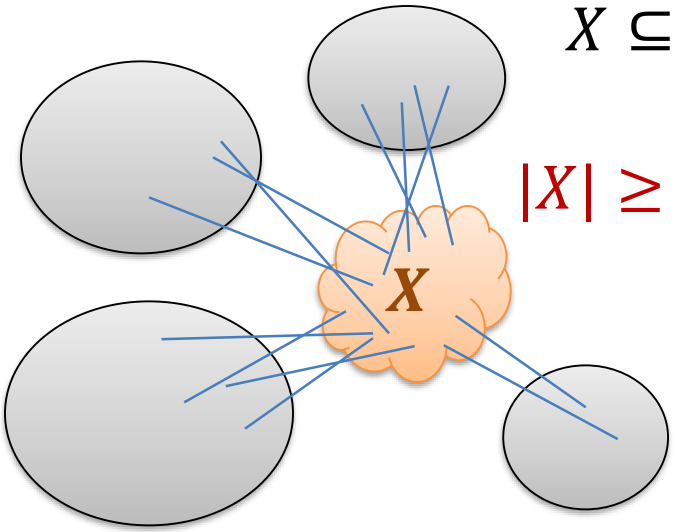
- Replace each node v by v_{in} and v_{out}
- Min edge cut corresponds to min vertex cut in G

Vertex Connectivity

Definition: A graph $G = (V, E)$ is k-vertex connected for an integer $k \geq 1$ if the sub-graph $G[V \setminus X]$ induced by $V \setminus X$ is connected for every edge set

$X \subseteq V, |X| \leq k - 1.$

Need to remove $\geq k$ edges to disconnect G



$|X| \geq k =: \kappa(G)$

Vertex Connectivity $\kappa(G)$

max k such that G is k-vertex connected.

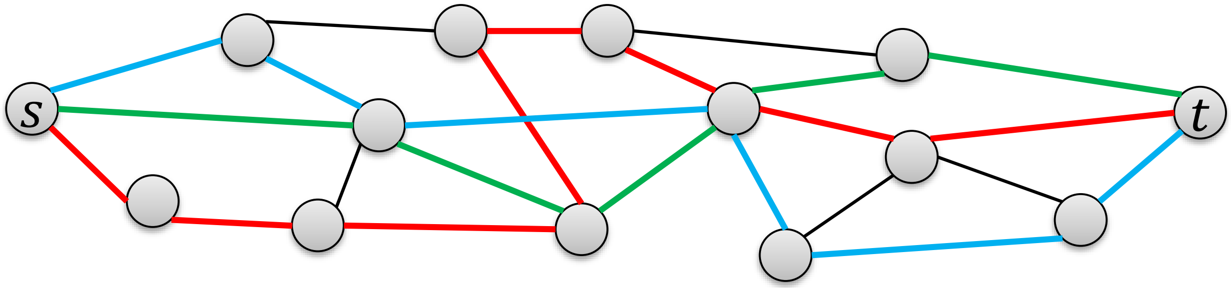
Goal: Compute vertex connectivity $\kappa(G)$ of G
(and node set X of size $\kappa(G)$ that divides G into ≥ 2 parts)

- Compute minimum s-t vertex cut for all s and all $t \neq s$ such that t is not a neighbor of s $\implies$ running time $O(m \cdot n^3)$

Edge-Disjoint Paths

Given: Graph $G = (V, E)$ with nodes $s, t \in V$

Goal: Find as many edge-disjoint s-t paths as possible



Solution:

- Find max s-t flow in G with edge capacities $c_e = 1$ for all $e \in E$

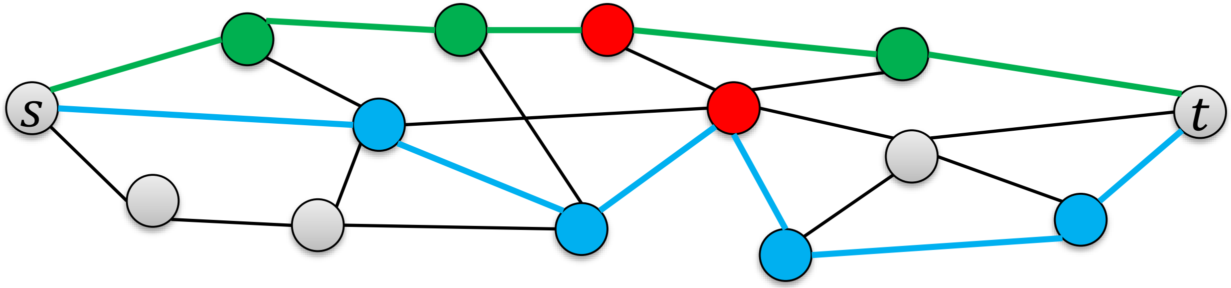
Flow f induces $|f|$ edge-disjoint paths:

- Integral capacities → can compute integral max flow f
- Get $|f|$ edge-disjoint paths by greedily picking them
- Correctness follows from flow conservation $f^{\text{in}}(v) = f^{\text{out}}(v)$

Vertex-Disjoint Paths

Given: Graph $G = (V, E)$ with nodes $s, t \in V$

Goal: Find as many internally vertex-disjoint s-t paths as possible



Solution:

- Find max s-t flow in G with node capacities $c_v = 1$ for all $v \in V$

Flow f induces $|f|$ vertex-disjoint paths:

- Integral capacities → can compute integral max flow f
- Get $|f|$ vertex-disjoint paths by greedily picking them
- Correctness follows from flow conservation $f^{\text{in}}(v) = f^{\text{out}}(v)$

Menger's Theorem

Theorem: (edge version)
For every graph $G = (V, E)$ with nodes $s, t \in V$, the size of the minimum s-t (edge) cut equals the maximum number of pairwise edge-disjoint paths from s to t.

Theorem: (node version)
For every graph $G = (V, E)$ with non-adjacent nodes $s, t \in V$, the size of the minimum s-t vertex cut equals the maximum number of pairwise internally vertex-disjoint paths from s to t.

- Both versions can be seen as a special case of the max flow min cut theorem