C&O 355
Mathematical Programming
Fall 2010
Lecture 21

[N. Harvey]

Topics

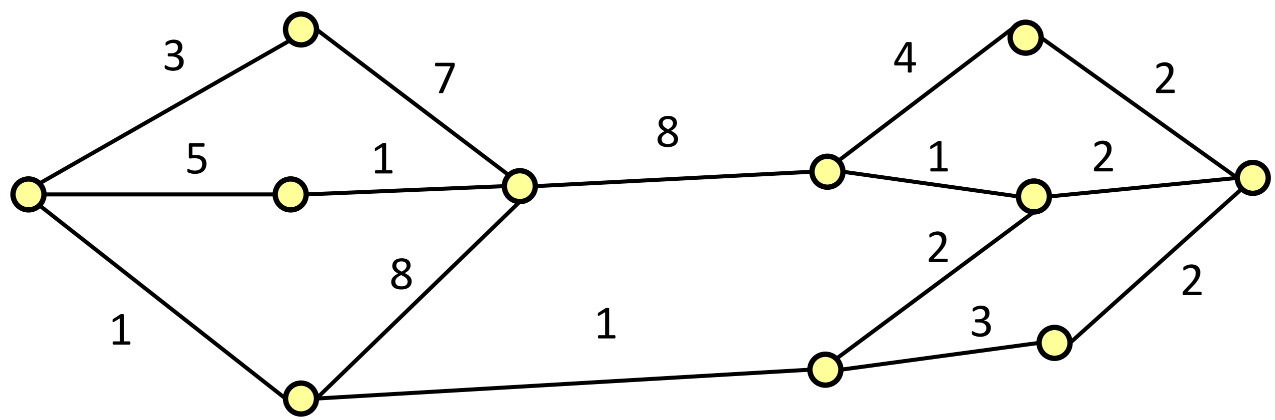
- Max Weight Spanning Tree Problem
- Spanning Tree Polytope
- Separation Oracle using Min s-t Cuts

Warning!

The point of this lecture is to do things in an unnecessarily complicated way.

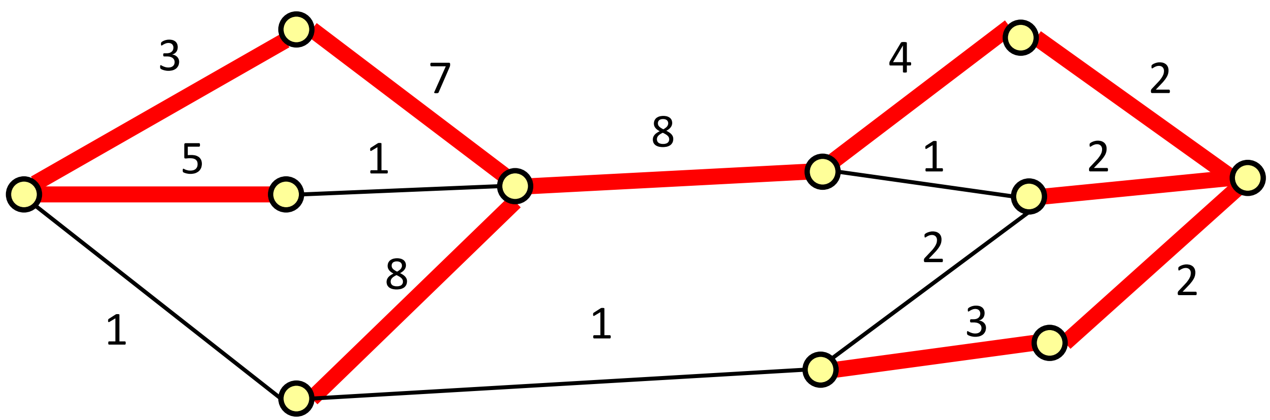
Spanning Tree

- Let G = (V,E) be a connected graph, n=|V|, m=|E|
- Edges are weighted: $w_e \in \mathbb{R}$ for every $e \in E$
- **Def:** A set $T \subseteq E$ is a **spanning tree** if (these are equivalent)
  - |T|=n-1 and T is acyclic
  - T is a maximal acyclic subgraph
  - T is a minimal connected, spanning subgraph



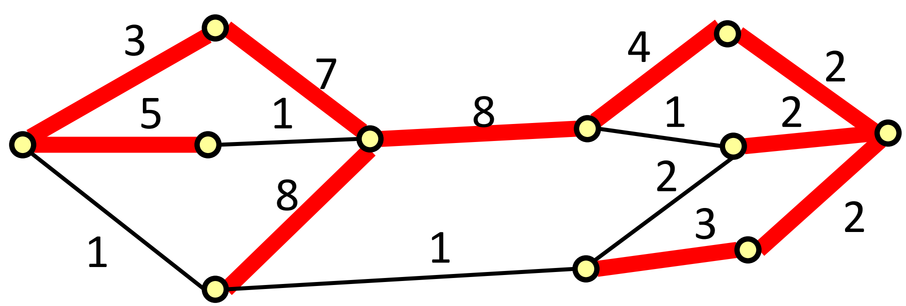
Spanning Tree

- Let $G = (V,E)$ be a connected graph, $n=|V|$, $m=|E|$
- Edges are weighted: $w_e \in \mathbb{R}$ for every $e \in E$
- **Def:** A set $T \subseteq E$ is a **spanning tree** if (these are equivalent)
  - $|T|=n-1$ and T is acyclic
  - T is a maximal acyclic subgraph
  - T is a minimal connected, spanning subgraph



Spanning Tree

- Let G = (V,E) be a connected graph, $n=|V|$, $m=|E|$
- Edges are weighted: $w_e \in \mathbb{R}$ for every $e \in E$
- **Def:** A set $T \subseteq E$ is a **spanning tree** if (these are equivalent)
  - $|T|=n-1$ and T is acyclic
  - T is a maximal acyclic subgraph
  - T is a minimal connected, spanning subgraph
- **Def:** $T \subseteq E$ is a **max weight spanning tree** if it maximizes $\sum_{e \in T} w_e$ over all spanning trees.



A Simple Properties of Trees

- For any $C \subseteq E$, let $\kappa(C)$ = # connected components in (V,C)
- **Examples:** $\kappa(E)=1$ and $\kappa(\emptyset)=n$
- **Claim:** Suppose T is a spanning tree.
  For every $C \subseteq E$, $|T \cap C| \leq n-\kappa(C)$.
- **Proof:** Let the connected components of (V,C) be (V_1,C_1), (V_2,C_2), ...
- So $V = \cup_i V_i$ and $C = \cup_i C_i$.
- Since $T \cap C_i$ is acyclic, $|T \cap C_i| \leq |V_i|-1$.
- So $|T \cap C| = \sum_{i=1}^{\kappa(C)} |T \cap C_i| \leq \sum_{i=1}^{\kappa(C)} (|V_i|-1) = n - \kappa(C).$ $\blacksquare$

Characteristic Vectors

- **Notation:** We consider vectors x assigning real numbers to the edges in E. We write this as $x \in \mathbb{R}^E$.
- **Notation:** For $C \subseteq E$, let $x(C) = \sum_{e \in C} x_e$.
- **Examples:**
  - the edge weights are $w \in \mathbb{R}^E$
  - For $T \subseteq E$, the **characteristic vector** of T is $x \in \mathbb{R}^E$ where

$$x_e = \begin{cases} 1 & (\text{if } e \in T) \\ 0 & (\text{otherwise}) \end{cases}$$

    For any $C \subseteq E$, $x(C) = |T \cap C| \leq n - \kappa(C)$.

    This is a **linear inequality** in x: $\sum_{e \in C} x_e \leq n - \kappa(C)$

Spanning Tree Polytope

- Since we know all these linear inequalities, why not assemble them into a polyhedron?
- Let $P_{ST} = \left\{ \begin{array}{rcl} x(E) & = & n-1 \\ x(C) & \leq & n-\kappa(C) \quad \forall C \subseteq E \\ x & \geq & 0 \end{array} \right\}$
- **Note:**
  - P_{ST} is a polyhedron, because x(E) and x(C) are linear functions of x
  - P_{ST} is a polytope, because $x_e = x(\{e\}) \leq 1$, so P_{ST} is bounded
  - If x is the characteristic vector of a spanning tree, then $x \in P_{ST}$

The Main Theorems

- **Theorem 1:** The LP max { $w^T x : x \in P_{ST}$ } can be solved in polynomial time.
  (In fact, we'll do this by the ellipsoid method!)

- **Theorem 2:** [Edmonds '71]
  The extreme points of P_{ST} are precisely the characteristic vectors of spanning trees of G.

- **Corollary:** A max weight spanning tree can be found in polynomial time.

- **Proof:** Solve the LP and find an extreme point x.

  x is the characteristic vector of a tree T of weight $w^T x$.

  Since $w^T x \geq w^T y$ for any other extreme point y,
  it follows that T is a max weight spanning tree. ■

"Polynomial Time"

- There are many algorithms for finding maximum weight spanning trees

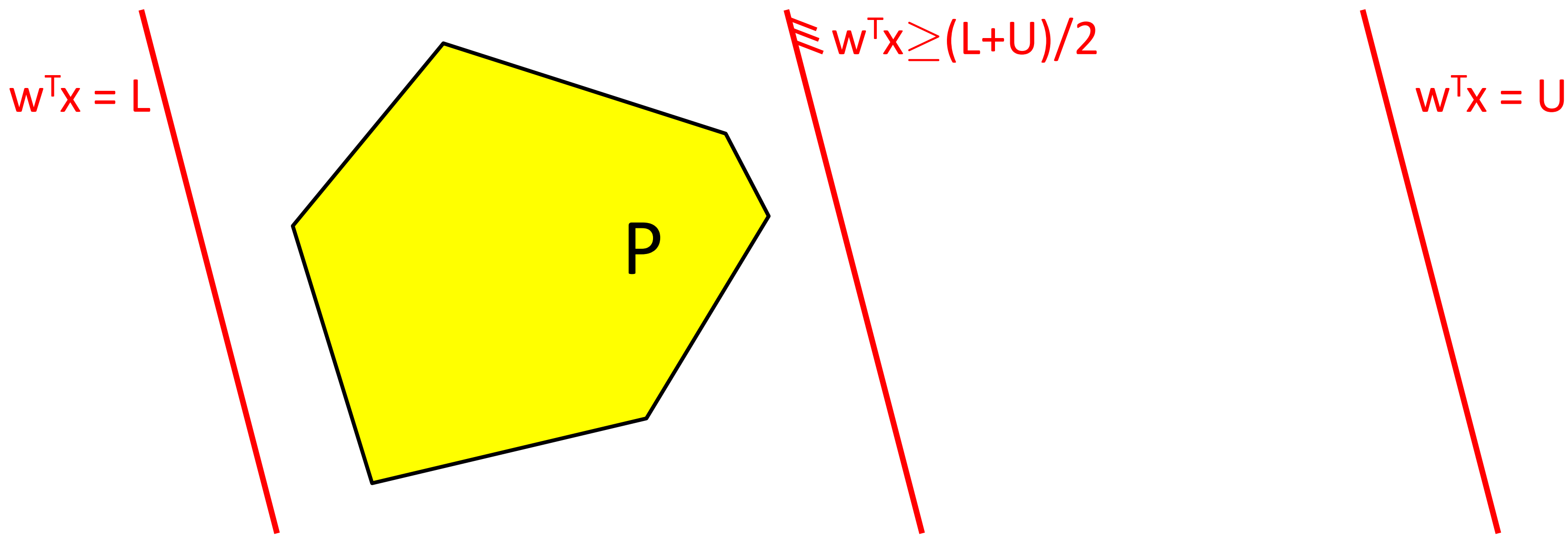
Name	Running Time
Prim's Algorithm	O(m log n)
Kruskal's Algorithm	O(m log n)
...with fancy data structures	O(m + n log n)
...even fancier data structures	O(m α(m,n))
Karger-Klein-Tarjan Algorithm	O(m) (randomized)
Pettie-Ramachandaran Algorithm	Optimal determistic alg, but runtime is unknown

- Our algorithm has running time something like O(m^{12})
- Hopelessly impractical! But illustrates important ideas.

Ellipsoid Method for Solving LPs

(from Lecture 8)

- Ellipsoid method can find a feasible point in P i.e., it can solve a system of inequalities
- But we want to optimize, i.e., solve max { $w^T x : x \in P$ }
- **One approach:** Binary search for optimal value
  - Suppose we know optimal value is in interval [L,U]
  - Add a new constraint $w^T x \geq (L+U)/2$
  - If LP still feasible, replace L with (L+U)/2 and repeat
  - If LP not feasible, replace U with (L+U)/2 and repeat



Applying the Ellipsoid Method

- By binary search, we need to decide feasibility of

$$P_{ST} = \left\{ \begin{array}{rcl} w^{\mathsf{T}} x & \geq & W \\ x(E) & = & n-1 \\ x(C) & \leq & n - \kappa(C) \quad \forall C \subseteq E \\ x & \geq & 0 \end{array} \right\}$$

- **Main obstacle:** Huge number of constraints! (2^m)
- **Recall:** Ellipsoid method works for **any convex set** P, as long as you can give a separation oracle.

Separation Oracle

Is $z \in P$?

If not, find a vector a s.t. $a^{\mathsf{T}}x < a^{\mathsf{T}}z \;\; \forall x \in P$

Applying the Ellipsoid Method

- By binary search, we need to decide feasibility of

$$P_{ST} = \left\{ \begin{array}{rcl} w^{\mathsf{T}} x & \geq & W \\ x(E) & = & n-1 \\ x(C) & \leq & n - \kappa(C) \quad \forall C \subseteq E \\ x & \geq & 0 \end{array} \right\}$$

- **Main obstacle:** Huge number of constraints! (2^m)
- **Recall:** Ellipsoid method works for **any convex set** P, as long as you can give a separation oracle.

> **Separation Oracle**
> Is $z \in P$?
> If not, find a **violated constraint**.

- How quickly can we test **these** constraints?
- **We'll show:** This can be done in time polynomial in n.

Ellipsoid method inside Ellipsoid method

Everything runs in polynomial time!

Minimum Spanning Tree Problem

Solve by Ellipsoid Method
Separation oracle uses…

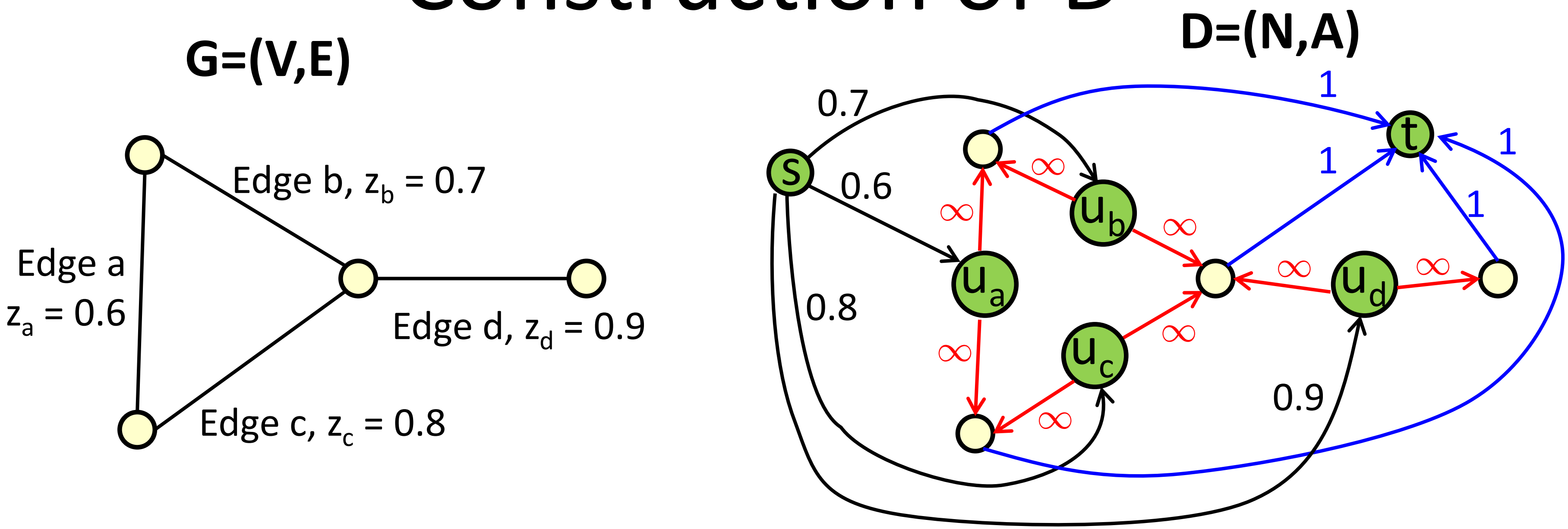
Minimum S-T Cut Problem

Solve by Ellipsoid Method!

Separation Oracle: Game Plan

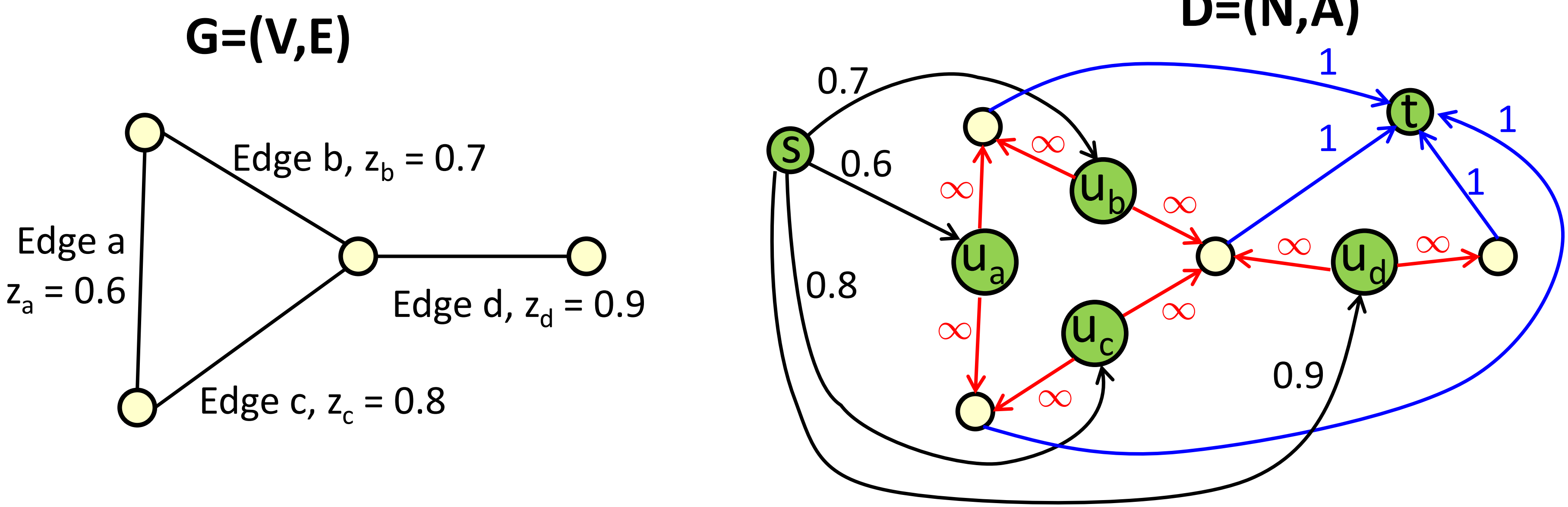
- We have graph $G=(V,E)$ and a point $z \in \mathbb{R}^E$
- We construct digraph $D=(N,A)$ with capacities $c \in \mathbb{R}^A$
- If s-t min cut in D is:
  - **Small:** this shows that z violates a constraint of P
  - **Large:** this shows that z is feasible for P

Construction of D



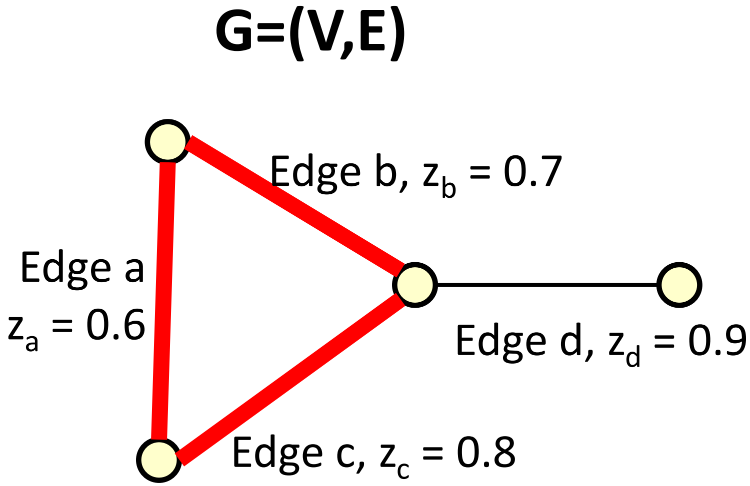
- **Nodes of D:** $N = V \cup \{s,t\} \cup \{ u_e : e \in E \}$
- **Arcs of D:**
  - Arc (s,u_e) of capacity z_e for every edge $e \in E$
  - Arc (v,t) of capacity 1 for every node $v \in V$
  - Infinite capacity arcs $(u_{\{v,w\}},v)$ and $(u_{\{v,w\}},w)$ for all $\{v,w\} \in E$

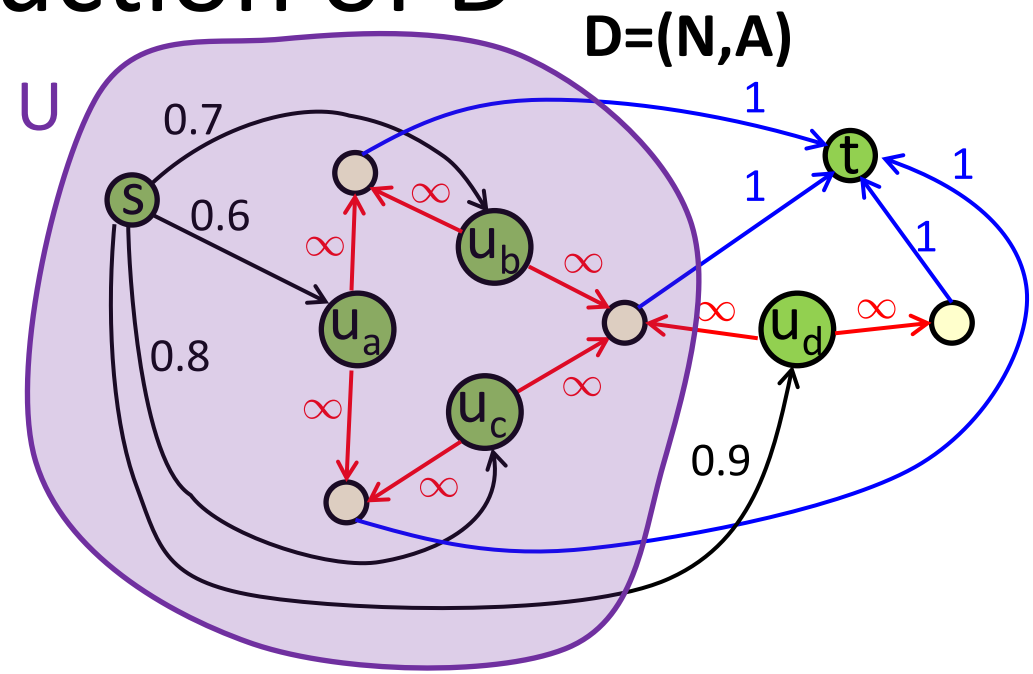
Construction of D



- **Lemma:** z is feasible $\Leftrightarrow$ every s-t cut has capacity $\geq$ n (except for the cut with only black edges).

Construction of D





- **Lemma:** z is feasible $\Leftrightarrow$ every s-t cut has capacity $\geq$ n (except for the cut with only black edges).
- **Example:**
  - $z(C) = 2.1 > n - \kappa(C) = 2$, so z is infeasible.
  - The s-t cut $\delta^+(U)$ in D has capacity $3.9 < n = 4$.

Separation Oracle Summary

- **Input:** $G=(V,E)$ and $z\in\mathbb{R}^E$
- Construct the graph $D=(N,A)$ and arc capacities
- For each $v\in V$
  - Temporarily add an infinity capacity arc (s,v)
  - Compute the s-t min cut value q
    (by the Ellipsoid Method)
  - If $q<n$
    - We obtain a set $S\subseteq V$ s.t. $z(E[S]) > |S|-1$
    - **Halt:** return this violated constraint
  - Remove the temporary arc
- End for
- **Halt:** z is feasible

Summary

- Some combinatorial objects are described by LPs of exponential size
- Even if an LP has exponential size, the ellipsoid method might be able to solve it "efficiently", if a separation oracle can be designed
- The separation oracle might use ellipsoid method too
- Ellipsoid method gives impractical algorithms, but these can be a "proof of concept" for realistic algorithms