# C&O 355
# Lecture 22

N. Harvey

# Topics

- Integral Polyhedra
- Minimum s-t Cuts via Ellipsoid Method
- Weight-Splitting Method
- Shortest Paths

# Minimum s-t Cuts

**Theorem:** Every optimal BFS of (LP) is optimal for (IP). (Lecture 21)

(IP)

$$\begin{aligned} \min \quad & \sum_{a \in A} c_a \cdot y_a & & \\ \text{s.t.} \quad & \sum_{a \in p} y_a & \geq 1 & \quad \forall p \in \mathcal{P} \\ & y_a & \in \{0,1\} & \quad \forall a \in A \end{aligned}$$

(LP)

$$\begin{aligned} \min \quad & \sum_{a \in A} c_a \cdot y_a & & \\ \text{s.t.} \quad & \sum_{a \in p} y_a & \geq 1 & \quad \forall p \in \mathcal{P} \\ & y_a & \geq 0 & \quad \forall a \in A \end{aligned}$$

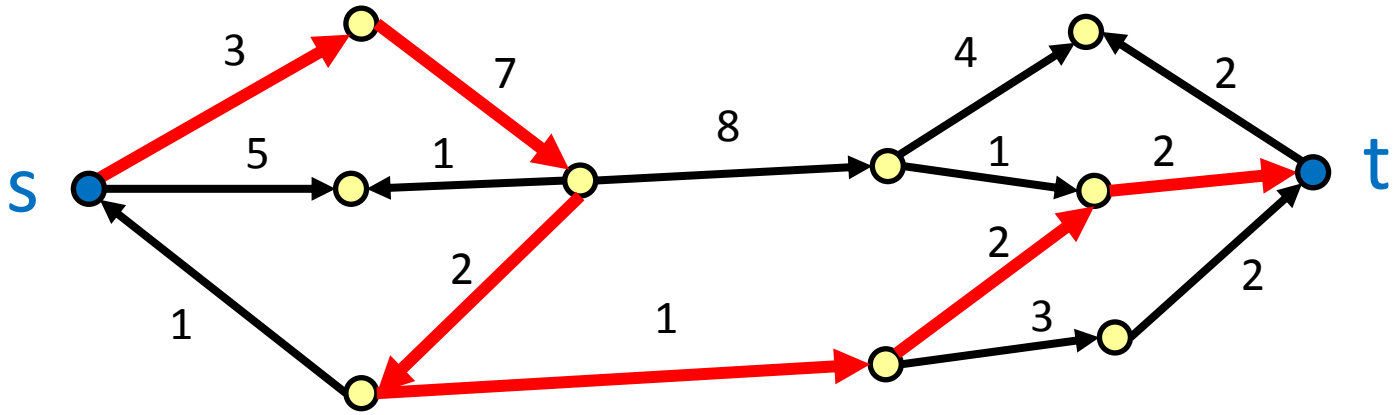
- So to solve (IP), we can just solve (LP) and return an optimal BFS.
- To solve (LP), the separation oracle is: (Lecture 12)

  Check if $y_a<0$ for any $a \in A$. If so, the constraint "$y_a \geq 0$" is violated.

  Check if $dist_y(s,t)<1$. If so, let p be an s-t path with $length_y(p)<1$. Then the constraint for path p is violated.
- So to compute min s-t cuts, we just need an algorithm to compute shortest dipaths!

# Shortest Paths in a Digraph

- Let G=(V,A) be a directed graph. Every arc a has a "length" $w_a$>0.
- Given two vertices s and t, find a path from s to t of minimum total length.




These edges form a **shortest s-t path**

# Shortest Paths in a Digraph

- Let b be vector with $b_s=1$, $b_t=-1$, $b_v=0$ $\forall v \in V \setminus \{s,t\}$
- Consider the IP:

$$\begin{aligned} \min \quad & \sum_{a \in A} w_a \cdot x_a & & \\ \text{s.t.} \quad & \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a & = b_v & \forall v \in V \\ & x_a & \in \{0,1\} & \forall a \in A \end{aligned}$$

- And the LP relaxation:

$$\begin{aligned} \min \quad & \sum_{a \in A} w_a \cdot x_a & & \\ \text{s.t.} \quad & \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a & = b_v & \forall v \in V \\ & 0 \leq x_a \leq 1 & & \forall a \in A \end{aligned}$$

**Claim:** Every optimal solution of (IP) is a shortest s-t path.

(Assignment 5)

**Theorem:** Every optimal BFS of (LP) is optimal for (IP).

# Our Min s-t Cut Algorithm

Minimum S-T Cut Problem

Solve by Ellipsoid Method

Separation oracle is…

Shortest Path Problem

Solve by Ellipsoid Method!

- Inner LP
  - Has $|V|$ constraints, $|A|$ variables.
  - Our analysis in Lecture 12: roughly $O(|A|^6)$ iterations (actually, depends on # bits to represent lengths w)
- Outer LP
  - Has $|\mathcal{P}|$ constraints, $|A|$ variables
  - Can show $O(|A|^2 (\log^2 |A| + \log^2 c_{max}))$ iterations suffice
- Total
  - Roughly $O(|A|^8)$ iterations, each taking roughly $O(|A|^3)$ time
  - Total running time: roughly $O(|A|^{11})$
  - Best-known algorithm has running time: $O(|A|^{1.5})$

# Combinatorial Algorithms

- We've used the ellipsoid method to prove several problems are solvable in "polynomial time"
  - LPs, Maximum Bipartite Matching, Max Weight Perfect Matching, Min s-t Cut, Shortest Paths
  - Approximate solutions to SDPs and some Convex Programs
- In practice, no one uses the ellipsoid method.
- It should be viewed as a "proof of concept" that efficient algorithms exist
- For many combinatorial optimization problems, combinatorial algorithms exist and are much faster
- Next: a slick way to design combinatorial algorithms, based on **weight splitting**.

# Weight-Splitting Method

- Let $C \subset \mathbb{R}^n$ be set of feasible solutions to some optimization problem.
- Let $w \in \mathbb{R}^n$ be a "weight vector".
- x is "optimal under w" if x optimizes min { $w^T y : y \in C$ }

- **Lemma:** Suppose $w = w_1 + w_2$. Suppose that x is optimal under $w_1$, and x is optimal under $w_2$. Then x is optimal under w.

- **Proof:** Let z be optimal under w. Then:

$$w^T x = w_1^T x + w_2^T x \leq w_1^T z + w_2^T z = w^T z$$

So x is also optimal under w. ■

Frank

Hassin

Bar-Yehuda

Even

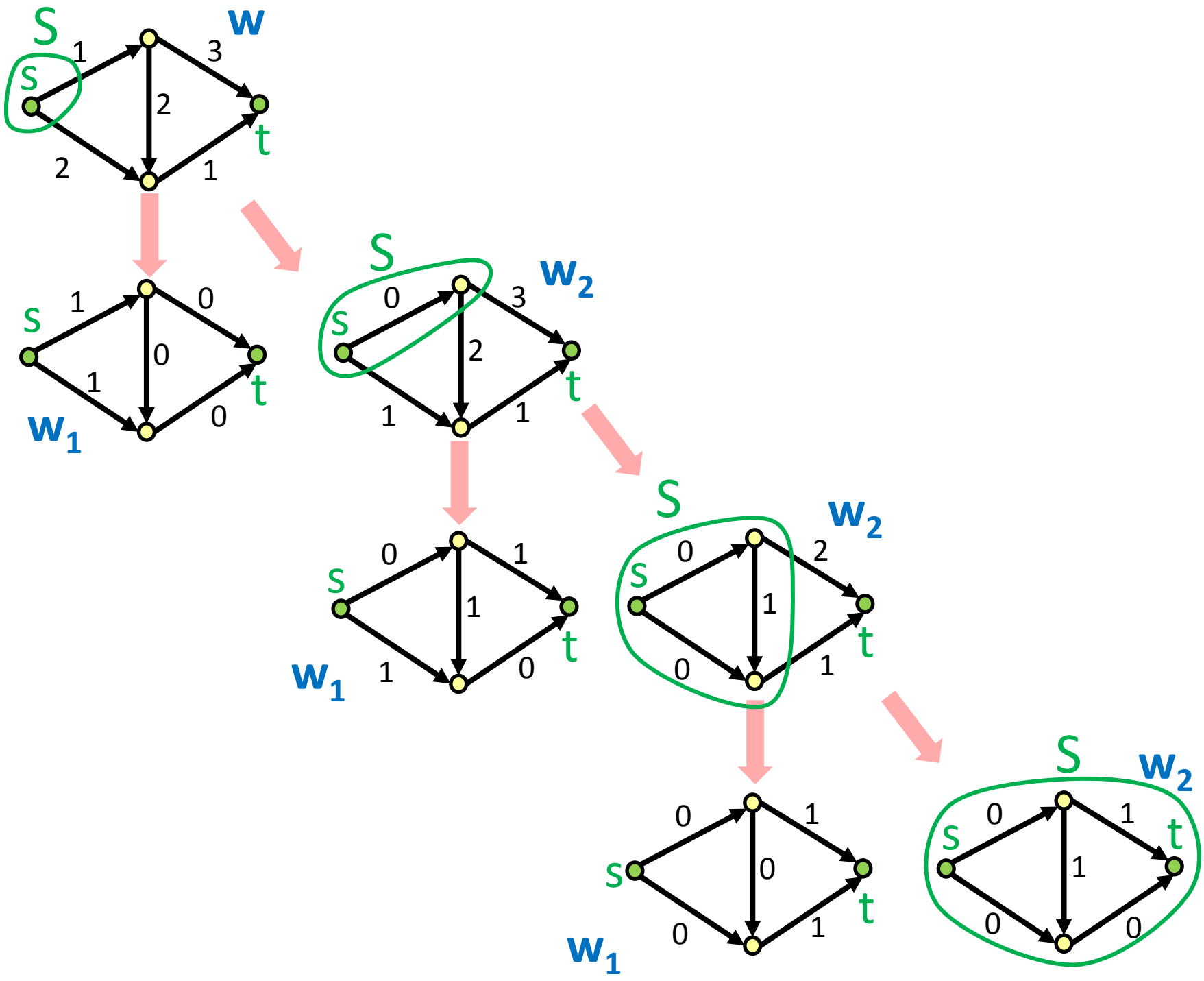
**ShortestPath**( G, S, t, w )

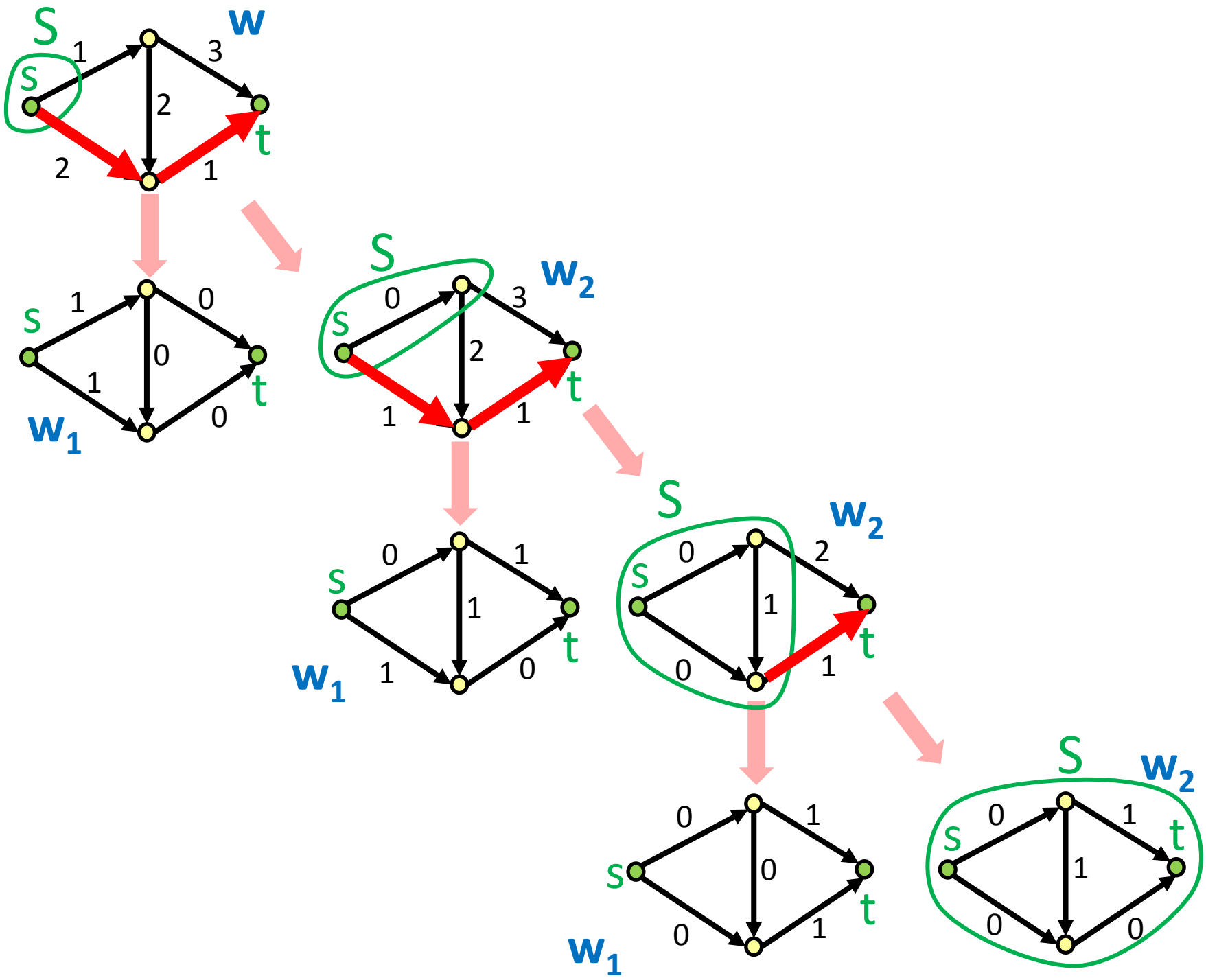
**Input:** Digraph G = (V,A), source vertices $S \subseteq V$, destination vertex $t \in V$, and integer lengths $w_a$, such that $w_a > 0$, unless both endpoints of a are in S.

**Output:** A shortest path from some $s \in S$ to t.

- If $t \in S$, return the empty path p=()
- Set $w_1(a)=1$ for all $a \in \delta^+(S)$, and $w_1(a)=0$ otherwise
- Set $w_2 = w - w_1$.
- Set $S' = S \cup \{ u : \exists s \in S$ with $w_2( (s,u) ) = 0 \}$
- Set $p' = (v_1,v_2\ldots,t) =$ **ShortestPath**( G, S', t, $w_2$ )
- If $v_1 \in S$, then set p=p'
- Else, set $p=(s,v_1,v_2,\ldots,t)$ where $s \in S$ and $w_2( (s,v_1) )=0$
- Return path p

To find shortest s-t path, run ShortestPath(G, {s}, t, w)


S
w
s
1
3
2
2
1
t
s
1
0
0
1
0
t
w1
S
w2
s
0
3
2
1
1
t
s
0
1
1
1
0
t
w1
S
w2
s
0
2
1
0
1
t
s
0
1
0
0
1
t
w1
S
w2
s
0
1
1
0
0
t


S
w
s
t
1
3
2
2
1
w1
w2

# Correctness of Algorithm

- **Claim:** Algorithm returns a shortest path from S to t.
- **Proof:** By induction on number of recursive calls.
- If $t \in S$, then the empty path is trivially shortest.
- Otherwise, p' is a shortest path from S' to t under $w_2$.
- So p is a shortest path from S to t under $w_2$.
  (Note: $\text{length}_{w2}(p)=\text{length}_{w2}(p')$, because if we added an arc, it has $w_2$-length 0.)
- Note: p cannot re-enter S, otherwise a subpath of p would be a shorter S-t path. So p uses exactly one arc of $\delta^+(S)$.
- So $\text{length}_{w1}(p)=1$. But **any** S-t path has length **at least** 1 under $w_1$. So p is a shortest path from S to t under $w_1$.
- By Weight-Splitting Lemma, p is a shortest S-t path under arc-lengths w. ■