CS 577- Intro to Algorithms

Network Flow (Part 2)

Dieter van Melkebeek

October 29, 2020

Recap - notions

Recap - notions

Network

- a digraph (V, E)
- edge capacities $c : E \to [0, \infty)$
- the source $s \in V$, which has indegree 0, and
- the sink $t \in V$, which has outdegree 0.

Recap - notions

Network

- a digraph (V, E)
- edge capacities $c : E \to [0, \infty)$
- the source $s \in V$, which has indegree 0, and
- the sink $t \in V$, which has outdegree 0.

Flow

A mapping $f : E \to [0, \infty)$ satisfying

- [capacity constraints] $(\forall e \in E)\, f(e) \leq c(e)$
- [conservation constraints] $(\forall v \in V \setminus \{s, t\})\, f_{\text{in}}(v) = f_{\text{out}}(v)$
  where $f_{\text{in}}(v) \doteq \sum_{u \in V : e \doteq (u,v) \in E} f(e)$ and
  $f_{\text{out}}(v) \doteq \sum_{u \in V : e \doteq (v,u) \in E} f(e)$.

Recap - notions

Network

- a digraph (V, E)
- edge capacities $c : E \to [0, \infty)$
- the source $s \in V$, which has indegree 0, and
- the sink $t \in V$, which has outdegree 0.

Flow

A mapping $f : E \to [0, \infty)$ satisfying

- [capacity constraints] $(\forall e \in E)\, f(e) \leq c(e)$
- [conservation constraints] $(\forall v \in V \setminus \{s, t\})\, f_{\text{in}}(v) = f_{\text{out}}(v)$
  where $f_{\text{in}}(v) \doteq \sum_{u \in V : e \doteq (u,v) \in E} f(e)$ and
  $f_{\text{out}}(v) \doteq \sum_{u \in V : e \doteq (v,u) \in E} f(e)$.

st-Cut

A partition (S, T) of V such that $s \in S$ and $t \in T$.

Recap - computational problems

Recap - computational problems

Max flow

Input: network $N = (V, E, c, s, t)$

Output: flow f such that $\nu(f) \doteq f_{\text{out}}(s)$ is maximized

Recap - computational problems

Max flow

Input: network $N = (V, E, c, s, t)$

Output: flow f such that $\nu(f) \doteq f_{\text{out}}(s)$ is maximized

Min cut

Input: network $N = (V, E, c, s, t)$

Output: st-cut (S, T) such that $c(S, T) \doteq \sum_{e \in S \times T} c(e)$ is minimized

Recap - properties

Recap - properties

- Invariance: For every flow f and st-cut (S, T)

$$f_{\text{out}}(S) - f_{\text{in}}(S) = \nu(f)$$

Recap - properties

- Invariance: For every flow f and st-cut (S, T)

$$f_{\text{out}}(S) - f_{\text{in}}(S) = \nu(f)$$

- Weak duality: For every flow f and st-cut (S, T)

$$\nu(f) \leq c(S, T)$$

Recap - properties

- Invariance: For every flow f and st-cut (S, T)

$$f_{\text{out}}(S) - f_{\text{in}}(S) = \nu(f)$$

- Weak duality: For every flow f and st-cut (S, T)

$$\nu(f) \leq c(S, T)$$

  Equality $\nu(f) = c(S, T)$ holds iff

Recap - properties

- Invariance: For every flow f and *st*-cut (S, T)

$$f_{\text{out}}(S) - f_{\text{in}}(S) = \nu(f)$$

- Weak duality: For every flow f and *st*-cut (S, T)

$$\nu(f) \leq c(S, T)$$

  Equality $\nu(f) = c(S, T)$ holds iff
    - $(\forall e \in E \cap S \times T)\, f(e) = c(e)$

Recap - properties

- Invariance: For every flow f and st-cut (S, T)
$$f_{\text{out}}(S) - f_{\text{in}}(S) = \nu(f)$$
- Weak duality: For every flow f and st-cut (S, T)
$$\nu(f) \leq c(S, T)$$
Equality $\nu(f) = c(S, T)$ holds iff
  - $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - $(\forall e \in E \cap T \times S)\, f(e) = 0.$

Recap - properties

- Invariance: For every flow f and *st*-cut (S, T)

$$f_{\text{out}}(S) - f_{\text{in}}(S) = \nu(f)$$

- Weak duality: For every flow f and *st*-cut (S, T)

$$\nu(f) \leq c(S, T)$$

  Equality $\nu(f) = c(S, T)$ holds iff
  - $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - $(\forall e \in E \cap T \times S)\, f(e) = 0.$

- Strong duality (today):

$$\max_{\text{flow } f} \nu(f) = \min_{st\text{-cut } (S,T)} c(S, T)$$

Residual Network

Consider a flow f in $N = (V, E, c, s, t)$.

Definition

The residual network $N_f = (V, E_f, c_f, s, t)$ has:

- For each $e \in E$ with $f(e) < c(e)$, an edge e in E_f with $c_f(e) \doteq c(e) - f(e)$.
- For each $e = (u, v) \in E$ with $f(e) > 0$, an edge $e' \doteq (v, u)$ in E_f with $c_f(e') \doteq f(e)$.

Path Augmentation

Scheme

1. Start with $f \equiv 0$.
2. While there is an st-path in N_f
   - Pick such a path P.
   - $f \leftarrow f +$ flow along P of value $\min_{e \in P}(c_f(e))$
3. Return f.

Path Augmentation

Scheme

1. Start with $f \equiv 0$.
2. While there is an st-path in N_f
   - Pick such a path P.
   - $f \leftarrow f +$ flow along P of value $\min_{e \in P}(c_f(e))$
3. Return f.

Soundness

If the algorithm produces an output, it is correct.

Path Augmentation

Scheme

1. Start with $f \equiv 0$.
2. While there is an st-path in N_f
   - Pick such a path P.
   - $f \leftarrow f +$ flow along P of value $\min_{e \in P}(c_f(e))$
3. Return f.

Soundness

If the algorithm produces an output, it is correct.

- f always is a valid flow.

Path Augmentation

Scheme

1. Start with $f \equiv 0$.
2. While there is an st-path in N_f
   - Pick such a path P.
   - $f \leftarrow f +$ flow along P of value $\min_{e \in P}(c_f(e))$
3. Return f.

Soundness

If the algorithm produces an output, it is correct.

- f always is a valid flow.
- If there is no st-path in N_f then $\nu(f)$ is maximized.

Path Augmentation

Scheme

1. Start with $f \equiv 0$.
2. While there is an st-path in N_f
   - Pick such a path P.
   - $f \leftarrow f+$ flow along P of value $\min_{e \in P}(c_f(e))$
3. Return f.

Soundness

If the algorithm produces an output, it is correct.

- f always is a valid flow.
- If there is no st-path in N_f then $\nu(f)$ is maximized.

Termination

Soundness

Soundness

Theorem

The following are equivalent:

Soundness

Theorem

The following are equivalent:

(1) f has maximum value.

Soundness

Theorem

The following are equivalent:

(1) f has maximum value.

(2) There is no st-path in N_f.

Soundness

Theorem

The following are equivalent:

(1) f has maximum value.

(2) There is no *st*-path in N_f.

(3) There exists an *st*-cut (S, T) such that
   - $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
   - $(\forall e \in E \cap T \times S)\, f(e) = 0.$

Soundness

Theorem

The following are equivalent:

(1) f has maximum value.

(2) There is no st-path in N_f.

(3) There exists an st-cut (S, T) such that
  - $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - $(\forall e \in E \cap T \times S)\, f(e) = 0$.

Proof

Soundness

Theorem

The following are equivalent:

(1) f has maximum value.

(2) There is no st-path in N_f.

(3) There exists an st-cut (S, T) such that

- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
- $(\forall e \in E \cap T \times S)\, f(e) = 0$.

Proof

(1) $\Rightarrow$ (2) Contrapositive follows by path augmentation.

Soundness

Theorem

The following are equivalent:

(1) f has maximum value.

(2) There is no st-path in N_f.

(3) There exists an st-cut (S, T) such that
   - $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
   - $(\forall e \in E \cap T \times S)\, f(e) = 0$.

Proof

(1) $\Rightarrow$ (2) Contrapositive follows by path augmentation.

(2) $\Rightarrow$ (3) Next slide.

Soundness

Theorem

The following are equivalent:

(1) f has maximum value.

(2) There is no st-path in N_f.

(3) There exists an st-cut (S, T) such that
- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
- $(\forall e \in E \cap T \times S)\, f(e) = 0$.

Proof

(1) $\Rightarrow$ (2) Contrapositive follows by path augmentation.

(2) $\Rightarrow$ (3) Next slide.

(3) $\Rightarrow$ (1) Follows from weak duality.

Soundness

Theorem

The following are equivalent:

(1) f has maximum value.

(2) There is no st-path in N_f.

(3) There exists an st-cut (S, T) such that

- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
- $(\forall e \in E \cap T \times S)\, f(e) = 0$.

Proof

(1) $\Rightarrow$ (2) Contrapositive follows by path augmentation.

(2) $\Rightarrow$ (3) Next slide.

(3) $\Rightarrow$ (1) Follows from weak duality.

Corollary: Strong duality

$$\max_{\text{flow } f} \nu(f) = \min_{st\text{-cut } (S,T)} c(S, T)$$

Construction of Minimum Cut

Construction of Minimum Cut

Suppose that there is no st-path in N_f.

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$
  - $t \in T$

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$
  - $t \in T$
- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$
  - $t \in T$
- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - Consider $e = (u, v) \in E$ with $u \in S$ and $f(e) < c(e)$.

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$
  - $t \in T$
- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - Consider $e = (u, v) \in E$ with $u \in S$ and $f(e) < c(e)$.
  - Then $(u, v) \in E_f$ so $v \in S$.

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$
  - $t \in T$
- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - Consider $e = (u, v) \in E$ with $u \in S$ and $f(e) < c(e)$.
  - Then $(u, v) \in E_f$ so $v \in S$.
- $(\forall e \in E \cap T \times S)\, f(e) = 0$

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$
  - $t \in T$
- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - Consider $e = (u, v) \in E$ with $u \in S$ and $f(e) < c(e)$.
  - Then $(u, v) \in E_f$ so $v \in S$.
- $(\forall e \in E \cap T \times S)\, f(e) = 0$
  - Consider $e = (u, v) \in E$ with $v \in S$ and $f(e) > 0$.

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$
  - $t \in T$
- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - Consider $e = (u, v) \in E$ with $u \in S$ and $f(e) < c(e)$.
  - Then $(u, v) \in E_f$ so $v \in S$.
- $(\forall e \in E \cap T \times S)\, f(e) = 0$
  - Consider $e = (u, v) \in E$ with $v \in S$ and $f(e) > 0$.
  - Then $e' \doteq (v, u) \in E_f$ so $u \in S$.

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$
  - $t \in T$
- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - Consider $e = (u, v) \in E$ with $u \in S$ and $f(e) < c(e)$.
  - Then $(u, v) \in E_f$ so $v \in S$.
- $(\forall e \in E \cap T \times S)\, f(e) = 0$
  - Consider $e = (u, v) \in E$ with $v \in S$ and $f(e) > 0$.
  - Then $e' \doteq (v, u) \in E_f$ so $u \in S$.
- $\therefore (S, T)$ is an st-cut with $c(S, T) = \nu(f)$.

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$
  - $t \in T$
- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - Consider $e = (u, v) \in E$ with $u \in S$ and $f(e) < c(e)$.
  - Then $(u, v) \in E_f$ so $v \in S$.
- $(\forall e \in E \cap T \times S)\, f(e) = 0$
  - Consider $e = (u, v) \in E$ with $v \in S$ and $f(e) > 0$.
  - Then $e' \doteq (v, u) \in E_f$ so $u \in S$.
- $\therefore (S, T)$ is an st-cut with $c(S, T) = \nu(f)$.
  By weak duality $\nu(f)$ is maximized and $c(S, T)$ minimized.

Construction of Minimum Cut

Suppose that there is no st-path in N_f. Consider

$$S \doteq \{v \in V : \text{ there exists an } sv\text{-path in } N_f\}.$$

- (S, T) with $T = V \setminus S$ is an st-cut in N.
  - $s \in S$
  - $t \in T$
- $(\forall e \in E \cap S \times T)\, f(e) = c(e)$
  - Consider $e = (u, v) \in E$ with $u \in S$ and $f(e) < c(e)$.
  - Then $(u, v) \in E_f$ so $v \in S$.
- $(\forall e \in E \cap T \times S)\, f(e) = 0$
  - Consider $e = (u, v) \in E$ with $v \in S$ and $f(e) > 0$.
  - Then $e' \doteq (v, u) \in E_f$ so $u \in S$.
- $\therefore (S, T)$ is an st-cut with $c(S, T) = \nu(f)$.
  By weak duality $\nu(f)$ is maximized and $c(S, T)$ minimized.
- Construction of (S, T) from f runs in linear time.

Path Augmentation - recap

Scheme

1. Start with $f \equiv 0$.
2. While there is an st-path in N_f
   - Pick such a path P.
   - $f \leftarrow f +$ flow along P of value $\min_{e \in P}(c_f(e))$
3. Return f.

Soundness

If the algorithm produces an output, it is correct.

- f always is a valid flow.
- If there is no st-path in N_f then $\nu(f)$ is maximized.

Termination

Termination and Running Time

Termination and Running Time

- Depends on the choice of augmenting path.

Termination and Running Time

- Depends on the choice of augmenting path.
- There exist instances and choices without termination.

Termination and Running Time

- Depends on the choice of augmenting path.
- There exist instances and choices without termination.

Integral capacities

If all capacities are integral, termination is guaranteed no matter how augmenting paths are picked.

Termination and Running Time

- Depends on the choice of augmenting path.
- There exist instances and choices without termination.

Integral capacities

If all capacities are integral, termination is guaranteed no matter how augmenting paths are picked.

- $\#$ augmentations $\leq F \doteq \max_{\text{flow } f} \nu(f)$

Termination and Running Time

- Depends on the choice of augmenting path.
- There exist instances and choices without termination.

Integral capacities

If all capacities are integral, termination is guaranteed no matter how augmenting paths are picked.

- $\#$ augmentations $\leq F \doteq \max_{\text{flow } f} \nu(f)$
- If each augmenting path is picked using linear-time graph traversal, running time is $O((n+m) \cdot F)$.

Termination and Running Time

- Depends on the choice of augmenting path.
- There exist instances and choices without termination.

Integral capacities

If all capacities are integral, termination is guaranteed no matter how augmenting paths are picked.

- $\#$ augmentations $\leq F \doteq \max_{\text{flow } f} \nu(f)$
- If each augmenting path is picked using linear-time graph traversal, running time is $O((n+m) \cdot F)$.
- Running time is *pseudopolynomial*, i.e., bounded by polynomial in size parameters (n and m) and *value* of numbers involved (capacities). [$F \leq \sum_{e \in E} c(e)$.]

Slow convergence

Better Algorithms

Better Algorithms

Augmentation along path of maximum residual capacity

Better Algorithms

Augmentation along path of maximum residual capacity

- $\#$ augmentations $= O(m \cdot \log F)$ for integral instances

Better Algorithms

Augmentation along path of maximum residual capacity

- $\#$ augmentations $= O(m \cdot \log F)$ for integral instances
- Finding one path: $O(n + m)$ time

Better Algorithms

Augmentation along path of maximum residual capacity

- $\#$ augmentations $= O(m \cdot \log F)$ for integral instances
- Finding one path: $O(n + m)$ time
- Overall running time: $O((n + m)m \cdot \log F)$

Better Algorithms

Augmentation along path of maximum residual capacity

- $\#$ augmentations $= O(m \cdot \log F)$ for integral instances
- Finding one path: $O(n+m)$ time
- Overall running time: $O((n+m)m \cdot \log F)$
- Running time is *polynomial*, i.e., bounded by polynomial in size parameters and *bitlength* of numbers involved.

Better Algorithms

Augmentation along path of maximum residual capacity

- $\#$ augmentations $= O(m \cdot \log F)$ for integral instances
- Finding one path: $O(n + m)$ time
- Overall running time: $O((n + m)m \cdot \log F)$
- Running time is *polynomial*, i.e., bounded by polynomial in size parameters and *bitlength* of numbers involved.

Augmentation along path with smallest number of edges

Better Algorithms

Augmentation along path of maximum residual capacity

- $\#$ augmentations $= O(m \cdot \log F)$ for integral instances
- Finding one path: $O(n+m)$ time
- Overall running time: $O((n+m)m \cdot \log F)$
- Running time is *polynomial*, i.e., bounded by polynomial in size parameters and *bitlength* of numbers involved.

Augmentation along path with smallest number of edges

- $\#$ augmentations $= O(nm)$ for all instances

Better Algorithms

Augmentation along path of maximum residual capacity

- $\#$ augmentations $= O(m \cdot \log F)$ for integral instances
- Finding one path: $O(n + m)$ time
- Overall running time: $O((n + m)m \cdot \log F)$
- Running time is *polynomial*, i.e., bounded by polynomial in size parameters and *bitlength* of numbers involved.

Augmentation along path with smallest number of edges

- $\#$ augmentations $= O(nm)$ for all instances
- Finding one path: $O(n + m)$ time (BFS)

Better Algorithms

Augmentation along path of maximum residual capacity

- # augmentations $= O(m \cdot \log F)$ for integral instances
- Finding one path: $O(n + m)$ time
- Overall running time: $O((n + m)m \cdot \log F)$
- Running time is *polynomial*, i.e., bounded by polynomial in size parameters and *bitlength* of numbers involved.

Augmentation along path with smallest number of edges

- # augmentations $= O(nm)$ for all instances
- Finding one path: $O(n + m)$ time (BFS)
- Overall running time: $O(nm(n + m))$

Better Algorithms

Augmentation along path of maximum residual capacity

- \# augmentations $= O(m \cdot \log F)$ for integral instances
- Finding one path: $O(n + m)$ time
- Overall running time: $O((n + m)m \cdot \log F)$
- Running time is *polynomial*, i.e., bounded by polynomial in size parameters and *bitlength* of numbers involved.

Augmentation along path with smallest number of edges

- \# augmentations $= O(nm)$ for all instances
- Finding one path: $O(n + m)$ time (BFS)
- Overall running time: $O(nm(n + m))$
- Running time is *strongly polynomial*, i.e., bounded by polynomial in size parameters only.

Better Algorithms

Augmentation along path of maximum residual capacity

- $\#$ augmentations $= O(m \cdot \log F)$ for integral instances
- Finding one path: $O(n+m)$ time
- Overall running time: $O((n+m)m \cdot \log F)$
- Running time is *polynomial*, i.e., bounded by polynomial in size parameters and *bitlength* of numbers involved.

Augmentation along path with smallest number of edges

- $\#$ augmentations $= O(nm)$ for all instances
- Finding one path: $O(n+m)$ time (BFS)
- Overall running time: $O(nm(n+m))$
- Running time is *strongly polynomial*, i.e., bounded by polynomial in size parameters only.

Other approaches: $O(nm)$ time for all instances