CS 577- Intro to Algorithms

Network Flow (Part 3)

Dieter van Melkebeek

November 3, 2020

Outline

Outline

Recap

- Notions: network, flow, cut
- Computational problems: max flow vs min cut

Outline

Recap

- Notions: network, flow, cut
- Computational problems: max flow vs min cut

Applications of max flow

- Bipartite matching
- Edge-disjoint paths
- Survey design

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 - notions

Integral network

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

Integral flow

A mapping $f : E \to \mathbb{N}$ 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

Complexity: time $O(nm)$

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

Complexity: time $O(nm)$

Note: integrality preserved

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

Complexity: time $O(nm)$

Note: integrality preserved

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 - 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

Complexity: time $O(nm)$

Note: integrality preserved

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

Complexity: time $O(n + m)$ when given max flow

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

Complexity: time $O(nm)$

Note: integrality preserved

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

Complexity: time $O(n + m)$ when given max flow

Duality

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

Bipartite Matching

Bipartite Matching

Definition

A matching M in a graph $G = (V, E)$ is a subset $M \subseteq E$ such that each $v \in V$ appears in at most one $e \in M$.

Bipartite Matching

Definition

A perfect matching M in a graph $G = (V, E)$ is a subset $M \subseteq E$ such that each $v \in V$ appears in exactly one $e \in M$.

Bipartite Matching

Definition

A perfect matching M in a graph $G = (V, E)$ is a subset $M \subseteq E$ such that each $v \in V$ appears in exactly one $e \in M$.

Computational problem

Input: bipartite graph G with bipartition (L, R)

Output: matching M such that $|M|$ is maximized

Bipartite Matching

Definition

A perfect matching M in a graph $G = (V, E)$ is a subset $M \subseteq E$ such that each $v \in V$ appears in exactly one $e \in M$.

Computational problem

Input: bipartite graph G with bipartition (L, R)

Output: matching M such that $|M|$ is maximized

Reduction to max flow

Bipartite Matching

Definition

A perfect matching M in a graph $G = (V, E)$ is a subset $M \subseteq E$ such that each $v \in V$ appears in exactly one $e \in M$.

Computational problem

Input: bipartite graph G with bipartition (L, R)

Output: matching M such that $|M|$ is maximized

Reduction to max flow

- Construction of integral network N such that

$$\text{matching } M \text{ in } G \quad \stackrel{\text{bijection}}{\longleftrightarrow} \quad \text{integral flow } f \text{ in } N$$

Bipartite Matching

Definition

A perfect matching M in a graph $G = (V, E)$ is a subset $M \subseteq E$ such that each $v \in V$ appears in exactly one $e \in M$.

Computational problem

Input: bipartite graph G with bipartition (L, R)

Output: matching M such that $|M|$ is maximized

Reduction to max flow

- Construction of integral network N such that

$$\begin{array}{ccc} \text{matching } M \text{ in } G & \xleftrightarrow{\text{bijection}} & \text{integral flow } f \text{ in } N \\ |M| & = & \nu(f) \end{array}$$

Bipartite Matching

Definition

A perfect matching M in a graph $G = (V, E)$ is a subset $M \subseteq E$ such that each $v \in V$ appears in exactly one $e \in M$.

Computational problem

Input: bipartite graph G with bipartition (L, R)

Output: matching M such that $|M|$ is maximized

Reduction to max flow

- Construction of integral network N such that

$$\begin{array}{ccc} \text{matching } M \text{ in } G & \stackrel{\text{bijection}}{\longleftrightarrow} & \text{integral flow } f \text{ in } N \\ |M| & = & \nu(f) \end{array}$$

- Resulting algorithm runs in time $O(nm)$.

Bipartite Matching – duality

Bipartite Matching – duality

Capacity of a cut in N

Bipartite Matching – duality

Capacity of a cut in N

- For any $A \subseteq V$, let $G(A)$ denote all the neighbors of A in G:
  $G(A) \doteq \{v : (\exists u \in A)\,(u, v) \in E\}$.

Bipartite Matching – duality

Capacity of a cut in N

- For any $A \subseteq V$, let $G(A)$ denote all the neighbors of A in G: $G(A) \doteq \{v : (\exists u \in A)\,(u, v) \in E\}$.
- $c(S, T)$ is finite iff $G(L \cap S) \subseteq S$.

Bipartite Matching – duality

Capacity of a cut in N

- For any $A \subseteq V$, let $G(A)$ denote all the neighbors of A in G: $G(A) \doteq \{v : (\exists u \in A)\,(u, v) \in E\}$.
- $c(S, T)$ is finite iff $G(L \cap S) \subseteq S$.
- If $c(S, T)$ is finite then $c(S, T) = |L \cap T| + |R \cap S|$.

Bipartite Matching – duality

Capacity of a cut in N

- For any $A \subseteq V$, let $G(A)$ denote all the neighbors of A in G: $G(A) \doteq \{v : (\exists u \in A)\,(u, v) \in E\}$.
- $c(S, T)$ is finite iff $G(L \cap S) \subseteq S$.
- If $c(S, T)$ is finite then $c(S, T) = |L \cap T| + |R \cap S|$.

Marriage theorem

A bipartite graph $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has a perfect matching iff

$$(\forall A \subseteq L)\,|G(A)| \geq |A|.$$

Bipartite Matching – duality

Capacity of a cut in N

- For any $A \subseteq V$, let $G(A)$ denote all the neighbors of A in G: $G(A) \doteq \{v : (\exists u \in A)\,(u, v) \in E\}$.
- $c(S, T)$ is finite iff $G(L \cap S) \subseteq S$.
- If $c(S, T)$ is finite then $c(S, T) = |L \cap T| + |R \cap S|$.

Marriage theorem

A bipartite graph $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has a perfect matching iff

$$(\forall A \subseteq L)\,|G(A)| \geq |A|.$$

Proof

Bipartite Matching – duality

Capacity of a cut in N

- For any $A \subseteq V$, let $G(A)$ denote all the neighbors of A in G: $G(A) \doteq \{v : (\exists u \in A)\,(u, v) \in E\}$.
- $c(S, T)$ is finite iff $G(L \cap S) \subseteq S$.
- If $c(S, T)$ is finite then $c(S, T) = |L \cap T| + |R \cap S|$.

Marriage theorem

A bipartite graph $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has a perfect matching iff

$$(\forall A \subseteq L)\,|G(A)| \geq |A|.$$

Proof

$\Rightarrow$: Contrapositive follows from definition of perfect matching.

Bipartite Matching – duality

Capacity of a cut in N

- For any $A \subseteq V$, let $G(A)$ denote all the neighbors of A in G: $G(A) \doteq \{v : (\exists u \in A)\,(u, v) \in E\}$.
- $c(S, T)$ is finite iff $G(L \cap S) \subseteq S$.
- If $c(S, T)$ is finite then $c(S, T) = |L \cap T| + |R \cap S|$.

Marriage theorem

A bipartite graph $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has a perfect matching iff

$$(\forall A \subseteq L)\, |G(A)| \geq |A|.$$

Proof

$\Rightarrow$: Contrapositive follows from definition of perfect matching.

$\Leftarrow$: Proof of contrapositive on next slide.

Proof of Marriage Theorem

- Suppose $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has no perfect matching.

Proof of Marriage Theorem

- Suppose $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has no perfect matching.
- $\nu(f) < n$ for every flow f in N.

Proof of Marriage Theorem

- Suppose $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has no perfect matching.
- $\nu(f) < n$ for every flow f in N.
- For every min cut (S, T) in N:

Proof of Marriage Theorem

- Suppose $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has no perfect matching.
- $\nu(f) < n$ for every flow f in N.
- For every min cut (S, T) in N:
  (1) $c(S, T) < n$

Proof of Marriage Theorem

- Suppose $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has no perfect matching.
- $\nu(f) < n$ for every flow f in N.
- For every min cut (S, T) in N:
  (1) $c(S, T) < n$
  (2) $G(L \cap S) \subseteq R \cap S$

Proof of Marriage Theorem

- Suppose $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has no perfect matching.
- $\nu(f) < n$ for every flow f in N.
- For every min cut (S, T) in N:
  (1) $c(S, T) < n$
  (2) $G(L \cap S) \subseteq R \cap S$
  (3) $c(S, T) = |L \cap T| + |R \cap S|$

Proof of Marriage Theorem

- Suppose $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has no perfect matching.
- $\nu(f) < n$ for every flow f in N.
- For every min cut (S, T) in N:
  (1) $c(S, T) < n$
  (2) $G(L \cap S) \subseteq R \cap S$
  (3) $c(S, T) = |L \cap T| + |R \cap S|$
- Consider $A \doteq L \cap S$.

Proof of Marriage Theorem

- Suppose $G = (V, E)$ with bipartition (L, R) and $|L| = |R| = n$ has no perfect matching.
- $\nu(f) < n$ for every flow f in N.
- For every min cut (S, T) in N:
  (1) $c(S, T) < n$
  (2) $G(L \cap S) \subseteq R \cap S$
  (3) $c(S, T) = |L \cap T| + |R \cap S|$
- Consider $A \doteq L \cap S$.

$$|G(A)| \stackrel{(2)}{\leq} |R \cap S| \stackrel{(3)}{=} c(S, T) - |L \cap T| \stackrel{(1)}{<} n - |L \cap T| = |L \cap S| = |A|$$

Edge-Disjoint Paths

Edge-Disjoint Paths

Computational problem

Input: digraph $G = (V, E)$; $s, t \in V$

Output: set C of edge-disjoint st-paths in G such that $|C|$ is maximized

Edge-Disjoint Paths

Computational problem

Input: digraph $G = (V, E)$; $s, t \in V$

Output: set C of edge-disjoint st-paths in G such that $|C|$ is maximized

Reduction to max flow

Edge-Disjoint Paths

Computational problem

Input: digraph $G = (V, E)$; $s, t \in V$

Output: set C of edge-disjoint st-paths in G such that $|C|$ is maximized

Reduction to max flow

- Integral network $N \doteq (V, E \setminus (V \times \{s\} \cup \{t\} \times V), c \equiv 1, s, t)$

Edge-Disjoint Paths

Computational problem

Input: digraph $G = (V, E)$; $s, t \in V$

Output: set C of edge-disjoint st-paths in G such that $|C|$ is maximized

Reduction to max flow

- Integral network $N \doteq (V, E \setminus (V \times \{s\} \cup \{t\} \times V), c \equiv 1, s, t)$

  set C of edge-disjoint st-paths in G $\longleftrightarrow$ integral flow f in N

Edge-Disjoint Paths

Computational problem

Input: digraph $G = (V, E)$; $s, t \in V$

Output: set C of edge-disjoint st-paths in G such that $|C|$ is maximized

Reduction to max flow

- Integral network $N \doteq (V, E \setminus (V \times \{s\} \cup \{t\} \times V), c \equiv 1, s, t)$

$$\begin{array}{ccc} \text{set } C \text{ of edge-disjoint } st\text{-paths in } G & \longleftrightarrow & \text{integral flow } f \text{ in } N \\ |C| & = & \nu(f) \end{array}$$

Edge-Disjoint Paths

Computational problem

Input: digraph $G = (V, E)$; $s, t \in V$

Output: set C of edge-disjoint st-paths in G such that $|C|$ is maximized

Reduction to max flow

- Integral network $N \doteq (V, E \setminus (V \times \{s\} \cup \{t\} \times V), c \equiv 1, s, t)$

  set C of edge-disjoint st-paths in $G \quad \longleftrightarrow \quad$ integral flow f in N

  $$|C| \quad = \quad \nu(f)$$

- Resulting algorithm runs in time $O(nm)$.

Edge-Disjoint Paths – duality

Edge-Disjoint Paths – duality

Capacity of a cut in N

Edge-Disjoint Paths – duality

Capacity of a cut in N

- $c(S, T) = |E \cap S \times T|$

Edge-Disjoint Paths – duality

Capacity of a cut in N

- $c(S, T) = |E \cap S \times T|$
- Removing $E \cap S \times T$ from G ensures no st-path remains.

Edge-Disjoint Paths – duality

Capacity of a cut in N

- $c(S, T) = |E \cap S \times T|$
- Removing $E \cap S \times T$ from G ensures no st-path remains.

Edge connectivity duality

The maximum number of edge-disjoint st-paths equals the minimum number of edges to be removed so no st-path remains.

Edge-Disjoint Paths – duality

Capacity of a cut in N

- $c(S, T) = |E \cap S \times T|$
- Removing $E \cap S \times T$ from G ensures no st-path remains.

Edge connectivity duality

The maximum number of edge-disjoint st-paths equals the minimum number of edges to be removed so no st-path remains.

Proof

Edge-Disjoint Paths – duality

Capacity of a cut in N

- $c(S, T) = |E \cap S \times T|$
- Removing $E \cap S \times T$ from G ensures no *st*-path remains.

Edge connectivity duality

The maximum number of edge-disjoint *st*-paths equals the minimum number of edges to be removed so no *st*-path remains.

Proof

- $\ell \doteq$ max number of edge-disjoint *st*-paths

Edge-Disjoint Paths – duality

Capacity of a cut in N

- $c(S, T) = |E \cap S \times T|$
- Removing $E \cap S \times T$ from G ensures no st-path remains.

Edge connectivity duality

The maximum number of edge-disjoint st-paths equals the minimum number of edges to be removed so no st-path remains.

Proof

- $\ell \doteq$ max number of edge-disjoint st-paths
- $r \doteq$ min number of edges to be removed so no st-path

Edge-Disjoint Paths – duality

Capacity of a cut in N

- $c(S, T) = |E \cap S \times T|$
- Removing $E \cap S \times T$ from G ensures no *st*-path remains.

Edge connectivity duality

The maximum number of edge-disjoint *st*-paths equals the minimum number of edges to be removed so no *st*-path remains.

Proof

- $\ell \doteq$ max number of edge-disjoint *st*-paths
- $r \doteq$ min number of edges to be removed so no *st*-path
- $\max_{\text{flow } f}(\nu(f)) = \ell \leq r \leq \min_{st\text{-cut } (S,T)}(c(S, T))$

Edge-Disjoint Paths – duality

Capacity of a cut in N

- $c(S, T) = |E \cap S \times T|$
- Removing $E \cap S \times T$ from G ensures no st-path remains.

Edge connectivity duality

The maximum number of edge-disjoint st-paths equals the minimum number of edges to be removed so no st-path remains.

Proof

- $\ell \doteq$ max number of edge-disjoint st-paths
- $r \doteq$ min number of edges to be removed so no st-path
- $\max_{\text{flow } f}(\nu(f)) = \ell \leq r \leq \min_{st\text{-cut } (S,T)}(c(S, T))$
- By duality, LHS = RHS so $\ell = r$.

Survey Design

Survey Design

Computational problem

Input: n customers $i \in [n]$
m products $j \in [m]$
$S_i \subseteq [m]$: products that customer $i \in [n]$ can survey
$c_i \in \mathbb{N}$: max number of surveys for customer $i \in [n]$
$p_j \in \mathbb{N}$: min number of surveys of product $j \in [m]$

Output: set $D \subset [n] \times [m]$ such that

- $(\forall (i,j) \in D)\, j \in S_i$
- $(\forall i \in [n])\, |\{j \in [m] : (i,j) \in D\}| \leq c_i$
- $(\forall j \in [m])\, |\{i \in [m] : (i,j) \in D\}| \geq p_j$

Survey Design

Computational problem

Input: n customers $i \in [n]$
m products $j \in [m]$
$S_i \subseteq [m]$: products that customer $i \in [n]$ can survey
$c_i \in \mathbb{N}$: max number of surveys for customer $i \in [n]$
$p_j \in \mathbb{N}$: min number of surveys of product $j \in [m]$

Output: set $D \subset [n] \times [m]$ such that

- $(\forall (i,j) \in D)\, j \in S_i$
- $(\forall i \in [n])\, |\{j \in [m] : (i,j) \in D\}| \leq c_i$
- $(\forall j \in [m])\, |\{i \in [m] : (i,j) \in D\}| \geq p_j$

Model

Survey Design

Computational problem

Input: n customers $i \in [n]$
m products $j \in [m]$
$S_i \subseteq [m]$: products that customer $i \in [n]$ can survey
$c_i \in \mathbb{N}$: max number of surveys for customer $i \in [n]$
$p_j \in \mathbb{N}$: min number of surveys of product $j \in [m]$

Output: set $D \subset [n] \times [m]$ such that

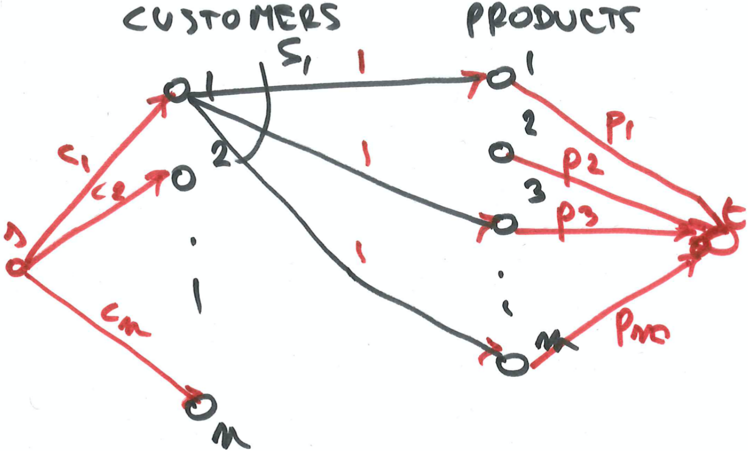
- $(\forall (i,j) \in D)\, j \in S_i$
- $(\forall i \in [n])\, |\{j \in [m] : (i,j) \in D\}| \leq c_i$
- $(\forall j \in [m])\, |\{i \in [m] : (i,j) \in D\}| \geq p_j$

Model

Represent each $(i,j) \in D$ as a unit of flow that passes through customer i and product j.

Survey Design – reduction to max flow

Survey Design – reduction to max flow



$$S_1 = \{1, 3, m\}$$

Survey Design – analysis

Survey Design – analysis

- Minimal survey D satisfies

$$(\forall j \in [m])\,|\{i \in [m] : (i,j) \in D\}| = p_j.$$

Survey Design – analysis

- Minimal survey D satisfies
$$(\forall j \in [m])\, |\{i \in [m] : (i,j) \in D\}| = p_j.$$
- Construction of integral network N such that
$$\text{minimal survey } D \quad \stackrel{\text{bijection}}{\longleftrightarrow} \quad \text{integral flow } f \text{ with } \nu(f) = \sum_{j \in [m]} p_j$$
$$D \quad = \quad \text{middle edges that carry flow}$$

Survey Design – analysis

- Minimal survey D satisfies

$$(\forall j \in [m])\, |\{i \in [m] : (i,j) \in D\}| = p_j.$$

- Construction of integral network N such that

$$\begin{aligned} \text{minimal survey } D \quad &\overset{\text{bijection}}{\longleftrightarrow} \quad \text{integral flow } f \text{ with } \nu(f) = \sum_{j \in [m]} p_j \\ D \quad &= \quad \text{middle edges that carry flow} \end{aligned}$$

- Resulting algorithm runs in time $O((n+m)(n+m+\sum_{i\in[n]} |S_i|))$.