CS 573: Algorithms, Fall 2014

Network Flow III – Applications

Lecture 13

October 9, 2014

Part I

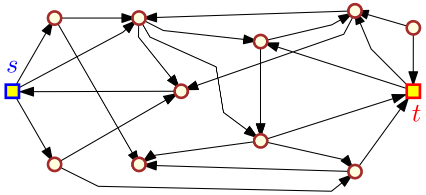
Edge disjoint paths

Edge disjoint paths

question

G: graph (dir/undir). s, t: vertices. k: parameter.

Task: Compute k paths from s to t that are ***edge disjoint***



1. Convert **G** into network flow H.
2. Capacities 1.
3. Compute max flow in **H**.
4. Value of flow = # of edge disjoint paths.

Edge Disjoint paths lemma

Lemma

$\exists k$ edge disjoint s-t paths in **G**
$\implies$ *max flow value H is at least k.*

Proof.

Given k such edge disjoint paths, push one unit of flow along each such path. The resulting flow is legal in h and it has value k. $\square$

Definition ($\mathbf{0/1}$-flow)

A flow f is a $\mathbf{0/1}$-***flow*** if every edge has either no flow on it, or one unit of flow.

0/1 flow

Lemma

f: $0/1$ flow in H with flow value μ. Then there are μ edge disjoint paths between s and t in H.

proof

1. Induction on # edges $\in H$ with 1 unit of flow on them.
   If $\mu = 0$...
2. Otherwise... Travel from s on edges with flow 1. Extract path. Repeat.
3. If reached t. Take path π. Reduce flow along π.
   H'/f': new network/flow
   $|f'| = \mu - 1$, H' has less edges,
4. By induction: has $\mu - 1$ edge disjoint paths in H' between s and t. With π this forms μ such paths.

0/1 flow proof continued

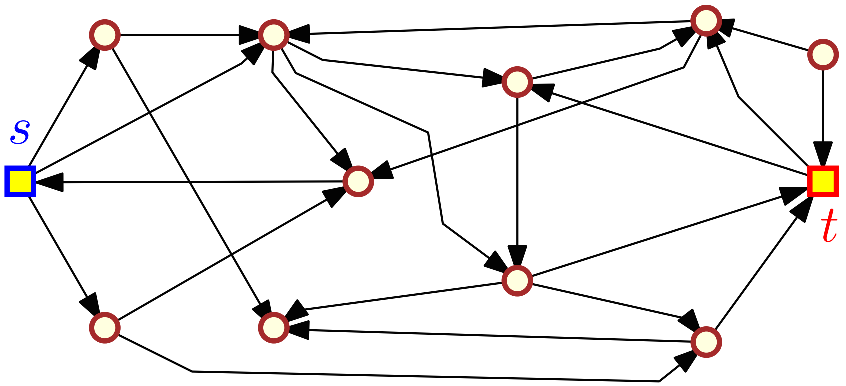
Lemma

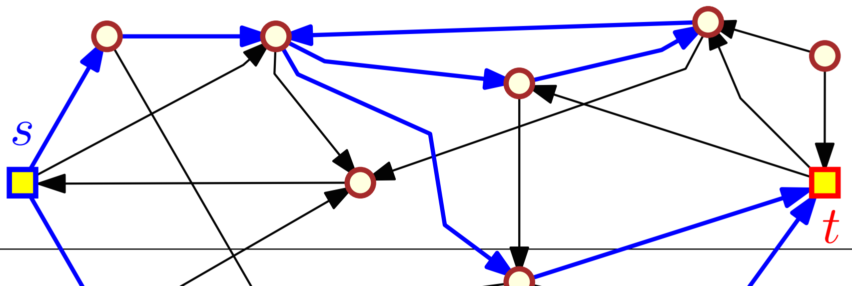
f: $0/1$ flow in H with flow value μ. Then there are μ edge disjoint paths between s and t in H.

Proof continued

1. If visit a vertex v for the second time (while extracting π).
2. Traversal contains a cycle C.
3. C edges in H have flow 1 on them.
4. Set flow along edges of C to 0.
5. Induction on the remaining graph.
6. Value of f did not change by removing C.
7. By induction $\exists \mu$ edge disjoint paths $s \rightsquigarrow t$ in H. ■

Example: Extracting paths





Extracting paths

1. $\mathbf{G}$ is simple
2. $\implies \leq n = |\mathbf{V(H)}|$ edges leaving s.
3. max flow in $\mathbf{H}$ is $\leq n$.
4. Ford-Fulkerson takes $O(mn)$ time.
5. Extraction of paths takes linear time (by proof).

Theorem

$\mathbf{G}$: directed graph, n vertices, m edges, s, t vertices. Compute max # edge disjoint paths from s to t in $O(mn)$ time.

$\#$ edge disjoint paths

Max-flow min-cut theorem strikes again!

Lemma

$\mathbf{G}$, s, t as above.

$\boxed{\text{Max \# edge disjoint } s-t \text{ paths}} = \boxed{\text{min \# edges whose removal separates } s \text{ from } t.}$

Proof.

1. U: set of edge-disjoint paths from s to t.
2. F: set of edges removing them separates s from t
3. every path in U contains edge of F. $\implies |U| \leq |F|$.
4. F: form a cut in $\mathbf{G}$ between s and t.
5. F minimal $= s - t$ min cut.
6. max-flow mincut theorem $|F| =$ max flow in $\mathbf{G}$
7. $\implies \exists \quad |F|$ disjoint paths in $\mathbf{G}$. $\implies |F| \leq |U|$. □

Edge-disjoint paths in undirected graphs

Problem

$\mathbf{G}$: undirected graph $\mathbf{G}$, s and t, find max $\#$ edge-disjoint paths between s and t.

1. Duplicate every edge in $\mathbf{G} \implies$ directed graph $\mathbf{H}$, apply algorithm for directed case.
2. Problem: flow f might use simultaneously edge in both directions: $(u \to v)$ and $(v \to u)$.
3. Solution: Remove $\mathbf{2}$-cycle! Repeat.
4. Then use algorithm for directed case...

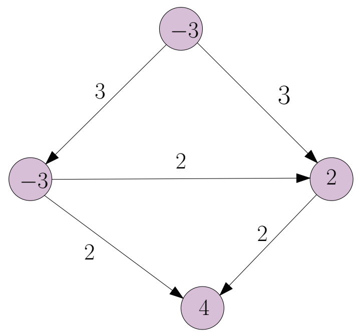
Lemma

$\exists \quad k$ edge-disjoint $s - t$ paths in undirected $\mathbf{G} \iff$ max flow in (directed) graph is at least k.
Paths in $\mathbf{G}$ computed in $O(mn)$ time (Ford-Fulkerson).

Part II

Circulations with demands

Circulations with demands



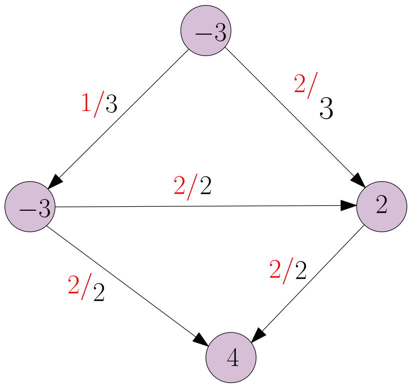
$\mathbf{G} = (\mathbf{V}, \mathbf{E})$.
$\forall v \in \mathbf{V}$ there is a ***demand*** d_v:

1. $d_v > 0$: sink requiring d_v flow into this node.
2. $d_v < 0$: source with $-d_v$ units of flow leaving it.
3. $d_v = 0$: regular node.

S set of source vertices
T: set of sink vertices.

A circulation with demands: example

A valid circulation for the given instance



Definition: Circulation with demands

Definition

circulation with demands $\{d_v\}$ is a function $f : E(G) \to \mathbb{R}^+$:

- Capacity condition: $\forall e \in E$: $f(e) \le c(e)$.
- Conservation condition: $\forall v \in V$: $f^{in}(v) - f^{out}(v) = d_v$.

Where:

1. $f^{in}(v)$ flow into v.
2. $f^{out}(v)$: flow out of v.

Problem

Is there a circulation that comply with the demand requirements?

Feasible circulation lemma

Lemma

If there is a feasible circulation with demands $\{d_v\}$*, then* $\sum_v d_v = 0$.

Proof.

1. Circulation $\implies \forall v \quad d_v = f^{in}(v) - f^{out}(v)$.
2. $\sum_{v \in V} d_v = \sum_v f^{in}(v) - \sum_v f^{out}(v)$
3. Flow on every edge is summed twice, one with positive sign, one with negative sign.
4. $\implies \sum_v d_v = \sum_v f^{in}(v) - \sum_v f^{out}(v) = 0$,

□

Computing circulations

$\exists$ feasible circulation only if

$$D = \sum_{v, d_v > 0} d_v = \sum_{v, d_v < 0} -d_v.$$

Algorithm for computing circulation

(A) $G = (V, E)$: input network with demands on vertices.
(B) Check $D = \sum_{v, d_v > 0} d_v = \sum_{v, d_v < 0} -d_v$.
(C) Create super source s. Connect to all v with $d_v < 0$. Set capacity $(s \to v)$ to $-d_v$.
(D) Create super sink t. Connect to all vertices u with $d_u > 0$. Set capacity $(u \to t)$ to d_u.
(E) H: new network flow. Compute max-flow f in H from s to t.
(F) If $|f| = D \implies \exists$ valid circulation. Easy to recover.

Result: Circulations with demands

Theorem

$\exists$ feasible circulation with demands $\{d_v\}$ in $\mathbf{G}$ $\iff$ max-flow in $\mathbf{H}$ has value D.

Integrality: *If all capacities and demands in $\mathbf{G}$ are integers, and there is a feasible circulation, then there is a feasible circulation that is integer valued.*

Part III

Circulations with demands and lower bounds

Circulations with demands and lower bounds

1. circulation and demands + for each edge a lower bound on flow.
2. $\forall e \in E(\mathbf{G})$: $\ell(e) \leq c(e)$.
3. Compute f such that $\forall e \quad \ell(e) \leq f(e) \leq c(e)$.
4. Be stupid! Consider flow: $\forall e \quad f_0(e) = \ell(e)$.
5. f_0 violates conservation of flow!

$$L_v = f_0^{in}(v) - f_0^{out}(v) = \sum_{e \text{ into } v} \ell(e) - \sum_{e \text{ out of } v} \ell(e).$$

6. If $L_v = d_v$, then no problem.
7. Fix-up demand: $\forall v \quad d'_v = d_v - L_v$.
   Fix-up capacity: $c'(e) = c(e) - \ell(e)$.
8. $\mathbf{G}'$: new network w. new demands/capacities (no lower bounds!)
9. Compute circulation f' on $\mathbf{G}'$.
   $\implies$ The flow $f = f_0 + f'$, is a legal circulation,

Circulations with demands and lower bounds

Lemma

$\exists$ feasible circulation in $\mathbf{G}$ $\iff$ $\exists$ feasible circulation in $\mathbf{G}'$.
Integrality: *If all numbers are integers $\implies$ $\exists$ integral feasible circulation.*

Proof.

Let f' be a circulation in $\mathbf{G}'$. Let $f(e) = f_0(e) + f'(e)$.
Clearly, f satisfies the capacity condition in $\mathbf{G}$, and the lower bounds.
$f^{in}(v) - f^{out}(v) = \sum_{e \text{ into } v}(\ell(e) + f'(e)) - \sum_{e \text{ out of } v}(\ell(e) + f'(e)) = L_v + (d_v - L_v) = d_v$.
f: valid circulation in $\mathbf{G}$. Then $f'(e) = f(e) - \ell(e)$ is a valid circulation for $\mathbf{G}'$. $\square$

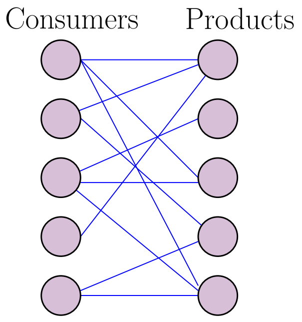
Part IV

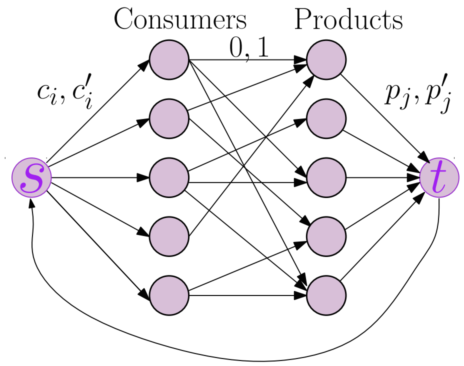
Applications

Survey design

1. Ask "Consumer $\boldsymbol{i}$: what did you think of product $\boldsymbol{j}$?"
2. $\boldsymbol{i}$th consumer willing to answer between $\boldsymbol{c_i}$ to $\boldsymbol{c'_i}$ questions.
3. For each product $\boldsymbol{j}$: at least $\boldsymbol{p_j}$ opinions, no more than $\boldsymbol{p'_j}$ opinions.
4. Full knowledge which consumers can be asked on which products.
5. Problem: How to assign questions to consumers?

Survey design...





Result...

Lemma

Given $\boldsymbol{n}$ consumers and $\boldsymbol{u}$ products with their constraints $\boldsymbol{c_1, c'_1, c_2, c'_2, \ldots, c_n, c'_n, p_1, p'_1, \ldots, p_u, p'_u}$ and a list of length $\boldsymbol{m}$ of which products where used by which consumers. An algorithm can compute a valid survey under these constraints, if such a survey exists, in time $\boldsymbol{O((n+u)m^2)}$.