CS 491G Combinatorial Optimization

Lecture Notes

Jingjing Chen

1. Review

Question: Given a directed graph G and an algorithm to find the maximum flow between r and s, how do we find the minimum capacity subset of edges such that removing them disconnect r from s**?**

(1) Find the max-flow $\vec{x}'$
(2) Construct $G(\vec{x}')$
(3) Find all vertices which can be reached from r call this set R, $\delta(R)$ is the min-cut as previous Max-Flow Min-Cut Theorem.

2. Minimum cuts in undirected graphs

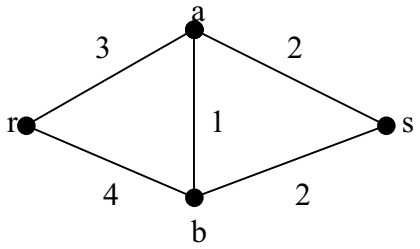
Now let us begin with a discussion of "global minimum cut" problem for undirected graphs.

2.1 Global Minimum Cuts

Minimum Cut Problem

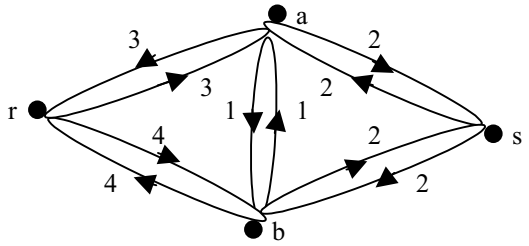
Given a connected, undirected graph $G = (V, E)$ and $u_e > 0$ for all $e \in$ E, find a set $\delta(S)$ such that $\Phi \subset S \subset V$ and $u(\delta(S))$ is minimum.

We say that nodes v, w of G are ***separated*** by a cut $\delta(S)$, if exactly one of v, w is in S. We surely can use flow techniques to solve above problem, now look at the following undirected graph G:



(G)

Replace each undirected edge by a pair of oppositely directed arcs and give them the same capacity as the edge shown on graph G_1 :



(G_1)

Claim: Every cut of *G* corresponds to a cut of G_1.

Proof: $\Rightarrow$ The fact that a cut of *G* corresponds to a cut of G_1 is obvious.

$\Leftarrow$ Suppose $\delta(S_1)$ is a cut of G_1. Every directed edge in $\delta(S_1)$ has a corresponding undirected edge in *G*. Denote these edges by $\delta(S)$. If $\delta(S)$ is not a cut of *G*, we can find a undirected path *P* from *r* to *s* in *G*, such that there must have a corresponding directed path P_1 from *r* to *s* in G_1, a contradiction with the fact that $\delta(S_1)$ is a cut of G_1. Done.

Corollary: Every min-cut of *G* corresponds to a min-cut of G_1.

If we fix one node *r* of G_1, every cut of G_1 is a *(r, s)*-cut for some node *s*. Since there are *n* nodes and we know that the running time of push-relabel algorithm is $O(n^3)$. Therefore, we can solve the min-cut problem by solving $n-1$ max-flow problems, and the running time is $O(n^4)$.

It is convenient to use the notation $\lambda(G)$ for the capacity of a minimum cut of *G*, and $\lambda(G;v,w)$ for the capacity of a minimum (*v, w*)-cut of *G*.

2.2 Node Identification

Let *v*, *w* be distinct nodes, then G_{vw} is obtained by ***identifying*** *v* with *w*. we put $V(G_{vw}) = (V \setminus \{v,w\}) \cup \{x\}$, where x is a new node, and $E(G_{vw}) = E \setminus \gamma(\{v,w\})$; for each edge $e \in E'$ and end p of e in G, p is an end of e in G_{vw} if $p \neq v,w,$ and otherwise x is an end of e in G_{vw}.The edges of G_{vw} have the same capacities as they had in G. Figure 2.2 shows the effect of identifying nodes c,d in the example of figure 2.1.

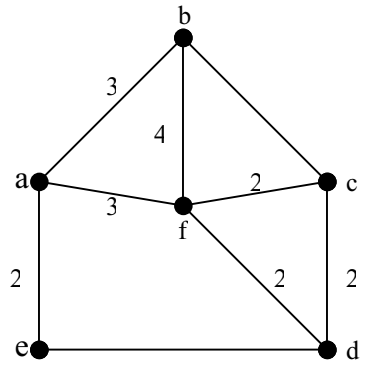


Figure 2.1

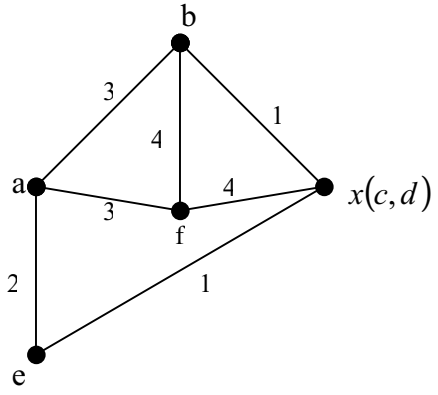


Figure 2.2

Note that the operation may create multiple edges, but no loop. We replace multiple edges e_{fc}, e_{fd} by a new edge e_{fx} with capacity equal to the sum of the capacities of the edges replaced.

3. Node Identification Algorithm

Proposition 3.1 Every cut of G_{vw} is a cut of *G*. Every cut of *G* that does not separate *v* from *w* is a cut of G_{vw}.
Proof: It is obvious that

$$\lambda(G) = \min(\lambda(G_{vw}), \lambda(G; v, w))$$

Definition 3.2 A ***legal ordering*** $v_1, v_2, ..., v_n$ of *G* is one in which
$Vi = \{ v_1, v_2, ..., v_n \}$,

$$u(\delta(V_{i-1}) \cap \delta(v_i)) \geq u(\delta(V_{i-1}) \cap \delta(v_j)) for 2 \leq i \leq j \leq n$$

We can choose any node to be v_1 and at step *i* we choose v_i to be the node that has the largest total capacity of edges joining it to the previously chosen nodes.

For example, in Figure 3.1,

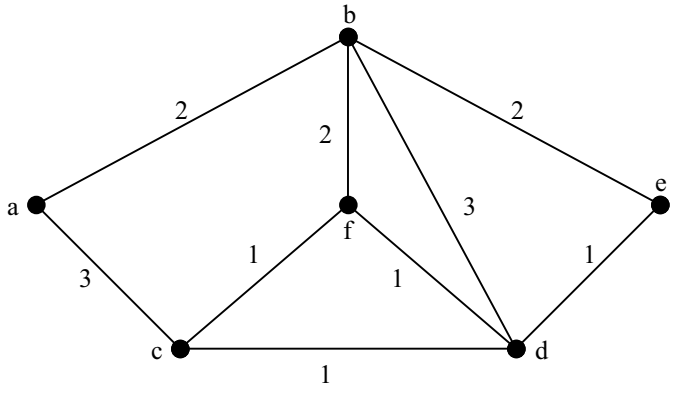


Figure 3.1 Legal Ordering

Let *a* be the first node $\{a\}$;
since $u(e_{ac}) > u(e_{ab})$ (3>2), we choose *c* as the second node $\{a, c\}$;

from *c,* we find node *b* has the largest total capacity of edges $\{a,c,b\}$;
then from *b*, for $u(e_{bd}) > u(e_{bf}) = u(e_{be})$ (3>2=2), we choose *d* $\{a,c,b,d\}$;
keep going , we get a legal ordering beginning with *a* is $\{a,c,b,d,f,e\}$.

The running time of finding a legal ordering is $O(n^2)$ by using a technique that is similar to Prim's Algorithm or Dijkstra's Algorithm.

Theorem 3.3 If $v_1, v_2, ..., v_n$ is a legal ordering of *G*, then $\delta(v_n)$ is a minimum (v_n, v_{n-1})-cut of *G*.

Assuming that Theorem 3.3 is correct, we get the following algorithm for finding the global min-cut of a graph *G.*

Node Identification Minimum Cut Algorithm

Initialize $M = \infty, A = undefined$;
While *G* has more than one node
  Find a legal ordering $v_1, v_2, ..., v_n$ of *G*;
  If $u(\delta(v_n)) < M$
    Replace *M* by $u(\delta(v_n))$, *A* by $\delta(v_n)$;
  Replace *G* by $G_{v_{n-1}v_n}$;
Return *A*.

The running time of above algorithm is $O(n^3)$, since there are *n* nodes and determining a legal ordering takes $O(n^2)$ running time.

Lemma 3.4 If $p,q,r \in V$, then $\lambda(G;p,q) \geq \min(\lambda(G;r,q), \lambda(G;p,r))$.

Proof:

Consider the minimum $(p,q)-cut$ $\delta(S)$ with $p \in S$.
If $r \in S$, then $\delta(S)$ is a $(r,q)-cut$ and so $u(\delta(S)) \geq \lambda(G;q,r)$.
Otherwise $\delta(S)$ is a $(p,r)-cut$ and so $u(\delta(S)) \geq \lambda(G;p,r)$.
The result follows.

Proof: *(of Theorem 3.3) All we need to show is that* $u(\delta(v_n)) \leq lambda(G; v_{n-1}, v_n)$ *in a legal ordering. We use induction on the number of vertices and edges. The statement is trivially true if* $|V| = 2$ *or* $|E| = 0$. *We use* δ' *to refer to* δ *on* G'. *Consider the following* 2 *cases:*

1. *Let* $e = v_n v_{n-1}$ *be an edge of* G *and let* $G' = G \setminus \{e\}$. *Note that the ordering* $v_1, v_2, \ldots, v_n$ *is still legal in* G'. *Now,*

$$\begin{aligned} u(\delta(v_n)) &= u(\delta'(v_n)) + u_e \\ &= \lambda(G'; v_{n-1}, v_n) + u_e \ (by\ induction) \\ &= \lambda(G; v_{n-1}, v_n) \end{aligned}$$

2. *Suppose that* v_n *and* v_{n-1} *are not adjacent. It suffices to show that*

$$u(\delta(v_n)) \leq \lambda(G; v_{n-2}, v_n) \tag{1}$$

and

$$u(\delta(v_n)) \leq \lambda(G; v_{n-2}, v_{n-1}) \tag{2}$$

Then by using Lemma (3.4), we can conclude that $u(\delta(v_n)) \leq \lambda(G; v_{n-1}, v_n)$.

To prove (1), apply induction to $G' = G \setminus v_{n-1}$. *Clearly,* $v_1, v_2, \ldots, v_{n-2}, v_n$ *is a legal ordering of* G'. *Now,* $u(\delta(v_n)) = u(\delta'(v_n)) = \lambda(G'; v_{n-2}, v_n)$ *(by induction)* $\leq \lambda(G; v_{n-2}, v_n)$.

To prove (2), apply induction on $G' = G \setminus v_n$. *Once again,* $v_1, v_2, \ldots, v_{n-1}$ *is a legal ordering of* G'. *Hence,* $u(\delta(v_n)) \leq u(\delta(v_{n-1})) = u(\delta'(v_{n-1})) = \lambda(G'; v_{n-2}, v_{n-1})$ *(by induction)* $\leq \lambda(G; v_{n-2}, v_{n-1})$.

□