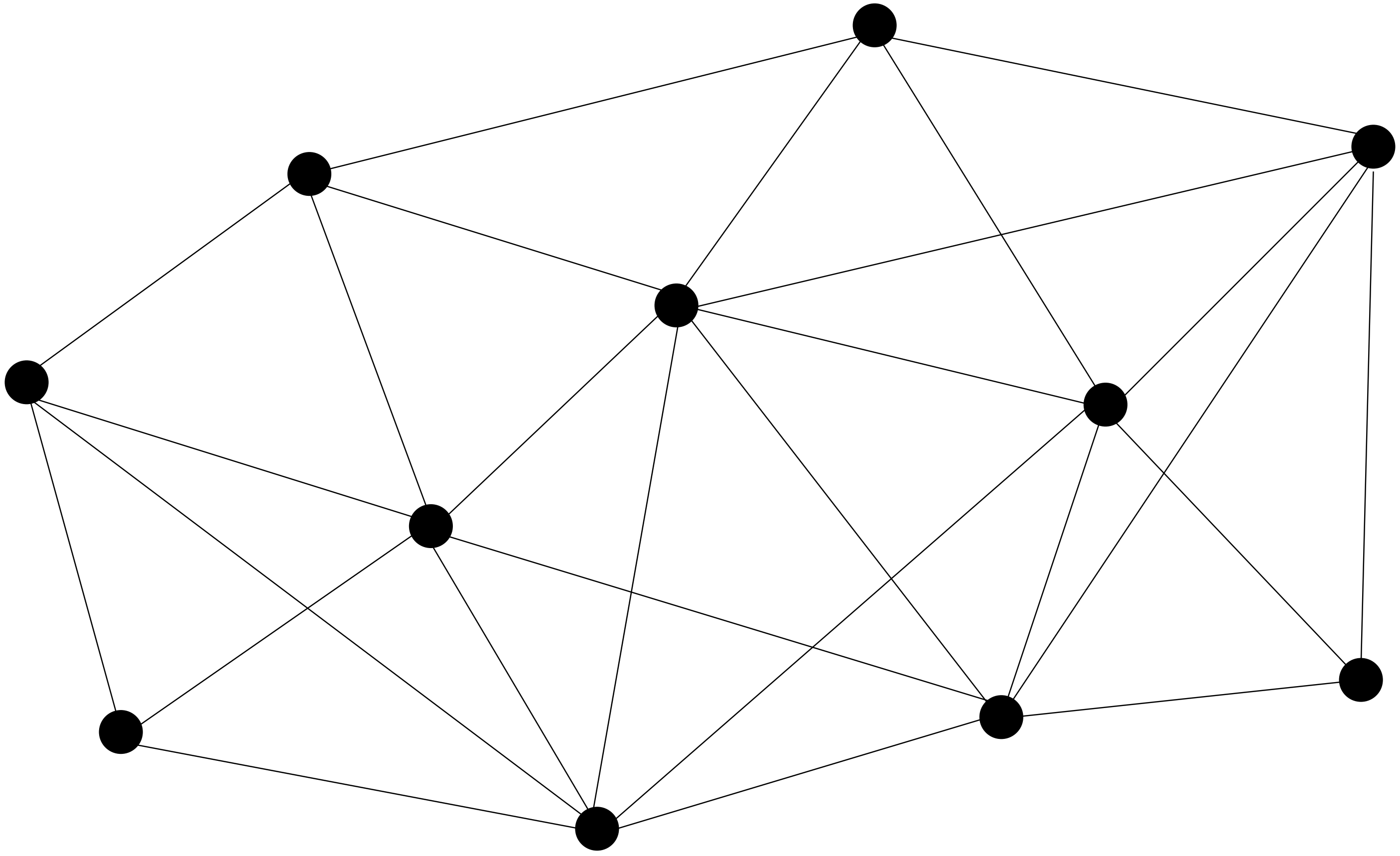
Minimum Cut Problem



Minimum Cut Problem

Input: $G = (V, E)$ undirected graph

Output: $V_1, V_2 \subseteq V$ such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$ and the number of edges between V_1 and V_2 is minimum.

$$c_{\min}(G) = \#\text{edges in a minimum cut of } G$$

A cut is often represented by the set of edges between V_1, V_2.

Weighted problem: Edge e has a weight $w(e)$.
Determine cut of minimum weight.

Reduction to flow problem: For all pairs $s, t \in V$, compute maximum (s,t)-flow. Time $O(n^5)$.

Randomized Algorithm

Multigraph: Between two nodes there may exist more than one edge.

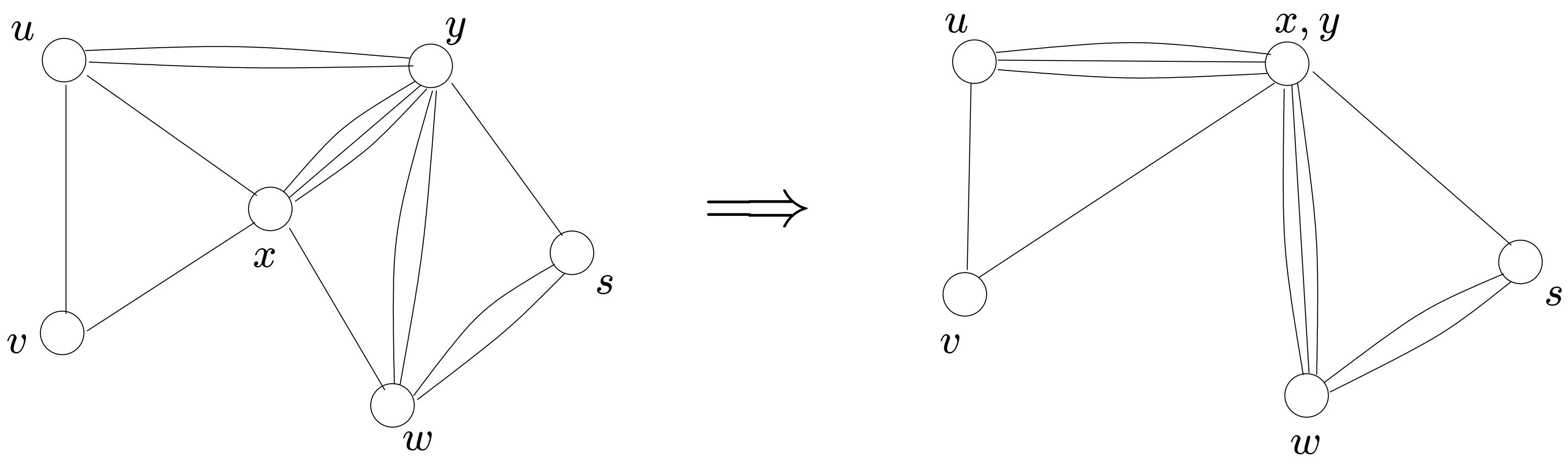
Basic operation: Contract edge $e = \{x, y\}$.

Replace x, y by meta-node z

For $v \notin \{x, y\}$ replace $\{v, x\}$ by $\{v, z\}$

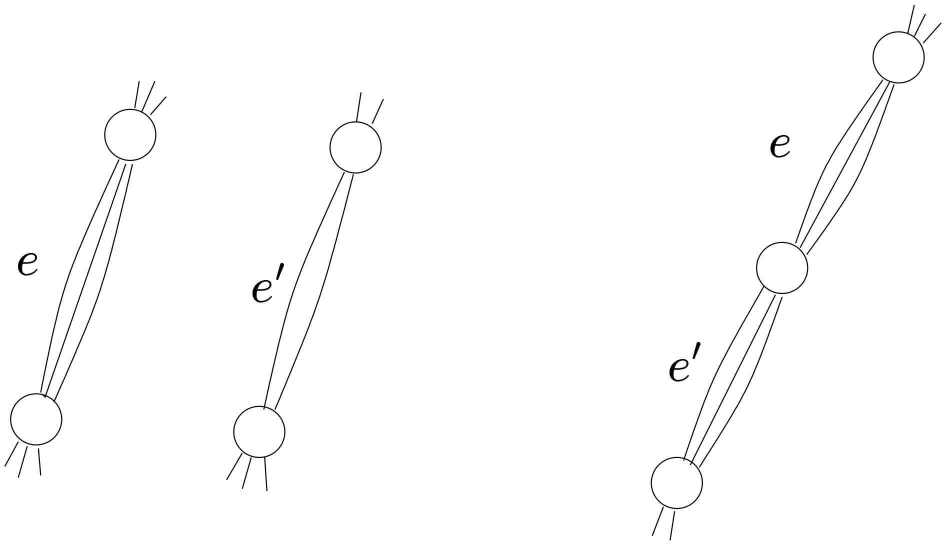
$\{v, y\}$ by $\{v, z\}$

$\Longrightarrow G \setminus \{e\}$



Contraction

Order of the contractions is irrelevant.



Can be implemented in $O(n)$ time.

For each meta-node, maintain number of edges to other nodes

For each original node, store name of meta-node it belongs to.

Algorithm *Contraction*

Algorithm *Contraction*

1. $H \longleftarrow G$;
2. **while** H has more than two nodes **do**
3. Choose edge e from H uniformly at random;
4. $H \longleftarrow H \setminus \{e\}$;
5. **endwhile**;
6. Let V_1, V_2 be the node sets represented by the last two nodes of H.

Running time: $O(n^2)$

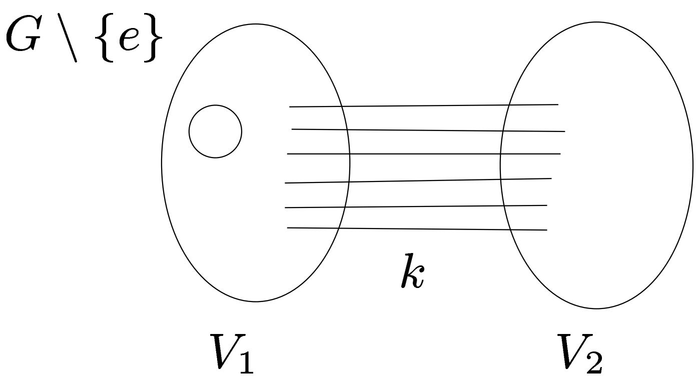
Properties

Lemma 1: Partition V_1, V_2 is the result of algorithm *Contraction* iff it does not contract an edge between V_1 and V_2.

Lemma 2: If $c_{\min}(G) = k$, then all nodes have degree $\geq k$ and G has $\geq nk/2$ edges.

Lemma 3: For any e in G, $c_{\min}(G) \leq c_{\min}(G \setminus \{e\})$.

Proof: Partition V_1, V_2 of $G \setminus \{e\}$ with k edges is also partition of G with k edges.



Probability of Success

Theorem 1: Let C be a minimum cut in G. *Contraction* outputs C with probability $\geq 2/n^2$.

Proof: i-th iteration of the while-loop

Corresponding graph H has $n_i = n - i + 1$ nodes

Suppose that during the first $i - 1$ iterations, no edge of C was contracted.

C is cut for H and $c_{\min}(H) = k$ by Lemma 3.

H has at least $n_i k/2$ edges by Lemma 2.

$$\text{Prob}[i\text{-th iteration contracts an edge of } C] \leq 2/n_i$$

$$\text{Prob}[i\text{-th iteration contracts no edge of } C] \geq 1 - 2/n_i$$

$$
\begin{aligned}
& \text{Prob}[C \text{ is output}] \\
\geq \quad & \prod_{i=1}^{n-2}(1 - \frac{2}{n_i}) = \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} = \prod_{j=3}^{n} \frac{j-2}{j} = \frac{1 \cdots (n-2)}{3 \cdots n} \\
= \quad & \frac{2}{n(n-1)} \geq \frac{2}{n^2} \quad \square
\end{aligned}
$$

Repeat *Contraction* $dn^2 \ln n$ times and choose smallest cut.

$$
\text{Prob}[C \text{ is not found}] \leq \left(1 - \frac{2}{n^2}\right)^{n^2 d \ln n} \leq e^{-2d \ln n} = n^{-2d}
$$

Improving the Running Time

Lemma 4: Let C be a minimum cut. Suppose *Contraction* terminates when there are exactly t nodes left.

Prob[no edge of C is contracted] $\geq \frac{t(t-1)}{n(n-1)}$.

Proof:

$$\prod_{i=1}^{n-t}(1-\frac{2}{n_i}) = \prod_{i=1}^{n-t}\frac{n-i-1}{n-i+1} = \prod_{j=t+1}^{n}\frac{j-2}{j} = \frac{(t-1)\cdots(n-2)}{(t+1)\cdots n}$$

$$= \frac{(t-1)t}{n(n-1)} \quad \square$$

Algorithm *Fast-Cut*

1. $n \longleftarrow |V|$;
2. **if** $n \leq 6$ **then**
3. Compute minimum cut using complete enumeration;
4. **else**;
5. $t \longleftarrow \lceil 1 + n/\sqrt{2} \rceil$; /* $\frac{t(t-1)}{n(n-1)} \geq \frac{1}{2}$
6. Apply *Contraction* twice so that t nodes remain each time and let H_1 and H_2 be the resulting graphs;
7. Apply *Fast-Cut* recursively to H_1, H_2;
8. Output the smaller cut;
9. **endwhile**.

Theorem 2: *Fast-Cut* has running time $O(n^2 \log n)$.

Proof: *Contraction* has running time $O(n^2)$.

$$T(n) = 2T(\lceil 1 + n/\sqrt{2} \rceil) + O(n^2) \quad \square$$

Probability of Success

Theorem 3: *Fast-Cut* finds minimum cut with probability $\geq \Omega(1/\log n)$.

Proof: Suppose that minimum cut C has k edges.
Fast-Cut returns C if
– during the contraction to H_1 or H_2, no edge of C is contracted
– *Fast-Cut* applied to H_i returns C.

$P(n) = \text{Prob}[\textit{Fast-Cut} \text{ finds min-cut in graph with } n \text{ nodes}]$

$$\begin{aligned}
P(n) &= 1 - \text{Prob}[\textit{Fast-Cut} \text{ finds } C \text{ in neither of the two trials}] \\
&= 1 - \prod_{i=1,2} \text{Prob}[\textit{Fast-Cut} \text{ does not find } C \text{ in trial related to } H_i] \\
&= 1 - \prod_{i=1,2} (1 - \text{Prob}[\textit{Fast-Cut} \text{ finds } C \text{ in trial related to } H_i]) \\
&= 1 - \left(1 - \frac{1}{2}P(t)\right)^2
\end{aligned}$$

$p(k)$ = lower bound on P when there are k recursive levels

$$p(k+1) = 1 - (1 - \frac{1}{2}p(k))^2 = p(k) - \frac{p(k)^2}{4}$$

$$p(0) = 1$$

We show that $p(k) \geq 1/d$ implies $p(k+1) \geq 1/(d+1)$

$f(x) = x - x^2/4$ monoton increasing in $[0,1]$
Thus $p(k) \geq 1/d$ and $p(k) \in [0,1]$ imply

$$p(k+1) \geq \frac{1}{d} - \frac{1}{4d^2} = \frac{d+1}{d+1} \cdot \frac{4d-1}{4d^2} = \frac{1}{d+1} \frac{4d^2+3d-1}{4d^2} \geq \frac{1}{d+1}$$

Since $p(0) = 1 \geq 1/1, \quad p(k) \geq 1/(k+1) = \Omega(1/k)$.
We have $O(\log n)$ recursive levels, $P(n) = \Omega(1/\log n)$ $\square$

Increasing the Probability of Success

Repeat *Fast-Cut* $d \ln^2 n$ times and take smallest cut.

$$\text{Prob}[C \text{ is not found}] \leq \left(1 - \frac{c}{\ln n}\right)^{d \ln^2 n} \leq e^{-cd \ln n} = n^{-cd}$$

Running time: $O(n^2 \log^3 n)$

Weighted Minimum Cuts

Weighted problem: Each edge e in $G = (V, E)$ has weight $w(e) \geq 0$.

A minimum (s, t)-cut, for some $s, t \in V$, is a cut $V_s, V_t \subseteq V$ with $V_s \cup V_t = V$, $V_s \cap V_t = \emptyset$, $s \in V_s, t \in V_t$ of minimum weight. Denoted by $c_{\min}(G, s, t)$

$G \setminus \{x, y\}$ = graph if x, y are contracted.
The weight of multiple edges adds up.

Theorem 4: Let $s, t \in V$. $c_{\min}(G) = \min\{c_{\min}(G, s, t),\ \ c_{\min}(G \setminus \{s, t\})\}$.

Deterministic Algorithm

Procedure *Some-*(s,t)*-cut*

1. $A \longleftarrow$ some node of V;
2. **while** $A \neq V$ **do**
3. Add $v \in V - A$ such that $w(v, A)$ is maximal;
4. **endwhile**;
5. Let s be the second to last and t be the last node added to A.

Algorithm *Min-Cut*

1. Min $\longleftarrow \infty$; nodes $\longleftarrow |V|$;
2. **while** nodes ≥ 2 **do**
3. Apply *Some-*(s,t)*-cut* to obtain s, t and a cut C of weight W;
4. **if** $W \leq$ Min **then** store C; Min $\longleftarrow W$ **endif**;
5. Contract s and t; nodes $\longleftarrow$ nodes-1;
6. **endwhile**;
7. Return Min and the last cut stored.

Analysis

Theorem 5: $V_t = \{t\}$ and $V_s = V - \{t\}$ computed by *Some-(s,t)-cut* is a minimum (s,t)-cut.

Proof:

Number nodes from 1 to n such that node i is the i-th node added to A.

$s = n - 1$ and $t = n$

Let C be an arbitrary (s,t)-cut.
C_i = edges in C having both endpoints in $\{1, \ldots, i\}$.
$w(C_i)$ = total weight of edges in C_i

Node i is active if i and $i-1$ are on different sides of the cut C.

Claim: For each active node i, $w(i, \{1, \ldots, i-1\}) \leq w(C_i)$.

n is active and hence $w(n, \{1, \ldots, n-1\}) \leq w(C_n) \leq w(C)$ $\square$

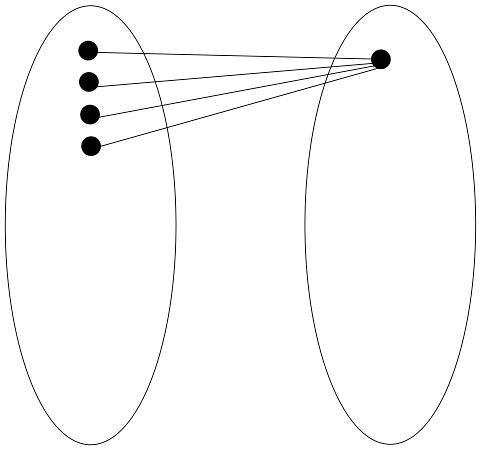
Analysis

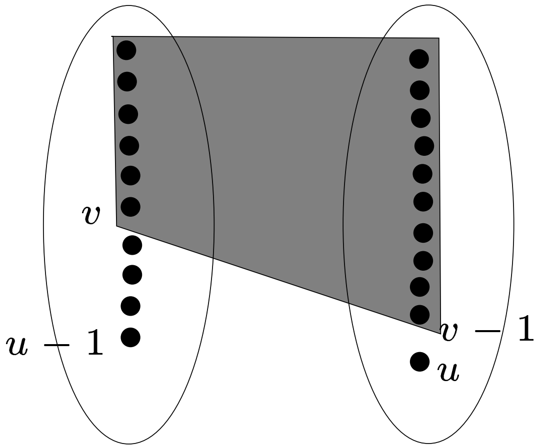
Claim: For each active node i, $w(i, \{1, \ldots, i-1\}) \leq w(C_i)$.

Proof: Statement holds for first active node.

Suppose statement holds for active node v and let u be the next active node.

$$\begin{aligned} w(u, \{1, \ldots, u-1\}) &= w(u, \{1, \ldots, v-1\}) + w(u, \{v, \ldots, u-1\}) \\ &\leq w(v, \{1, \ldots, v-1\}) + w(u, \{v, \ldots, u-1\}) \\ &\leq w(C_v) + w(u, \{v, \ldots, u-1\}) \\ &\leq w(C_u) \end{aligned}$$





Analysis

Theorem 6: *Min-Cut* computes a minimum cut in time $O(en + n^2 \log n)$.

Proof: Correctness: Induction on n.

Algorithm works correctly on multigraphs with $n = 2$ nodes.

$n \Longrightarrow n+1$: *Some-(s,t)-cut* computes s, t and $c_{\min}(G, s, t)$ correctly,

Min-Cut computes $c_{\min}(G \setminus \{s, t\})$ correctly.

Some-(s,t)-cut has running time $O(e + n \log n)$.

Keep a priority queue for $v \in V - A$, $\quad key(v) = w(v, A)$

n *DeleteMax* and e *IncreaseKey* operations $\quad \square$