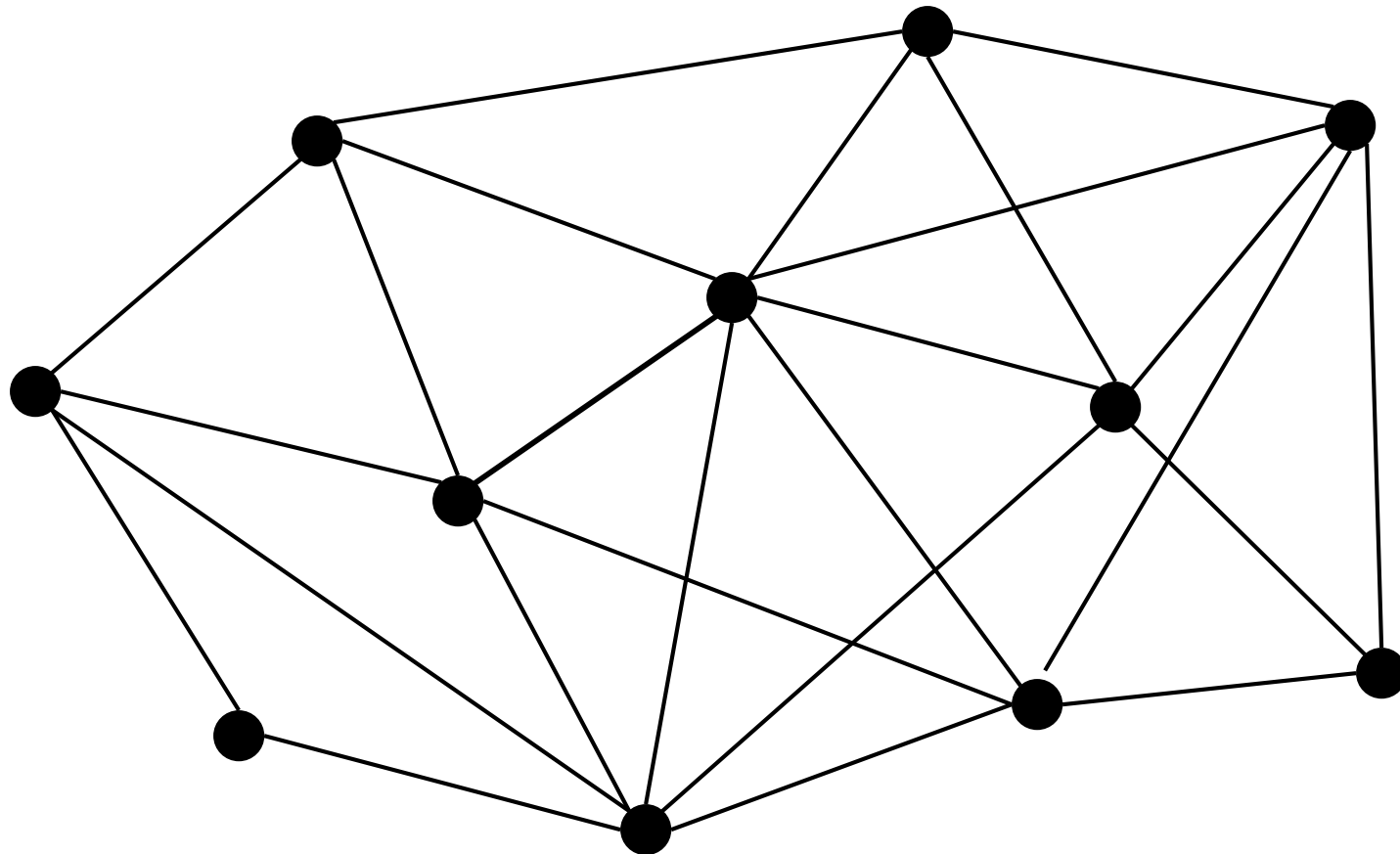


05 – Minimum Cuts



1. Minimum cuts



Minimum cuts

Input: Undirected graph $G = (V, E)$ $n = |V|$ $m = |E|$

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 as small as possible.

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

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

Weighted problem: Edge e has weight $w(e)$.

Find a cut of minimum weight.

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

2. Randomized algorithm

Multigraph: Multiple edges may exist between any two vertices.

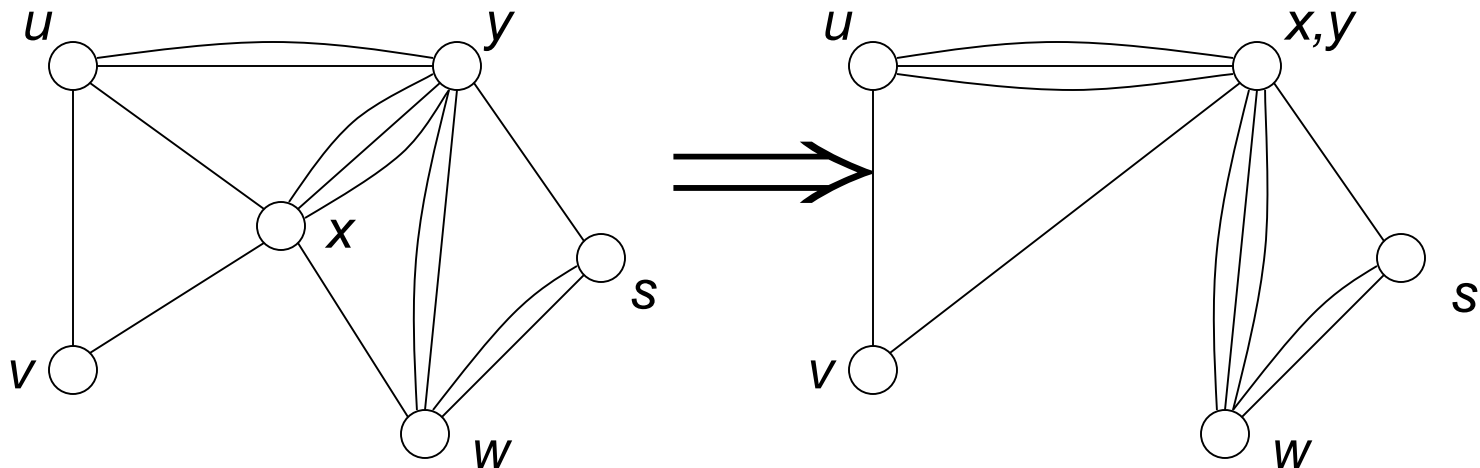
Basic operation: Edge contraction $e = \{x,y\}$.

Replace x,y by a **meta-vertex z** .

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

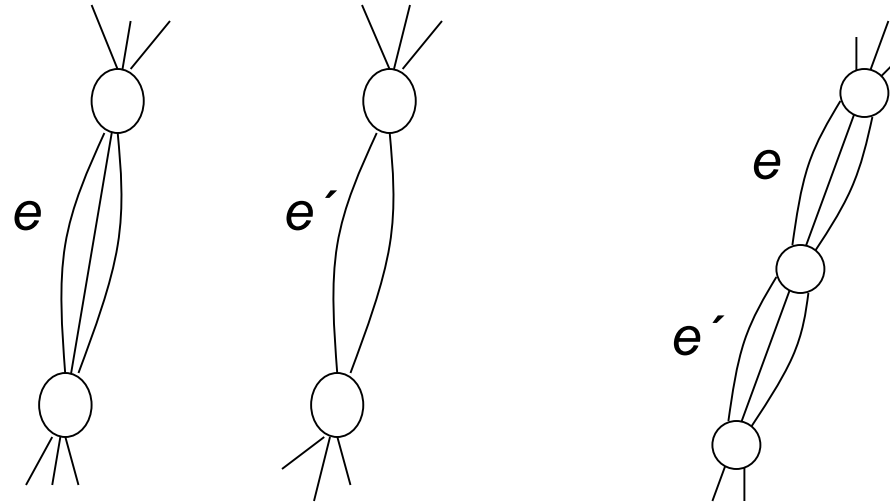
replace $\{v,y\}$ by $\{v,z\}$. No self-loops!

$\rightarrow G \setminus \{e\}$



Contractions

The **order** of the contractions is **irrelevant**.



Each edge contraction can be implemented in time $O(n)$ using (extended) adjacency lists or matrices.

For each meta-vertex

- store the number of edges to other meta-vertices,
- store the names of the original vertices it contains.

Algorithm Contraction

Algorithm Contraction;

1. $H := G$;
2. **while** H consists of more than two vertices **do**
3. Choose an edge e in H uniformly at random;
4. $H := H \setminus \{e\}$;
5. **endwhile**;
6. Let V_1, V_2 be the vertex sets represented by the last two vertices in H .

Running time: $O(n^2)$

Properties

Lemma 1: Partition V_1, V_2 is output by the *Contraction* if and only if no edge between V_1 and V_2 is ever contracted.

Proof: If an edge $\{v_1, v_2\}$ with $v_1 \in V_1$ and $v_2 \in V_2$ is contracted, partition V_1, V_2 cannot be output by the algorithm.

If no edge between V_1 and V_2 is ever contracted, then this partition indeed survives.

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

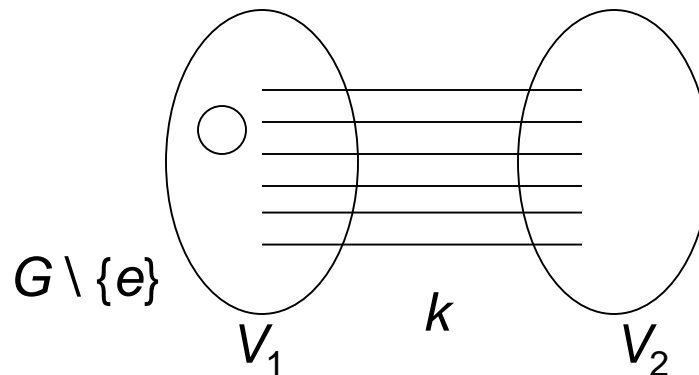
Proof: If there were a vertex v having degree $< k$, then $\{v\}$ and $V \setminus \{v\}$ would be a cut with less than k edges, and $c_{\min}(G) < k$.

If all vertices have degree $\geq k$, then the total edge degree, summed over all vertices, is at least nk . In this total edge degree, each edge is counted exactly twice.

Properties

Lemma 3: Let G be a multigraph. For each edge e in G there holds
 $c_{\min}(G) \leq c_{\min}(G \setminus \{e\})$.

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



Probability of success

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

Proof: Let $c_{\min}(G) = k$.

Consider the i -th iteration of the while-loop.

H has $n_i = n - i + 1$ vertices.

Suppose that the first $i - 1$ iterations do not contract an edge of C .

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

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

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

$$\text{Prob}[i\text{-th iteration does not contract an edge of } C] \geq 1 - 2/n_i$$

Probability of success

Prob[C is output]

$$\begin{aligned} &\geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n_i}\right) = \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} = \prod_{j=3}^n \frac{j-2}{j} = \frac{1 \cdots (n-2)}{3 \cdots n} \\ &= \frac{2}{n(n-1)} \geq \frac{2}{n^2} \end{aligned}$$

Increasing the success probability

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

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

3. Improved running time

Lemma 4: Let C be a minimum cut. Stop *Contraction* when exactly t vertices are left. There holds

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

Proof:

$$\begin{aligned} \prod_{i=1}^{n-t} \left(1 - \frac{2}{n_i}\right) &= \prod_{i=1}^{n-t} \frac{n-i-1}{n-i+1} = \prod_{j=t+1}^n \frac{j-2}{j} = \frac{(t-1) \cdots (n-2)}{(t+1) \cdots n} \\ &= \frac{(t-1)t}{n(n-1)} \end{aligned}$$

Algorithm *Fast-Cut*

Input: Multigraph $G = (V, E)$.

Output: Partition $V = V_1 \cup V_2$ or the respective edge set.

1. $n := |V|$;
2. **if** $n \leq 6$ **then**
3. Compute a minimum cut by complete enumeration;
4. **else**
5. $t := \left\lceil 1 + n / \sqrt{2} \right\rceil$ /* $\frac{t(t-1)}{n(n-1)} \geq \frac{1}{2}$
6. Execute *Contraction* twice so that each time exactly t vertices remain. Let H_1 and H_2 be the resulting graphs;
7. Apply *Fast-Cut* recursively to H_1 and H_2 ;
8. Output the smaller cut;
9. **endif**;

Algorithm *Fast-Cut*

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

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

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

Success probability

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

Proof: Let C be a minimum cut.

Fast-Cut returns C if

- during the reduction to H_1 or H_2 no edge of C is contracted and
- *Fast-Cut* applied to such an H_i returns C .

$P(n) = \text{Prob}[\textit{Fast-Cut} \text{ finds minimum cut } C \text{ in graphs with } n \text{ vertices}]$

Success probability

$$\begin{aligned} P(n) &= 1 - \text{Prob}[Fast - Cut \text{ does not find } C \text{ in any of the two trials}] \\ &= 1 - \prod_{i=1,2} \text{Prob}[Fast - Cut \text{ does not find } C \text{ in the trial on } H_i] \\ &= 1 - \prod_{i=1,2} \left(1 - \text{Prob}[Fast - Cut \text{ finds } C \text{ in the trial on } H_i]\right) \\ &\geq 1 - \left(1 - \frac{1}{2} P(t)\right)^2 \end{aligned}$$

Success probability

$p(l)$ = lower bound on P if there are l recursive levels

$$p(l+1) = 1 - \left(1 - \frac{1}{2} p(l)\right)^2 = p(l) - \frac{p(l)^2}{4}$$

$$p(0) = 1$$

We prove that $p(l) \geq 1/d$ implies $p(l+1) \geq 1/(d+1)$.

Since $p(0) = 1 \geq 1/1$ it follows $p(l) \geq 1/(l+1) = \Omega(1/l)$.

There are $O(\log n)$ recursive levels so that $P(n) = \Omega(1/\log n)$.

Success probability

$f(x) = x - x^2/4$ is monotonically increasing in $[0,1]$

Hence $p(l) \geq 1/d$, where $d \geq 1$, and $p(l) \in [0,1]$ imply

$$\begin{aligned} p(l+1) &\geq \frac{1}{d} - \frac{1}{4d^2} \\ &= \frac{d+1}{d+1} \cdot \frac{4d-1}{4d^2} \\ &= \frac{1}{d+1} \frac{4d^2+3d-1}{4d^2} \\ &\geq \frac{1}{d+1}. \end{aligned}$$

Increasing the success probability

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

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

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

4. Minimum weighted cuts

Weighted problem: Undirected graph $G = (V, E)$ $n = |V|$ $m = |E|$

Edge e in $G = (V, E)$ has **weight** $w(e) \geq 0$.

Output: $V_1, V_2, \subseteq V$ such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$ and $\sum_{e=(u,v) \in V_1 \times V_2} w(e)$ is as small as possible.

Let $c_{\min}(G)$ denote the weight of such a minimum cut.

A **minimum (s, t) -cut**, where $s, t \in V$, is a cut $V_s, V_t \subseteq V$ with $V_s \cup V_t = V$, $V_s \cap V_t = \emptyset$ and $s \in V_s, t \in V_t$ of minimum weight.

This weight is denoted by $c_{\min}(G, s, t)$.

Minimum weighted cuts

$G \setminus \{x,y\}$ = graph if x, y are contracted

The weights of multiple edges add up.

Lemma 5: Let $s, t \in V$ be arbitrary. There holds

$$c_{\min}(G) = \min\{c_{\min}(G, s, t), c_{\min}(G \setminus \{s, t\})\}.$$

Deterministic algorithm

Algorithm *Some-(s,t)-Cut*;

Input: Graph G

Output: s, t (along with a minimum (s,t) -cut)

1. $A := \{\text{arbitrary vertex of } V\}$;
2. **while** $A \neq V$ **do**
3. Add vertex $v \in V - A$ to A for which $w(v,A)$ is maximum;
4. **endwhile**;
5. Let s be the second to last and t be the last vertex added to A ;

$w(v,A)$ = total weight of edges between v and vertices in A

Deterministic algorithm

Algorithm *Minimum-Cut*,

1. $Min := \infty$; $n := |V|$;
2. **while** $n \geq 2$ **do**
3. Execute *Some-(s,t)-Cut*, and obtain s , t and a cut C of weight W ;
4. **if** $W < Min$ **then** store C ; $Min := W$; **endif**;
5. Contract s and t , $n := n - 1$;
6. **endwhile**;
7. Return Min and the cut stored last;

Analysis

Theorem 4: Sets $V_t = \{t\}$ and $V_s = V - \{t\}$ computed by *Some-(s,t)-Cut* are a minimum (s,t) -cut.

Proof:

Number the vertices from 1 to n so that vertex i is added to A in the i -th iteration.

$$s = n - 1 \text{ and } t = n$$

Let C be an arbitrary (s,t) -cut.

C_i = edges in C having both endpoints in $\{1, \dots, i\}$

$w(C_i)$ = total weight of edges in C_i

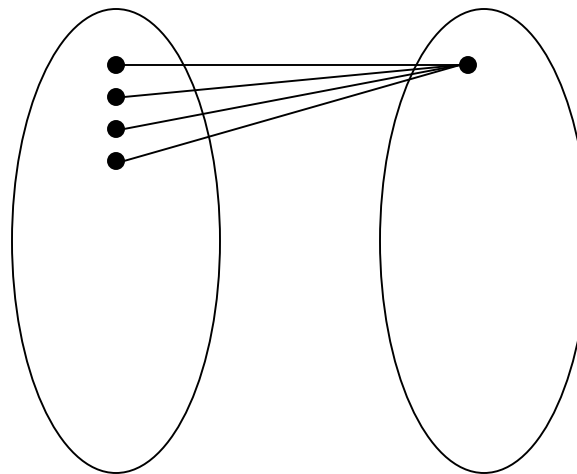
Vertex i is active (with respect to C) if i and $i - 1$ belong to different parts of C .

Claim: For each active vertex i there holds $w(i, \{1, \dots, i - 1\}) \leq w(C_i)$.

n is active and hence $w(n, \{1, \dots, n - 1\}) \leq w(C)$.

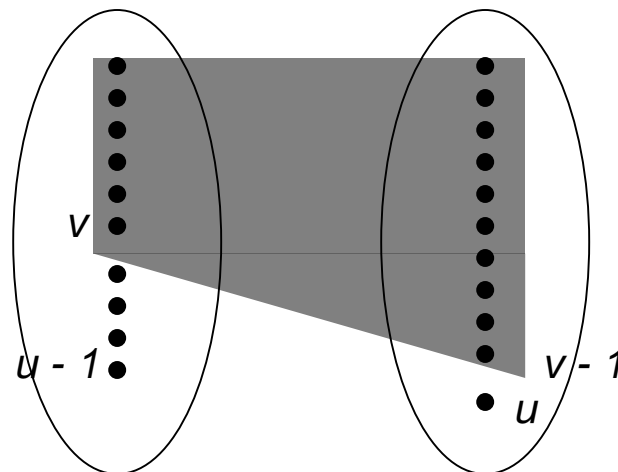
Claim: For each active vertex i there holds $w(i, \{1, \dots, i-1\}) \leq w(C_i)$.

Proof: The claim holds for the first active vertex.



Suppose the claim holds for **active vertex v** and the next active **vertex is u** .

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



Theorem 5: *Minimum-Cut* computes a minimum cut in time $O(mn + n^2 \log n)$.

Proof: Correctness: Induction on the number of vertices.
Algorithm works correctly for multigraphs with $n = 2$ vertices.

$n-1 \rightarrow n$: *Some-(s,t)-Cut* computes s, t and $c_{\min}(G, s, t)$ correctly.
MinimumCut computes $c_{\min}(G \setminus \{s, t\})$ correctly.

Some-(s,t)-Cut has a running time of $O(m + n \log n)$.

Maintain a priority queue for $v \in V - A$ with $key(v) = w(v, A)$.

n *DeleteMax* and m *IncreaseKey* operations (Fibonacci Heaps).