1 Graphs and the Contraction Algorithm

1.1 Graphs and Minimum Cuts

1.1.1 Graphs

Graph: represent pairwise relations between sets of bojects.

Two ingredients**:**

- vertices aka nodes (N)
- edges (E) = pairs of vertices
  - undirected [unordered pair]
  - directed/arcs [ordered pair]

1.1.2 Cuts and Graphs

Definition: a cut of a graph (V, E) is a partition of V into two non-empty sets A and B.

Definition: the crossing edges of a cut (A, B) are those with:

- one endpoint in each of (A, B) [undirected]
- tail in A, head in B [directed]

A graph with n vertices has 2^n cuts. One binary degree of freedom. Strictly speaking a cut can't have a non-empty set A or B, so the number is $2^n - 2$.

1.1.3 The Minimum Cut Problem

Input: an undirected graph $G = (V, E)$ [parallel edges allowed - video on representation of input]

Goal: Compute a cut with fewest number of crossing edges (a min-cut).

1.1.4 Applications

Identify weaknesses in a network/bottlenecks.

Community detection in social network.

Image segmentation

- input: graph of pixels
- use edge weights [(u, v) has large weight $\iff$ "expect" u, v to come from some object]
  hope: repeated min-cuts identifies the primary objects in picture.

1.2 Graph Representation

Consider an undirected graph that has n vertices, no parallel edges, and is connected (i.e. "in one piece"). The minimum and maximum number of edges that the graph could have respectively is:

$$\begin{aligned} minimum &= n - 1 \\ maximum &= \frac{n(n-1)}{2} = \binom{n}{2} \end{aligned}$$

1.2.1 Sparse vs. Dense Graphs

Let

$$\begin{aligned} n &= \text{number of vertices} \\ m &= \text{number of edges} \end{aligned}$$

In most (but not all) applications m is $\Omega(n)$ and $O(n^2)$.

In a sparse graph, m is $O(n)$ or close to it

In a dense graph, m is closer to $O(n^2)$

1.2.2 The Adjacency Matrix

Represent G by a $n \times n$, $0-1$ matrix A, where $A_{ij} = 1 \iff G$ has an $i-j$ edge.

Variants:

- $A_{ij} =$ number of i-j edges (if parallel edges)
- $A_{ij} =$ weight of i-j edge (if any)
- $A_{ij} = \begin{cases} +1 & \text{if } i \to j \\ -1 & \text{if } i \leftarrow j \end{cases}$

An adjacency matrix requires $\Theta(n^2)$ space as a function of the number n of vertices and the number m of edges.

1.2.3 Adjacency List

Ingredients:

- array (or list) of vertices [$\Theta(n)$ space]
- array (or list) of edges [$\Theta(m)$ space]
- each edge points ot its endpoints [$\Theta(m)$ space]
- each vertex points to edges incident on it [$\Theta(m)$ space]

The three $\Theta(m)$ categories have 1-to-1 correspondence between m, n. All together $\Theta(m+n)$ or $\Theta(max\,\{m, n\})$.

An adjacency list representation requires $\Theta(m+n)$ space as a function of the number n of vertices and the number m of edges.

Which is better? Depends on graph density, operatations needed.

1.3 Random Contraction Algorithm

[David Karger, early 90s]

```
While there are more than 2 vertices:
        pick a remaining edge (u, v) uniformly at random
        merge (or "contract") u and v into a single vertex
        remove self-loops
        return cut represented by final 2 vertices
```

1.3.1 Example

[2:45]

[6:40]

Algorithm sometimes identifies a min-cut and sometimes doesn't, depending on choice of vertex.

What is the probability of success?

1.4 Analysis of Contraction Algorithm

1.4.1 The Setup

Question: What is the probability of success?

Fix a graph $G = (V, E)$ with n vertices, m edges.

Fix any (if multiple) minimum-cut (A, B).

Let k =number of edges crossing (A, B). Call these edges F.

1.4.2 What Could Go Wrong?

Suppose one of the edges in F gets chosen for contraction at some point $\Rightarrow$algorithm will not output (A, B).

Suppose only edges inside A or inside B get contracted $\Rightarrow$algorithm will output (A, B).

Thus: $Pr\,[\text{output is (A, B)}] = Pr\,[\text{never contracts an edge of F}]$

Let S_i =event that an edge of F contracted in iteration i. Goal: compute $Pr[\neg S_1 \cap \neg S_2 \cap \neg S_3 \cap \cdots \cap \neg S_{n-2}]$.

The probability that an edge crossing the minimum cut (A, B) is chosen in the first iteration (as a function of the number of vertices n, the number of edges m, and the number k of crossing edges is $\frac{k}{m}$, because $Pr[S_1] = \frac{\text{\# of crossing edges}}{\text{\# of edges}} = \frac{k}{m}$.

1.4.3 The First Iteration

Key observation: degree of each vertex is at least k.

Reason: each vertex v defines a cut $(\{u\}, V - \{u\})$.

Since $\sum_v degree(v) = 2m \geq kn$, we have $m \geq \frac{kn}{2}$.

Since $Pr[S_1] = \frac{k}{n}$, $Pr[S_1] \leq \frac{2}{n}$.

1.4.4 The Second Iteration

Recall: $Pr[\neg S_1 \cap \neg S_2] = Pr[\neg S_2 | \neg S_1] \cdot Pr[\neg S_1]$ probability we don't screw up in first and second iteration.

$Pr[\neg S_2 | \neg S_1] = 1 - \frac{k}{\text{\# of remaining edges}}$ probability we not screw up in second iteration given that we didn't do it already.

$Pr[\neg S_1] \geq (1 - \frac{2}{n})$ probability we not screw up in first iteration.

How many remaining edges are there? Rewrite that denominator in terms of remaining vertices $(n - 1)$.

Note: all nodes in contracted graph define cutes in G (with at least k crossing edges) $\Rightarrow$all degrees in contracted graph are at least k.

So: the number of remaining edges $\geq \frac{1}{2}k(n - 1)$

So $Pr[\neg S_2 | \neg S_1] \geq 1 - \frac{2}{n-1}$

1.4.5 All Iterations

$$
\begin{aligned}
Pr[\neg S_1 \cap \neg S_2 \cap \neg S_3 \cap \cdots \cap \neg S_{n-2}] &= \\
Pr[\neg S_1]Pr[\neg S_2|\neg S_1]Pr[\neg S_3|\neg S_1 \cap \neg S_2] \cdot \cdots \cdot Pr[\neg S_{n-2}|\neg S_1 \cap \cdots \cap \neg S_{n-1}] &\geq \\
(1-\frac{2}{n})(1-\frac{2}{n-1})(1-\frac{2}{n-2})\ldots(1-\frac{1}{n-(n-4)})(1-\frac{1}{n-(n-3)} &= \\
\frac{n \cancel{-2}}{n}\frac{n \cancel{-3}}{n-1}\frac{n \cancel{-4}}{n \cancel{-2}}\cdots\frac{2}{\cancel{4}}\frac{1}{\cancel{3}} &= \\
\frac{2}{n(n-1)} &\geq \frac{1}{n^2}
\end{aligned}
$$

<u>**Problem**</u>: low success probability (but: non-trivial lower bound).

1.4.6 Repeated Trials

<u>Solution</u>: run the basic algorithm a large number N times, remember the smallest cut found.

<u>Question</u>: how many trials needed?

Let T_i =event that the cut (A, B) is found on the i^{th} try $\Rightarrow$by definition, different T_i's are independent.

So.:

$$
\begin{aligned}
Pr\,[\text{all N trials fail}] &= Pr[\neg T_1 \cap \neg T_2 \cap \cdots \cap \neg T_N] \\
(\textit{independent}) &= \prod_{i=1}^{N} Pr[\neg T_i] \\
&\leq (1-\frac{1}{n^2})^N
\end{aligned}
$$

Calculus fact: $\forall$ reall numbers x, $1+x \leq e^x$ [2:40].

So: if we take

$$N = n^2,\ Pr\,[\text{all fail}] \leq \left(e^{-\frac{1}{n^2}}\right)^{n^2} = \frac{1}{e}$$

If we take

$$\mathbf{N = n^2 \ln n},\ \mathbf{Pr}\,[\text{all fail}] \leq \left(\frac{\mathbf{1}}{\mathbf{e}}\right)^{\ln \mathbf{n}} = \frac{\mathbf{1}}{\mathbf{n}}$$

Running time: Polynomial in n and m but slow - $\Omega(n^2 m)$

But: can get big speedups (to roughly $O(n^2)$) with more ideas. Outside of the scope of this course.

1.5 Counting Minimum Cuts

<u>Note</u>: a graph can have multiple min-cuts.

e.g. a tree with n vertices has $(n-1)$ min-cuts

<u>Question</u>: what's the larges number of min-cuts that a graph with n vertices can have?

<u>Answer</u>: $\binom{n}{2} = \frac{n(n-1)}{2}$

1.5.1 The Lower Bound

Consider the n-cycle, $n = 8$ [2:00]

<u>Note</u>: each pair of the n edges defines a distinct minimum cut (with two crossing edges) $\Rightarrow$has $\geq \binom{n}{2}$ min cuts.

1.5.2 The Upper Bound

Let $(A_1, B_1), (A_2, B_2), \ldots, (A_t, B_t)$ be the min-cuts of a graph with n vertices. By the Contraction Algorithm analysis (without repeated trials):

$$Pr[output = (A_i, B_i)] \geq \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}} \qquad \forall_{i=1,2,\ldots,t}$$

$t =$the number of different min-cuts

Note: S_i's are disjoint events (i.e. only one can happen) $\Rightarrow$their probabilities sum to at most 1

Thus: $\frac{t}{\binom{n}{2}} \leq 1 \Rightarrow t \leq \binom{n}{2}$