Min-Cut

I/p : an undirected connected graph
$G = (V, E)$ {note that G can have parallel edges}

Our problem: Find a min-size subset E' of E such that the graph $G' = (V, E - E')$ is disconnected.
Such a subset E' is called a (global) min-cut.

Karger showed the following simple randomized algorithm for finding a min-cut.

1. Start with $G_0 = G$.

2. For $i = 1$ to $n-2$ do:
   - pick an edge uniformly at random from G_{i-1} and contract this edge.
   - call the new graph G_i.

3. Return the set of edges between the 2 vertices in G_{n-2} as our candidate min-cut.

$$G_0 \longrightarrow G_1 \longrightarrow G_2 \longrightarrow \cdots \longrightarrow G_{n-2}$$

Note that G_{n-2} has 2 "vertices" (these are actually subsets of V) with many parallel edges between these 2 vertices.

- Run the above algorithm on a simple example. Of course, this algorithm need not return the correct answer.

To analyze this algorithm, let us fix one particular min-cut (G may have many min-cuts) as our favourite min-cut.

- call this min-cut C. Let $C = \{e_1, e_2, \ldots, e_k\}$.

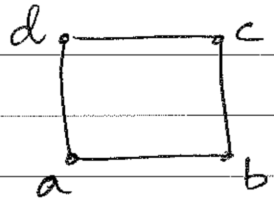
We want to estimate the probability that this algorithm returns C.

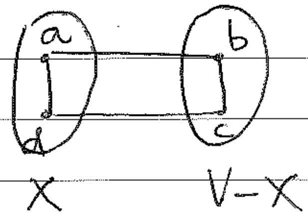
Since $|C| = k$, we know that G has at least $\frac{nk}{2}$ edges.

- thus an edge picked uniformly at random has probability $\leq \frac{2}{n}$ of belonging to C.

Suppose no edge of C got contracted in the entire algorithm. Then can we say that the set of edges between the 2 vertices in G_{n-2} is exactly C?

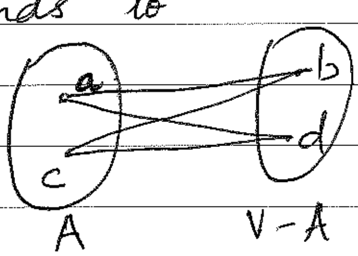
For example, consider





Here $C = \{(a,b), (c,d)\}$ is our favourite min-cut.

There is another cut $C' = \{(a,b), (b,c), (c,d), (a,d)\}$ which corresponds to the partition



$A = \{a, c\}$
$V-A = \{b, d\}$

Perhaps the algorithm returns a strict superset C' of C.

Exercise. Show this cannot happen. That is, if no edge of C got contracted in the algo. then the algo. returns C.

Thus if we are lucky in each iteration, i.e, if no edge of C got contracted in any iteration, then we have our favourite min-cut C at the end.

Success Probability

Let $\mathcal{E}_1$ be the event that no edge of C got contracted in the first iteration.
Let $\mathcal{E}_2$ be the event that no edge of C got contracted in the second iteration.

⋮

Let $\mathcal{E}_i$ be the event that no edge of C got contracted in the i-th iteration.

⋮

Our good event is $\mathcal{E}_1 \cap \mathcal{E}_2 \cap \ldots \cap \mathcal{E}_{n-2}$

$$\Pr(\mathcal{E}_1 \cap \mathcal{E}_2 \cap \ldots \cap \mathcal{E}_{n-2}) = \Pr(\mathcal{E}_1) . \Pr(\mathcal{E}_2 | \mathcal{E}_1) \ldots \Pr(\mathcal{E}_i | \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}) \ldots \Pr(\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}).$$

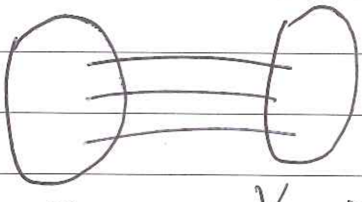
Let us estimate each of these probabilities now.

We have already seen that $\Pr(\mathcal{E}_1) \geq 1 - \frac{2}{n}$.

Let us now estimate $\Pr(\mathcal{E}_2 | \mathcal{E}_1)$.
For this, we need to look at the graph G_1 which is the graph G with one edge contracted.

- what is the number of vertices in G_1?
  - it is $\underline{n-1}$
- what about the number of edges in a min-cut in G_1?

Observe that every cut in G_1 is also a cut in G. Recall that a cut corresponds to a partition of the vertex set. Consider any cut in G_1: so there is a partition $(X, V_1 - X)$ where V_1 is the vertex set of G_1 that corresponds to this cut



$X \qquad V_1 - X$

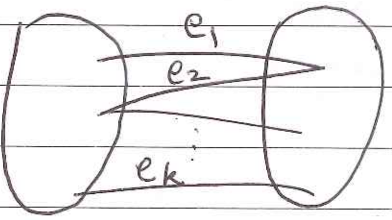
The above partition of V_1 is also a partition of V which is the vertex set of G.

- recall that V_1 is the same as V except for a "supervertex" → which is a set of 2 vertices.

Since every cut in G_1 is also a cut in G and because every cut in G has size $\geq k$, it follows that every cut in G_1 also has size $\geq k$.

- thus the number of edges in a min-cut of G_1 is at least k.
- hence the number of edges in G_1 $\geq \frac{(n-1) \cdot k}{2}$ (because there are $n-1$ vertices in G_1)

Recall that C is our favourite min-cut in G. $\mathcal{E}_1$ is the event that no edge of C got contracted in the first iteration of the algorithm.



$A \qquad V-A$

Let $(A, V-A)$ be the partition of V corr. to C.

Suppose event $\mathcal{E}_1$ occurred. This means both endpoints of e are in A or both are in $V-A$, where e is the edge picked in the first iteration of the algo.

So if $\mathcal{E}_1$ happens, then C is a valid cut in the graph G_1 also. We want to estimate $Pr(\mathcal{E}_2 | \mathcal{E}_1)$, which is the probability that no edge of C got contracted in the second iteration, given that C is preserved in G_1.

We have $Pr(\mathcal{E}_2 | \mathcal{E}_1) \geq \dfrac{|C|}{\text{number of edges in } G_1}$

$$\geq \frac{k}{(n-1).k/2} = \frac{2}{(n-1)}$$

Exercise. Show that $Pr(\mathcal{E}_i | \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}) \geq 1 - \dfrac{2}{n-i+1}$

So we get $Pr(\mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-2}) \geq \left(\dfrac{n-2}{n}\right) \cdot \left(\dfrac{n-3}{n-1}\right) \cdots \dfrac{2}{4} \cdot \dfrac{1}{3}$

Thus the success probability of this algorithm is rather low $\rightsquigarrow$ only $1/\binom{n}{2}$. $= \dfrac{2}{n(n-1)}$

We need to improve the success probability to $3/4$

- Repeat the algorithm independently n^2 times.
- Return the least sized candidate cut output in these n^2 repetitions.

$$Pr(\text{a cut of size } k \text{ is not returned}) \leq \left(1 - \frac{2}{n^2}\right)^{n^2} \leq \frac{1}{e^2} \simeq 0.12$$

So we have improved the success probability to 0.88 by repeating the basic algorithm n^2 times and returning the best cut among the n^2 candidate cuts.

So we have improved the success probability of the total algorithm to a value $\geq 3/4$, however this has come at a price.

The running time of the total algorithm is $n^2 \cdot$ (running time of the basic algorithm)

$= n^2 \cdot n \cdot$ (time to choose an edge uniformly at random & contract it)

<u>Choosing an edge uniformly at random</u>

- first choose a vertex u with prob. $\frac{deg(u)}{2m}$ and then choose an edge incident to u with prob. $\frac{1}{deg(u)}$. Here $m = |E|$.

So probability of an edge (a,b) getting chosen

$$= \frac{deg(a)}{2m} \cdot \frac{1}{deg(a)} + \frac{deg(b)}{2m} \cdot \frac{1}{deg(b)} = \frac{1}{m}.$$

Maintain the graphs in terms of their adjacency matrix. At the beginning, there are n vertices.

$$\begin{array}{cc} & \begin{matrix}1 & 2 & \cdots & n\end{matrix} \\ \begin{matrix}1\\2\\\vdots\\n\end{matrix} & \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}\end{array}$$

So we have an $n \times n$ matrix and the (i,j)-th entry = number of edges between i & j.

Contracting an edge (a,b) amounts to deleting the row of a and replacing the row of b with the sum of row a and row b. Similarly with the columns. This takes $O(n)$ time.

Hence the running time of the total algorithm is $O(n^2 \cdot n \cdot n) = O(n^4)$.