CSE 373: Analysis of Algorithms

Topic: Randomized Min-Cut

Nov 03, 2003

Lecturer: Piyush Kumar Scribed by: Piyush Kumar

Please Help us improve this draft.
If you read these notes and find any errors or have an idea to improve it, please send feedback.

This was the most unkindest cut of all
Shakespeare(Julius Caesar Act III)

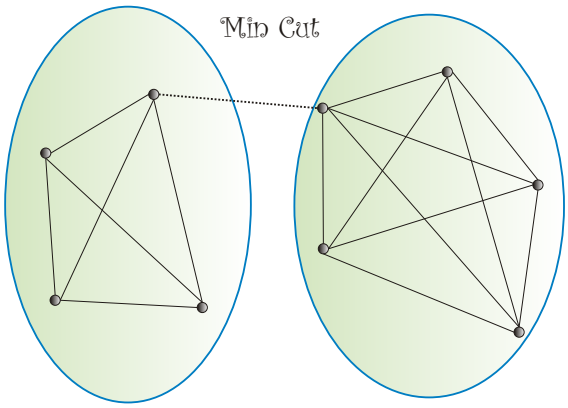
1 Randomized Algorithm

A randomized algorithm is an algorithm that is allowed to access a source of independent, unbiased random bits. It uses these bits to influence its computation. Randomized algorithms can be broadly classified in two types.

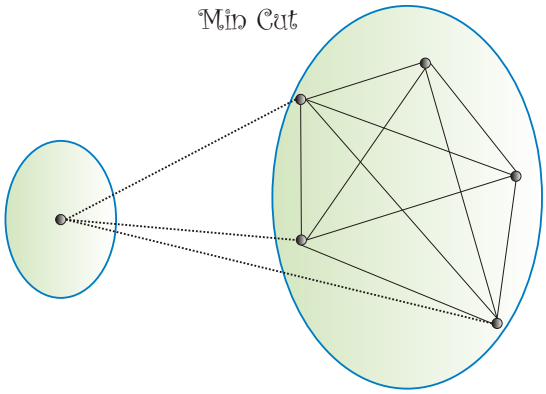
- *Monte Carlo:* A randomized algorithm that runs for a fixed number of steps for each input and produces an answer that is correct with a bounded probability. (Example: Randomized Min-Cut)
- *Las Vegas:* A randomized algorithm that always produces a correct answer, but its runtime for each input is a random variable whose expectation is bounded. (Example that we have already seen: Randomized Quick Sort)

2 Monte-Carlo Min-Cut Algorithm

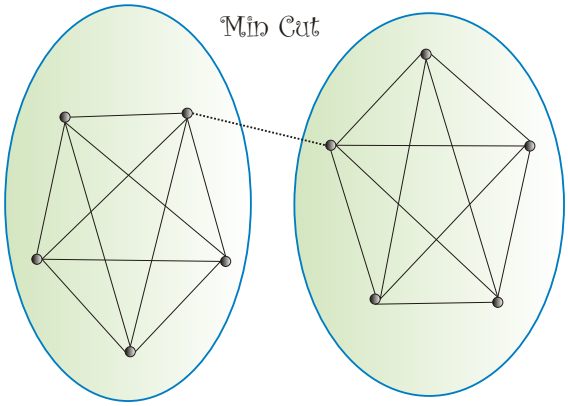
Let $G = (V, E)$ be a multigraph with n vertices and m edges. Given G, a *cut set* or a *cut* is a set of edges whose removal splits (or cuts) the graph into two connected components. The *minimum cut* is the cut of minimum size. For instance, say this graph represents a network, and we want to know how "robust" it is in the sense of the the minimum number of links whose failure causes the network to become disconnected. The minimum cut problem is to find a cut of minimum size. Here is an example.



Lemma 1 *The minimum cut has size at most the minimum degree of any node.*



Note: that the minimum degree can be much larger than the size of the minimum cut.



Lemma 2 *Removing self loops does not have any effect on the size of the min-cut.*

We have already seen Boruvka's algorithm in which we used edge contractions. How can you contract an edge in $O(n)$ time?

Lemma 3 *Edge Contractions can only increase the size of the min-cut.*

Proof: If an edge is not in the min-cut, then it's contraction does not change the min-cut. If an edge of the min-cut is contracted, then the min-cut can increase in size but never decrease (Why?). ■

Let us assume that the size of the min cut is equal to k. If there are more than one min-cut, just pick up one of those. Then every vertex should have degree at least k because otherwise we could cut the vertex out of the graph with less than k edges and that would contradict the fact that the graph has a min-cut of size k. Thus the number of vertex-edge pairs is at least kn (Each vertex is incident on at least k edges). Since every edge is incident to exactly two vertices, G must have at least $\frac{kn}{2}$ edges. Hence,

$$|E| = m \geq \frac{kn}{2}$$

The Super Greedy Step: Pick an edge $e \in E$ at random. What is the probability that it is in the min-cut.

$$Pr(e \in \text{min-cut}(G)) = \frac{k}{|E|} \leq \frac{k}{\frac{nk}{2}} \leq \frac{2}{n}$$

$$Pr(e \notin \text{min-cut}(G)) \geq 1 - \frac{2}{n}$$

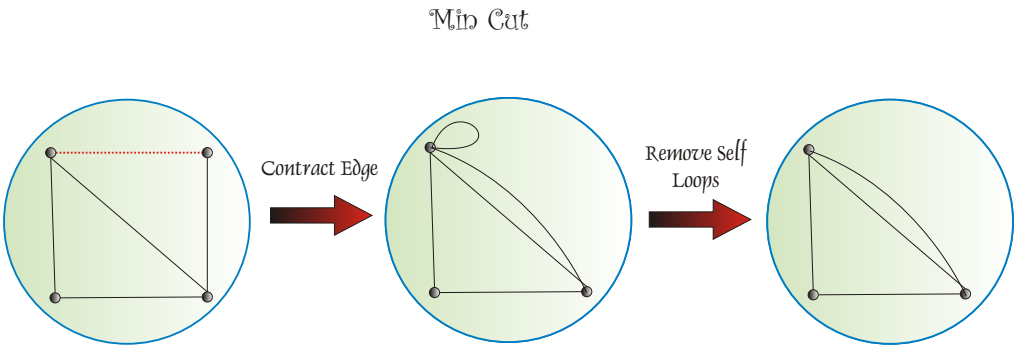
How do we use the facts that we have already learnt to design an algorithm. We know that if we keep contracting edges not in the min-cut, we will reach to a point where the graph has only two nodes left and the min-cut is the edges connecting those two nodes. But we do not know the min-cut to only contract edges not in the min-cut. We do know how to pick up an edge not in the min-cut with non-zero probability. This is the fact that we are going to use to design an algorithm which works!

Algorithm 1 Outputs the size of the min-cut

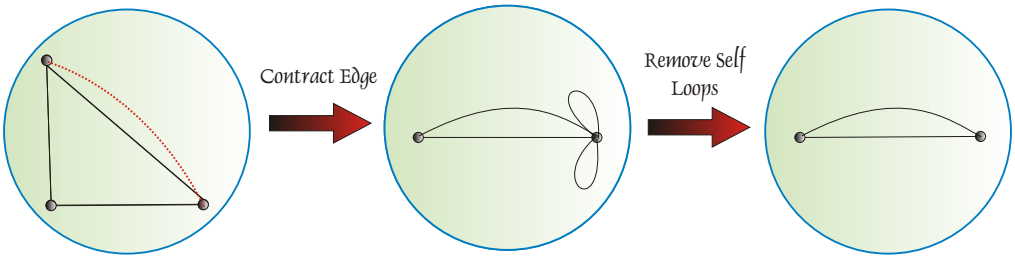
Require: A Multigraph G

1: **while** $|V| \geq 2$ **do**
2: Pick an edge e randomly and contract it
3: Remove Self Loops
4: **end while**
5: return $|E|$;

Once we have collapsed the first random edge, the rest of the algorithm proceeds recursively on the remaining $(n-1)$-node graph. Here is a lucky example.



1st iteration of the while loop



2nd iteration of the while loop

Look at what happens to the red edges after they are contracted. They become self loops and are removed. After the i-th iteration of the while loop, or the i-th contraction, how many nodes remain: $n - i$. Let us assume that no minimum-cut edge got contracted. (We want to know what is the probability of this 'lucky event' happening) Each node in the contracted graph has degree $\geq k$ (Why? Because contractions do not decrease the min-cut). Thus after the i-th iteration, the total number of edges in the graph is $\geq \frac{k(n-i)}{2}$.

$$Pr(\text{Choosing an edge in the min-cut in } i\text{-th contraction}) = \frac{k}{\text{Remaining Edges}} \leq \frac{k}{\frac{k(n-i)}{2}} = \frac{2}{n-i}$$

$$Pr(\text{Not Choosing an edge in the min-cut in } i\text{-th contraction}) \geq 1 - \frac{2}{n-i}$$

In the whole while loop of Algorithm 1, we have $n - 2$ contractions.

$$Pr(\text{We do not choose any min-cut edge in the while loop of Algorithm 1})$$

is equal to the probability that we do not choose an edge in the min-cut in the first contraction *AND* we do not choose an edge in the min-cut in the second contraction *AND* ... we do not choose a min cut edge in the $n-2$-th contraction. If we do not choose any edge from the min-cut, Algorithm 1 gives us the correct answer. Hence, Mathematically ,

$$\begin{aligned} Pr(\text{Success}) &\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{3}{5}\right)\left(1-\frac{2}{4}\right)\left(1-\frac{1}{3}\right) \\ &= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(1-\frac{3}{5}\right)\left(1-\frac{2}{4}\right)\left(1-\frac{1}{3}\right) \\ &= \frac{2}{n(n-1)} \geq \frac{2}{n^2} \end{aligned}$$

Hence, the Probability that algorithm 1 succeeds is at least $\frac{2}{n^2}$. How do we amplify this probability? By running Algorithm 1 many times and picking up the best answer(The run that gives us the minimum number of edges between the last two vertices left).

Note that Algorithm 1 and 2 can easily be modified to return the actual min-cut instead of the size of the min-cut. What is the probability of error for algorithm 2 if we run it for some positive constant $c > 1$? Let us analyze it for $c = 100$.

Algorithm 2 Outputs the size of the min-cut

Require: A Multigraph G

1: MinCutSize $= \infty$
2: **for** $i = 1$ to $\frac{cn^2}{2}$ **do**
3: Run Algorithm 1. Let x be the answer returned
4: if $x \leq$ MinCutSize then MinCutSize$= x$
5: **end for**
6: return x

$$Pr(\text{Error}) = Pr(\text{Min-Cut contracted each time in the For loop}) \leq \left(1 - \frac{2}{n^2}\right)^{\frac{cn^2}{2}} \approx \left(\frac{1}{e}\right)^c$$

This comes from the fact that

$$\lim_{x \to \infty} \left(1 - \frac{1}{x}\right)^x = \frac{1}{e}$$

What is the running time of Algorithm 1? We can implement an edge contraction in $O(n)$ time. Edge contraction is called for $n-2$ edges in Algorithm 1. Hence algorithm 1 takes $O(n^2)$ time to run. Algorithm 2 runs Algorithm 1 for $\frac{cn^2}{2}$ times. So the total cost of running Algorithm 2 is $\frac{cn^2}{2} \times O(n^2) = O(n^4)$. Its not very hard to modify algorithm 2 to run in $O(n^2 \text{polylog}(n))$. Note that randomized min-cut is a monte-carlo algorithm.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2nd ed., 2001.

[2] R. Motwani and P. Raghavan, *Randomized Algorithms*, CUP, Stanford, 1995.