princeton univ. F'15 cos 521: Advanced Algorithm Design

Lecture 2: Karger's Min Cut Algorithm

Lecturer: *Pravesh Kothari* Scribe:*Pravesh*

(These notes are a slightly modified version of notes from previous offerings of the class scribed by Sanjeev.)

Today's topic is simple but gorgeous: Karger's min cut algorithm and its extension. It is a simple randomized algorithm for finding the *minimum cut* in a graph: a subset of vertices S in which the set of edges leaving S, denoted $E(S, \overline{S})$ has minimum size among all subsets. You may have seen an algorithm for this problem in your undergrad class that/ uses maximum flow. Karger's algorithm is elementary and and a great introduction to randomized algorithms.

1 Karger's Algorithm

The basic subroutine in Karger's algorithm is *edge-contraction*: given an edge $e = \{u, v\}$ in a graph G with vertices V (of size n) and edges E, contraction of e produces a new graph $G' = G \setminus e$ with $n-1$ size vertex set $V \setminus \{u, v\} \cup S_{u,v}$ where $S_{u,v}$ is a *super-node* obtained by merging u and v. In this process, we remove all edges incident to either u or v and make them incident to $S_{u,v}$ instead, remove self-loops and retain parallel edges (so at any point in the sequence of edge-contractions, we will have a multigraph in general).

Consider the following general procedure to obtain a cut: Choose an edge e, contract it and repeat. Each edge contraction decreases the number of vertices in the graph by 1. Thus, in $n-2$ steps, the number of vertices remaining in the graph is 2. The two supernodes remaining then define a cut in the original graph given by the partition corresponding to the supernodes. We want to choose a sequence of edge contractions so that the 2 supernodes remaining define a min-cut in the graph. Karger's algorithm makes this choice extremely simple- just choose an edge uniformly at random from the remaining edges in the graph!

Formally, Karger's algorithm is:

1. Repeat the following until 2 supernodes are left:
2. Pick a uniformly random edge e and perform edge-contraction with e.
3. Output the cut corresponding to the supernodes as the guess for a min-cut in the graph.

Why should this algorithm work? The intuition is that a particular cut survives $n-2$ contractions if every one of the $n-2$ edges chosen in Karger's algorithm are not from the cut. If you pick a random edge, it is more likely to come from parts of the graph that contain more edges and thus, since a min-cut should contain a small number of edges, the probability that an edge is picked from it should be small. In fact, this algorithm provides a great heuristic to try on all kinds of real-life graphs, where one wants to *cluster* the nodes into "tightly-knit"portions. For example, social networks may cluster into communities;

graphs capturing similarity of pixels may cluster to give different portions of the image (sky, grass, road etc.). Thus instead of continuing Karger's algorithm until you have two supernodes left, you could stop it when there are k supernodes and try to understand whether these correspond to a reasonable clustering.

Today we will first see that the above version of the algorithm yields the optimum min cut with probability at least $2/n^2$. Thus we can repeat it say $20n^2$ times, and output the smallest cut seen in any iteration. The probability that the optimum cut is not seen in any repetition is at most $(1 - 2/n^2)^{20n^2} < 0.01$.

Unfortunately, this simple version has running time about n^4 which is not great.

So then we see a better version with a simple tweak that brings the running time down to closer to n^2. The idea is that roughly that *repetition ensures fault tolerance*. The real-life advice of making two backups of your hard drive is related to this: the probability that both fail is much smaller than one does. In case of Karger's algorithm, the overall probability of success is too low. But if run part of the way until the graph has $n/\sqrt{2}$ supernodes, the chance that the mincut hasn't changed is at least $1/2$. So you make two independent runs that go down to $n/\sqrt{2}$ supernodes, and recursively solve both of these. Thus the expected number of instances that will yield the correct mincut is $2 \times \frac{1}{2} = 1$. (Unwrapping the recursion, you see that each instance of size $n/\sqrt{2}$ will generate two instances of size $n/2$, and so on.) Simple induction shows that this 2-wise repetition is enough to bring the probability of success above $1/\log n$.

As you might suspect, this is not the end of the story but improvements beyond this get more hairy. If anybody is interested I can give more pointers.

Also this algorithm forms the basis of other algorithms for other tasks. Again, talk to me for pointers.

2 Analysis of Karger's algorithm

Clearly, the two supernodes at the end correspond to a cut of the original graph, so the algorithm does always return a cut.

Let's analyze what happens to a given cut under a sequence of edge contractions. The following observation is easy to verify but quite useful.

Observation 1
Let G' be obtained by a sequence of edge contractions of G. Then, there's a one-one correspondence between every cut $(Y, \bar{Y})$ of G' and the cut $(X, \bar{X})$ obtained by taking the union of nodes of G in the supernodes in Y.

Let's now formalizing the intuition that an edge is likely *not* to be from a min-cut. In the following, the notation $e \sim E$ will denote a uniformly random draw of an edge e from E.

Lemma 1
Let $(X, \bar{X})$ be a min-cut of a graph G on n vertices. Let G' be a graph obtained by a sequence of t edge contractions for $n - 2 \geq t \geq 0$ obtained by a sequence of edge contractions where each of the chosen edges are not from the cut $(X, \bar{X})$. Then, $\Pr_{e \sim E(G')}[e \in (X, \bar{X})] \leq 2/n$.

PROOF: It is easy to show this lemma for the case of $t = 0$ (i.e. when $G' = G$.) Suppose that the size of the min-cut in the graph G is k. Then, observe that the degree of every vertex must be at least k (otherwise, the vertex v forms a cut $(v, V \setminus v)$ of size smaller than k). Thus, the total number of edges in the graph must be at least $nk/2$ and a uniformly random edge has probability of being inside a given min-cut is at most $2/n$.

For the general case, we only need observe that the min-cut of G' is of size exactly k. This is because by Observation 1 above, every cut in G' corresponds to a cut in G which has size at least k and the cut $(X, \bar{X})$ has a corresponding cut in G' by the assumption about the edges contracted in the process of obtaining G. □

The above estimate shows that as edge-contraction process succeeds, it becomes more and more likely that we pick an edge from a given min-cut. Nevertheless, we can obtain a decent bound on the probability that we never pick an edge from a given min-cut in all of the $n - 2$ steps.

LEMMA 2
Suppose we run Karger's algorithm till the point that there are ℓ supernodes left for $\ell \geq 2$. Then, the probability that no edge is contracted from a given min-cut $(X, \bar{X})$ in this process is at least $\frac{\binom{\ell}{2}}{\binom{n}{2}}$. In particular (by choosing $\ell = 2$), the cut at the end is a minimum cut of the original cut with probability at least $\frac{2}{n(n-1)}$.

PROOF: Let $e_1, e_2, \ldots, e_{n-\ell}$ be the sequence of edges contracted in the algorith. We can estimate this probability as: $\Pr[e_1 \notin (X, \bar{X})] \cdot \Pr[e_2 \notin (X, \bar{X}) \mid e_1 \notin (X, \bar{X})] \cdots \Pr[e_{n-\ell} \notin (X, \bar{X}) \mid e_1, e_2, \ldots, e_{n-\ell-1} \notin (X, \bar{X})]$.

Using Lemma 1, the above product can be estimated from below by:

$$(1 - \frac{2}{n}) \cdot (1 - \frac{2}{n-1}) \cdots (1 - \frac{2}{\ell+1})$$

The above expression can be easily seen to be equal to $\frac{\binom{\ell}{2}}{\binom{n}{2}}$. □

Thus, repeating the algorithm K times for $K = t\binom{n}{2}$ and taking the smallest of all the cuts found in the K runs will yield a min-cut with probability at least $1-(1-\frac{2}{n(n-1)})^K \geq 1-e^{-t}$. By making $t = \log(1/\epsilon)$, we can make the probability of Karger's algorithm succeeding in finding a min-cut to be at least $1 - \epsilon$.

(Aside: The sequence $(1 - 1/K)^K$ is monotonnically increasing and converges to $1/e$ very fast - for large enough K, we can thus approximate it by $1/e$. This is a useful technical tool and will appear in later lectures.)

It is relatively easy using data structures you learnt in undergrad algorithms to implement each repetition of the algorithm in $O(n^2)$ time. So the overall running time is $O(n^4)$.

Aside: We have proven a stronger result than we had needed to: every minimum cut remains at the end with probability at least $\frac{2}{n(n-1)}$. This implies in particular that the number of minimum cuts in an undirected graph is at most $\binom{n}{2}$ (Note that the number of cuts in the graph is the set of all nonempty subsets, which is $2^n - 1$, so this implies only a tiny number of all cuts can be minimum cuts.) This upper bound has had great impact in

subsequent theory of algorithms, though we will not have occasion to explore that in this course.

3 Improvement by Karger-Stein

Karger and Stein improved the algorithm to run in $O(n^2 \log^2(n))$ time. The idea is roughly that *repetition* ensures *fault tolerance*. The real-life advice of making two backups of your hard drive is related to this: the probability that both fail is much smaller than one does. In case of Kargers algorithm, the overall probability of success is too low at $\frac{2}{n(n-1)}$. But if run part of the way until the graph has then Lemma 1 shows that the probability that the mincut has survived (i.e. no edge in it has been contracted) is at least $\frac{1}{2}$. So you make two independent runs that go down to $n/\sqrt{2}$ supernodes, and recursively solve both of these with the same Karger-Stein algorithm. Then return the smaller of the two cuts returned by the recursive calls. The running time for such an algorithm satisfies the recurrence

$$T(n) = O(n^2) + 2T(n/\sqrt{2}) \tag{1}$$

One can now use the Master theorem [1] to show that the above recurrence has the solution: $T(n) = O(n^2 \log(n))$. As you might suspect, this is not the end of the story but improvements beyond this get more hairy. If anybody is interested I can give more pointers.

The major remaining claim of the analysis is to estimate that a given min-cut survives a run of the Karger-Stein algorithm.

LEMMA 3
Let $(X, \bar{X})$ be a min-cut of G. Then, the probability that Karger-Stein algorithm outputs $(X, \bar{X})$ is at least $\frac{c}{\log(n)}$ for some constant $c > 0$.

Thus repeating the algorithm $Cc \log(n))$ times gives a success probability at least $1 - e^{-c}$ from a calculation as before and a running time of $O(n^2 \log^2(n))$.
PROOF: To prove the lemma, we again write a recurrence relationship for the probability of survival of a min-cut $(X, \bar{X})$ in a Karger-Stein run.

If $P(n)$ is this probability, then it must satisfy:

$$P(n) \geq 1 - (1 - \frac{1}{2}P(n/\sqrt{2}))^2 \tag{2}$$

where $(1 - \frac{1}{2}P(n/\sqrt{2}))^2$ represents the probability of the event that a minimum cut survived in the shrinkage to $n/\sqrt{2}$ 2 vertices, and the recursive call then recovered this minimum cut.

To see that this has the solution $P(n) \geq c/\log(n)$, we can do a simple induction, where the inductive step needs to verify that:

$$\frac{1}{\log(n)} \leq 1 - (1 - \frac{1}{2}\frac{1}{\log(n) - 0.5})^2 = \frac{1}{\log(n) = 0.5} - \left(\frac{1}{4(\log(n) - 0.5)}\right)^2$$

[1] Hush, hush, dont tell anybody, but most researchers dont use the Master theorem, even though it was stressed a lot in undergrad algorithms. When we need to solve such recurrences, we just unwrap the recurrence a few times and see that there are $O(\log(n))$ levels, and each involves $O(n^2)$ time operations

which is true using the approximation:

$$\frac{1}{\log(n) = 0.5} \approx \frac{1}{\log(n)} + \frac{0.5}{\log^2(n)}$$

□

Bibliography

1. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. David Karger, Proc. ACM-SIAM SODA 1993.

2. A new approach to the minimum cut problem. David Karger and Cliff Stein, JACM 43(4):601640, 1996.