CSE 203A: Karger's Min-Cut Algorithm (Lecture Date: 5/11/06)

Evan Ettinger

May 19, 2006

1 General Min-Cut Problem

There are typically 2 main types of cut problems. The s-t cut problem asks how we can find the minimum cut that separates specified nodes s and t. This problem can be solved efficiently using Ford-Fulkerson. The general min-cut problem asks how we can divide the graph into two sets that has minimal cut value across the sets (i.e. without specified terminals s,t).

Instance: $G = (V, E)$ weighted, undirected

Want: $S,\ T$ cut, $S \neq \emptyset,\ T \neq \emptyset$.

Goal: minimize the value of the cut:

$$\min \sum_{\substack{e=(u,v),\\ u \in S,\ v \in T}} w(e)$$

A naive way to solve this problem combinatorially is to fix the source s and then run Ford-Fulkerson on every possible sink t and take the overall minimum cut found. However, this is an expensive algorithm that runs on the order of $n \cdot \mathcal{O}(|f^*|m)$. Even on the restricted case of unweighted graphs this procedure results in a running time of $\mathcal{O}(n^2 m) \sim \mathcal{O}(n^4)$ for dense graphs. We ask the question: Can we develop a more efficient randomized algorithm for general min-cut?

2 Karger's Randomized Min-Cut

The main idea behind Karger's algorithm is to randomly select edges to not place in the cut. However, we don't select each edge with uniform probability, but instead weight the probability that we will select an edge e by $w(e)$ (i.e. select large weight edges more often to not be in our cut). Before we present the first version of Karger's algorithm, let's first make a definition:

Definition 1. *Let $G_{uv} = (V_{uv}, E_{uv})$ be called a contracted graph if $V_{uv} = V \setminus \{u, v\} \cup c$ and has edges such that:*

- *Edges not incident to u or v are kept with the same weight.*
- *If there is an edge from u or v to $a \neq u, v$ we insert an edge (c, a) of weight $w((u, a)) + w((v, a))$.*
- *Edge between (u, v) is contracted (i.e. disappears).*

Observe that it follows from this definition that the $\text{MinCut}(G_{uv}) = \text{MinCut}(G)$ when u and v both fall on the same side of the cut in $\text{MinCut}(G)$. We are now ready to present Karger's basic algorithm.

Karger1.0(G):

1. If $|V| = 2$ then let e be the edge connecting the two nodes and return $w(e)$.
2. Pick $e = (u, v)$ at random with probability $\frac{w(e)}{\sum_{e' \in E} w(e')}$.
3. Karger1.0(G_{uv}).

Note that when $|V| = 2$ each node represents a set of contracted nodes from the original G that are now on either side of our cut. If we use the correct data structures here (i.e. tree for edge probabilities, union-find for the sets, etc.) then this algorithm will run in linear time.

Lemma 1. $P[MinCut(G_{uv}) > MinCut(G)] \leq \frac{2}{n}$

Proof. Let $W = MinCut(G)$, let S, T be the cut sets, and let $e = (u, v)$ be a random edge we've selected to contract. Then, $P[MinCut(G_{uv}) > MinCut(G)] = P[MinCut(G_{uv}) \neq MinCut(G)] < P[e \in Cut(S, T)]$. We now bound $P[e \in Cut(S, T)]$.

$$\begin{aligned} P[e \in Cut(S,T)] &= \frac{\sum_{e \in Cut(S,T)} w(e)}{\sum_{e' \in E} w(e')} \\ &= \frac{W}{\sum_{e' \in E} w(e')} \end{aligned}$$

Since W is the min-cut of G we know each single vertex isolating cut must be

greater than W, that is $\forall v \ \sum_{e'=(v,v')} w(e') \geq W$.

$$2\sum_{e' \in E} w(e) = \sum_{v \in V} \sum_{e'=(v,v')} w(e') \geq Wn$$

$$\Leftrightarrow$$

$$\sum_{e' \in E} w(e') \geq \frac{Wn}{2}$$

$$\Rightarrow$$

$$P[e \in Cut(S,T)] = \frac{W}{\sum_{e' \in E} w(e')} \leq \frac{2}{n}$$

□

We can also immediately prove a more powerful corollary using exactly the same proof technique.

Corollary 1. *Let S,T be any cut within a β factor of the optimal min-cut. Then $P[e \in Cut(S,T)] \leq \frac{2\beta}{n}$.*

Now we investigate the chance that the cut Karger1.0 finds is in fact the optimal min-cut. Let's define a sequence of graphs $G = G_0,\ G_1 = G_{uv},\ G_2 = G_{1_{uv}}, \ldots$

$$\begin{aligned} P[\text{final cut is optimal}] &= \prod_{i=1}^{|V|-2} P[MinCut(G_{i+1}) = MinCut(G_i)] \\ &\geq (1 - \frac{2}{n})(1 - \frac{2}{n-1}) \cdot \cdots \cdot (1 - \frac{2}{3}) \\ &= \frac{n-2}{n}\frac{n-3}{n-1} \cdot \cdots \cdot \frac{1}{3} \\ &= \frac{2}{n(n-1)} \in \Omega(1/n^2) \end{aligned}$$

This is a low probability of success, but if we run Karger1.0 $n^2 \ln n$ times, then with high probability we will find the optimal solution.

$$P[n^2 \ln n \text{ runs fail to find optimal}] \leq (1 - \frac{1}{n^2})^{n^2 \ln n} \leq e^{-\ln n} = \frac{1}{n}$$

However, observe that the running time of this algorithm, $\mathcal{O}(n^3 \log n)$, wouldn't be much of an improvement over the brute force combinatorial approach introduced in the first section. Also, observe that we have the highest probability of selecting an edge and placing it incorrectly when the graphs are small. This last observation will lead us to our first improvement.

3 Karger2.0

To improve the running time of Karger's algorithm, we present the 2.0 version that runs the basic step of the algorithm only $\frac{n}{2}$ and repeats this process 4 times. Performing an identical calculation as above we get:

$$P[\text{we don't misplace an edge}] = \frac{n-2}{n}\frac{n-3}{n-1}\cdots\frac{n/2-1}{n/2+1}\frac{n/2-2}{n/2} \approx \frac{1}{4}$$

Here is the algorithm:

Karger2.0(G):

1. Run Karger1.0 variant until you get 4 graphs G_1, G_2, G_3, G_4 each of size $|V|/2$.
2. Run Karger2.0 on each of $G_1,\ G_2,\ G_3,$ and G_4.

The main idea here is to create a tree like structure where we have more runs of smaller sized graphs than larger sized graphs since the chance of us placing an edge incorrectly in the cut is small when the graphs are large. The improvement in running time can be expressed by the following recurrence: $T(n) = 4T(n/2) + \mathcal{O}(n^2)$, and solving gives $T(n) \in \mathcal{O}(n^2 \log n)$.

We now ask what the probability is of Karger2.0 finding the min-cut of G. Let p_n be the minimum probability that Karger2.0 finds the min-cut.

$$\begin{aligned} p_n &\geq 1 - (1 - \frac{1}{4}p_{n/2})^4 \\ &= p_{n/2} - \frac{3}{8}p_{n/2}^2 + \frac{1}{16}p_{n/2}^3 - \frac{1}{256}p_{n/2}^4 \\ &\geq p_{n/2} - \frac{3}{8}p_{n/2}^2 \end{aligned}$$

Russell Solves Here...

We find $p_n \in \mathcal{O}(1/\log n)$ which isn't very good for a single run, but we can now create Karger2.1 where we run Karger2.0 $\log^2 n$ times and output the best cut of these runs. Karger2.1 has running time $\mathcal{O}(n^2 \log^3 n)$ which is significantly better than the naive combinatorial implementation first presented and will output the min-cut with high probability.

4 β-Factor Min-Cuts

Let $\beta > 1$. An interesting consequence of Karger's algorithm is that we can find a bound on the number of cuts a graph has that are within β of optimal.

Theorem 1. *There are at most $\mathcal{O}(n^{2\beta})$ cuts within β of optimal.*

This is rather surprising since there are an exponential (2^n) number of cuts in a graph. To prove the theorem we present an algorithm, Karger3.0, that outputs all sets S,T that define cuts that are within β of the min-cut.

Karger3.0(G):

1. Repeat cn^k times for some $c > 0,\ k > 2\beta$:
   (a) Run Karger1.0 until you have a graph of size $2\beta + 1$.
   (b) In graph of size $2\beta + 1$ output all $2^{2\beta+1}$ cuts.

Let S, T be any cut within a factor of β of the min-cut. The probability that one run of Karger3.0 finds this cut is:

$$\begin{aligned} P[\text{one loop of Karger3.0 finds S,T}] &\geq (1 - \frac{2\beta}{n})(1 - \frac{2\beta}{n-1}) \cdots (1 - \frac{2\beta}{2\beta+1}) \\ &\geq \frac{(2\beta)(2\beta-1)}{n(n-1)\cdots(n-2\beta+1)} \\ &\geq \frac{1}{n^{2\beta}} \end{aligned}$$

Theorem 1 follows immediately from this result.