Second Level Algorithms

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 08

Lecture 39

Welcome to the 39th lecture of the second-level algorithms course. In the last class, we have seen how we can use the maximum st-flow algorithms to compute a minimum cut in a graph, and we have also seen an example of the execution of Karger's min-cut algorithm. So, in this lecture, let us begin with the pseudocode of Karger's min-cut algorithm. So, here is the pseudocode of Karger's min-cut algorithm. While the number of vertices in G is more than 2, pick an edge E uniformly randomly from the set of edges of G, then we contract G, that is it, and then this loop terminates when we have only 2 vertices. So, let $(X, V \setminus X)$ be the cut induced by the two vertices of G after the while loop terminates, then it returns this particular cut, ok. So, a couple of important points about this algorithm: first of all, this is a Monte Carlo-type randomized algorithm. That means it may not always output correctly. So, we need to see what is the probability that it can make an error, which we will see, but what other kind of randomized algorithms can there be? So, we have seen or we know the randomized quicksort algorithm, which picks a pivot uniformly at random, and that algorithm is a Las Vegas kind of randomized algorithm.

Those kind of randomized algorithms always output the correct answer but the runtime can be different. Those are Las Vegas kind of randomized algorithms. Examples are randomized quicksort. On the other hand, for Karger's, for this algorithm Karger's min cut, this is a Monte Carlo type of randomized algorithm. That means, its runtime is more or less fixed, but it may not always output the correct answer.

So, an important question is what is the error probability or probability of success what is the probability that Karger's min cut algorithm indeed outputs a min cut so to analyze that error that success probability let us fix a min cut say $(X, V \setminus X)$ of G, F be the set of cut edges and suppose the cardinality of F which is the value of the min cut that number let it be k. So, a crucial observation is that the algorithm outputs $(X, V \setminus X)$ if and only if it

never picks any edge from F to contract, ok. So, probability of success which is probability that a min cut is output. This is greater than equal to probability that the algorithm outputs this particular min cut $(X, V \setminus X)$. This is so because the minimum cut of the graph G may not be unique; there could be more than one min cut. But this probability is the same as the probability that no edge from F is ever picked by the algorithm for contraction. OK, but this is the same as this probability by the chain rule: the probability that no edge from F is contracted in the first iteration times the probability that no edge from F is contracted in the second iteration given no edge from F is contracted in the first iteration, the probability that no edge from F is contracted in the ith iteration given the probability that no edge from F is contracted in 1 to $i-1$ iterations, and so on up to Karger's min-cut algorithm makes $n-2$ iterations; the number of vertices decreases by 1 in every iteration. So, in the $(n-2)$-th iteration, given no edge from F is contracted from 1 to $(n-3)$-th iteration.

So, we will calculate these probabilities, bound these probabilities, and plug the value here. So, first, let us see what is the probability that no edge from F is contracted in the first iteration. So, for that, let us assume that the minimum degree of any vertex of is d. Then we have k must be less than or equal to d because if there is a vertex with degree d then that means, there is a cut with size d where that particular vertex of degree d is in one part of the bipartition and all the other vertices is in the other part of the bipartition. So, now, let us find out the probability that no edge from if is contracted in the first iteration So, because edges are picked uniformly at random, this probability is same as the number of edges in F by total number of edges. But the number of edges in F we have assumed is k by total number of edges.

But what is total number of edges? we know the minimum degree is d. So, the sum of degrees is at least $d \times n$ and sum of degrees is twice the number of edges. So, this is greater than equal to $\frac{k}{nd/2}$. This is so because cardinality E is half times sum of degrees of all the vertices and this is greater than equal to $d \times n/2$. ok.

So, we get this is equal to $2\frac{k}{nd}$, but now we know that $k \leq d$. So, because $k \leq d$ this implies $\frac{d}{k}$, sorry, no edge from F is contracted in the first iteration is $1-t$. not cardinality $\frac{|F|}{|E|}$ is the probability that one edge is contracted in the first iteration this is one minus and this is also one minus now it is going in the same direction because of the negative sign and the denominator and so this implies that $1-2\frac{k}{d}$ this implies $\frac{k}{d}$ is less than equal to $1-\frac{k}{d}\frac{(n-2)}{}\geq 1$. So, this is greater than equal to $1-\frac{2}{n}$ which is $\frac{(n-2)}{n}$. So, with probability at least $\frac{(n-2)}{}$, the Karger's min cut algorithm does not pick any edge from the minimum

cut that we have fixed for the analysis in the first iteration. In the second iteration, the number of vertices drops by 1 and so, if I replace n with $n-1$, I get the bound with the bound for the probability for the second iteration and so on. That is why I derived a bound which depends only on n not on d or because the minimum degree d can change in every iteration. you know although number of vertices also changes in every iteration I know it drops by exactly one it behaves much more predictably than the behavior of d we know we do not know how the minimum degree of the graph evolves over the iterations.

So, probability that the Karger's min cut algorithm outputs this particular min cut $(X, V \setminus X)$ is at least it does not pick any edge from f in the first iteration that happens with probability $\frac{(n-2)}{n}$ at least this probability it does not pick any edge from f in the second iteration also i just need to replace n with $n-1$ and get the bound this chain of multiplications telescope and this the last two numerator and the first two denominator remains so this is $\frac{2}{n(n-1)}$ you see the success probability is not so much actually it is quite small it decreases as n increases.

So, we remedy this by repeating the algorithm many times and output the best min-cut found. The success probability is only $\frac{2}{n(n-1)}$. We improve this bound by boosting technique what is boosting technique we repeat the algorithm say l number of times we will let our analysis tell us what should be the value of L and that we will fill up later, but for now let us repeat the algorithm L number of times and return the minimum of this L min cuts ok. Because it is the randomized algorithm every run of the algorithm will return different may be different min cuts it not be same. So, if you run L times you may get L different min cuts. among all those min cuts you find out the minimum and output that.

So, now, let us find out the probability of success probability of success after l repetitions L repeats. So, the failure prob so, this is greater than equal to 1 minus the failure probability. So, for this repeated algorithm to fail in all of the L runs which are independent of each other, it should not output a min cut. So, it should fail in all of the l runs and the failure probability of one run is at most 1 - $\frac{2}{n(n-1)}$ and we repeat it l times they are independent.

So, to fail all the l times the probability of failure for consecutive L times is 1 - $\frac{2}{n(n-1)}$ to the power L. So, now, we have to pick L, but we need a more workable bound. So, for that, we use a bound which is $1+x \leq e^x$ for all real number x. So, this is using that bound we have this is greater than equal to $\left(1-e\right)^{\frac{2L}{n(n-1)}}$ ok.

And so, now, if we pick L to be no $n(n-1)/2$ times any constant c, then this success probability becomes 1 - e^{-c}. There is a minus sign here because this is a negative sign here. Ok, so by choosing appropriate constant c, we can boost the success probability of Karger's min-cut algorithm to an arbitrary constant less than 1. What is the runtime of the algorithm?

Each run of the algorithm, which involves some contraction procedure, can be executed in $O(n^2)$ time using adjacency matrix representation of the graph. So, the time complexity of the algorithm with boosting, of course, is big O of, in each iteration takes $O(n^2)$ time using adjacency matrix, and we repeat it $O(n(n-1)/2)$ times. So, the overall time complexity is big O of n to the We see that using a much simpler algorithm, we can find a minimum cut with the same time complexity as the best algorithm that we have seen for finding a min-cut using maximum flow.

The advantage of this algorithm is that it is much simpler; the disadvantage is that it is a randomized algorithm, whereas those algorithms are deterministic algorithms. Okay, so let us stop here. Thank you.