CS 466/666 Algorithm Design and Analysis, Spring 2019. Waterloo.

Lecture 4: Approximation algorithms

We will see different ways of using randomization in designing approximation algorithms for graph problems. The first example is graph sparsification, with applications in designing fast algorithms for graph problems. The second and third examples are by randomized rounding, both on linear programs - for the congestion minimization problem and the set cover problem.

Graph Sparsification

Given an edge weighted undirected graph $G=(V,E,w)$, for a subset of vertices $S \subseteq V$, let $\delta_G(S)$ be the set of edges with one endpoint in S and another endpoint in $V-S$, and let $w(\delta_G(S))$ be the total weight of the edges in $\delta_G(S)$.

We say H is a $(1\pm\varepsilon)$-cut-approximator of G if $(1-\varepsilon)w(\delta_G(S)) \leq w(\delta_H(S)) \leq (1+\varepsilon)w(\delta_G(S))$ for all $S \subseteq V$. Note that H is on the same vertex set but may have different edge weights.

We are interested in finding a $(1\pm\varepsilon)$-cut-approximator of G that is sparse (having few edges). This is an interesting problem that leads to many surprising and beautiful results.

Today we will just see a simple first result on this problem (but it already has some surprising applications). If we have time at the end of this course, we may see the more interesting results.

Assumption: We consider a simple setting in which G is unweighted and has min-cut value $\Omega(\log n)$.

Algorithm: In this simple setting, the algorithm is very simple. Set a sampling probability p. For every edge $e \in E(G)$, put it in H with weight $\frac{1}{p}$ with probability p.

The idea is to set the expectation right - so that we expect to have p fraction of edges, and hence every cut the expected weight is the same as in the original graph.

Of course, it is not enough to just have the expected values right, as we need to ensure that <u>all</u> cuts have approximately the same weights simultaneously, for which we will use Chernoff bound and the assumption that the min-cut value is at least $\Omega(\log n)$ (recall that $\Omega(\log n)$ is the regime that we can expect tight concentration). The following is the precise statement.

Theorem Set $p = 9\ln n/(\varepsilon^2 c)$ where c is the min-cut value of G.

Then H is a $(1\pm\varepsilon)$-cut-approximator of G with $O(p\cdot|E(G)|)$ edges with prob $\geq 1-\frac{4}{n}$.

Proof Consider a subset $S\subseteq V$. Say $\delta_G(S)$ has k edges. Note that $k\geq c$ by definition.

By linearity of expectation, $E[|\delta_H(S)|] = pk$ and thus $E[w(\delta_H(S))] = p\cdot k\cdot\frac{1}{p} = k = |\delta_G(S)|$.

So, the expected value is correct.

Next, we bound the probability that the actual value is "far" from the expected value.

Since each edge is an independent 0-1 random variable, by Chernoff, we have.

$$\Pr\left[\left|\,|\delta_H(S)| - pk\,\right| > \varepsilon pk\right) \leq 2e^{-pk\varepsilon^2/3} = 2e^{-\left(\frac{9\ln n}{\varepsilon^2 c}\right)\left(\frac{k\varepsilon^2}{3}\right)} = 2e^{-\frac{3k\ln n}{c}}.$$

Since $k\geq c$ by definition, it follows that this probability is at most $2/n^3$.

So, the probability that one cut is violated is pretty small but there are exponentially many cuts, and a naive union bound would not work.

The important observation is that the probability that a large cut is violated is much smaller, and there are not many small cuts!

Claim: The number of cuts with at most αc edges for $\alpha\geq 1$ is at most $n^{2\alpha}$.

Proof: We have seen the proof for $\alpha=1$. The proof for the general case is similar but requires a slight modification. We leave it as an exercise. □

Now, with the claim, we can bound

$$\begin{aligned}
&\Pr(\text{Some cut } S \text{ is violated})\\
&\leq \sum_{S\subseteq V} \Pr(\text{cut } S \text{ is violated}) && \text{// union bound}\\
&= \int_{\alpha\geq 1} \sum_{S\subseteq V:\,|\delta_G(S)|=\alpha c} \Pr(\text{cut } S \text{ is violated}) && \text{// grouped by size}\\
&\leq \int_{\alpha\geq 1} n^{2\alpha}\cdot\Pr(\text{cut } S \text{ is violated} \mid |\delta_G(S)|=\alpha c) && \text{// by the claim}\\
&\leq \int_{\alpha\geq 1} n^{2\alpha}\cdot 2e^{-3\alpha\ln n} && \text{// by Chernoff}\\
&= \int_{\alpha\geq 1} 2n^{-\alpha} \leq 4/n.
\end{aligned}$$

Therefore, with probability $\geq 1-\frac{4}{n}$, all cuts are within $(1\pm\varepsilon)$-factor as in the original graph.

Finally, again by Chernoff bound, it is easy to show that the number of edges in H is $O(p|E(G)|)$.

This completes the proof. □

Remark: Just using Chernoff bound and the union bound can already solve many interesting problems.

Applications: One natural application is to design fast approximation algorithms for cut problems.

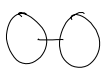
Take your favorite cut problem, say minimum s-t cut.

The running time of the algorithms usually depends on $|E|$, which could be $\Omega(|V|^2)$.

To speedup, we can first "sparsify" G by constructing a $(1\pm\varepsilon)$-cut approximator H with fewer edges.

Then, a minimum s-t cut in $S \subseteq V(H)$ is a $(1+3\varepsilon)$-approximation of the minimum s-t cut in G.

This gives a tradeoff between the approximation guarantee and the running time.

Improvements: Without the minimum cut assumption, then it is easy to see that a uniform sampling algorithm won't work, e.g. in  , it is very likely that the cut edge is not picked.

(optional)

Benczur and Karger designed a very clever non-uniform sampling algorithm, where the sampling probability for each edge is inversely proportional to the "connectivity" of the two endpoints.

They defined a notion called "strong connectivity" and proved that sampling inversely proportional to it will result in a $(1\pm\varepsilon)$-cut-approximator with $O(n\log n)$ edges.

Furthermore, they showed that there is an almost linear-time algorithm to estimate the strong connectivity which is good enough for the purpose, leading to an $\widetilde{O}(n^2)$-algorithm for approx min s-t cut.

Minimum Cut in Near Linear Time (Sketch, optional)

It is an amazing result that uses graph sparsification to design faster exact algorithms.

We consider the problem of finding a global minimum cut in an undirected graph, i.e.

to remove a minimum number of edges to disconnect the input graph.

Let k be the optimal value of this problem. So, the input graph is k-edge-connected.

Karger cleverly used a classical result on spanning tree packing.

Theorem (Tutte) If a graph is k-edge-connected, then it has $\lfloor k/2 \rfloor$ edge disjoint spanning trees.

Suppose we have edge disjoint spanning trees $T_1, T_2, \ldots, T_{k/2}$. Then, at least one tree T that

crosses a minimum cut $S \subseteq V$ at most two times.

Karger observed that having such a T would help find a minimum cut efficiently, as one

just needs to consider the cuts formed by removing at most two edges from T.

He showed that the minimum cut over these cuts can be computed in $\tilde{O}(m)$ time using dynamic programming (the case that T only crosses a minimum cut once is easier).

Okay, but how to find such a tree T?

There is a known algorithm to find c edge disjoint spanning trees in $\tilde{O}(m \cdot c)$ time.

But this is not fast enough for this purpose.

The idea here, of course, is graph sparsification.

Just using the above graph sparsification result, we can sparsify the graph so that its min-cut is $O(\log n)$ while approximately preserving all the cuts.

Now, we can compute edge-disjoint spanning trees $T_1, \ldots, T_{O(\log n)}$ in the sparsifier, and one of these trees would cross a minimum cut at most two times.

So, doing dynamic programming on each tree would work, and total complexity is $\tilde{O}(m)$.

This is just a sketch of the main ideas, with many missing details.

Congestion minimization

This is probably the first example of doing randomized rounding on linear programming solution.

Input : A directed or undirected graph $G = (V, E)$, k source-sink pairs (s_i, t_i)

Task : For each pair (s_i, t_i), find a path P_i connecting s_i to t_i, so that each edge e is used by at most C paths.

Objective : Minimize the congestion C.

Integer linear program formulation

This problem is NP-complete and we will design an approximation algorithm for it.

The general approach is to formulate the problem as an integer linear program and then "relax" it.

Let $\mathcal{P}_i$ be the set of all s_i-t_i paths in the graph. (There could be exponentially many.)

For each path $P \in \mathcal{P}_i$, we create a variable f_P^i, with the intention of setting it to be one if we choose this path to connect s_i and t_i, otherwise we set it to zero.

The congestion minimization problem can be formulated as an integer linear program.

minimize C

subject to $\sum_{P \in \mathcal{P}_i} f_P^i = 1 \quad \forall\, 1 \le i \le k$ // exactly one path in P_i is chosen to connect s_i and t_i

$\sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i : e \in P} f_P^i \le C \quad \forall e \in E$ // each edge e is used in at most C times

$f_P^i \in \{0,1\} \quad \forall i \; \forall p \in \mathcal{P}_i$

Since this is an exact formulation of the NP-complete problem, we do not know how to solve it in polyti

Linear programming relaxation

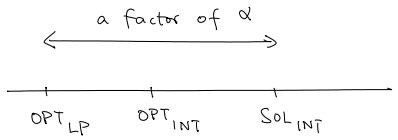
So, we relax the discrete constraint $f_P^i \in \{0,1\}$ by $0 \le f_P^i \le 1$, and then the linear program can be solved in polynomial time. (Even though there are exponentially many variables, as there is an equival linear program with only polynomial number of variables. Not our focus ; details omitted.)

However, in the relaxation, we allow "fractional solutions" and these may not correspond to the solution that we wanted (i.e. s_i-t_i path).

Rounding linear programming solution

Nevertheless, the optimum of this LP serves as a lower bound on the optimum of the original problem.

Suppose we could find an integral solution which has congestion at most $\alpha \cdot OPT_{LP}$, then it is a k-approximate solution to the original problem.



$\Rightarrow \frac{SOL_{INT}}{OPT_{LP}} \le \alpha \Rightarrow \frac{SOL_{INT}}{OPT_{INT}} \le \alpha.$

Randomized rounding : The idea is to interpret f_P^i as the probability of choosing path P to connec s_i and t_i. Then, the algorithm is for each i, pick one path P according to the probability f_P^i.

Theorem The maximum congestion is $O(C \ln n / \ln\ln n)$ with probability $\ge 1 - \frac{1}{n}$.

proof Let $X_e^i = \begin{cases} 1 & \text{if the path connecting } s_i \text{ and } t_i \text{ uses the edge } e \\ 0 & \text{otherwise} \end{cases}$.

Let $X_e = \sum_{i=1}^{k} X_e^i$, the congestion on e.

Note that $E[X_e^i] = \sum_{P \in \mathcal{P}_i : e \in P} f_P^i$.

And thus $E[X_e] = \sum_{i=1}^{k} E[X_e^i] = \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i : e \in P} f_P^i \le C$ by the LP constraint.

Since each X_e^i is an independent 0-1 variable, we can use Chernoff bound to obtain

$$\Pr(X > (1+\delta)C) < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^C \leq \frac{e^\delta}{(1+\delta)^{1+\delta}} \quad \text{as} \quad C \geq 1.$$

If we set $\delta = \Theta(\ln n / \ln\ln n)$, then $(1+\delta)\ln(1+\delta) = \Omega(\ln n)$.

We can choose the hidden constant to be large enough so that $(1+\delta)^{1+\delta} \geq n^4$.

This implies that each edge has congestion $\Omega(c\ln n/\ln\ln n)$ with probability $\leq \frac{1}{n^3}$.

By the union bound, some edge has congestion $\Omega(c\ln n/\ln\ln n)$ is $\leq \frac{1}{n}$, as there are at most n^2 edges in the graph. □

Open question: This is still the best approximation algorithm for the congestion minimization problem.

References:

- Karger, Random sampling in cut, flow and network design problems, 1994.
- Benczur and Karger, Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time, 1996.
- Karger. Minimum cuts in near linear time, 2000.
- Raghavan and Tompson. Randomized rounding: A technique for provably good algorithms and algorithmic proof, 1987.