CS 761 : Randomized Algorithms , Spring 2023, Waterloo

Lecture 4 : Approximation Algorithms

We see some applications of Chernoff bound in designing approximation algorithms.

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.

$$\text{minimize} \quad C$$

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

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

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

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

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 equivalent linear program with only polynomial number of variables. Not our focus ; details omitted.)

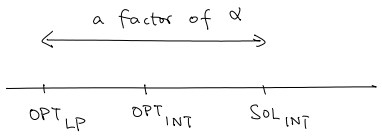
However, in the relaxation, we allow "fractional solutions" and these may not correspond to the solution that we wanted (i.e. s_i-t_i path).

Rounding linear programming solution

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.



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

Randomized rounding : The idea is to interpret f_P^i as the probability of choosing path P to connect 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 $\geq 1 - \frac{1}{n}$.

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

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

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

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. Can you beat it ?

Graph Sparsification

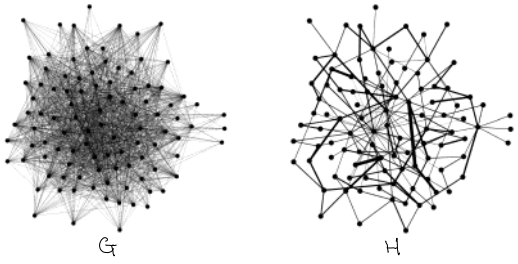
Given an undirected graph $G = (V, E)$ with a weight $w(e)$ on each edge $e \in E$, for a subset of vertices $S \subseteq V$, let $\delta_G(S)$ be the set of edges with one endpoint in S and one endpoint in $V - S$, and let $w(\delta_G(S)) = \sum_{e \in \delta_G(S)} w(e)$ be the total weight of the edges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

and let $w(\delta_G(S)) = \sum_{e \in \delta_G(S)} w(e)$ be the total weight of the edges in $\delta_G(S)$.

We are interested in finding a "sparse" graph H that approximates the cut structures of G well.

Definition ($(1\pm\varepsilon)$-cut approximator) We say $H=(V,F)$ is an $(1\pm\varepsilon)$-cut approximator of $G=(V,E)$ if for all $S \subseteq V$ we have $(1-\varepsilon)\, w(\delta_G(S)) \le w(\delta_H(S)) \le (1+\varepsilon)\, w(\delta_G(S))$.

Note that G and H are defined on the same vertex set but could have different edge sets.



(picture from Nick Harvey's notes)

This turns out to be an important and beautiful problem with many interesting and surprising results.

Today we will see the first and simplest result in this direction, and mention one surprising application in designing near-linear time algorithm for minimum cut.

This first result needs a somewhat strong assumption on the input graph.

Assumption: The input graph $G=(V,E)$ is unweighted and has min-cut value $\Omega(\log|V|)$.

With this assumption, there is a very simple uniform sampling algorithm to sparsify a dense input graph.

Algorithm: Set a sampling probability p.

For every edge $e \in E(G)$, with probability p, put e in H with edge weight $w_e = \frac{1}{p}$.

The idea is to set the expectation right, so that we expect to choose p-fraction of edges, and make their weight to be $\frac{1}{p}$, so that the expected total weight in each cut in H is the same as that in G.

But, of course, it is not enough to have the expected values to be correct, as we need to ensure that all cuts in H have approximately the same weight as that in G simultaneously.

For this, we will use Chernoff bound and crucially the assumption that the min-cut value is $\Omega(\log n)$, where we recall that expected value $\Omega(\log n)$ is the regime that we can achieve tight concentration.

Theorem (Karger) Set $p = \frac{15 \ln n}{\varepsilon^2 c}$, where c is the min-cut value of G.

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

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

By linearity of expectation, $\mathbb{E}[|\delta_H(S)|] = \sum_{e \in \delta_G(S)} \mathbb{E}[x_e] = \sum_{e \in \delta_G(S)} (p \cdot 1 + (1-p) \cdot 0) = p\,|\delta_G(S)| = pk$ where x_e

is an indicator variable where $x_e = 1$ if the edge e is added to H and $x_e = 0$ otherwise.

Similarly, $\mathbb{E}[w(\delta_H(S))] = \frac{1}{p} \cdot p \cdot |\delta_G(S)| = |\delta_G(S)| = k$, and so the expected values are what we want.

Next, we consider the probability that the actual value of $|\delta_H(S)|$ is "far" from the expectation.

Since $|\delta_H(S)|$ can be written as a sum of independent 0-1 variables (i.e. $\sum_{e \in \delta_G(S)} x_e$), we can apply Chernoff bound and get $\Pr\left(\left||\delta_H(S)| - p \cdot k\right| \geq \varepsilon p k\right) \leq 2e^{-\varepsilon^2 pk/3} = 2e^{-\frac{5k}{c}\ln n} = 2n^{-\frac{5k}{c}}$,

where we used our choice of $p = 15\ln n / (\varepsilon^2 c)$.

Recall that $k \geq c$ by definition, so the probability that $\delta_H(S)$ violates the requirement of an $(1 \pm \varepsilon)$-cut approximator is at most n^{-5}. which is pretty small.

But there are exponentially many subsets of vertices $S \subseteq V$, and so a naive union bound won't work.

The important observation is that the probability that a large cut (i.e. k large) is violated is much smaller (i.e. $n^{-\frac{5k}{c}}$), and there are not many small cuts!

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

proof We have done the case when $\alpha = 1$. The general case is similar with a modification. The proof is left as an exercise or a homework problem. □

With this lemma, we can do a more careful union bound based on the size of the cuts:

$$
\begin{aligned}
&\Pr(\text{some cut is violated}) \\
&\leq \sum_{S \subseteq V} \Pr(\text{cut } S \text{ is violated}) && \text{// union bound} \\
&= \sum_{\alpha = 1,2,4,8,\ldots,\ S \subseteq V:\ \alpha c \leq |\delta_G(S)| \leq 2\alpha c} \Pr(\text{cut } S \text{ is violated}) && \text{// dividing into cases based on the cut size; grouped into } 2^i c \leq |\delta_G(S)| \leq 2^{i+1} c \\
&\leq \sum_{\alpha = 2^i :\ 0 \leq i \leq \infty} n^{4\alpha} \cdot \Pr\left(\text{cut } S \text{ is violated} \mid \alpha c \leq |\delta_G(S)| \leq 2\alpha c\right) && \text{// by the lemma} \\
&\leq \sum_{\alpha = 2^i :\ 0 \leq i \leq \infty} n^{4\alpha} \cdot 2n^{-5\alpha c/c} && \text{// by the Chernoff bound above} \\
&= \sum_{\alpha = 2^i :\ 0 \leq i \leq \infty} 2n^{-\alpha} \\
&\leq 4/n && \text{// first term dominated}
\end{aligned}
$$

Therefore, with probability at least $1 - \frac{4}{n}$, H is an $(1 \pm \varepsilon)$-cut approximator of G.

Finally, by a simple application of Chernoff bound, H has $O(p \cdot |E(G)|)$ edges with high probability.

This completes the proof. □

Remark Using Chernoff bound and union bound can already solve many interesting problems.

Example: This result is restrictive, since it assumes the graph has a somewhat large min-cut value.

But for graphs that are essentially complete (i.e. $c=\Omega(n)$), the theorem says that there is an $(1\pm\varepsilon)$-cut approximator with $O(n\log n/\varepsilon^2)$ edges, a significant sparsification from $\Omega(n^2)$ edges in the input graph.

Applications: One natural application is to design fast approximation algorithms for cut problems.
Take our favorite problem about cuts, say the minimum s-t cut problem.
The running time of the algorithms usually depend on $|E|$, which could be $\Omega(|V|^2)$.
To speedup, we first sparsify G by constructing an $(1\pm\varepsilon)$-cut approximator H with fewer edges.
Then, run an (exact) algorithm on H to find a minimum s-t cut $S\subseteq V$, and return S as our approximate min s-t cut on G, and it can be shown that it is a $(1+3\varepsilon)$-approximation.
This gives us a tradeoff between runtime and approximation guarantee.

Improvement by Benczur and Karger: Without the minimum cut assumption, it is easy to see that the same uniform sampling algorithm won't work, e.g. in  , it is very likely the cut edge is not chosen.
Benczur and Karger designed a very clever non-uniform sampling algorithm, where each edge is sampled with probability inversely proportional to the "connectivity" of the two endpoints.
They defined a notion called "strong connectivity" and proved that sampling with probability inversely proportional to it will result in an $(1\pm\varepsilon)$-cut approximator with $O(n\log n/\varepsilon^2)$ edges for any graph.
Furthermore, they showed how to compute this ε-cut sparsifier in near linear time, and this for example gives the first $\tilde{O}(n^2)$-time algorithm for computing approximate min s-t cut.

We will not discuss this result in more detail. Instead, we will study a stronger result on spectral sparsification that uses linear algebra to solve the problem.

Minimum cuts in near linear time (optional, sketch)

An amazing application of Karger's sparsification result is a near-linear time algorithm for minimum cuts.
Let c be the min-cut value of the input graph G. So, G is c-edge-connected.
Karger cleverly used a classical result about spanning tree packing, by Tutte and also by Nash-Williams.

Theorem: If G is c-edge-connected, then G has at least $\lfloor c/2 \rfloor$ edge-disjoint spanning trees.

Suppose we have edge-disjoint spanning trees $T_1, T_2, T_3, \ldots, T_{\lfloor c/2 \rfloor}$.
Then, we know that there exists some tree T_i that crosses a minimum cut S at most 2 times.
Karger observed that having such a tree T_i would be very helpful in finding that minimum cut S.

because one just needs to search over all cuts formed by removing two edges from T_i.

He came up with a sophisticated dynamic programming $\tilde{O}(m)$ time algorithm to find a minimum cut formed by removing at most two edges from T_i (the case of removing one edge is easier)

Okay, but how to find such a tree T_i?

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

But this is too slow for our purpose.

The idea here, of course, is to do graph sparsification.

Using karger's graph sparsification result, we can sparsify the graph so that its min-cut value is $O(\log n)$ while preserving the value of each cut approximately.

Now, we can compute $T_1, T_2, \ldots, T_{O(\log n)}$ in the sparsifier, and one of these trees will cross a minimum cut at most two times.

So, doing dynamic programming on each of these $O(\log n)$ trees would work, with total complexity $\tilde{O}(m)$!

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