4
Applications of Concentration Inequalities

4.1 Graph Cut Sparsification

In this lecture we show how to construct *cut sparsifiers* of graphs. More formally, we show how given an unweighted graph $G = (V, E)$, one can construct an edge-weighted graph $H = (V, E, w)$ with $O(\frac{1}{\epsilon^2} n \log n)$ edges such that for every $S \subseteq V$ one has

$$(1-\epsilon)|\delta_G(S)| \leq w(\delta_H(S)) \leq (1+\epsilon)|\delta_G(S)|\,.$$

Here $\delta_G(S)$ and $\delta_H(S)$ stands for the set of edges in the cut $(S, V \setminus S)$ in graphs G and H, respectively. Moreover, we use $w(F)$ to denote the total weight of edges in F in graph H. So H approximate cuts in G, up to an error $1 \pm \epsilon$, even though H is only contains roughly linearly many edges.

The first such result was due to Benczur and Karger, who showed that sampling edges with probabilities proportional to their connectivity strengths yields a sparsifier with $O(n \log n/\epsilon^2)$ edges. A different approach was later given by Spielman and Srivastava, who showed that sampling edges based on their effective resistances also produces a stronger notion of spectral sparsifier of similar size. More recently, a breakthrough result by Batson, Spielman, and Srivastava showed the existence of spectral sparsifiers with a near-optimal $O(n/\epsilon^2)$ edges, though their construction method is considerably more complex.

4.1.1 Sparsifying the Complete Graph

To see that this is even possible let us consider the simplest possible example, i.e. when the graph $G = (V, E)$ is a clique. Let $p = \frac{C \log n}{\epsilon^2 n}$ for a sufficiently large absolute constant $C > 0$. For every edge $e \in E$ include e into graph H independently with probability p, giving it weight $1/p$ if it is included. Why does this produce a cut sparsifier with high probability? First, observe that cut sizes are exactly correct

in expectation. Indeed, for every $S \subseteq V$ one has

$$\mathbb{E}[w(\delta_H(S))] = \sum_{e \in \delta_G(S)} \frac{1}{p} \cdot \Pr[e \text{ included in } H] = \delta_G(S)\,.$$

We can use the Chernoff bound to show that for subset $S \subseteq V$ one has

$$(1-\epsilon)|\delta_G(S)| \leq w(\delta_H(S)) \leq (1+\epsilon)|\delta_G(S)| \qquad (*)$$

with high probability.

Theorem 4.1. *(Chernoff Bound) let $X = \sum_{i=1}^{n} X_i$ where X_i's are independent bernoulli random variables, such that $X_i = 1$ with probability p_i and 0 otherwise. Also, let μ_X denote $\mathbb{E}(X)$. Then for any $\delta \in (0,1)$,*

$$\Pr[|X - \mu_X| > \delta\mu_X] \leq 2e^{-\frac{\delta^2\mu_X}{3}}$$

For an edge $e \in E$, let X_e be the indicator random variable that is 1 if e is included in H and 0 otherwise. Fix a cut $S \subset V$, and let $X = \sum_{e \in \delta(S)} X_e$. We have $\mu_X = \mathbb{E}[X] = p \cdot |\delta(S)| = p|S|(|V| - |S|) = pr(n-r)$, where we let $r = |S|$ for convenience. Note that it's enough to consider $r \leq n/2$ by symmetry. By Theorem 4.1 we have

$$\begin{aligned}\Pr[|X - \mu_X| > \epsilon\mu_X] &\leq 2e^{-\frac{\epsilon^2\mu_X}{3}} \\ &= 2e^{-\epsilon pr(n-r)/3} \\ &\leq 2e^{-(C/6)r\log n},\end{aligned}$$

where in the last transition we lower bounded $n - r$ by $n/2$ and used the definition of p. We now take a union bound over all cuts $(S, V \setminus S)$ with $r \leq n/2$ vertices on the smaller side, getting

$$\begin{aligned}\Pr[\exists S \subseteq V, |S| = r \text{ such that (*) fails for } S] &\leq \binom{n}{r} \cdot 2e^{-(C/6)r\log n} \\ &\leq n^r \cdot 2e^{-(C/6)r\log n} \\ &= 2n^{-(C/6-1)r}.\end{aligned}$$

A union bound over all $r = 1, 2, \ldots, n/2$ then gives that (*) is satisfied *for all* S with probability at least

$$1 - \sum_{r=1}^{n/2} 2n^{-(C/6-1)r} \geq 1 - \sum_{r=1}^{\infty} 2n^{-(C/6-1)r} \leq 1 - n^{-1}$$

as long as C is a sufficiently large absolute constant.

4.1.2 *A Simple Uniform Sampler*

The argument above only applies to the clique, as it is easy to get a good upper bound on the number of cuts of a given size in a clique.

A seminal result of Karger in 1993 shows how to generalize this argument to graphs with a large min-cut. Suppose that G has min-cut k (think of k as large). We proved how to bound the number of min-cuts in lecture #1 and we bounded the number of cuts of size αk in the exercises. In particular, we have that the number of cuts of size bounded by αk is at most

$$\binom{n}{2\alpha} 2^{2\alpha-1} \leq (2n)^{2\alpha}.$$

Now to construct a sparsifier of G we include every edge $e \in E$ into H independently with probability $p = \frac{C \log n}{\epsilon^2 k}$ for a sufficiently large constant $C > 0$. We can now take a union bound over cuts of all sizes, getting that $\Pr[\exists S \subseteq V : \text{(*) fails for } (S, V \setminus S)]$ is upper bounded by

$$\begin{aligned}
&\sum_{\alpha=2}^{\infty} \Pr[\text{a fixed cut of size } \geq (\alpha-1)k \text{ fails}] \cdot \#(\text{cuts of size } \leq \alpha k) \\
&\quad \leq 2 \sum_{\alpha=2}^{\infty} e^{-\epsilon^2(\alpha-1)kp/3} \cdot (2n)^{2\alpha} \\
&\quad \leq 2 \sum_{\alpha=2}^{\infty} n^{-(C/3)(\alpha-1)} \cdot (2n)^{2\alpha} \\
&\quad \leq 2 \sum_{\alpha=2}^{\infty} n^{-(C/6)\alpha} \cdot (2n)^{2\alpha} \\
&\quad \leq 1/n
\end{aligned}$$

as long as C is a sufficiently large absolute constant.

4.1.3 *Near-Optimal Sparsification*

The uniform sampling method is effective for graphs with a large minimum cut, but its performance depends on the size of that cut. The first result showing that it is possible to construct sparse cut approximators for any graph was due to Benczur and Karger, who gave a randomized algorithm for constructing a sparsifier with $O(n \log n/\epsilon^2)$ edges. We now outline a simpler approach that follows the ideas of Khanna, Putterman, and Sudan, which yields a near-optimal sparsifier with $O(n \cdot \text{poly}(\log n)/\epsilon^2)$ edges.

The core of the method is a procedure that combines two techniques: isolating and preserving edges from small cuts, and sampling edges from the remaining graph which is guaranteed to have large cuts.

A Single Sparsification Step. Let's first consider a single step of the process. Given an unweighted graph $G = (V, E)$ and a parameter Z:

1. **Peel Small Cuts:** We repeatedly find non-trivial minimum cuts in the graph of size at most Z. We remove the edges of these cuts

and add them to a set E_{small}. This process continues until the remaining graph has no non-trivial cuts of size less than Z. The remaining graph is a collection of components $G_1, G_2, \ldots$. Since each cut removal increases the number of connected components by at least one, this phase involves at most $n-1$ cuts. Thus, the total number of edges collected is $|E_{\text{small}}| \leq (n-1)Z = O(nZ)$.

2. **Sparsify the Core:** Each remaining component G_i is now guaranteed to have a minimum cut size greater than Z. We can therefore effectively apply Karger's sampling technique from the previous section to each of these components. We form a graph H_{core} by including each edge from the components with probability $p = \frac{C \log n}{\epsilon^2 Z}$ for a sufficiently large constant C. If an edge is selected, it is given a weight of $1/p$. The expected number of edges in H_{core} is $O\left(\frac{m \log n}{\epsilon^2 Z}\right)$.

The resulting sparsifier H is the union of the unweighted edges from E_{small} and the weighted edges from H_{core}. The total number of edges in H is:

$$|E(H)| = |E_{\text{small}}| + |E(H_{\text{core}})| = O\left(nZ + \frac{m \log n}{\epsilon^2 Z}\right)$$

To minimize this quantity, we balance the two terms by setting $nZ \approx \frac{m \log n}{\epsilon^2 Z}$, which gives an optimal choice of $Z = \sqrt{\frac{m \log n}{n\epsilon^2}}$. This yields a sparsifier with $O\left(\frac{\sqrt{mn \log n}}{\epsilon}\right)$ edges.

A Recursive Algorithm. While an improvement for sparse graphs, the $O(\sqrt{mn})$ dependency is not ideal, especially for dense graphs. The key insight is to apply this idea recursively. A direct recursion is problematic: the procedure above takes an unweighted graph and produces a weighted one, which the procedure is not equipped to handle as input.

A more sophisticated recursive strategy is needed:

1. Given a graph G, decompose it into two subgraphs: A, containing edges from small cuts, and B, the "core" with large cuts.
2. **Recurse on A:** Generate a sparse approximation $\hat{A}$ by calling the sparsification algorithm recursively on A.
3. **Sample and Recurse on B:** Subsample the edges of B to create a smaller (but still dense in terms of cut properties) multigraph B'. Then, recursively call the sparsification algorithm on B' to get $\hat{B}'$.
4. **Combine:** The final sparsifier $\hat{H}$ is formed by the union of $\hat{A}$ and $\hat{B}'$. The edge weights in $\hat{B}'$ must be scaled up to account for the sampling probability.

By carefully setting the error parameters at each level of the recursion, the cumulative error remains bounded by $(1 \pm \epsilon)$. A full analysis of this recursive method shows that it produces a sparsifier with $O(n \cdot \text{polylog}(n)/\epsilon^2)$ edges, closely approaching the optimal bound and matching the Benczur-Karger result up to logarithmic factors.

4.2 Power of Two Choices

The classic "balls and bins" problem considers the process of throwing n balls into n bins, where each ball selects a bin uniformly and independently at random. In this scenario, it is well-known that the maximum number of balls in any single bin, known as the maximum load, is approximately $\frac{\log n}{\log \log n}$ with high probability. While a simple round-robin allocation (placing ball i into bin i) would achieve a perfect maximum load of 1, such a strategy requires centralized state and coordination, making it unsuitable for dynamic or distributed systems where requests (balls) may arrive and depart unpredictably.

A remarkably effective alternative was proposed by Azar, Broder, Karlin, and Upfal in 1994, known as the "Power of Two Choices." The algorithm is simple: for each ball, we select two bins independently and uniformly at random, and then place the ball in the bin that is currently less loaded. This minor change has a dramatic effect on the maximum load.

Theorem 4.2. *When placing n balls into n bins using the Power of d Choices (i.e., probing d random bins and choosing the least loaded), the maximum load is*

$$\frac{\ln \ln n}{\ln d} + O(1)$$

with high probability. For the standard case of $d = 2$, this simplifies to $\ln \ln n + O(1)$.

Notably, these results are essentially optimal. For any "local" and "symmetric" load balancing algorithm, these bounds are tight. While asymmetric algorithms can achieve slightly better constants, the double-logarithmic dependency remains a fundamental characteristic of this process.

4.2.1 Intuition for the Main Result

To understand why this exponential improvement occurs, we can develop an intuition by analyzing the heights of bins and balls.

- A bin is said to have *height h* if its final load is at least h.
- A ball is said to have *height h* if, at the moment it was placed, its destination bin already contained $h - 1$ balls.

From these definitions, a few key facts emerge. A ball can only attain height h if both of its random probes land in bins that already have a height of at least $h-1$. This is because if either choice were a bin with fewer than $h-1$ balls, the algorithm would have placed the ball there. Consequently, if a bin reaches height h, it must contain at least one ball of height h.

Let f_h denote the fraction of bins with height at least h. The probability that a new ball has height h is the probability that both of its probes land in bins of height at least $h-1$. This probability is at most $(f_{h-1})^2$ and for the purpose of the informal arguments here let's say:

$$\Pr[\text{ball has height } h] \approx (f_{h-1})^2$$

The expected number of balls of height h is then $n \cdot (f_{h-1})^2$. Since for every bin of height h there must be at least one ball of height h, the number of bins of height h is at most the number of balls of height h. This gives us an approximate relationship for the fractions:

$$\mathbb{E}[f_h] \leq (f_{h-1})^2$$

This recurrence relation leads to a rapid decay. If we start with the observation that $f_2 \leq 1/2$ (at most half the bins can have load 2 or more, otherwise we'd have more than n balls), we can see the trend:

$$\begin{aligned}
\mathbb{E}[f_3] &\leq (f_2)^2 \leq (1/2)^2 = 1/4 \\
\mathbb{E}[f_4] &\leq (f_3)^2 \leq (1/4)^2 = 1/16 \\
&\vdots \\
\mathbb{E}[f_h] &\leq 2^{-2^{h-2}}
\end{aligned}$$

This doubly exponential decay implies that the expected fraction of bins with a high load becomes vanishingly small very quickly. For a height of $h \approx \log_2 \log_2 n + O(1)$, the expected number of bins with that height becomes less than 1. This suggests that the maximum load is unlikely to exceed this value.

However, this argument is heuristic and relies on expectations. A formal proof requires more rigorous concentration bounds, such as Chernoff bounds, applied carefully in an inductive argument over the heights of the bins.

4.2.2 *Power of Two Choices: A Random Graphs Proof*

Another way to show that the maximum load is $O(\log \log n)$—note that the constant is worse—is to use an first-priciples analysis based on properties of random graphs. We build a random graph G as follows: the n vertices of G correspond to the n bins, and the edges

correspond to balls—each time we probe two bins we connect them with an edge in G. For technical reasons, we'll just consider what happens if we throw fewer balls (only $m = n/C$ balls) into n bins—also, let's imagine that each ball chooses two distinct bins each time.

The constants can be optimized a bit further, but I am keeping it simple here.

Theorem 4.3. *Let $C \geq 2e^2$. If we throw n/C balls into n bins using the best-of-two-bins method, the maximum load of any bin is $O(\log \log n)$ whp.*

Hence for n balls and n bins, the maximum load should be at most C times as much, whp. (It's as though after every n/C balls, we forget about the current loads and zero out our counters—not zeroing out these counters can only give us a more evenly balanced allocation; I'll try to put in a formal proof later.)

To prove the theorem, we need two results about the random graph G obtained by throwing in n/C random edges into n vertices. Both the proofs are simple but surprisingly effective counting arguments, they appear at the end.

Lemma 4.4. *The size of G's largest connected component is $O(\log n)$ whp.*

Lemma 4.5. *For all subsets S of the vertex set, the induced graph $G[S]$ contains at most $3|S|$ edges, and hence has average degree at most 6, whp.*

Given the graph G, suppose we repeatedly perform the following operation in rounds:

> In each round, remove all vertices of degree ≤ 12 in the current graph.

We stop when there are no more vertices of small degree.

Lemma 4.6. *This process ends after $O(\log \log n)$ rounds whp.*

Proof. Condition on the events in the two previous lemmas. Any component C in the current graph has average degree at most 5; by Markov at least half the vertices have degree at most 12 and will be removed, and we halve the component size. But the size of each component was $O(\log n)$ to begin, so this takes $O(\log \log n)$ rounds. □

Lemma 4.7. *If a node/bin survives i rounds before it is deleted, its load due to edges that have already been deleted is at most $12i$. If a node/bin is never deleted, its load is at most $12i^*$, where i^* is the total number of rounds.*

Proof. Consider the nodes removed in round 1: their degree was at most 12, so even if all those balls went to such nodes, their final load would be at most 12. Now, consider any node x that survived this round. While many edges incident to it might have been removed in this round, we claim that at most 12 of those would have contributed

to x's load. Indeed, the each of the other endpoints of those edges went to bins with final load at most 12. So at most 12 of them would choose x as their less loaded bin before it is better for them to go elsewhere.

Now, suppose y is deleted in round 2: then again its load can be at most 24: twelve because it survived the previous round, and 12 from its own degree in this round. OTOH, if y survives, then consider all the edges incident to y that were deleted in previous rounds. Each of them went to nodes that were deleted in rounds 1 or 2, and hence had maximum load at most 24. Thus at most 24 of these edges could contribute to y's load before it was better for them to go to the other endpoint. The same inductive argument holds for any round $i \leq i^*$. Finally, the process ends when each component is a singleton, and hence there are no more balls to assign. □

By Lemma 4.6, the number of rounds is $i^* = O(\log\log n)$ whp, so by Lemma 4.7 the maximum load is also $O(\log\log n)$ whp.

4.2.3 *Missing Proofs of Lemmas*

Lemma 4.8. *The size of G's largest connected component is $O(\log n)$ whp.*

This fact actually holds for a random graph with n nodes and any $m < \frac{1}{2}n$ edges; see the Frieze and Karoński book.

Proof. We have a graph with n vertices and $m = n/C$ edges where we connect vertices at random. If there is a component of at least k vertices, there must be a spanning tree with at least $k-1$ edges, and hence some subset S of k vertices must have had some $k-1$ edges fall into it. For any of the $\binom{n}{k}$ choices of S, and $\binom{m}{k-1}$ choices of the edge set, the probability of these edges choosing both endpoints in that set is $[(k/n)^2]^{k-1}$. (For $k \ll m$, we can upper bound $\binom{m}{k-1}$ by $\binom{m}{k}$—this simplifies some calculations.) Now a union bound says that the "bad event" of some component of size k is at most

$$\binom{n}{k} \cdot \binom{m}{k} \cdot \left(\frac{k}{n}\right)^{2(k-1)} \leq n^2 \cdot \left(\frac{e^2}{C}\right)^k \leq 1/\operatorname{poly}(n),$$

for any $k = c\log n$ with a large enough constant c; here we used the standard approximation that $\binom{n}{k} \leq (\frac{ne}{k})^k$, and also our assumption that $C \geq 2e^2$. □

Lemma 4.9. *For all subsets S of the vertex set, the induced graph $G[S]$ contains at most $3|S|$ edges, and hence has average degree at most 6, whp.*

Proof. This proof is very similar in spirit to the one above, with a couple more steps. Recall that we have $m = n/C$ edges and n nodes. The probability that some set S of size k gets some $3k$ edges falling

into it is again

$$\binom{n}{k} \cdot \binom{m}{3k} \cdot \left(\frac{k}{n}\right)^{6k} \leq \left(\frac{n\,e}{k}\right)^{k} \cdot \left(\frac{n\,e}{3Ck}\right)^{3k} \cdot \left(\frac{k}{n}\right)^{6k} = \left(\frac{e^4}{(3C)^3} \cdot \frac{k^2}{n^2}\right)^{k}.$$

Again, we used the approximations for the binomial, and that C is large enough. Now a union bound over all sizes $k \geq 2$ completes the argument. □

Bibliographic Notes: The original Balanced Allocations paper of Azar, Broder, Karlin, and Upfal uses a delicate layered induction argument via Chernoff bounds. The random graph analysis is in the paper Efficient PRAM Simulation on a Distributed Memory Machine by Karp, Luby, and Meyer auf der Heide; I learned it from Satish Rao; here are his notes.

One can get slightly better constaints than the naïve power-of-two-choices using using the brilliant ***Always-go-left*** algorithm due to How Asymmetry Helps Load Balancing by Berthold Vöcking. Here's a survey on the various proof techniques by Mitzenmacher, Sitaraman and Richa.