CSC 5450 Randomness and Computation

Week 3 : Chernoff Bounds

Plan ① Chernoff bounds, proofs

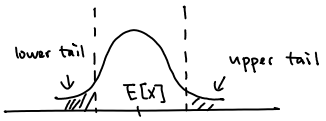
② Applications : packet routing, congestion minimization, graph sparsification

---

Reminder : Please complete the course questionnaire (in the homework page of the course homepage)

Those who take this course are required to do it, and can vote for the special topics.

Tail Inequalities [MU 4]



The general question is to bound $Pr(X > (1+\varepsilon)E[X])$ (upper tail) and $Pr(X < (1-\varepsilon)E[X])$ (lower tail).

We consider the situation when X is the sum of many independent random variables.

The law of large number asserts that the sum of n independent identically distributed variables is approximately $n\mu$, where μ is a typical mean.

The central limit theorem says that $\frac{X - n\mu}{\sqrt{n\sigma^2}} \to N(0,1)$. the deviations from $n\mu$ are typically of the order $\sqrt{n}$.

Chernoff bounds give us quantitative estimates of the probabilities that X is far from $E[X]$ for any (large enough) value of n.

Consider a simple setting where there are n coin flips, each is head with probability p.

The expected number of heads is np.

To bound the upper tail, in principle we just need to compute $Pr(X \geq k) = \sum_{i \geq k} \binom{n}{i} p^i (1-p)^{n-i}$, and show that it is very small when k is much larger than np (say $k \geq (1+\varepsilon)np$), but this sum is not easy to work with and this method is not easy to be generalized.

Instead, we extend the approach of using Markov's inequality. The Markov's inequality is often too weak, but recall in the proof of Chebyshev's inequality we can strengthen it if we know the second moment of X.

To extend this, one can use the fourth moment or any $2k$-th moment to get

$$Pr(|X - E[X]| > a) = Pr((X - E[X])^{2k} > a^{2k}) \leq E[(X - E[X])^{2k}] / a^{2k}.$$

The idea in proving the Chernoff bounds is to consider :

The idea in proving the Chernoff bounds is to consider:

$$\Pr(X \geq a) = \Pr(e^{tX} \geq e^{ta}) \leq E[e^{tX}] / e^{ta} \quad \text{for any } t > 0.$$

There are at least two reasons that we consider e^{tX}:

- Let $M_X(t) = E[e^{tX}] = E[\sum_{i \geq 0} \frac{t^i}{i!} X^i] = \sum_{i \geq 0} \frac{t^i}{i!} E[X^i]$. If we have $M_X(t)$, to compute $E[X^i]$, we can just compute $M_X^{(k)}(0)$, where $M_X^{(k)}(0)$ is the k-th derivative of $M_X(t)$ evaluated at $t=0$.
  So, $M_X(t)$ contains all the moments information, and is called the moment generating function.
  It gives a strong bound when applying Markov's inequality, as the denominator is exponentially large.
- If $X = X_1 + X_2$ and X_1, X_2 are independent, then $E[e^{tX}] = E[e^{tX_1} e^{tX_2}] = E[e^{tX_1}] E[e^{tX_2}]$.
  So, this function is easy to compute when X is the sum of independent random variables.

Chernoff Bounds [MU 4.2]

Roughly speaking, Chernoff-type bounds are the bounds obtained by $\Pr(X \geq a) \leq E[e^{tX}] / e^{ta}$.

Let us consider a useful case when X is the sum of independent heterogenous coin flips.

Heterogenous coin flips:

Let $X_1, \ldots, X_n$ be independent random variables with $X_i = 1$ with probability p_i and $X_i = 0$ otherwise.

Let $X = \sum_{i=1}^n X_i$. Let $\mu = E[X] = \sum_{i=1}^n E[X_i] = \sum_{i=1}^n p_i$.

Then $E[e^{tX}] = E[e^{tX_1} e^{tX_2} \cdots e^{tX_n}] = \prod_{i=1}^n E[e^{tX_i}]$ by independence

$$= \prod_{i=1}^n (p_i e^{t \cdot 1} + (1-p_i) e^{t \cdot 0}) = \prod_{i=1}^n (1 + p_i(e^t - 1)) \leq \prod_{i=1}^n e^{p_i(e^t-1)} = e^{\mu(e^t-1)}.$$

We put in some specific parameters to get some useful bounds.

Theorem In the heterogenous coin flipping setting, we have:

① for $\delta > 0$, $\Pr(X \geq (1+\delta)\mu) < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$.

② for $0 < \delta < 1$, $\Pr(X \geq (1+\delta)\mu) < e^{-\delta^2 \mu / 3}$.

③ for $R \geq 6\mu$, $\Pr(X \geq R) \leq 2^{-R}$.

proof ① $\Pr(X \geq (1+\delta)\mu) \leq E[e^{tX}] / e^{t(1+\delta)\mu} \leq e^{\mu(e^t-1)} / e^{t(1+\delta)\mu}$

By elementary calculus, we find out that this term is minimized when $t = \ln(1+\delta)$, and this implies that $\Pr(X \geq (1+\delta)\mu) \leq e^{\mu\delta} / (1+\delta)^{(1+\delta)\mu}$, proving ①.

② When $0 < \delta < 1$, it holds that $e^\delta / (1+\delta)^{1+\delta} \leq e^{-\delta^2/3}$.

This can be verified by taking log of both sides and letting $f(\delta) = \delta - (1+\delta)\ln(1+\delta) + \frac{\delta^2}{3}$,

and show that $f'(\delta) \le 0$ in the interval $[0,1]$, and thus $f(\delta) \le 0$ in this interval since $f(0) = 0$, and this implies the claim. (see MU Theorem 4.4 for details.)

③ Let $R = (1+\delta)\mu$. When $R \ge 6\mu$, we have $\delta \ge 5$.

Hence, $\Pr(X \ge (1+\delta)\mu) \le \left(e^{\delta}/(1+\delta)^{1+\delta}\right)^{\mu} \le \left(e/(1+\delta)\right)^{(1+\delta)\mu} \le (e/6)^{R} \le 2^{-R}$. □

Similar bounds hold for the lower tail: very similar proof (by setting $t < 0$). (see MU Thm 4.5)

Theorem In the heterogenous coin flipping setting, we have for $0 < \delta < 1$

① $\Pr(X \le (1-\delta)\mu) \le \left(e^{-\delta}/(1-\delta)^{1-\delta}\right)^{\mu}$

② $\Pr(X \le (1-\delta)\mu) \le e^{-\mu\delta^2/2}$.

Corollary In the heterogenous coin flipping setting, $\Pr(|X-\mu| \ge \delta\mu) \le 2e^{-\mu\delta^2/3}$ for $0 < \delta < 1$.

Hoeffding extension The same bounds hold when each X_i is a random variable taking values in $[0,1]$ with mean p_i. This is because the function e^{tx} is convex, and thus it always lies below the straight line joining the endpoints $(0,1)$ and $(1,e^t)$. This line has the equation $y = \alpha x + \beta$ for $\alpha = e^t - 1$ and $\beta = 1$. Therefore, $E[e^{tX_i}] \le E[\alpha X_i + \beta] = p_i(\alpha+\beta) + (1-p_i) = 1 + p_i(e^t - 1)$, and the same calculations as above follow.

Remarks:

- The same method holds for other random variables, e.g. Poisson r.v., Gaussian r.v., etc.
- It is often an easier way to compute the moments by computing the moment generating functions.

Basic Examples:

① Coin Flips: Consider n independent fair coin flips, so $\mu = n/2$.

$\Pr(|\#\text{heads} - \mu| \ge \delta\mu) \le 2e^{-\delta^2\mu/3} = 2e^{-\frac{\delta^2 n}{6}}$.

So, by setting $\delta = \sqrt{6\ln n / n}$, this probability is at most $2/n$.

Therefore, we conclude that $\Pr\left(|\#\text{heads} - \frac{n}{2}| \ge \frac{1}{2}\sqrt{6n\cdot\ln n}\right) \le \frac{2}{n}$.

So, with high probability, the number of heads is within $O(\sqrt{n})$ of the expected value, and this $\sqrt{n}$ term is something to remember, as it comes up in different places.

Recall that Markov's inequality implies that $\Pr(\#\text{heads} \ge \frac{3n}{4}) \le \frac{2}{3}$. Chebyshev's inequality implies that $\Pr(\#\text{heads} \ge \frac{3n}{4}) \le \frac{4}{n}$.

Chernoff's bound implies that $\Pr(\#\text{ heads} \geq \frac{3n}{4}) \leq e^{-(\frac{n}{2})(\frac{1}{4})^2/3} = e^{-n/96}$, exponentially small.

② Probability amplification:

Recall that the success probability of a randomized algorithm with one-sided error can be amplified easily: say the algorithm is always correct when it says NO and is correct with prob p when it says YES. To decrease the failure probability, we just repeat the algorithm k times or until it says NO, then the failure probability is at most $(1-p)^k$ when it says YES k times for a NO instance. For constant p, repeating $\log n$ times will decrease the failure probability to $O(1/n)$.

Suppose the randomized algorithm is two-sided error, say it has 60% of giving the correct answer, but it could make mistakes when it says YES or NO. To decrease the failure probability, we run the algorithm for k times and output the majority answer. Say the instance is a YES instance. The majority answer is wrong when the randomized algorithm outputs NO for more than k/2 times. But the expected number of answering NO is equal to 0.4k by our assumption. So, by chernoff bound, the majority answer is wrong is

$$\Pr(\#\text{NO} > (1+\tfrac{1}{4})\,E[\#\text{NO}]) \leq e^{-\mu\delta^2/3} = e^{-0.4k(1/4)^2/3} = e^{-k/120}.$$

Therefore, by repeating $k = O(\log n)$ times, the failure probability is at most $O(1/n)$.

This is of the same order as in the case of one-sided error.

This $O(\log n)$ term is another quantity to remember, and it will also come up in different places, e.g. in power of two choices in last week.

③ Quicksort: We can use Chernoff bound to show that the probability that the randomized quicksort will take say $50n\log n$ is at most $1/n^2$.

The following is a sketch. See L3 of 2011 for more details.

We would like to bound the probability that the recursion tree is of height at least $c\log n$.

Call a node in the recursion tree is "good" if the pivot element splits the problem into $(1/3, 2/3)$.

Note that there can be at most $\log_{3/2} n < 2\log n$ good nodes in a root-to-leaf path.

If the height of the tree is at least $c\log n$, then there is a root-to-leaf path with $\geq (c-2)\log n$ bad nodes.

Note that the probability that a node is bad is 2/3, and thus the expected number of

good nodes in such a path is $c\log n/3$.

By Chernoff bound, if c is large enough, then the probability is at most $e^{-\mu\delta^2/3}$.

Now, since $\mu = \Theta(\log n)$ and δ is a constant, this probability is $O(1/n^k)$ for some $k>1$.

Packet Routing [MU 4.5, MR 4.2]

General question: Design a sparse network with efficient communication time.

Communication network: An undirected graph, each vertex is a routing switch, each edge is a channel.

Synchronous model: At each time unit, an edge can carry at most one packet, and a packet can traverse at most one edge.

Routing switch: - can store packets waiting to be transmitted
- use first-in-first-out (FIFO) policy

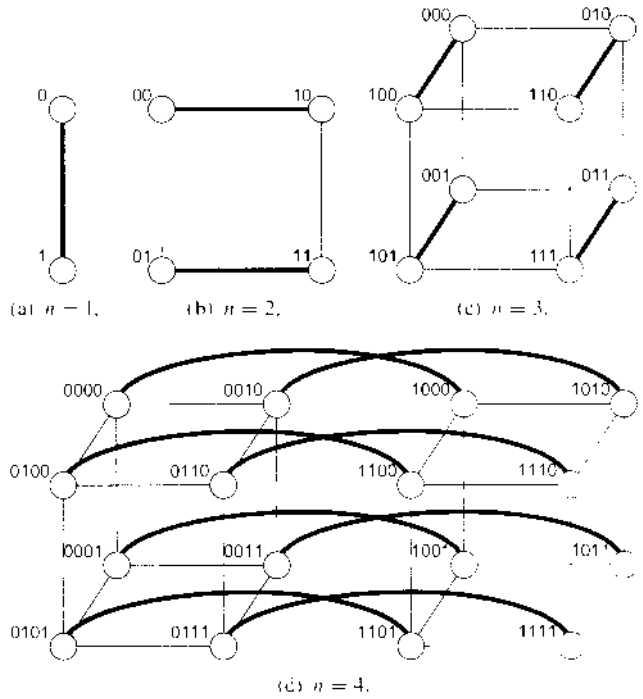
Permutation routing: - each node sends exactly one packet
- each node receives exactly one packet

Goal: Design ① a sparse network, and
② a routing scheme,
such that any permutation routing can be finished in a small number of steps.

Remark: - in a complete graph, any permutation routing can be done in one step, but the problem is that it requires a quadratic number of edges (i.e. expensive to build).
- given a permutation, it may be possible to compute an optimal routing scheme. But even if it can be done, it is not desirable in the distributed setting to coordinate the communication network to achieve it. It would be much better if there is a simple routing scheme that can be implemented without communications and yet efficient.

Next, we present the two components: the network and the routing scheme.

Hypercube: An undirected graph with $N=2^n$ nodes, each node corresponds to a distinct n-bit string, and two nodes have an edge iff their corresponding n-bit strings differed by exactly one bit.



picture from MU

Figure 4.1: Hypercubes of dimensions 1, 2, 3, and 4.

Routing Algorithm: We use the simple bit-fixing algorithm. If a node $(a_1, a_2, \ldots, a_n)$ sends a packet to another node $(b_1, b_2, \ldots, b_n)$, then the bit-fixing algorithm sends the packet using the path $(a_1, a_2, \ldots, a_n) \Rightarrow (b_1, a_2, \ldots, a_n) \Rightarrow (b_1, b_2, \ldots, a_n) \Rightarrow \ldots \Rightarrow (b_1, b_2, \ldots, b_n)$, e.g. if we send a packet from $(1,1,1,0)$ to $(0,1,0,1)$, then we use the path $(1,1,1,0) \Rightarrow (0,1,1,0) \Rightarrow (0,1,0,0) \Rightarrow (0,1,0,1)$. Note that this can be easily implemented in the routing switch $(a_1, a_2, \ldots, a_n)$: given a destination $(b_1, b_2, \ldots, b_n)$, just sends the packet along the edge corresponding to their first bit differed.

First attempt: Just use the bit-fixing algorithm to send the packets for each pair, but there exists a permutation for which the algorithm would take a long time to finish. (HW?)

Second attempt: For each pair, choose a random ordering of the bits independently, then use the bit-fixing algorithm in that ordering to route the packet (in other words, for each pair, the bits are fixed in a random order). Unfortunately, there still exists a permutation for which the algorithm takes a long time to finish. (Harder, HW?)

Roughly speaking, for the above algorithms, there exist some permutations for which a small part of the network is used heavily.

On the other hand, if the permutation is random, then one can show that the (deterministic) bit-fixing algorithm would work with high probability, but we want a routing algorithm that works for any permutation. Interestingly, we can adapt this idea to work for any permutation.

Third attempt: For each source, first sends the packet to a random intermediate node, and then send from this intermediate node to the destination.

Theorem: For any permutation, this "two-phase" routing algorithm routes all packets to their destinations in $O(n) = O(\log N)$ steps, with probability at least $1 - O(\frac{1}{N})$.

proof: The proof consists of three steps.

For each pair, there is a unique bit-fixing path connecting them. We will prove:

① each such bit-fixing path is used by at most $6n$ packets with very high probability;

② by a union bound, all N bit-fixing paths are used by at most $6n$ packets with high probability;

③ then a deterministic argument would conclude that the routing algorithm finishes in $O(n)$ steps.

① Consider a path $P = (v_0, v_1, \dots, v_m)$ where $e_i = (v_{i-1}, v_i)$ and $m \le n$.

Call a packet active at e_i if it will use e_i.

Let $v_{i-1} = (b_1, \dots, b_{j-1}, a_j, a_{j+1}, \dots, a_n)$ and $v_i = (b_1, \dots, b_{j-1}, b_j, a_{j+1}, \dots, a_n)$

Note that an active packet at e_i will use e_i to fix its j-th bit.

Therefore, ⓐ this packet must come from source with address $(*, *\dots, *, a_j, a_{j+1}, \dots a_n)$, since bit j to bit n have not been fixed yet.

ⓑ this packet must go to an intermediate node with address $(b_1, b_2, \dots, b_{j-1}, *, \dots *)$ since the first $j-1$ bits have already been fixed.

So, there are at most 2^{j-1} packet using this edge e_i (because there are at most 2^{j-1} possible sources), and each packet will use e_i with probability $2^{-(j-1)}$ (because the random intermediate node must have the first j bits as $b_1, \dots b_{j-1}$, which happens with probability $2^{-(j-1)}$).

Therefore, $E[\#\text{ active packets at } e_i] \le 2^{(j-1)} \cdot 2^{-(j-1)} = 1$.

Hence, $E[\#\text{ active packets at } P] \le m \cdot 1 \le n$.

Since total # of active packets at P is a sum of 0-1 independent variables (although the probability of each packet may not be the same), we can apply Chernoff bound ($\Pr(X \ge R) \le 2^{-R}$ for $R \ge 6\mu$) to establish that

$$\Pr(\#\text{ of active packets at } P > 6n) \le 2^{-6n}.$$

② Since there are at most $2^n \cdot 2^n$ pairs and thus 2^{2n} paths to consider,

$\Pr(\text{some path with } > 6n \text{ active packets}) \le 2^{2n} \cdot 2^{-6n} = 2^{-4n}$

③ Finally, we conclude with a deterministic argument that each packet takes at most $7n$ steps to reach their intermediate node.

Observation: if a packet leaves a bit-fixing path, then it will not come back again.

Proof: say the packet leaves the path in i-th step, then it will not come back because the i-th is different.

Now, consider a packet, it got "delay" only because a new packet joins its path (convince yourself).

Since a packet can join its path at most once (as it will not come back) and there are at most $6n$ active packets on a path, it got delayed at most $6n$ steps.

Since a path is of length at most n, it takes at most $7n$ steps to reach its intermediate node.

By the same argument, it takes at most $7n$ steps from the intermediate nodes to the destinations.

Therefore, with probability at least $1 - 2^{-4n}$, the algorithm finishes in $O(n)$ steps.

Congestion minimization (randomized rounding)

Input: A graph G, n source-sink pairs (s_i, t_i).

Task: Find a path P_i for each pair (s_i, t_i) s.t. each edge is used at most C times.

Objective: Minimize C

This is somewhat similar to the above problem, but the graph is given and the objective is slightly different.

This problem is NP-hard, but we will design an approximation algorithm for it.

First formulate this problem as an integer linear program.

Let $\mathcal{P}_i$ be the set of all s_i-t_i paths in G.

Let f_p^i be a variable indicating whether we pick $P \in \mathcal{P}_i$ or not.

Then the problem can be formulated as follows:

minimize C

s.t. $\sum_{\mathcal{P}_i} f_p^i = 1 \quad \forall i$ (exactly one path from s_i to t_i is chosen)

$$\sum_i \sum_{P \in \mathcal{P}_i \text{ s.t. } P \ni e} f_P^i \le C \quad \forall e \quad \text{(congestion on } e \text{ is at most } C\text{)}$$

$$f_P^i = \{0,1\} \quad \forall i, P$$

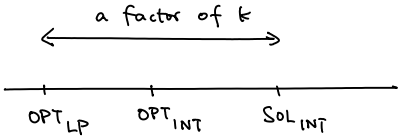
Solving this integer program is NP-complete, instead we "relax" the constraint

$$f_P^i = \{0,1\} \quad \text{to} \quad 0 \le f_P^i \le 1.$$

This is a linear program and can be solved in polynomial time (although this linear program has exponentially many variables, it can still be solved in polynomial time, one way is to formulate an equivalent LP with polynomially many variables)

However, this linear program allows some "fractional solutions", which do not correspond to the solutions we wanted (s_i-t_i path).

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 $k \cdot OPT_{LP}$, then it is a k-approximate solution to the original problem.



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

How to interpret the fractional solutions?

Idea: Think of f_P^i as the probability of picking path P for s_i-t_i

Randomized rounding: For each i, pick P with probability f_P^i.

Theorem: The congestion on each edge is at most $O(C \ln n / \ln\ln n)$ with probability $\ge 1 - 1/n$

Proof: Let $X_i^e = \begin{cases} 1 & \text{if pair } i \text{ uses } e \\ 0 & \text{otherwise} \end{cases}$

Let $X^e = \sum_i X_i^e$

$$E[X_i^e] = \sum_{P \in \mathcal{P}_i \text{ s.t. } P \ni e} f_P^i$$

$$E[X^e] = \sum_i \sum_{P \in \mathcal{P}_i \text{ s.t. } P \ni e} f_P^i \le C \quad \text{(by the constraint in the LP)}$$

Since each X_i^e is an 0-1 independent random variable,

by Chernoff bound, $Pr(X^e > (1+\delta)C) < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^C \le \frac{e^\delta}{(1+\delta)^{1+\delta}}$ as $C \ge 1$

Set $\delta = \Theta(\ln n / \ln\ln n)$. then $\ln(1+\delta)^{1+\delta} = (1+\delta)\ln(1+\delta) = \Omega(\ln n)$.

Therefore, $(1+\delta)^{1+\delta} > n^4$, and thus with prob. $\leq 1/n^3$, an edge has congestion $\Omega(c \ln n/\ln\ln n)$.

By a union bound, as there are at most n^2 edges in the graph, every edge has congestion $O(c \ln n / \ln\ln n)$ with probability at least $1 - 1/n$. □

Graph sparsification

General question: Find a sparse graph to approximate a dense graph

Cut-sparsifier: Given an undirected graph G, find an H with few edges s.t. the weight of each cut in H is roughly the same as the weight of the corresponding cut in G.

More precisely, let $w_G(e)$ and $w_H(e)$ be the weight of edge e in G and H, let $w_G(\delta(S))$ and $w_H(\delta(S))$ be the total weight of the edges in $\delta(S)$ in G and H. Then H is an $(1\pm\varepsilon)$-approximation of G if for every cut $\delta(S)$, we have $(1-\varepsilon) w_G(\delta(S)) \leq w_H(\delta(S)) \leq (1+\varepsilon) w_G(\delta(S))$.

Additional assumption: the min-cut in G is $\Omega(\log n)$.

Constructing sparsifier: Let $p = \frac{3(d+2)\ln n}{\varepsilon^2 c}$ where c is the min-cut value.

For each edge $e \in G$, put e in H with weight $1/p$ with probability p.

Theorem: H is an $(1\pm\varepsilon)$-approximation of G with prob. $\geq 1 - O(n^{-d})$

Proof: Consider a cut $\delta(S)$ with k edges.

$E(\#\text{ edges of } \delta(S) \text{ in } H) = kp$

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

$\Pr(|\#\text{ edges of } \delta(S) \text{ in } H - kp| > \varepsilon kp) \leq 2e^{-(kp)\varepsilon^2/3}$

Since $k \geq c$, this is at most $2n^{-(d+2)}$

However, there are exponentially many cuts in G, and thus we could not apply a union bound to finish the proof.

Important observation : There are at most $n^{2\alpha}$ cuts with αc edges.

Proof: when $\alpha = 1$, we proved this in week 1 using the randomized contraction algorithm. The proof for $\alpha > 1$ is the same. □

Now, $\bigcup_{S \subseteq V} \Pr\left(|\#\text{ edges of } \delta(S) \text{ in } H - |\delta(S)| \cdot p| > \varepsilon |\delta(S)| \cdot p \right)$

$$\leq \sum_{\alpha \geq 1} \sum_{S \subseteq V,\ |\delta(S)| \leq \alpha c} \Pr\left(|\#\text{ edges of } \delta(S) \text{ in } H - |\delta(S)| \cdot p| > \varepsilon |\delta(S)| \cdot p \right)$$

$$\leq \sum_{\alpha \geq 1} n^{2\alpha} \cdot 2e^{-p\alpha c \varepsilon^2/3} \quad (\text{assuming } n^{2\alpha} \text{ cuts with } \alpha c \text{ edges})$$

Since $p = \frac{3(d+2)\ln n}{\varepsilon^2 c}$, this is at most $\sum n^{2\alpha} \cdot 2^{-\alpha(d+2)\ln n} = \sum_{\alpha \geq 1} n^{-\alpha d}$

This is at most $O(n^{-d})$.

By putting a weight of $1/p$ on each edge, H is an $(1 \pm \varepsilon)$-approximation of G. □

Applications: ① fast approximate s-t cut

② fast approximate s-t flow

Since the applications are mostly based on deterministic arguments and also some technical algorithmic details, we just try to give the high-level idea.

① min s-t cut : fix ε (say 0.01), construct H, with high probability H has $\tilde{O}(m/c)$ edges (by Chernoff bound). Find a min s-t cut in H (roughly c times faster than G). Return this as an approximate min s-t cut in G.

② max s-t flow : randomly color each edge by $1/p$ colors. With high probability each color subgraph has $O(pm)$ edges and the s-t max-flow in each color subgraph is $(1-\varepsilon)p \cdot OPT$ where OPT is the max-s-t-flow value in G. (Each color subgraph has the same distribution as sampling with probability p.) Find a s-t max-flow in each color subgraph, and combine them to give a $(1-\varepsilon)$-approximation to s-t max-flow in G.

Significant improvement: [Benczur, Karger]

Significant improvement: [Benczur, Karger]

For any undirected graph, there is an $(1\pm\varepsilon)$-approximation with $O(n \log n)$ edges (without any assumption about min-cut value)

Furthermore, the sparsifier can be constructed in $\tilde{O}(n)$ time.

Thus, if we don't mind $(1+\varepsilon)$-approximation, we can replace m by n.

Idea: sample edges with probability proposional to their "connectivities".

(Potential topic later, see L11 in 2011)
