Polling

Outline

- Set has size u, contains n "special" elements
- goal: count number of special elements
- sample with probability $p = c(\log n)/\epsilon^2 n$
- with high probability, $(1 \pm \epsilon)np$ special elements
- if observe k elements, deduce $n \in (1 \pm \epsilon)k$.
- Problem: what is p?

Related idea: Monte Carlo simulation

- Probability space, event A
- easy to test for A
- goal: estimate $p = \Pr[A]$.
- Perform n trials (sampling with replacement).
  - expected outcome pn.
  - *estimator* $\frac{1}{n}\sum I_i$
  - prob outside $\epsilon < \exp(-\epsilon^2 np/3)$ ($\epsilon < 1$)
  - for prob. δ, need

$$n = O\left(\frac{\log 1/\delta}{\epsilon^2 p}\right)$$

- what if p unknown?
- What if p is small?

Handling unknown p

- Sample n times till get $\mu_{\epsilon,\delta} = O(\log \delta^{-1}/\epsilon^2)$ hits
- w.h.p, $p \in (1 \pm \epsilon)\mu_{\epsilon,\delta} n$

Transitive closure

Problem outline

- databases want size
- matrix multiply time
- compute reachibility set of each vertex, add

Sampling algorithm

- generate vertex samples until $\mu_\epsilon \delta$ reachable from v
- deduce size of $v's$ reachibility set.
- reachability test: $O(m)$.
- number of sample: n/size.
- $O(mn)$ per vertex—ouch!

Pipeline for all vertices simultaneously

- increase mean to $O(\log n/\epsilon^2)$,
- so $1/n^2$ failure
- $O(mn)$ for *all* vertices (still ouch).

Avoid wasting work

- after $O(n \log n)$ samples, every vertex has $\log n$ hits. No more needed.
- Send at most $\log n$ samples over an edge: $\tilde{O}(m)$

Minimum Cut

deterministic algorithms

- Max-flow
- Gabow

Min-cut implementation

- data structure for contractions
- alternative view—permutations.
- deterministic leaf algo

Recursion:

$$\begin{aligned} p_{k+1} &= p_k - \frac{1}{4}p_k^2 \\ q_k &= 4/p_k + 1 \\ q_{k+1} &= q_k + 1 + 1/q_k \end{aligned}$$

Minimum Cut

Min-cut

- saw RCA, $\tilde{O}(n^2)$ time
- Another candidate: Gabow's algorithm: $\tilde{O}(mc)$ time on m-edge graph with min-cut c
- nice algorithm, if m and c small. But how could we make that happen?
- Similarly, for those who know about it, augmenting paths gives $O(mv)$ for max flow. Good if m, v small. How make happen?
- Sampling! What's a good sample? (take suggestions, think about them.
- Define $G(p)$—pick each edge with probability p

Intuition:

- G has m edges, min-cut c
- $G(p)$ hss pm edges, min-cut pc
- So improve Gabow runtime by p^2 factor!

What goes wrong? (pause for discussion)

- expectation isn't enough
- so what, use chernoff?
  - min-cut has c edges
  - expect to sample $\mu = pc$ of them
  - chernoff says prob. off by ϵ is at most $2e^{-\epsilon^2\mu/4}$
  - so set $pc = 8\log n$ or so, deduce with high probability, no min-cut deviates.
- (pause for objections)
- yes, a problem: exponentially many cuts.
- so even though Chernoff gives "exponentially small" bound, accumulation of union bound means can't bound probability of small deviation over all cuts.

Surprise! It works anyway.

- Theorem: if min cut c and build $G(p)$, then "min expected cut" is $\mu = pc$. Probability any cut deviates by more than ϵ is $O(n^2 e^{-\epsilon^2\mu/3})$.
  - So, if get μ around $12(\log n)/\epsilon^2$, all cuts within ϵ of expectation with high probability.
  - Do so by setting $p = 12(\log n)/c$

- Application: min-cut approximation.
- Theorem says a min-cut will get value at most $(1+\epsilon)\mu$ whp
- Also says that any cut of original value $(1+\epsilon)c/(1-\epsilon)$ will get value at most $(1+\epsilon)\mu$
- So, sampled graph has min-cut at most $(1+\epsilon)\mu$, and whatever cut is minimum has value at most $(1+\epsilon)c/(1-\epsilon) \approx (1+2\epsilon)c$ in original graph.
- How find min-cut in sample? Gabow's algorithm
- in sample, min-cut $O((\log n)/\epsilon^2)$ whp, while number of edges is $O(m(\log n)/\epsilon^2 c)$
- So, Gabow runtime $\tilde{O}(m/\epsilon^2 c)$
- constant factor approx in near linear time.

Proof of Theorem

- Suppose min-cut c and build $G(p)$
- Lemma: bound on number of α-minimum cuts is $n^{2\alpha}$.
  - Base on contraction algorithm
- So we take as given: number of cuts of value less than αc is at most $n^{2\alpha}$ (this is true, though probably slightly stronger than what you proved. If use $O(n^{2\alpha})$, get same result but messier.
- First consider n^2 smallest cuts. All have expectation at least μ, so prob any deviates is $e^{-\epsilon^2\mu/4} = 1/n^2$ by choice of μ
- Write larger cut values in increasing order $c_1, \ldots$
- Then $c_{n^{2\alpha}} > \alpha c$
- write $k = n^{2\alpha}$, means $\alpha_k = \log k / \log n^2$
- What prob c_k deviates? $e^{-\epsilon^2 p c_k/4} = e^{-\epsilon^2 \alpha_k \mu/4}$
- By choice of μ, this is k^{-2}
- sum over $k > n^2$, get $O(1/n)$