CS294 Partition Functions **Fall 2020**

Lecture 5: September 10

Instructor: Alistair Sinclair

Disclaimer: *These notes have not been subjected to the usual scrutiny accorded to formal publications. They may be distributed outside this class only with the permission of the Instructor.*

5.1 Approximate Counting via Basic Monte Carlo

In this lecture we look at some basic approximate counting algorithms based on the classical Monte Carlo method. It is rare that simple methods like this work, and we will soon turn to more robust methods with wider applicability.

Suppose we want to estimate the size of a set Ω, which lies inside a larger set $\mathcal{U}$ (the "universe") of known size. The classical Monte Carlo method simply takes a sequence of t independent random samples from $\mathcal{U}$, letting X_i be the indicator r.v. of the event that the ith sample lies in Ω, and outputting the estimate $Z = \frac{\sum_i X_i}{t}|\mathcal{U}|$. Clearly this is an unbiased estimator of $|\Omega|$, in the sense that $\mathrm{E}(Z) = |\Omega|$. And as the sample size t increases the variance of the estimator decreases, improving the confidence of the estimate.

As we saw in the last lecture, the "Unbiased Estimator Theorem" tells us that the sample size required to ensure that

$$\Pr\big[|Z - |\Omega|| \le \varepsilon|\Omega|\big] \ge \frac{3}{4}$$

is $t = \frac{4}{\varepsilon^2}\frac{\mathrm{Var}(X_i)}{\mathrm{E}(X_i)^2} \le \frac{4}{\varepsilon^2}\frac{1}{p}$, where $p = \frac{|\Omega|}{|\mathcal{U}|}$. Hence this approach will be efficient provided $\frac{|\Omega|}{|\mathcal{U}|}$ is not too small.

Unfortunately, for almost all interesting counting problems this method fails because the fraction $\frac{|\Omega|}{|\mathcal{U}|}$ can be exponentially small. We now look at one example where a clever twist on the method actually works!

5.2 Counting satisfying assignments of a DNF formula

A boolean formula is in *Disjunctive Normal Form (DNF)* if it is a disjunction of one or more terms, each of which is a conjunction of one or more literals. For example,

$$\varphi = (x_1 \wedge x_2 \wedge \bar{x}_3) \vee (\bar{x}_2 \wedge x_5) \vee \ldots$$

Finding a satisfying assignment of a DNF formula is trivial: we need only satisfy one of the terms, so a satisfying assignment is easily obtained as long as there exists a term in which no two literals are negations of one another. If not, then the formula is not satisfiable.

However, the #DNF problem (computing the number of satisfying assignments to a DNF formula) is much harder. To see this, suppose φ is a formula in Conjunctive Normal Form (as in SAT) with n variables. Note that the formula $\neg\varphi$ can be converted in polynomial time to DNF using de Morgan's laws. Moreover, we clearly have

$$\#\mathrm{SAT}(\varphi) = 2^n - \#\mathrm{DNF}(\neg\varphi).$$

Hence, if we could evaluate #DNF in polynomial time, we would also have a polynomial time algorithm for #SAT. We have thus exhibited a polynomial time Turing reduction from #SAT to #DNF, so #DNF is $\#\mathcal{P}$-complete.

The following remarkable result, due to Karp and Luby [KL83], shows that we can get a very simple fpras for #DNF.

Theorem 5.1. [KL83] *There exists a fpras for #DNF.*

Before describing the Karp/Luby algorithm, let's first see why the naive Monte Carlo method above does not work. Let $\mathcal{U}$ be the set of all possible assignments (so $|\mathcal{U}| = 2^n$), and let $\Omega = \bigcup_i S_i$ (the set of assignments satisfying φ). Our goal is to estimate $\frac{|\Omega|}{|\mathcal{U}|}$. By the Unbiased Estimator Theorem, we would require $\sim \frac{4}{\varepsilon^2 \mu}$ trials to get an error of $1 \pm \varepsilon$ with probability $\geq \frac{3}{4}$. Recall that $\mu = \frac{|\Omega|}{|\mathcal{U}|}$, so in this case, $\mu = \frac{|\bigcup_i S_i|}{2^n}$; unfortunately this may be exponentially small, so we would require an exponentially large number of trials.

Incidentally, another natural approach to this problem is via inclusion-exclusion, using the formula

$$|\bigcup_{i=1}^{m} S_i| = \sum_{i=1}^{m} |S_i| - \sum_{i \neq j} |S_i \cap S_j| + \sum_{i \neq j \neq k} |S_i \cap S_j \cap S_k| - \dots$$

This summation contains exponentially many terms, so we cannot hope to evaluate it explicitly. We might, however, try to truncate it after a small number of terms and get an approximation. Unfortunately, one can come up with examples to show that any polynomial-length truncation of the sum fails to give a good approximation in general.

The reason that the naive Monte Carlo method fails is that the universe $\mathcal{U}$ is too large with respect to our set of satisfying assignments. The key idea of the Karp/Luby algorithm is to apply the Monte Carlo method, but with respect to a much smaller universe $\mathcal{U}$.

Proof of Theorem 5.1. Let φ be a DNF formula with n variables and m terms. Let S_i be the set of satisfying assignments for the i^{th} term. The goal is to compute the size of the union $\bigcup_i S_i$. Let $\mathcal{U} = \{(a, i) : a \text{ satisfies the } i^{th} \text{ term}\}$.

Enumerating the satisfying assignments of φ as $a_1, \dots, a_s$ (where $s = |\bigcup_i S_i|$), we can depict the elements of $\mathcal{U}$ in a chart as follows (marking satisfying pairs with $\times$ and $\otimes$):

	a_1	a_2	a_3	$\dots$	a_s
S_1	$\otimes$		$\otimes$	$\dots$	
S_2	$\times$	$\otimes$		$\dots$	
S_3			$\times$	$\dots$	$\otimes$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
S_m	$\times$			$\dots$	$\times$

For $(a, i) \in \mathcal{U}$, say (a, i) is *special* if $i = \min\{i' : (a, i') \in \mathcal{U}\}$. In the example chart above, these special pairs are marked with $\otimes$. Note that special pairs are always the topmost symbol in their column, so each a_k is in exactly one special pair. Therefore, the number of special pairs is exactly $s = |\bigcup_i S_i|$. Now, we can apply the standard Monte Carlo algorithm to the universe $\mathcal{U}$ with $\Omega = \{(a, i) \in \mathcal{U} : (a, i) \text{ is special}\}$:

for $j = 1$ to t **do**
    pick (a, i) uniformly at random from $\mathcal{U}$

$$\text{set } X_j = \begin{cases} 1 & \text{if } (a,i) \text{ is special;} \\ 0 & \text{otherwise} \end{cases}$$
$$\textbf{output } \left(\tfrac{1}{t}\sum_{j=1}^{t} X_j\right) \cdot |\mathcal{U}|$$

Note that $E(X_j) = \frac{|\Omega|}{|\mathcal{U}|} = \frac{|\bigcup_i S_i|}{|\mathcal{U}|}$, so the algorithm is an unbiased estimator for $|\bigcup_i S_i|$ as desired.

The issues left to cover in the above algorithm are:

The value of t: This is the crucial point. The total number of pairs in $\mathcal{U}$ is at most ms (the size of the above chart), and exactly s of them are special. Hence $\mu = \mathrm{E}[X_j] \geq \frac{s}{ms} = \frac{1}{m}$. Thus by the Unbiased Estimator Theorem, the number of trials required for a $(1 \pm \varepsilon)$ approximation is only

$$t = \frac{4}{\epsilon^2 \mu} \leq \frac{4m}{\epsilon^2}.$$

Computing the size of $\mathcal{U}$: Given a term i with q_i literals on distinct variables, the number of satisfying assignments $|S_i|$ is 2^{n-q_i} since q_i variables are "fixed" and the remaining variables can take two values each. Summing over all terms gives $|\mathcal{U}|$.

Picking an element of $\mathcal{U}$ uniformly at random: We pick term i with probability $\frac{|S_i|}{\sum_i |S_i|}$, and then pick a random satisfying assignment for this term. This is done by making the necessary assignments to the variables in the term and assigning (uniformly) random values to the remaining variables.

Complexity: Computing $|\mathcal{U}|$ requires $O(n)$ work for each term, which is $O(nm)$ work overall. Checking to see if a particular pair (a,i) is special requires checking for all $j < i$ if (a,j) is in $\mathcal{U}$. This requires $O(nm)$ work also. Generating a random pair (a,i) takes $O(n+m)$ work once each $|S_i|$ has been computed. So the total running time per trial is $O(nm)$, and since there are $O(m\varepsilon^{-2})$ trials, the total running time for the entire algorithm is $O(nm^2/\epsilon^2)$. Thus we have a FPRAS. □

We conclude with a few additional remarks:

1. With a slightly cleverer implementation, the running time of the above algorithm can be improved to $O(nm\frac{1}{\varepsilon^2})$; see [KLM89].
2. The algorithm can be used to compute the size of the union of any collection of finite sets S_i for which the above operations (computing the size of each S_i, and picking a random element of each S_i) can be carried out efficiently.
3. The algorithm easily generalizes to the "probabilistic DNF" problem, in which we are given a DNF formula φ together with a probability p_i for each variable X_i. Our goal is to compute the probability that φ is satisfied, assuming that each variable is set to true with probability p_i, independently of the other variables. The proof is left as an **exercise**. We shall see an interesting application of probabilistic DNF in the next section.

5.3 Network Reliability

Consider a connected undirected graph G with n vertices and m edges, where each edge has some probability p of failing. What is the probability that G becomes disconnected under random, independent edge failures? This can also be viewed as a counting problem, except that each item now has an associated weight: in particular, we want to compute the sum of the weights of all disconnected subgraphs of G, where the weight

of a subgraph with t fewer edges than G is $p^t(1-p)^{m-t}$. We can also generalize the problem to allow a different failure probability for each edge.

> **Input:** A connected graph $G = (V, E)$, and edge failure probabilities p_e for each edge $e \in E$.
>
> **Goal:** Compute $p_{\text{fail}} = \Pr[\text{G becomes disconnected when each edge } e \text{ fails independently with probability } p_e]$.

This problem is $\#\mathcal{P}$-hard, even in the special case $p_e = p = 1/2\ \forall e \in E$ [PB83]. We will look at an fpras for it, due to Karger [Kar95], that makes use of the Karp-Luby algorithm above as well as other ingredients.

Theorem 5.2 (Karger [Kar95])**.** *There exists a fpras for network reliability (for any set of edge-dependent failure probabilities $\{p_e\}_{e\in E}$).*

We begin with a high-level sketch of the algorithm. For simplicity, we will restrict attention to the case that all edge probabilities are equal, i.e., $p_e = p\ \forall e \in E$. Let c be the size of a minimum cut in G. Then clearly we have $p_{\text{fail}} \geq p^c$, since if all of the edges of any cut fail, G becomes disconnected. Then, we can make the following observations:

1. If $p^c \geq \frac{1}{n^4}$, then a naive Monte Carlo approach works. Namely, suppose we simply pick a random subgraph of G by removing each edge independently with probability p, and set $X_i = 1$ if the subgraph is disconnected, and $X_i = 0$ otherwise. Then X_i is an unbiased estimator of $\mu = p_{\text{fail}}$, and by the Unbiased Estimator Theorem, we only need $O(\frac{1}{\mu\varepsilon^2}) = O(n^4\varepsilon^{-2})$ trials to achieve the desired error bound.

2. Otherwise, if $p^c < \frac{1}{n^4}$, then, for $\alpha = 2 + \frac{1}{2}\log_n(2/\epsilon)$, we will prove that

$$\Pr[\text{some cut of size} \geq \alpha c \text{ fails}] \leq \varepsilon p^c \leq \varepsilon p_{\text{fail}}.$$

   Therefore, we can effectively ignore cuts of size $\geq \alpha c$, by absorbing the resulting error into ϵ. (More precisely, we will get an estimate within $(1 \pm \varepsilon)^2$, which is no worse than $(1 \pm 3\varepsilon)$; so we just replace ε by 3ε.)

3. We say a cut is an **α-minimum cut** if it has size $\leq \alpha c$. Then we have the following claim and corollary:

   Claim 5.3. *There are at most $n^{2\alpha} = \frac{2n^4}{\varepsilon}$ α-minimum cuts, and these cuts can be enumerated in time polynomial in $(n, \frac{1}{\varepsilon})$.*

   Corollary 5.4. *The probability that an α-minimum cut fails can be expressed as the solution to a probabilistic DNF problem as follows:*

$$\Pr[\textit{some } \alpha\textit{-minimum cut fails}] = \Pr\Big[\bigvee_{i=1}^{t}(x_{e_{i_1}} \wedge x_{e_{i_2}} \wedge \ldots \wedge x_{e_{i_r}})\Big],$$

   where the OR is over all $t \leq 2n^4/\varepsilon$ α-minimum cuts, $\{e_{i_1}, \ldots, e_{i_r}\}$ are the edges of the i^{th} cut and

$$x_{e_j} = \begin{cases} T & \text{with probability } p; \\ F & \text{otherwise.} \end{cases}$$

   Moreover, this formula can be constructed in time polynomial in $(n, \frac{1}{\varepsilon})$.

   With the above corollary, we can apply the Karp/Luby algorithm for probabilistic DNF from Theorem 5.1 to get a fpras for computing the probability that an α-minimum cut fails. In light of items 1 and 2, this gives us an fpras for Network Reliability.

It remains to prove the key Claim 5.3, and also the statement made in item 2 above (which in fact also follows from Claim 5.3). We prove the claim in the next subsection.

5.3.1 Proof of Claim 5.3

We first describe a randomized algorithm (also due to Karger [Kar93]) for finding a minimum cut in a connected graph $G = (V, E)$. (Note that this algorithm can be used to find the value of c that is required in step 1 of the above algorithm.) We'll call this algorithm *RMinCut*. From it, we will easily obtain an upper bound on the number of minimum cuts in any graph.

while $|V| > 2$ **do**
  Choose an edge $\{u, v\} \in E$ uniformly at random.
  Merge u and v, maintaining all edges from either of them to other vertices.
Return the remaining cut.

Note that the algorithm actually maintains a multigraph, since during a merge operation multiple edges may be created. When a random edge is picked, we view multiple edges as distinct.

Theorem 5.5. *Let $C \subset E$ be any minimum cut. Then* $\Pr[RMinCut \text{ returns } C] \geq \binom{n}{2}^{-1}$.

Proof. Say that C is *hit* at stage i if one of its edges $\{u, v\}$ is selected and collapsed at stage i in *RMinCut*. Observe that no vertex of G can have fewer than c neighbors; otherwise the cut disconnecting just that vertex would have size less than c. Hence, $|E(G)| \geq \frac{nc}{2}$ and

$$\Pr[C \text{ is hit in round } 1] \leq \frac{c}{nc/2} = \frac{2}{n}.$$

Similarly,

$$\Pr[C \text{ is hit in round } i+1 \mid C \text{ survives rounds } 1, ..., i] \leq \frac{c}{(n-i)c/2} = \frac{2}{n-i}.$$

Therefore,

$$\begin{aligned}
\Pr[C \text{ survives all rounds}] &\geq \left(1 - \frac{2}{n}\right) \times \left(1 - \frac{2}{n-1}\right) \times ... \times \left(1 - \frac{2}{3}\right) \\
&= \frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \times ... \times \frac{1}{3} \\
&= \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}.
\end{aligned}$$

□

Note that $O(n^2)$ trials of *RMinCut* are required to be confident that we have found a minimum cut. An obvious implementation takes time $O(n^2)$ per trial, for an overall running time of $O(n^4)$.

Exercise (tough): *Devise a cleverer, recursive implementation of RMinCut which achieves $O(n^2 \log n)$ overall running time.*
One may compare this with the approach of finding an s-t minimum cut for any fixed s and all t using a network flow algorithm, which naively would take overall time about $O(n^4 \log n)$ (for $O(n)$ flow computations each of cost $O(n^3 \log n)$).

For our purposes, a more important consequence of the above is the following.

Corollary 5.6. *For any graph G, the number of minimum cuts is at most $\binom{n}{2}$.*

This follows immediately from the fact that each distinct minimum cut is output with probability at least $\binom{n}{2}^{-1}$, and these are disjoint events.

Now let us focus on α-minimum cuts.

Corollary 5.7. *The number of α-minimum cuts is at most $n^{2\alpha}$ in any graph G.*

Proof. We know that by definition, an α-minimum cut C has size at most αc. As in the proof of Theorem 5.5, we have

$$\Pr[C \text{ is hit in round 1}] \leq \frac{\alpha c}{nc/2} = \frac{2\alpha}{n}$$

and

$$\Pr\big[C \text{ is hit in round } i+1 \mid C \text{ survives rounds } 1, ..., i\big] \leq \frac{\alpha c}{(n-i)c/2} = \frac{2\alpha}{n-i},$$

so that

$$\Pr[C \text{ survives until } 2\alpha \text{ vertices remain}] \geq \binom{n}{2\alpha}^{-1}.$$

(It is necessary to employ Γ functions to make sense of $\binom{n}{2\alpha}$ if 2α is not an integer, but we won't dwell on this detail here.)

Now define a process $\mathcal{P}$ as follows:

- apply the $RMinCut$ routine until 2α vertices remain
- pick a random cut in the remaining multigraph

Then we have

$$\begin{aligned}\Pr[C \text{ survives } \mathcal{P}] &= \Pr[C \text{ survives until } 2\alpha \text{ vertices remain}] \times \Pr[C \text{ survives random cut}] \\ &\geq \frac{1}{\binom{n}{2\alpha}} \times \frac{1}{2^{2\alpha-1}} \\ &\geq \frac{(2\alpha)!}{2^{2\alpha}} \cdot \frac{1}{n^{2\alpha}} \geq \frac{1}{n^{2\alpha}},\end{aligned}$$

which proves the corollary. □

This is the first part of Claim 5.3 of the algorithm. It remains only to enumerate the α-minimum cuts of G. This is achieved via the coupon-collector paradigm: if balls are thrown independently and u.a.r. into n bins, how many balls must be thrown to ensure that every bin contains at least one ball (with high probability)? We have

$$\Pr[\text{at least } (n \log n + an) \text{ balls must be thrown}] \leq e^{-a}$$

for any fixed a, so with very high probability $O(n \log n)$ balls suffice. Thus, if we repeatedly run process $\mathcal{P}$ above, after $O(n^{2\alpha} \log(n^{2\alpha})) = O(n^4(\log n + \log \epsilon^{-1})/\epsilon)$ attempts we will have enumerated all α-minimum cuts with high probability. (The small probability of failure can be absorbed into the error probability of our fpras.)

5.3.2 Justification of item 2

To complete the analysis of Karger's algorithm, we need to prove the claim in item 2 in the high-level sketch given earlier. For convenience, let $\delta > 2$ be such that $p^c = n^{-(2+\delta)}$. Let $C_1, C_2, ...$ be an enumeration of the cuts of size at least αc, and for each i, let $c_i = |C_i|$. We will assume that $c_1 \leq c_2 \leq c_3 \leq ....$ We divide the analysis into two parts.

(i) Consider the first $n^{2\alpha}$ cuts, then the remainder in sequence. We can also assume, without loss of generality, that there are at least $n^{2\alpha}$ cuts; otherwise, the argument below gives an even better bound. Note that for each $i \leq n^{2\alpha}$, Pr[all edges in C_i fail] $\leq p^{\alpha c}$. Hence,

$$\Pr\left[\bigvee_{i=1}^{n^{2\alpha}} C_i \text{ fails}\right] \leq n^{2\alpha} p^{\alpha c} = n^{2\alpha} n^{-(2+\delta)\alpha} = n^{-\delta\alpha}.$$

This takes care of the initial sequence of cuts.

(ii) For any $\beta > 0$, we know that there are no more than $n^{2\beta}$ cuts of size at most βc, by Corollary 5.7; that is, $c_{n^{2\beta}} \geq \beta c$. Writing $k = n^{2\beta}$, this translates to $c_k \geq \frac{c}{2} \log_n k$, so

$$p^{c_k} \leq p^{\frac{c}{2}\log_n k} = n^{-(2+\delta)(\log_n k)/2} = k^{-(1+\delta/2)}.$$

Thus,

$$\Pr\left[\bigvee_{i>n^{2\alpha}} C_i \text{ fails}\right] \leq \sum_{k>n^{2\alpha}} k^{-(1+\delta/2)} \leq \int_{n^{2\alpha}}^{\infty} x^{-(1+\delta/2)} dx = \frac{2}{\delta} n^{-\delta\alpha} < n^{-\delta\alpha}.$$

Putting (i) and (ii) together, we get

$$\Pr[\text{some cut of size} \geq \alpha c \text{ fails}] \leq 2n^{-\delta\alpha}.$$

Now with our choice of $\alpha = 2 + \frac{1}{2}\log_n(2/\epsilon) \geq 1 + \frac{2}{\delta} + \frac{1}{\delta}\log_n(2/\epsilon)$, this yields

$$\Pr[\text{some cut of size} \geq \alpha c \text{ fails}] \leq \epsilon n^{-(2+\delta)} = \epsilon p^c.$$

This concludes the argument for item 2 and the analysis of the algorithm claimed in Theorem 5.2.

Remarks:

1. A different, theoretically more efficient algorithm for the same problem, incorporating some of the same ideas but dispensing with the expensive step of cut enumeration, was recently obtained by Karger [Kar16].
2. This algorithm can be derandomized in every aspect except Part 1, the naive Monte Carlo routine. It is possible to derandomize both the enumeration of cuts and the Karp-Luby algorithm in polynomial time, but it is not known how or if the simple Monte Carlo method can be derandomized efficiently.
3. Note that this fpras does not provide a good estimate of $p_{\text{success}} = (1 - p_{\text{fail}})$, which is interesting when the probability p of edge failure is large (a less important scenario in the Network Reliability context, but mathematically interesting). Very recently, Guo and Jerrum have provided a fpras for p_{success}, using quite different methods [GJ19]. The same result actually follows as a consequence of the more general algorithm for approximately counting bases of a matroid due to Anari *et al.* [ALOV18] which we will discuss later in the course.

References

[ALOV18] N. Anari, K. Liu, S. Oveis Gharan, and C. Vinzant. Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 35–46, 2018.

[GJ19] H. Guo and M. Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. *SIAM Journal on Computing*, 48:964–978, 2019.

[Kar93] D.R. Karger. Global min-cuts in RNC and other ramifications of a simple min-cut algorithm. *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 21–30, 1993.

[Kar95] D.R. Karger. A randomized fully polynomial approximation scheme for the all terminal network reliability problem. *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–17, 1995.

[Kar16] D.R. Karger. A fast and simpler unbiased estimator for network (un)reliability. *Proceedings of the 48th Annual Symposium on the Foundations of Computer Science (FOCS)*, pages 635–644, 2016.

[KL83] R.M. Karp and M. Luby. Monte-Carlo algorithms for enumeration and reliability problems. *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC)*, pages 56–64, 1983.

[KLM89] R.M. Karp, M. Luby, and N. Madras. Monte Carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10:429–448, 1989.

[PB83] J.S. Provan and M.O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12:777–788, 1983.