Advanced Algorithms 10/17, 2017

Lecture 5: FPRAS

Lecturer: Mohsen Ghaffari *Scribe: Michal Švagerka*

This is first rough draft, which has not been checked by the lecturer.

1 Fully polynomial randomised approximation schemes

In this lecture, we relax the notion of approximation schemes introduced in the previous lecture by allowing randomness in the sense of Monte Carlo algorithms.

Definition 1. *Let $f : L \to \mathbb{R}_0^+$ be a function of words from some language L representing a counting problem. Let $\mathcal{A}$ be a probabilistic algorithm, which takes as input an instance of the problem $x \in L$, and a parameter $\varepsilon > 0$. We call $\mathcal{A}$ a **fully polynomial randomised approximation scheme** or short **FPRAS**, if it has the following properties*

- *the algorithm returns value within required precision with constant probability:*

$$Pr[|\mathcal{A}(x, \epsilon) - f(x)| \leq \varepsilon f(x)] \geq \frac{3}{4}$$

- *the running time of algorithm $\mathcal{A}$ is polynomial both in $|x|$, the size of the input, and ε^{-1}, the required precision.*

Remark Note that the constant 3/4 is somewhat arbitrary, the only requirement is that this constant is bounded away from 1/2. This success probability can be augmented by repeating the algorithm multiple times and taking the median as an answer. It can be easily shown that $\mathcal{O}(\log \frac{1}{\delta})$ suffice to achieve a probability of success at least $1 - \delta$ for an arbitrary constant $\delta > 0$.

We study three classical counting problems: DNF counting, Network reliability and Graph colourings.

2 DNF counting

Definition 2. *Let $x_1, x_2, \ldots, x_n$ be a collection of boolean variables. A literal can be either a variable x_i or its negation $\neg x_i$. A clause is a conjunction of a positive amount of literals. A formula in disjunctive normal form is a disjunction of a positive amount of clauses.*

An assignment is a mapping from the set of variables into TRUE/FALSE values. We say that assignment satisfies clause or formula, if the boolean function described by clause or formula evaluates to TRUE. One can easily see that deciding satisfiability of a DNF formula is in class **P** – as such formula is satisfiable if and only if any of its clauses is satisfiable, and a clause is satisfiable if and only if it does not contain both a variable and its negation. In the following we will assume that the formula contains no such clauses and hence it is satisfiable.

We will be interested in the number of satisfying assignments. This problem is known as DNF counting and it has been shown that it is **#P**-complete. Let S_i be the set of assignments satisfying i-th clause and S the set of assignments satisfying at least one of the clauses. If we assume that the clause contains q unique literals, then simply $|S_i| = 2^{n-q}$, as the assignment of the variables present in the clause is dictated by the clause, and the rest can be picked arbitrarily.

Calculation via PIE: A naïve approach to calculate the cardinality of a union is to use the principle of inclusion and exclusion. Indeed, we can calculate the amount precisely as

$$|S| = \left|\bigcup_{i=1}^{m} S_i\right| = \sum_{i=1}^{m} |S_i| - \sum_{i<j\in[m]} |S_i \cap S_j| + \sum_{i<j<k\in[m]} |S_i \cap S_j \cap S_k| + \cdots + (-1)^{m-1} \left|\bigcap_{i=1}^{m} S_i\right|$$

The drawback of this method is that there are exponentially many terms in this large sum and therefore this is not an efficient way. Furthermore, it has been shown that the method of truncation PIE does not work for this particular problem, as it can produce answers that are too far from the actual amount.

Simple Monte Carlo method: Let us examine another approach based on a Monte Carlo method that is a step towards a FPRAS. The algorithm works in a number of iterations depending on the size of input $|x|$ and the parameter ε. In each iteration, it samples an assignment uniformly at random from the set of all assignments and checks whether it satisfies the formula.

Let Y_i be a indicator random variable that is 1 if the i-th assignment satisfies the formula and 0 otherwise and let s be the number of sampled assignments. The algorithm outputs

$$Y := 2^n \frac{\sum_{i=1}^{s} Y_i}{s}.$$

We can see that this is an unbiased estimator of the number of satisfying assignments. The problem is that the sum of indicator variables

$$\mathbb{E}\left[\sum Y_i\right] = s\frac{|S|}{2^n}$$

may be exponentially small (if most of the assignments are not satisfying) and therefore it may not be well concentrated. Indeed, it may happen that the algorithm outputs 0 with high probability, even though there are plenty (e.g. $2^{\frac{n}{2}}$) satisfying assignments, since the number of iterations is only polynomial. We definitely need a smarter approach to tackle this problem.

Refined Monte Carlo method: Firstly, we take a graphical view of the situation. Imagine a table whose rows represent the m clauses, and the columns represent the satisfying assignments a_i, whose number is yet unknown to us. In each cell, there is 1 if the assignment corresponding to the column satisfies the clause in the given row, and 0 otherwise. The table may look like this:

	a_1	a_2	a_3	$\cdots$	a_k
S_1	**1**	0	0	$\cdots$	0
S_2	1	0	**1**	$\cdots$	**1**
S_3	0	**1**	0	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
S_m	0	1	0	$\cdots$	1

The bolded number represents the topmost 1 in every column and its significance will become evident soon. First note that every column has such one, since only satisfying assignments are included as columns. This, however, means, that the quantity we're after is the number of topmost ones – as this is exactly the number of satisfying assignments.

Furthermore, we know the number of ones in this table – it is equal to $\sum_{i=1}^{m} |S_i|$. By using the Monte Carlo method we can estimate the ratio of topmost ones to all ones, effectively calculating the number of satisfying assignments. For us to be able to do that, we need to argue the following:

- We can efficiently sample a "1" from the table uniformly at random. To achieve this, we first sample a clause with probability proportional to $|S_i|$. Next, we generate an assignment satisfying the sampled clause uniformly at random. One can see that this samples every "1" in the table with the same probability.
- We can efficiently decide whether a "1" is a topmost "1": This is also trivial – one just evaluates all clauses using the assignment, and checks whether the sampled clause is also the first satisfied clause. This step takes $\mathcal{O}(nm)$ time.
- Finally, it is evident that since each column contains a topmost "1", the fraction of the topmost ones among the general population of ones is at least $\frac{1}{m}$. Hence, using Chernoff, one can prove a concentration around expectation using $\mathcal{O}(\frac{m}{\varepsilon^2})$ iterations.

Combining these results yields a $\mathcal{O}(nm^2\varepsilon^{-2})$ time FPRAS for DNF counting.

3 Network Reliability

Let's consider an undirected graph $G = (V, E)$, where each edge $e \in E$ has a certain probability of failing, namely p_e, independent of other edges. In the Network Reliability problem, we are interested in calculating the probability that network becomes disconnected.

Note that we aim to obtain a $(1 \pm \varepsilon)$-approximation of the quantity $P := \Pr[G \text{ is disconnected}]$. Such an approximation is not necessarily a $(1 \pm \varepsilon)$-approximation of the probability of the converse, $1 - P = \Pr[G \text{ is connected}]$ and vice versa, since P may be very close to 1. However, as we will also discuss later, networks with rather large probability of getting disconnected are not useful in practice. For simplicity we study the symmetrical case, where $p_e = p$ for every edge $e \in E$.

Observation 3. *For an edge cut of size k edges, the probability of its disconnection is p^k.*

Reduction to DNF counting: In [KL83] Karp and Luby demonstrated a simple method of reduction to DNF counting – each edge is represented by a variable, and every clause corresponds to a cut postulating that all of its edges have failed. The probability of disconnecting can then be inferred from the fraction of satisfying assignments.

Unfortunately, there are exponentially many cuts in a general graph and this method is thus inefficient. The reduction we present here is due to Karger [Kar01]. We discuss two cases based on the minimum cut size c:

- When $p^c \geq \frac{1}{n^4}$, this means that the probability of the network disconnection is rather high. As mentioned previously, this is not the case of a large interest, since the motivation behind studying this problem is the understanding how to build reliable networks, and network with rather small cut is not. Nevertheless, since the probability of disconnection is rather high, Monte-Carlo method of sampling subsets of edges and checking whether they are cuts is sufficient, since we only need $\tilde{\mathcal{O}}(n^4)$ samples to achieve concentration.

- When $p^c \leq \frac{1}{n^4}$, we show that the large cuts do not contribute to the probability of disconnection too much, and therefore they can be safely ignored. Recall that the number of cuts of size αc for $\alpha \geq 1$ is at most $\mathcal{O}(n^{2\alpha})$. When one selects a threshold $\gamma = \mathcal{O}(\log_n \varepsilon^{-1})$ on the cut size, we can express the contribution of large cuts to the overall probability as

$$\int_{\gamma}^{\infty} n^{2\alpha} \cdot p^{\alpha c} \, d\alpha < \varepsilon p^c$$

  which is within ε factor of the lower bound on the probability of failure – p^c.

  The number of cuts smaller than γc is clearly a polynomial of n and therefore the reduction to DNF-counting is efficient. The only remaining thing is finding those – Karger's contraction algorithm provides us with a way of sampling those cuts and we can thus use Coupon collector to enumerate those.

4 Graph Colourings

Finally, we study the problem of calculating graph colourings. Let $G = (V, E)$ be an undirected graph with n vertices and m edges, and k a positive integer. The goal is to find the number of proper colouring of graph G using k colours.

The problem of deciding whether a general graph is colourable with k colours is **NP**-complete for every $k \geq 3$. An $(1 \pm \epsilon)$ approximation algorithm would be able to decide this decision problem. We thus only study graphs that are guaranteed to have at least one such colouring. To ensure that graph is k-colourable, we can simply set up $k > \Delta$, where Δ is the maximum degree of a vertex. In this setup, a colouring can be constructed using greedy algorithm. It has been shown that counting $(\Delta+1)$-colourings is hard, we present an FPRAS for coutning $(2\Delta + 1)$-colourings. It is conjectured that the following approach is also sufficient for $k = \Delta + 2$.

Generating a random colouring: To sample a random $(2\Delta + 1)$-colouring of a given graph, we will use the following iterative approach. First, we pick a greedy colouring as a starting point. Then we will repeatedly sample a vertex and sample its new colour from the set of colours not used by any of its neighbours, uniformly at random. This can be modelled using a Markov chain.

The Markov chain at hand as all the nice properties that we would require for this process to be successful at sampling colourings. Firstly, the chain is aperiodic. This is due to the fact that the colour of a vertex can be retained while resampling (since it is not used in any of its neighbours). As a consequence, the graph is aperiodic. Furthermore, we can transform between any two colourings using at most $2n$ steps, thus the chain is irreducible. This combined implies that the chain converges to its stationary distribution. As the chain is clearly symmetric, the stationary distribution is uniform. Furthermore, it can be shown that the chain is rapidly mixing, with mixing time $\mathcal{O}(n \log n)$. Therefore, the distribution of the graphs sampled by it, given enough iterations, will be point-wise at most ε from the uniform distribution.

Estimating the number of colourings: Armed by this, we can investigate how to estimate the number of colourings. In order to do that, we number the edges of the graph as $e_1, e_2, \ldots, e_m$ in arbitrary order. Let $G_0 = (V, \emptyset)$ be a graph on n isolated vertices, and $G_i = G_{i-1} \cup \{e_i\}$. We can see that $G_m = G$. Let $P(H, k)$ be the number of colourings of graph H using k colours. The answer we seek lies in the following telescopic product:

$$P(G, k) = \frac{P(G, k)}{P(G_{m-1}, k)} \cdot \frac{P(G_{m-1}, k)}{P(G_{m-2}, k)} \cdots \frac{P(G_2, k)}{P(G_1, k)} \cdot \frac{P(G_1, k)}{P(G_0, k)} \cdot P(G_0, k)$$

Note that $P(G_0, k)$ is simply k^n, as the graph has no edges, every colouring is a proper colouring. If we can approximate every fraction with ratio $1 \pm \frac{\epsilon}{m}$ with failure probability at most $\frac{\delta}{m}$, then we can approximate $P(G_k)$ to ratio $(1 \pm \epsilon)$ with failure probability at most δ. Estimating each fraction is easy, we just sample a proper colouring of G_{i-1} using the above Markov chain, and check whether it is a proper colouring of G_i.

We further argue that this fraction is not too small and can be thus the value we obtain is concentrated around its expectation. Since any improper colouring of G_i we obtain must be a proper colouring of G_{i-1}, the only conflict can happen along the edge e_i. We can thus fix the colouring by sampling the colour of a vertex v_i from e_i that has smaller label. Note that this is an injective mapping, and as we have at least Δ options for the colour of v_i, we get

$$\frac{P(G_i, k)}{P(G_{i-1}, k)} \geq 1 - \frac{1}{\Delta}\,.$$

References

[Kar01] David R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM Rev.*, 43(3):499–522, March 2001.

[KL83] L Karp and M Luby. Monte-carlo algorithms for enumeration and reliability. In *Proceedings, 24th IEEE Foundations of Computer Science Symposium*, pages 56–64. IEEE, 1983.