CSC2420 - Algorithm Design, Analysis and Theory **Fall 2022**

Lecture 8: Randomized Algorithm

Lecturer: Allan Borodin *Scribe: Kevin Gao*

8.1 The Why of Randomized Algorithms

There are some problem settings (e.g. simulation, cryptography, interactive proofs, sublinear algorithms) where randomization is necessary. We can also use randomization to improve approximation ratios.

In complexity theory, a fundamental question is how much can randomization lower the time complexity of a problem. For decision problems, there are three polynomial time randomized classes ZPP (zero-sided), RP (one-sided), and BPP (two-sided) error. The big question in **fine-grained complexity theory** is BPP $\stackrel{?}{=}$ P.

One important aspect of randomized algorithms in an offline setting is that the probability of success can be amplified by repeated independent trials of the algorithm. We will present some problems with randomized polynomial time algorithms but with no known deterministic polynomial algorithms.

Randomization also plays a central role in online algorithms as the online settings is particularly vulnerable to worset case adversarial inputs.

8.2 Polynomial Identity Testing

We are implicitly given two multivariate polynomials and wish to determine if they are identical. One way we could be implicitly given these polynomial is by an arithmetic circuit. A specific case is the following symbolic determinant problem.

Problem 8.1 (Symbolic Determinant Problem)**.**

Input: $n \times n$ matrix $A = (a_{i,j})$ whose entries are polynomials of total degree d in m variables, say with integer coefficients.
Output: Whether or not $\det(A) = 0$ where

$$\det(A) = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} a_{i,\pi(i)}.$$

Lemma 8.2 (Schartz-Zipple Lemma)**.** *Let $P \in \mathbb{F}[x_1, \ldots, x_m]$ be a non-zero polynomial over a field $\mathbb{F}$ of total degree at most d. Let S be a finite subset of $\mathbb{F}$. Then,*

$$\Pr_{r_i \in S}[P(r_1, \ldots, r_m) = 0] \leq \frac{d}{|S|}$$

The Schartz-Zipple lemma is a multivariate generalization that a univariate polynomial of degree d can have at most d zeros.

For the randomized algorithm for symbolic determinant, pick a large set S of integers such that $|S| \geq 2nd$. Randomly choose $r_i \in S$. Evaluate $x_i = r_i$. Then, we have a matrix of integers.

The determinant can then be calculated in $O(n^3)$ arithmetic operations. We are computing a degree nd polynomial $\det(A)$ at random $r_i \in S$. Since $|S| \geq 2nd$, then

$$\Pr[\det(A') = 0] = \frac{1}{2}.$$

The probability of correctness con be amplifed by choosing a bigger S or by repeated trials.

8.3 Exact Max-k-SAT

Recall that in **exact** max-k-SAT, we are given a CNF formula in which every clause has exactly k literals. We consider the weighted case in which clauses have weights. The goal is to find a satisfying assignment that maximizes the size (or weight) of clauses that are satisfied. In the general **max-SAT** problem, there is no restriction on the number of literals in each clause.

The naive randomized online algorithm for max-k-SAT is to randomly set each variable to TRUE or FALSE with equal probability.

Proof: Each clause C_i has k variables. The probability that a random assignment of the literals in C_i will set the clause to be satisfied is exactly $\frac{2^k-1}{2^k}$. Hence,

$$\mathbb{E}[w(\text{satisfied clauses})] = \frac{2^k-1}{2^k}\sum_i w_i$$

■

8.3.1 Derandomizing the Naive Randomized Algorithm

This can be derandomized into a deterministic algorithm using the method of **conditional expectations**. For notation simplicity, let T = 1 and F = 0. Let $w(F)|\tau$ denote the weighted sum of satisfied clauses given truth assignment τ.

Let x_j be any variable. We express

$$\mathbb{E}[w(F)|_{x_i\in\{0,1\}}] = \frac{1}{2}\mathbb{E}[w(F)|_{x_i\in\{0,1\}}|x_j = 1] + \frac{1}{2}\mathbb{E}[w(F)|_{x_i\in\{0,1\}}|x_j = 0]$$

This implies that one of the choices of x_j will yield an expectation at least as large as the overall expectation. Hence, to derandomize this algorithm, we can set each variable x_j to the value that maximizes the overall expectation. We can continue to do this for each variable and obtain a deterministic solution whose weight is at least the overall expected value of the naive randomized algorithm.

More formally, if $\mathbb{E}[w(F)|x_i = 1] \geq \mathbb{E}[w(F)|x_i = 0]$, we set x_i to True and False otherwise. Then, the deterministic choice of the truth assignment to x_i guarantees an expected value no less than the expected value of the randomized algorithm because

$$\max\{\mathbb{E}[w(F)|x_i = 1], \mathbb{E}[w(F)|x_i = 0]\} \geq \mathbb{E}[w(F)].$$

For $k \geq 3$, Hasatad proved using PCP that it is NP-hard to improve upon the $\frac{2^k-1}{2^k}$ approximation ratio for max-k-SAT. For max-2-SAT, the $\frac{3}{4}$ ratio can be improved by the use of semi-definite programming and randomized rounding. However, the naive randomized algorithm only achieves an approximation ratio of 1/2 for the general max-SAT problem.

8.4 Max-SAT

The analysis for exact max-k-SAT needs the fact that all clauses have k literals. Next, we consider **Johnson's algorithm** for the general max-SAT problem where there could be unit clauses.

JOHNSON'S ALGORITHM

```
1  for all clauses C_i, w'_i = w_i/2^{|C_i|}
2  L = set of clauses in formula F
3  X = set of variables
4  for x ∈ X or until L == ∅
5        P = {C_i ∈ L | x occurs positively}
6        N = {C_j ∈ L | x occurs negatively}
7        if Σ_{C_i∈P} w'_i ≥ Σ_{C_j∈N} w'_j
8              x = T
9              L = L \ P
10             for C_r ∈ N
11                   w'_r = 2w'_r
12       else
13             x = F
14             L = L \ N
15             for C_r ∈ P
16                   w'_r = 2w'_r
17       delete x from X
```

Yannakakis observed that for arbitrary max-SAT, the approximation ratio (not totality ratio) of Johnson's algorithms is at best $\frac{2}{3}$. Consider the 2-CNF: $F = (x \vee \neg y) \wedge (\neg x \vee y) \wedge \neg y$ when variable x is first set to true; otherwise, consider $F = (x \vee \neg y) \wedge (\neg x \vee y) \wedge y$.

Suppose that we assign each variable x_j with a probability value p_j.

RANDOMIZED-MAX-SAT

```
1  L = set of clauses in formula F
2  X = set of variables
3  let u_i = Pr[C_i is not satisfied] = (∏_{¬x_j∈C_i} p_j) · (∏_{x_j∈C_i} (1 − p_j))
4  for x_j ∈ X or until L == ∅
5        P = {C ∈ L | x_j occurs positively}
6        N = {C ∈ L | x_j occurs negatively}
7        if p_j · Σ_{C_i∈P} w(C)u_i ≥ (1 − p_j) Σ_{C_i∈N} w(i)u_i
8              x_j = T
9              L = L \ P
10             for C_r ∈ N
11                   u_r = u_r/p_j
12             ⋮
13       delete x from X
```

Observe that Johnson's algorithm is the derandomization of this randomized algorithm with $p_i = 1/2$ for all variables x_i (Yannakakis, 1994).

In proving the $\frac{2}{3}$ approximation ratio for Johnson's max-SAT algorithm, Chen et al. asked whether or not the approximation ratio could be improved by using a random ordering of the propositional variables. Costello, Shapira and Tetali in 2011 showed that in the ROM model, Johnson's algorithm achieves approximation $(2/3 + \epsilon)$ for $\epsilon \approx 0.003653$. Poloczek and Schnitger showed that the approximation ratio for Johnson's algorithm in ROM is at most $2\sqrt{15} - 7 \approx 0.746 < 3/2$, noting that $3/4$ is the ratio first obtained by Yannakakis's IP/LP approximation that we will soon present.

8.4.1 Max-SAT in Random Order Model

To precisely model the max-SAT problem within an online or priority framework, we need to specify the input model. We consider the following models in increasing order of information provided:

M0. Each propositional variable x is represented by the names of the positive and negative clauses in which it appears.

M1. Each propositional variable x is represented by the length of each clause C_i in which x appears positively, and for each clause C_j in which it appears negatively.

M2. In addition to M1, we provide for each C_i and C_j, a list of the other variables in that clause.

M3. The variable x is represented by a complete specification of each clause in which it appears.

Poloczek and Schnitger first considered a "canonical randomization" of Johnson's algorithm. The canonical randomization algorithm sets a variable x_i to True with probability $\frac{w_i'(P)}{w_i'(P)+w_i'(N)}$ where $w_i'(P)$ and $w_i'(N)$ is the current combined weight of clauses in which x_i occurs positively and negatively, respectively. Their substantial idea is to adjust the random setting so as to **better account for the weight of unit clauses** in which only one variable occurs.

8.4.2 Randomized Rounding for Max-SAT

We will reformulate the weighted max-SAT as a 0/1 IP: suppose that there are n variables and m clauses.

$$\begin{array}{lll} \max & \sum_j w_j z_j & \\ \text{s.t.} & \sum_{x_i \in C_j} y_i + \sum_{\neg x_i \in C_j} (1 - y_i) \geq z_j & \forall j \in \{1, \ldots, m\} \\ & y_i \in \{0, 1\} & \forall i \in \{1, \ldots, n\} \\ & z_j \in \{0, 1\} & \forall j \in \{1, \ldots, m\} \end{array}$$

For this integer program, $y_i = 1$ iff variable x_i is T in the corresponding max-SAT problem. $z_j = 1$ iff clause C_j is satisfied in the corresponding max-SAT problem. Since IP is NP-hard, we need to relax the constraints on y and z. In particular, we relax the constrants to $y_i \geq 0$ and $z_j \in [0, 1]$ (note that it is equivalent to say $y_i \in [0, 1]$).

Let y^* and z^* be an optimal solution of the relaxed LP. We obtain a random IP solution by setting each $\tilde{y_i} = 1$ independently with probability y^* and $\tilde{y_i} = 0$ with probability $1 - y^*$.

Theorem 8.3. *Let C_j be a clause with k literals. Then,*

$$\Pr[C_j \text{ is satisfied}] = \left[1 - \left(1 - \frac{1}{k}\right)^k\right] \cdot z_j^*$$

Proof: Recall the arithmetic-geometric mean inequality (AM-GM ineq.).

$$\left(\prod_{i=1}^{k} a_i\right)^k \leq \frac{1}{k}\sum_{i=1}^{k} a_i.$$

Pick an arbitrary clause C_j. Then,

$$\Pr[C_j \text{ not satisfied}] = \prod_{x_i \in C_j}(1-y_i^*)\prod_{\neg x_i \in C_j} y_i^* \leq \left[\frac{1}{k}\left(\sum_{x_i\in C_j}(1-y_i^*)+\sum_{\neg x_i\in C_j} y_i^*\right)\right]^k.$$

Note that by rearranging the terms,

$$\left[\frac{1}{k}\left(\sum_{x_i\in C_j}(1-y_i^*)+\sum_{\neg x_i\in C_j} y_i^*\right)\right]^k = \left[1-\frac{1}{k}\left(\sum_{x_i\in C_j} y_i^*+\sum_{\neg x_i\in C_j}(1-y_i^*)\right)\right]^k$$

so by the LP constraint that $\sum_{x_i\in C_j} y_i + \sum_{\neg x_i\in C_j}(1-y_i) \geq z_j$, we have

$$\Pr[C_j \text{ not satisfied}] \leq \left(1-\frac{z_j^*}{k}\right)^k$$

By concavity of $1-(1-\frac{z_j^*}{k})^k$,

$$\Pr[C_j \text{ satisfied}] \geq 1-\left(1-\frac{z_j^*}{k}\right)^k \geq \left[1-\left(1-\frac{1}{k}\right)^k\right]\cdot z_j^*.$$

■

It follows immediately from this theorem that for sufficiently large k, the probability that C_j is satisfied is lower bounded by $1-1/e$. Therefore, the expected solution of the rounded LP is at least $1-1/e$ the OPT of the original IP.

8.5 Weighted Max-Cut and Max-Di-Cut

Given a graph $G=(V,E)$, the max-cut objective is to partition the vertices into subsets S and T so as to maximize $|\{(u,v)\in E \mid u\in S,\, v\in T\}$.

In the weighted max-cut problem, there is a weight function $w:\ E\to\mathbb{R}^+$ and the objective is to maximize

$$\sum_{\substack{(u,v)\in E\\ u\in S\\ v\in T}} w(u,v).$$

In the max-di-cut problem, we are givena directed graph instead of an undirected graph.

Consider max-cut. We randomly and inependently set each u to be in S with probability $\frac{1}{2}$. For any edge (u,v), it is easy to see that the probability that the random assignment assigns different values to u and v is exactly $\frac{1}{2}$. Hence, the expected approximation ratio is $\frac{1}{2}$ and this is a totality ratio.

8.6 Approximation for USM

```
1  X_0 = ∅
2  Y_0 = U
3  for i = 1 to n
4        a_i = f(X_{i-1} ∪ {u_i}) − f(X_{i-1})
5        b_i = f(Y_{i-1} \ {u_i}) − f(Y_{i-1})
6        if a_i ≥ b_i
7              X_i = X_{i-1} ∪ {u_i}
8              Y_i = Y_{i-1} \ {u_i}
```

Buchbinder et al. showed that the natural randomization of this deterministic algorithm achieves approximation ratio 1/2. The randomized algorithm chooses to either add $\{u_i\}$ to X_{i-1} with probability $\frac{a'_i}{a'_i+b'_i}$ or to delete $\{u_i\}$ from Y_{i-1} with probability $\frac{b'_i}{a'_i+b'_i}$ where $a'_i = \max\{a_i, 0\}$ and $b'_i = \max\{b_i, 0\}$. If $a_i = b_i = 0$, then add $\{u_i\}$ to X_{i-1}.