CSC2420 Fall 2012: Algorithm Design, Analysis and Theory
An introductory (i.e. foundational) level graduate course.

Allan Borodin

November 1, 2012; Lecture 8

Randomized algorithms

The why of randomization:

- There are some problem settings (e.g. simulation, cryptography, interactive proofs, sublinear time algorithms) where randomization is necessary.
- We can use randomization to improve approximation ratios.
- Even when a given algorithm can be derandomized, there is often conceptual insight to be gained from the initial randomized algorithm.
- 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 (1-sided) and BPP (2-sided) error. The big question (and conjecture?) is BPP $=$ P?
- One important aspect of randomized algorithms is that the probability of success can be amplified by repreated independent trials of the algorithm.

Comment on Umesh Vazarani's DLS talk

- In his DLS talk, Umesh Vazarani made an interesting suggestion: Will quantum computation be a 2010s analog to the impact of randomization in say the 1980s. (I would say the 1970s and 80's.)
- For example, there is a necessary use of randomization in interactive proofs. Probabisitically checkable proofs (PCPs) gave rise to insights (e.g. hardness results) for approximation bounds.
- We will see today how a randomized algorithm can suggest algorithms and simplify analysis.
- So even if say true randomness does not exist (i.e. no source of true random bits), the computational use of randomization has great conceptual benefits. (Moreover, in practice, pseudo randomness works).
- Vazarani gave an example where quantum concepts have suggested new classical algorithms (in the context of matrix weighted majority) in addition to adding insight into semi definite programming.

Some problems in randomized polynomial not known to be in polynomial time

1. The symbolic determinant problem.
2. Given n, find a prime in $[2^n, 2^{n+1}]$
3. Estimating volume of a convex body given by a set of linear inequalitiies.
4. Solving a quadratic equation in $Z_p[x]$ for a large prime p.

Polynomial identity testing

- The general problem concerning polynomial identities is that we are implicitly given two multivariate polynomials and wish to determine if they are identical. One way we could be implicitly given these polynomials is by an arithmetic circuit. A specific case of interest is the following symbolic determinant problem.
- Consider an $n \times n$ matrix $A = (a_{i,j})$ whose entries are polynomials of total degree (at most) d in m variables, say with integer coeficients. The determinant $det(A) = \sum_{\pi \in S_n} (-1)^{sgn(\pi)} \prod_{i=1}^{n} a_{i,\pi(i)}$, is a polynomial of degree nd. The symbolic determinant problem is to determine whether $det(A) \equiv \mathbf{0}$, the zero polynomial.

Schwartz Zipple Lemma

Let $P \in \mathbf{F}[x_1, \ldots, x_m]$ be a non zero polynomial over a field $\mathbf{F}$ of total degree at most d. Let S be a finite subset of $\mathbf{F}$. Then $Prob_{r_i \in_u S}[P(r_1, \ldots . r_m) = 0] \leq \frac{d}{|S|}$

Schwartz Zipple is clearly a multivariate generalization of the fact that a univariate polynomial of degree d can have at most d zeros.

Polynomial identity testing and symbolic determinant continued

- Returning to the symbolic determinant problem, suppose then we choose a suffciently large set of integers S (for definiteness say $|S| \geq 2nd$). Randomly choosing $r_i \in S$, we evaluate each of the polynomial entries at the values $x_i = r_i$. We then have a matrix A' with (not so large) integer entries.
- We know how to compute the determinant of any such integer matrix $A'_{n\times n}$ in $O(n^3)$ arithmetic operations. (Using the currently fastest, but not necessarily practical, matrix multiplication algorithm the determinant can be computed in $O(n^{2.38})$ arithmetic operations.)
- That is, we are computing the $det(A)$ at random $r_i \in S$ which is a degree nd polynomial. Since $|S| \geq 2nd$, then $Prob[det(A') = 0] \leq \frac{1}{2}$ assuming $det(A) \not\equiv \mathbf{0}$. The probability of correctness con be amplifed by choosing a bigger S or by repeated trials.
- In complexity theory terms, the problem (is $det(A) \equiv \mathbf{0}$) is in co-RP.

The naive randomized algorithm for exact Max-k-Sat

We continue our discussion of randomized algorthms by considering the use of randomization for improving approximation algorithms. In this context, randomization can be (and is) combined with any type of algorithm.
Warning: For the following discussion of Max-Sat, we will follow the prevailing convention by stating approximation ratios as fractions $c < 1$.

- Consider the exact Max-k-Sat problem where we are given a CNF propositional 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.
- Since exact Max-k-Sat generalizes SAT, it is clearly an NP hard problem for $k \geq 3$. It is interesting to note that while 2-SAT is polynomial time, Max-2-Sat is still NP hard.
- The naive randomized (online) algorithm for Max-k-Sat is to randomly set each variable to *true* or *false* with equal probability.

Analysis of naive Max-k-Sat algorithm continued

- Since the expectation of a sum is the sum of the expectations, we just have to consider the probability that a clause is satisfied to determine the expected weight of a clause.
- Since 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 $\mathbf{E}$ [weight of satisfied clauses] $= \frac{2^k-1}{2^k} \sum_i w_i$
- Of course, this probability only improves if some clauses have more than k literals. It is the small clauses that are the limiting factor in this analysis.
- This is not only an approxination ratio but moreover a "totality ratio" in that the algorithms expected value is a factor $\frac{2^k-1}{2^k}$ of the sum of all clause weights whether satisfied or not.
- We can hope that when measuring againt an optimal solution (and not the sum of all clause weights), small clauses might not be as problematic as they are in the above analysis of the naive algorithm.

Derandomizing the naive algorithm

We can derandomize the naive algorithm by what is called the method of conditional expectations. Let $F[x_1, \ldots, x_n]$ be an exact k CNF formula over n propositional variables $\{x_i\}$. For notational simplicity let $true = 1$ and $false = 0$ and let $w(F)|\tau$ denote the weightea sum of satisfied clauses given truth assignment τ.

- Let x_j be any variable. We express $\mathbf{E}[w(F)|_{x_i \in_u \{0,1\}}]$ as $\mathbf{E}[w(F)|_{x_i \in_u \{0,1\}} | x_j = 1] \cdot (1/2) + \mathbf{E}[w(F)|_{x_i \in_u \{0,1\}} | x_j = 0] \cdot (1/2)$
- This implies that one of the choices for x_j will yield an expectation at least as large as the overall expectation.
- It is easy to determine how to set x_j since we can calculate the expectation clause by clause.
- We can continue to do this for each variable and thus obtain a deterministic online solution whose weight is at least the overall expected value of the naive randomized algorithm.
- For exact Max-2-Sat (resp. Max-3-Sat), the approximation (and totality) ratio is $\frac{3}{4}$ (resp. $\frac{7}{8}$). For $k \geq 3$, using PCPs, Hastad proves 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.

Johnson's Max-Sat Algorithm

Johnson's [1974] algorithm

For all clauses C_i, $w_i' := w_i/(2^{|C_i|})$
Let L be the set of clauses in F and X the set of variables
For $x \in X$ (or until L empty)
  Let $P = \{C_i \in L$ such that x occurs positively$\}$
  Let $N = \{C_j \in L$ such that x occurs negatively$\}$
  If $\sum_{C_i \in P} w_i' \geq \sum_{C_j \in N} w_j'$
    $x := true; L := LP$
    For all $C_r \in N, \quad w_r' := 2w_r'$ **End For**
  Else
    $x := false; L := L - N$
    For all $C_r \in P, \quad w_r' := 2w_r'$ **End For**
  End If
  Delete x from X
End For

Johnson's algorithm is the derandomized algorithm

- Yannakakis [1994] presented the naive algorithm and showed that Johnson's algorithm is the derandomized naive algorithm.
- Yannakakis also observed that for arbitrary Max-Sat, the approximation of Johnson's algorithm is at best $\frac{2}{3}$. For example, consider the 2-CNF $F = (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge \bar{y}$ when variable x is first set to true.
- Chen, Friesen, Zheng [1999] showed that Johnson's algorithm achieves approximation ratio $\frac{2}{3}$ for arbitrary weighted Max-Sat.
- For arbitrary Max-Sat (resp. Max-2-Sat), the current best approximation ratio is .797 (resp. .931) using semi-definite programming and randomized rounding.

Modifying Johnson's algorithm for Max-Sat

- In proving the (2/3) approximation ratio for Johnson's Max-Sat algorithm,, Chen et al asked whether or not the ratio could be improved by using a random ordering of the propositional variables (i.e. the input items). This is an example of the random order model (ROM), a randomized variant of online algorithms.
- To precisely model the Max-Sat problem within the priority framework, we need to specify the input model.
- In increasing order of providing more information (and possibly better approximation ratios), the following input models can be considered:
  1. 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.
  2. In addition, for each C_i and C_j, a list of the other variables in that clause is specified.
  3. The variable x is reprsented by a complete specification of each clause it which it appears.
- We note that Johnson's algorithm can be viewed as a priority algorithm using the weakest of these three input models.

Improving on Johnson's algorithm

- The question asked by Chen et al was answered by Costello, Shapira and Tetali [2011] who showed that in the ROM model, Johnson's algorithm achieves approximation $(2/3 + \epsilon)$ for $\epsilon \approx .003653$
- Poloczek and Schnitger [same SODA 2011 conference] show that the approximation ratio for Johnsons algorithm in the ROM model is at most $2\sqrt{157} \approx .746 < 3/4$, the ratio obtained by Yannakakis' IP/LP approximation that we will soon present.
- Poloczek and Schnitger first consider a "canonical randomization" of Johnson's algorithm"; namely, the canonical randomization sets a variable $x_i = \mathit{true}$ with probability $\frac{w_i'(P}{w_i'(P)+w_i'(N)}$ where $w_i'(P)$ (resp. $w_i'(N)$) is the current combined weight of clauses in which x_i occurs positively (resp. negatively). Their substantial additional idea is to adjust the random setting so as to better account for the weight of unit clauses in which a variable occurs.

A few comments on the Poloczek and Schnitger algorithm

- The new Poloczek and Schnitger algorithm is called Slack and has approximation ratio $= 3/4$.
- In terms of priority algorithms this is a randomized online algorithm (i.e. adversary chooses the ordering) where the variables are represented in the second (middle power) input model.
- This approximation ratio is in contrast to Azar et al [2011] proving that no randomized online algorithm can achieve approximation better than $2/3$ when the input model is the weakest of the input models.
- Finally (in this regard), Poloczek [2011] shows that no deterministic priority algorithm can achieve a $3/4$ approximation within the second (middle) input model. This provides a strong sense in which one can prove that the Poloczek and Schnitger Slack algorithm cannot be derandomized.
- The best deterministic priority algorithm in the third (most powerful) model remains an open problem as does the best randomized priority algorithm.

Yannakakis' IP/LP *randomized rounding* algorithm for Max-Sat

- We will formulate the weighted Max-Sat problem as a $\{0,1\}$ IP.
- Relaxing the variables to be in $[0,1]$, we will treat some of these variables as probabilities and then round these variables to 1 with that probability.
- Let F be a CNF formula with n variables $\{x_i\}$ and m clauses $\{C_j\}$. The Max-Sat formulation is :
  maximize $\sum_j w_j z_j$
  subject to $\sum_{\{x_i \text{ is in } c_j\}} y_i + \sum_{\{\bar{x}_i \text{ is in } c_j\}} (1 - y_i) \geq z_j$
  $y_i \in \{0,1\}$; $z_j \in \{0,1\}$
- The y_i variables correspond to the propositional variables and the z_j correspond to clauses.
- The relaxation to an LP is $y_i \geq 0$; $z_j \in [0,1]$. Note that here we cannot simply say $z_j \geq 0$.

Randomized rounding of the y_i variables

- Let $\{y_i^*\}, \{z_j^*\}$ be the optimal LP solution,
- Set $\tilde{y}_i = 1$ with probability y_i^*.

Theorem

Let C_j be a clause with k literals and let $b_k = 1 - (1 - \frac{1}{k})^k$. Then $Prob[C_j$ is satisifed $]$ is at least $b_k z_j^*$.

- The theorem shows that the contribution of the j^{th} clause C_j to the expected value of the rounded solution is at least $b_k w_j$.
- Note that b_k converges to (and is always greater than) $1 - \frac{1}{e}$ as k increases. It follows that the expected value of the rounded solution is at least $(1 - \frac{1}{e})$ LP-OPT $\approx .632$ LP-OPT.
- Taking the max of this IP/LP and the naive randomized algorithm results in a $\frac{3}{4}$ approximation algorithm that can be derandomized.