Lecture 3: Averaging argumets in randomized algorithms; derandomization; randomized rounding; relaxation

Bogdan Warinschi

January 29, 2001

The theme of this class is: randomness can be an essential tool for devising algorithms, even when it later can be eliminated from them without much cost.

Simple random algorithm for $MAXSAT$

Consider a set of literals:$X = \{x_1, x_2, ..., x_n\} \cup \{\neg x_1, \neg x_2, ..., \neg x_n\}$. A clause over this set is a boolean formula $l_1 \vee l_2 \vee ... \vee l_k$. The **MAXSAT** problem is defined as follows:

- **Input**: A set of clauses $\mathcal{C}$, $|\mathcal{C}| = m$ from a set of variables X;
- **Output**: A boolean assignment to the variables in X that satisfies the maximum number of clauses in $\mathcal{C}$;

It is well known that this problem is NP-hard, and even more, it is NP-hard to approximate within some constant. However some simple approximation algorithms exist.

For example, assigning random boolean values to the literals yields a $(1 - \frac{1}{2^k})$-approximation algorithm for **MAX** $\geq$ k-**SAT**(a version of the **MAX-SAT** where all clauses have size $\geq k$). This can be seen easily, since each clause is satisfied with probability $\geq 1 - \frac{1}{2^k}$. The expected number of clauses to be satisfied, E, is $(1 - \frac{1}{2^k})m$ which is obviously grater or equal to $(1 - \frac{1}{2^k})$ fraction of the optimum. Moreover, if $p = Pr\{\#satisfied\ clauses\ \geq\ (1 - \frac{1}{2^k})m\}$ then $p > \frac{1}{m}$. This follows from the fact that $E \leq pm + ((1-p)(1 - \frac{1}{2}^k)m - 1)$.

Derandomizing the algorithm

Derandomizing using k-way independence properties of modular polynomials

A method that is commonly used in derandomizing algorithms is the following. Fix a prime $p > n > k$, and pick $a_0, a_1, ...a_{k-1}$ at random from $\{1, 2, ..., p-1\}$. Then the modular polynomial $h(X) = a_{k-1}X^{k-1} + ... + a_1 X + a_0 (mod\ p)$ has the following property (k-wise independence): let $x_i = h(i)$ mod 2. Then knowing $x_{i_1}, x_{i_2}, ...x_{i_{k-2}}$, gives no information about the value of $x_{i_{k-1}}$, which is still uniformly distributed (possibly with a slight bias of $1/p$).

The previous algorithm can be derandomized by assigning values to $x_1, ...x_n$ according to this method (using such a polynomial), combined with exhaustive search over the space of such polynomials (space which is of order $O(n^k)$). That the probability of a clause being satisfied is $\geq 1 - 1/2^k$ depends only on the first k variables in the clause, so it is still (approximately) true for the above k-wise independent setting of the variables.

Derandomizing using the method of conditional expectations

For a fixed set of clauses $\mathcal{C}$, for a variable x we denote by $\mathcal{C}_x$ the set of clauses obtained from $\mathcal{C}$ by assigning to x the value true. Similarly, denote by $\mathcal{C}_{\overline{x}}$ the set obtained from $\mathcal{C}$ by assigning to x the value false. Also, denote by $\#_x$ the number of clauses which are satisfied if x is assigned true, and $\#_{\overline{x}}$ the number of clauses that are satisfied if x is assigned false.

We can express the expected number of clauses to be satisfied in two ways: $E(\mathcal{C}) = \sum_{C\in\mathcal{C}}(1 - \frac{1}{2^{|C|}})$, and also $E(\mathcal{C}) = \frac{1}{2}(E(\mathcal{C}_x) + \#_x)) + \frac{1}{2}(E(\mathcal{C}_{\overline{x}}) + \#_{\overline{x}})$.

This leads to the following algorithm:

(1) pick a variable x
(2) compute $E_1 = E(\mathcal{C})$, $E_2 = E(\mathcal{C}_x) + \#_x$,$E_3 = E(\mathcal{C}_{\overline{x}}) + \#_{\overline{x}})$
(3) if $E_1 \leq E_2$ then set x to true else set x to false;
(4) go to (1) if there are unassigned variables.

This deterministic algorithm always achieves at least the expectation, $(1 - 1/2^k)m$ clauses satisfied, because the expected number of satisfied clauses only increases from step to step. But it is still based on a probabilistic analysis.

Linear programming relaxation algorithm for $MAXSAT$; randomized rounding

The idea is to relax the condition of assigning boolean values to the variables, to having the variables mapped to values in [0..1] and then use the assigned values as probabilities to assign 0/1. This leads to the following algorithm:

Use variables $y_{x_1}, y_{x_2}, ..., y_{x_n}$ to replace variables x_i, $1 \leq i \leq n$, and variables $z_1, ..., z_m$ to indicate if a clause has been satisfied. Consider $y_{\overline{x_i}} = 1 - y_{x_i}$. The goal is to find:

$$\mathbf{max} \sum z_i, \text{ subject to } 0 \leq y_x, z_i \leq 1 \text{ for every literal } x \text{ and } 1 \leq i \leq m \text{ and } \sum_{l\in C_j} y_l \geq z_j, \text{ for all } C_j \in \mathcal{C}$$

This defines a linear programming program which is known to be polynomial time solvable. If OPT_r denotes the solution for the relaxed solution and OPT the solution for the unrelaxed problem then, obviously, $OPT_r \geq OPT$. Once we find the solution for the relaxed problem, we find an approximate solution for the unrelaxed version by assigning x_i true with probability y_{x_i}.

Claim Say that C_j is a clause of size k, then $Pr[C_j\ is\ satisfied] \geq z_j(1 - (1 - \frac{1}{k})^k)$.

Proof the clause C_j is not satisfied if each variable is assigned value 0. This happens with probability $\prod_{i=1}^{k}(1 - y_{l_i})$. We have to show that $1 - \prod_{i=1}^{k}(1 - y_{l_i}) \geq z_j(1 - (1 - \frac{1}{k})^k)$. Given that $\sum y_{l_i} \geq z_j$, the minimum of the left hand of the inequality is reached when $y_{l_i} = z_j/k$ for all i, therefore it suffices to show that $1-(1-\frac{z}{k})^k) \geq (1-(1-\frac{1}{k})^k)z$ for all positive integers k and $0 \leq z \leq 1$. Consider the difference of the two functions; it is zero at $z = 0$ and at $z = 1$. If it were negative anywhere in this interval, it would have to have a local minimum z_0, where the first derivitive were 0 and the second derivative positive. But the second derivitive is $-(k-1)/k(1 - \frac{z}{l}k)^{k-2}$ which is negative for $0 < z < 1$ and $k \geq 2$. (The case $k = 1$ is trivial: both sides are z.)

We can combine the first and the last algorithm: a variable x_i is assigned the truth value either either with probability 1/2, or with the probability y_{x_i}. The expected solution is then within 3/4 of the optimal solution. The 3/4 is obtained as follows: First let $\gamma_k = 1 - 1/2^k$ and $\beta_k = 1 - (1 - 1/k)^k$. By calculation, we can show that γ_k and β_k average to at least 3/4 for all k, with equality at $k = 2$, when both are 3/4. Let $Z_k = \sum_{C_j, |C_j|=k} z_j$. Then the one algorithm has expectation at least $\sum_k Z_k \beta_k$ and the other at least $\sum_k Z_k \gamma_k$, so at least one has expectation at least $\sum_k Z_k(\gamma_k + \beta_k)/2 \geq 3/4 \sum_k Z_k = 3/4 \sum_j z_j = 3/4 Opt_R \geq 3/4 Opt$.