ALGORITHMS IN ACTION - SOLVING SAT (RANDOMIZED)

TOMER BINCOVICH

1. Introduction

We present two randomized algorithms for the 3-SAT problem (the PPZ algorithm is more general and solves the k-SAT problem), then state two conjectures about lower bounds.

2. Schöning's algorithm

2.1. **The algorithm.** This algorithm is based on a random walk. It consists of running a "try" procedure (which tries to find a satisfying assignment) s times, and if no run succeeded in finding a satisfying assignment, *unsat* is returned. The algorithm:

```
Walk3SAT(F: clause set):
repeat s times:
    A ← random assignment
    repeat 3n times:
        if A is satisfying return sat
        choose C ∈ F which is not satisfied
        choose uniformly at random a literal l in C
        flip the value of the variable of l in A
return unsat
```

Or, more concisely:

```
Walk3SAT(F: clause set)
repeat s times: "try"
return unsat
```

We remark that if the boolean formula (with the clause set F) is unsatisfiable, the algorithm always returns *unsat*, while if the formula is satisfiable, the algorithm may return *unsat* with some probability. Next we bound this probability.

2.2. **The success probability of a *try*.** Assume the formula is satisfiable. Fix some satisfying assignment A^*, and let A_t be the assignment after t steps of the *try* procedure. Note that A_0 is the initial random assignment. Let $X_t = d(A_t, A^*)$, meaning the number of variables that are given different values according to A_t and A^*. It can be seen that $X_0 \sim Bin\left(n, \frac{1}{2}\right)$. A *try* is successful if $X_t = 0$ for some $t \leq 3n$: it means that $A_t = A^*$, which says we found a satisfying assignment (pay attention that the probability for a *try* to succeed can be higher; here we analyze the probability to find a specific satisfying assignment - A^*, but there can be more the just one).

Based on Lectures by Haim Kaplan and Uri Zwick.

Let us focus on the variable X_t. X_t changes by ∓ 1 each time we change the assignment, and decreases with probability $\geq \frac{1}{3}$ (in every unsatisfied clause, at least 1 out of the 3 variables gets a different value than in A^* - otherwise the clause would be satisfied). However, the *exact* probability depends on the history.

As A^* is a satisfying assignment, there exists for every clause a literal satisfied by A^*. Mark such a literal for each clause. Define the variable Y_t: $Y_0 = X_0$, Y_t changes by -1 if the algorithm picks the marked literal (of the unsatisfied clause C), otherwise it changes by $+1$.

Lemma 1. *$Y_t \geq X_t$.*

Proof. By induction. First we have $Y_0 = X_0$. It remains to show that whenever X_t goes up, so does Y_t (so if we have inductively that $Y_t \geq X_t$, if X_t changes by $+1$ so does Y_t, and otherwise X_t changes by -1, and Y_t cannot go down by more than 1. In each case the inequality holds). But X_t goes up only when we flip a variable whose values in A_t and A^* are the same. Assume to the contrary that Y_t goes down, so we picked the marked literal, and by definition it is satisfied by A^*. But as seen, the chosen variable got the same value as in A^*, which implies that the chosen clause was satisfied - a contradiction. □

Thus, $p = Pr\left[\exists t \leq 3n, Y_t = 0\right] \leq Pr\left[\exists t \leq 3n, X_t = 0\right]$, and we want a lower bound on p. Y_t describes a random walk on the line (of non-negative integers), with probability $\frac{1}{3}$ to go left and $\frac{2}{3}$ to go right.

We wish to bound $p_j = Pr\left[\exists t \leq 3n, Y_t = 0 \,|Y_0 = j\,\right]$. We can compute $q_j = Pr\left[\exists t, Y_t = 0 \,|Y_0 = j\,\right]$, and $p_j \leq q_j$, but we actually need a lower bound:

$$q_0 = 1$$

$$q_1 = \frac{1}{3} + \frac{2}{3}\left(q_1\right)^2 \Rightarrow q_1 = \frac{1}{2}$$

$$6q_j = \frac{1}{3}q_{j-1} + \frac{2}{3}q_{j+1} \Rightarrow q_{j+1} = \frac{3}{2}q_j - \frac{1}{2}q_{j-1} \Rightarrow q_j = \left(\frac{1}{2}\right)^j$$

so $p_j \leq q_j = \left(\frac{1}{2}\right)^j$. We also have, for every k such that $j + 2k \leq 3n$:

$$p_j \geq \binom{j+2k}{k}\left(\frac{2}{3}\right)^k\left(\frac{1}{3}\right)^{j+k}$$

since this is the probability that we have $j + k$ moves left and k move right, in a sequence of $j + 2k$ moves. Now:

$$p_j \geq \binom{3j}{j}\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j}$$

Stirling:

$$\sqrt{2\pi m}\left(\frac{m}{e}\right)^m \leq m! \leq 2\sqrt{2\pi m}\left(\frac{m}{e}\right)^m$$

$$\binom{3j}{j} = \frac{(3j)!}{j!\,(2j)!} \geq \frac{\sqrt{2\pi 3j}\left(\frac{3j}{m}\right)^{3j}}{2\sqrt{2\pi j}\left(\frac{j}{e}\right)^j 2\sqrt{2\pi 2j}\left(\frac{2j}{e}\right)^{2j}} = \frac{\sqrt{3}}{8\sqrt{\pi j}}\left(\frac{27}{4}\right)^j$$

$$p_j \geq \binom{3j}{j}\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j} \geq \frac{c}{\sqrt{j}}\left(\frac{27}{4}\right)^j\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j} = \frac{c}{\sqrt{j}}\left(\frac{1}{2}\right)^j$$

and we get

$$\frac{c}{\sqrt{n}}\left(\frac{1}{2}\right)^j \le \frac{c}{\sqrt{j}}\left(\frac{1}{2}\right)^j \le p_j \le q_j = \left(\frac{1}{2}\right)^j$$

(the leftmost inequality holds even for $j=0$).
Now, we return to p:

$$p = Pr\left[\exists t \le 3n, Y_t = 0\right] = \sum_{j=0}^{n} Pr[Y_0 = j]p_j \ge \sum_{j=0}^{n}\binom{n}{j}\left(\frac{1}{2}\right)^n \frac{c}{\sqrt{n}}\left(\frac{1}{2}\right)^j =$$

$$= \left(\frac{1}{2}\right)^n \frac{c}{\sqrt{n}}\left(\sum_{j=0}^{n}\binom{n}{j}\left(\frac{1}{2}\right)^j\right) = \left(\frac{1}{2}\right)^n \frac{c}{\sqrt{n}}\left(\sum_{j=0}^{n}\binom{n}{j}\left(\frac{1}{2}\right)^j 1^{n-j}\right)$$

$$= \left(\frac{1}{2}\right)^n \frac{c}{\sqrt{n}}\left(\frac{1}{2}+1\right)^n = \left(\frac{1}{2}\right)^n \frac{c}{\sqrt{n}}\left(\frac{3}{2}\right)^n = \frac{c}{\sqrt{n}}\left(\frac{3}{4}\right)^n$$

2.3. **Analysis of the full algorithm.** We set $s = \frac{\alpha}{p}$ (α is a constant; recall the Walk3SAT algorithm in section 2.1), and conclude that if there is a satisfying assignment we fail to find it with probability $\le (1-p)^{\frac{\alpha}{p}} \overbrace{\le}^{1-x\le e^{-x}} e^{-\alpha}$. The running time is $\mathrm{O}^*\left(\frac{1}{p}\right) = \mathrm{O}^*\left(\left(\frac{4}{3}\right)^n\right)$ (i.e. we ignore polynomial factors).

3. PATURI-PUDLAK-ZANE ALGORITHM

3.1. **The algorithm.** This algorithm resembles the SAT solver with unit propagation, but traverses the variables in a random order, and pick a random value when cast with a decision.

```
PPZ(F: clause set):
repeat s times:
    Pick random π ∈ S_n
    x ← ∅ (the assignment)
    for i = 1 to n:
        if (x_π(i)) ∈ F then
            {x_π(i) = 1; F ← F[x_π(i) = 1]}
        else if (x̄_π(i)) ∈ F then
            {x_π(i) = 0; F ← F[x_π(i) = 0]}
        else {pick α ∈ {0,1} at random;
            x_π(i) = α; F ← F[x_π(i) = α]}
    if x is satisfying return sat
return unsat
```

Again, we "try" to find a satisfying assignment s times (restarting after each time), and we will prove a lower bound on the success probability of a *try*.

3.2. **Analysis of the full algorithm.** As before, let p be the probability that a *try* finds a specific satisfying assignment (assuming the formula is satisfiable). We again set $s = \frac{\alpha}{p}$, and remark that we fail to find the assignment when there is one with probability $(1-p)^t \le e^{-\alpha}$.

Theorem 2. *For k-SAT,* $p \ge \left(\frac{1}{2^{1-\frac{1}{k}}}\right)^n$.

So we repeat the *try* procedure $\approx \left(2^{\left(1-\frac{1}{k}\right)}\right)^n$ times. The values for small k are:

k	3	4	5	6	7	8
$2^{1-\frac{1}{k}}$	1.58	1.68	1.74	1.78	1.81	1.83

3.3. **Proof of the theorem - the success probability of a *try*.** Fix a satisfying assignment x. We call a variable *critical* if when we flip its value, the assignment we get (from x) is no longer satisfying. Let $j(x)$ be the number of *critical vars*, and $s(x) = n - j(x)$.

Lemma 3. $\sum_{x \in sat} \frac{1}{2^{s(x)}} \geq 1$.

Proof. Induction on n (the number of vars).

Base: $n = 1$.

Case 1: Only one satisfying assignment x. Then flipping the value of the only variable leads to a non-satisfying assignment, so $j(x) = 1$, $s(x) = 0$.

Case 2: Two satisfying assignments x^1, x^2, so the value of the variable does not matter, and

$$j\left(x^1\right) = j\left(x^2\right) = 0$$

$$s\left(x^1\right) = s\left(x^2\right) = 1$$

Induction step: split the satisfying assignments into two sets

$$sat_0 = \{x \in S \,| x_n = 0\}$$

$$sat_1 = \{x \in S \,| x_n = 1\}$$

Case 1: $sat_0 = \emptyset$ (the case $sat_1 = \emptyset$ is analogous). There is a 1-1 correspondence between assignments x and assignments x' of $F[x_n = 1]$ (x_n must be 0 in x), x_n is critical in x so $s_{F[x_n=1]}\left(x'\right) = s(x)$. Apply the induction hypothesis to $F[x_n = 1]$.

Case 2: $sat_0 \neq \emptyset$ and $sat_1 \neq \emptyset$. There is a 1-1 correspondence between assignments $x \in sat_0$ and assignments x' of $F[x_n = 0]$, so $s_{F[x_n=0]}(x') \geq s(x) - 1$ (as x_n may not be critical in x). Similarly, there is a 1-1 correspondence between assignments $x \in sat_1$ and assignments x' of $F[x_n = 1]$, so $s_{F[x_n=1]}(x') \geq s(x) - 1$. Thus

$$\begin{aligned}\sum_{x \in sat} \frac{1}{2^{s(x)}} &= \sum_{x \in sat_0} \frac{1}{2^{s(x)}} + \sum_{x \in sat_1} \frac{1}{2^{s(x)}} \\ &\geq \sum_{x' \in sat(F[x_n=0])} \frac{1}{2^{s_{F[x_n=0]}(x')+1}} \\ &+ \sum_{x' \in sat(F[x_n=1])} \frac{1}{2^{s_{F[x_n=1]}(x')+1}} \geq \frac{1}{2} + \frac{1}{2} = 1\end{aligned}$$

□

In addition, let $r(x, \pi) \leq j(x)$ be the number of critical vars that are last in some *critical clause* (a clause that becomes unsatisfied when we flip the value of a critical var) by π (i.e. vars $x_{\pi(i)}$ such that when the algorithm gets to $x_{\pi(i)}$ its value is forced to be the correct value by unit propagation). Observe that the only way that we find x when using π is to guess correctly the values for the vars which are not counted in $r(x,\pi)$, and the algorithms is then forced to set the other values correctly. Thus

$$P[\text{Alg finds } x \text{ when using } \pi] = \frac{1}{2^{n-r(x,\pi)}}$$

$$\begin{aligned} P[\text{Alg finds } x] &= \sum_{\pi} P[\text{Alg finds } x \text{ when using } \pi]\frac{1}{n!} \\ &= \sum_{\pi} \frac{1}{2^{n-r(x,\pi)}}\frac{1}{n!} = \frac{1}{2^n}\sum_{\pi}\frac{1}{n!}2^{r(x,\pi)} = \frac{1}{2^n}E\left(2^{r(x,\pi)}\right) \\ &\geq \frac{1}{2^n}2^{E(r(x,\pi))} \end{aligned}$$

The last inequality is due to Jensen's inequality (as the function 2^x is convex).

$$\begin{aligned} E(r(x,\pi)) &= \sum_{x_i \text{ critical}} P(x_i \text{ last in critical clause in } \pi) \\ &\geq \sum_{x_i \text{ critical}} \frac{1}{k} = \frac{j(x)}{k} \end{aligned}$$

And

$$\begin{aligned} P[\text{Alg finds } x] &\geq \frac{1}{2^n}2^{E(r(x,\pi))} \geq \frac{1}{2^n}2^{\frac{j(x)}{k}} \\ &\geq \frac{1}{2^{n-\frac{n}{k}}}2^{\frac{j(x)}{k}-\frac{n}{k}} \\ &\geq \left(\frac{1}{2^{1-\frac{1}{k}}}\right)^n \frac{1}{2^{\frac{s(x)}{k}}} \geq \left(\frac{1}{2^{1-\frac{1}{k}}}\right)^n \frac{1}{2^{s(x)}} \end{aligned}$$

Finally

$$\begin{aligned} p = \sum_{x\in sat} P[\text{Alg finds } x] &\geq \left(\frac{1}{2^{1-\frac{1}{k}}}\right)^n \sum_{x\in sat}\frac{1}{2^{s(x)}} \\ &\geq \left(\frac{1}{2^{1-\frac{1}{k}}}\right)^n \end{aligned}$$

$\square$

3.4. **Summary.** For 3-SAT we get running time of 1.58^n. It was improved (PPSZ) to 1.36^n, and the current record is 1.308^n. The first algorithm we saw (section 2.1) whose running time is about 1.33^n beats PPZ.

4. ETH AND SETH

Two famous conjectures that capture the following beliefs (Exponential Time Hypothesis and its Strong variant):

(ETH) There is no algorithm for 3-SAT that runs in $2^{o(n)}$ time
(SETH) There is no algorithm for SAT that runs in $(2-\varepsilon)^n$ time

It can be shown that SETH implies ETH, and those conjectures have been used to derive many (conditional) lower bounds.