CSE 632: Analysis of Algorithms II: Randomized Algorithms **Spring 2024**

Lecture 18: 2-SAT

Lecturer: Zongchen Chen

1 2-SAT

Consider the 2-SAT problem. Given a Boolean formula f in CNF where every clause contains exactly two literals, our goal is to find a satisfying assignment for f.

Recall that 2-SAT can be solved in polynomial time via a reduction to strongly connected components of directed graphs. Here we present a simple randomized algorithm. Algorithm 1 applies to the general SAT problem and we shall analyze it for 2-SAT.

Algorithm 1 Randomized algorithm for SAT

```
Input: σ initial assignment
1: repeat T times
2:     Choose an arbitrary unsatisfied clause c
3:     Choose a literal in c u.a.r.
4:     Flip the truth value of the chosen variable in σ (to satisfy c)
5:     if σ is a satisfying assignment then
6:         return σ & halt
7:     end if
8: end
9: return Unsatisfiable
```

Remark 1.

1. We can pick an arbitrary assignment σ as initialization, e.g., taking the all-true or all-false assignment. Alternatively, we can apply random initialization: start with a uniformly random assignment σ.
2. In Line 2, we can choose the unsatisfied clause c arbitrarily, e.g., picking the one of the smallest index, or choosing one uniformly at random.
3. In Line 3, we have to choose a literal uniformly at random for our analysis to work.

If f is unsatisfiable, then clearly Algorithm 1 will output Unsatisfiable. Suppose f is satisfiable, and let τ be a satisfying assignment. Define σ_t to be the assignment at time t of Algorithm 1, and X_t to be the number of variables that agree between σ_t and τ. If $X_t = n$, then $\sigma_t = \tau$ and the algorithm finds a satisfying assignment. Note that $X_t \in \{0, 1, \dots, n\}$ can be thought of as a random walk moving between adjacent integers where either $X_{t+1} = X_t + 1$ or $X_{t+1} = X_t - 1$.

Claim 2. *For each $t < T$ and $i \in \{0, 1, \dots, n-1\}$, we have*

$$\Pr\left(X_{t+1} = i + 1 \mid X_t = i\right) \geq \frac{1}{2}.$$

Proof. Suppose we pick clause c in the update from σ_t to σ_{t+1}, and the two variables in c are x_1 and x_2 without loss of generality. Then, c is not satisfied by σ_t, but is satisfied by τ. This means that $\sigma_t(x_j) \neq \tau(x_j)$ for at least one $j \in \{1, 2\}$. If $\sigma_t(x_1) \neq \tau(x_1)$ and $\sigma_t(x_2) = \tau(x_2)$, then $X_{t+1} = X_t + 1$ with probability $1/2$. Similarly for the case $\sigma_t(x_1) = \tau(x_1)$ and $\sigma_t(x_2) \neq \tau(x_2)$. If $\sigma_t(x_1) \neq \tau(x_1)$ and $\sigma_t(x_2) \neq \tau(x_2)$, then $X_{t+1} = X_t + 1$ always. The claim then follows. □

We want to show that the number of steps for X_t to reach n (in which case $\sigma_t = \tau$) is $\mathrm{poly}(n)$ in expectation. Consider a slowed-down process (Y_t) where $Y_0 = X_0$ and

$$\Pr\left(Y_{t+1} = i+1 \mid Y_t = i\right) = \frac{1}{2}.$$

So, (Y_t) is an unbiased random walk on $\{0, 1, \dots, n\}$. It suffices to show (via a simple coupling argument) that the number of steps for Y_t to reach n is $\mathrm{poly}(n)$ in expectation.

Define H_j to be the number of steps for the random walk (Y_t) to reach n when starting at $Y_0 = j$. Let $h_j = \mathbb{E}[H_j]$ be its expectation.

Lemma 3. *For all $j \in \{0, 1, \dots, n\}$, we have*

$$h_j = n^2 - j^2.$$

In particular, $h_j \le h_0 = n^2$.

Proof. By definition, we have the following recurrence:

$$\begin{cases} h_0 = h_1 + 1; \\ h_j = \frac{1}{2} h_{j-1} + \frac{1}{2} h_{j+1} + 1, \quad 1 \le j \le n-1; \\ h_n = 0. \end{cases}$$

Thus, we have

$$h_j - h_{j+1} = h_{j-1} - h_j + 2 = h_{j-2} - h_{j-1} + 4 = \cdots = h_0 - h_1 + 2j = 2j + 1,$$

and

$$h_j = h_j - h_n = \sum_{i=j}^{n-1} h_i - h_{i+1} = \sum_{i=j}^{n-1} 2i + 1 = n^2 - j^2,$$

as claimed. □

If we run Algorithm 1 with $T = 2n^2$ rounds, then it holds

$$\begin{aligned} & \Pr\left(\text{Algorithm 1 outputs Unsatisfiable}\right) \\ \le\ & \Pr\left(X_t \text{ does not reach } n \text{ for } t \le 2n^2\right) \\ \le\ & \Pr\left(Y_t \text{ does not reach } n \text{ for } t \le 2n^2\right) \\ \le\ & \Pr\left(H_0 > 2n^2\right) \\ \le\ & \frac{h_0}{2n^2} = \frac{1}{2}. \qquad \text{(Markov's Inequality)} \end{aligned}$$

2 3-SAT

In the 3-SAT problem, we are given a Boolean formula f in CNF where every clause contains exactly three literals, and our goal is to find a satisfying assignment for f. Recall that 3-SAT is NP-complete. Moreover, the Exponential Time Hypothesis (ETH) states that 3-SAT cannot be solved in $2^{o(n)}$ time.

The trivial algorithm for 3-SAT is to enumerate all 2^n assignments and check for each of them if it is satisfying or not. The running time of such a brute-force algorithm is $2^n \mathrm{poly}(n)$. Our goal is to obtain a faster algorithm running in time $a^n \mathrm{poly}(n)$ for some $a < 2$ as small as possible. In fact, we show that Algorithm 1 with *random initialization* can achieve $a = 4/3$ for 3-SAT.

As before, we assume f is satisfiable and let τ be a satisfying assignment. Define σ_t to be the assignment at time t of Algorithm 1, and X_t to be the number of variables that agree between σ_t and τ. Analogously to Claim 2, we have for each $t < T$ and $i \in \{0, 1, \ldots, n-1\}$ that

$$\Pr\left(X_{t+1} = i+1 \mid X_t = i\right) \geq \frac{1}{3}.$$

The slowed-down version (Y_t) is then defined as $Y_0 = X_0$ and

$$\Pr\left(Y_{t+1} = i+1 \mid Y_t = i\right) = \frac{1}{3}.$$

If h_j denotes the expected number of steps for (Y_t) to reach n starting at $Y_0 = j$, then we have the recurrence

$$\begin{cases} h_0 = h_1 + 1; \\ h_j = \frac{2}{3} h_{j-1} + \frac{1}{3} h_{j+1} + 1, \quad 1 \leq j \leq n-1; \\ h_n = 0. \end{cases}$$

Solving the recurrence gives

$$h_j = 2^{n+2} - 2^{j+2} - 3(n-j).$$

Note that $h_0 = 2^{n+2} - 4 - 3n = \Theta(2^n)$ (for worst-case initialization) and $h_{n/2} = 2^{n+2} - 2^{n/2+2} - 3n/2 = \Theta(2^n)$ (for random initialization). Therefore, the previous argument for 2-SAT could only give a $2^n \mathrm{poly}(n)$ time algorithm for 3-SAT. In fact, even for $j = n-1$ we have $h_{n-1} = 2^{n+2} - 2^{n+1} - 3 = 2^{n+1} - 3 = \Theta(2^n)$; namely, even if we start with an assignment σ_0 which differ from τ at only one variable, the number of steps to reach τ could be $\Theta(2^n)$ in expectation. However, the probability of reaching τ in the first step is at least $1/3$; that is, $\Pr\left(Y_1 = n \mid Y_0 = n-1\right) = 1/3$. This indicates that, instead of analyzing the expected number of steps to reach n (and then applying the Markov's inequality), we should look at directly the probability of reaching n within T steps.

The following fact is helpful to us.

Fact 4. *For any $n \in \mathbb{N}$ and $\alpha \in (0,1)$ such that αn is an integer, we have*

$$\binom{n}{\alpha n} \geq \frac{1}{n+1} \left(\frac{1}{\alpha}\right)^{\alpha n} \left(\frac{1}{1-\alpha}\right)^{(1-\alpha)n}.$$

We set $T = 3n$ in Algorithm 1. For any $1 \leq j \leq n$, we deduce that

$$\begin{aligned} & \Pr\left(Y_t \text{ reaches } n \text{ for } t \leq 3n \mid Y_0 = n-j\right) \\ \geq\ & \Pr\left(\substack{\text{in the first } 3j \text{ steps, } 2j \text{ of them} \\ \text{are ``+1'' and } j \text{ of them are ``}-1\text{''}} \mid Y_0 = n-j\right) \\ =\ & \binom{3j}{j} \left(\frac{1}{3}\right)^{2j} \left(\frac{2}{3}\right)^{j} \\ \geq\ & \frac{1}{3j+1} (3)^j \left(\frac{3}{2}\right)^{2j} \left(\frac{1}{3}\right)^{2j} \left(\frac{2}{3}\right)^{j} && \text{(Fact 4)} \\ \geq\ & \frac{1}{4n} \left(3 \cdot \frac{3^2}{2^2} \cdot \frac{1}{3^2} \cdot \frac{2}{3}\right)^j \\ =\ & \frac{1}{4n} \cdot \frac{1}{2^j}. \end{aligned}$$

Recall that in random initialization, we pick the initial assignment σ_0 uniformly at random. Hence, $X_0 = Y_0$, the number of variables that agree in σ_0 and τ, is a binomial random variable with parameters n and $1/2$. We then deduce that

$$
\begin{aligned}
& \Pr(\text{Algorithm 1 finds a satisfying assignment}) \\
\geq\ & \Pr(X_t \text{ reaches } n \text{ for } t \leq 3n) \\
\geq\ & \Pr(Y_t \text{ reaches } n \text{ for } t \leq 3n) \\
=\ & \sum_{j=0}^{n} \Pr(Y_0 = n - j) \Pr(Y_t \text{ reaches } n \text{ for } t \leq 3n \mid Y_0 = n - j) \\
\geq\ & \sum_{j=0}^{n} \frac{\binom{n}{j}}{2^n} \cdot \frac{1}{4n} \cdot \frac{1}{2^j} \\
=\ & \frac{1}{4n} \cdot \frac{1}{2^n} \sum_{j=0}^{n} \binom{n}{j} \frac{1}{2^j} \\
=\ & \frac{1}{4n} \cdot \frac{1}{2^n} \left(1 + \frac{1}{2}\right)^n \\
=\ & \frac{1}{4n} \left(\frac{3}{4}\right)^n.
\end{aligned}
$$

Therefore, repeating Algorithm 1 for $(\frac{4}{3})^n \text{poly}(n)$ times allows us to find a satisfying assignment with probability at least $1/2$. The overall running time is $(\frac{4}{3})^n \text{poly}(n)$, which is much better than 2^n.