# Randomized Algorithms

## Lecture 12: 2-SAT Randomized Algorithm

Raul Garcia-Patron

School of Informatics
University of Edinburgh

# Logical Formulae and the "satisfiability" question

## Definition

- Propositional logical variables $x_1, \ldots, x_n$ for varying $n$.
- A *literal* is any expression which is either $x_i$ or $\bar{x}_i$, for some $i \in [n]$.
- A *clause* is any *disjunction* of a number of literals (ex: $x_i \vee x_j$).
- We say a propositional formula $\phi : \{0,1\}^n \to \{0,1\}$ is in *Conjunctive Normal Form (CNF)* if it is of the form

$$C_1 \wedge C_2 \ldots \wedge C_h,$$

  where every $C_j$ is a *clause*.
- The formula $\phi : \{0,1\}^n \to \{0,1\}$ is in *$k$-CNF* if it is in CNF and every clause contains *exactly* $k$ literals.

The *SAT problem*, *$k$-SAT problem* is the problem of examining a given CNF (or $k$-CNF) expression and deciding whether or not it has a *satisfying assignment.*

# Examples of SAT, *k*-SAT

Example of a SAT question:

$$(x_1 \vee x_8 \vee \bar{x_6}) \wedge (\bar{x_4} \vee \bar{x_7}) \wedge (x_5 \vee x_7 \vee x_4 \vee x_2).$$

- For the formula above, easy to see there is a satisfying assignment (any with $x_1 = 1, x_4 = 0, x_2 = 1$ would do).
- In general, the *SAT* problem is NP-complete (we believe there is no polynomial-time algorithm).

Example of a 2-SAT question:

$$(x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee \bar{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \bar{x_3}) \wedge (x_4 \vee \bar{x_1}).$$

- There is a *polynomial-time* algorithm (either *randomized*, as we see today, or *deterministic*) to solve 2-SAT.
- The 3-SAT problem, and *k*-SAT for all $k > 3$, are all NP-complete.

# 2-SAT Randomized Algorithm

We will design a simple *randomized algorithm* for 2-SAT, and analyse its performance by analogy to a *Markov chain*.

**Algorithm** $\text{2SATR}\textsc{andom}(n; C_1 \wedge C_2 \wedge \ldots \wedge C_\ell)$

1. Assign *arbitrary* values to each of the $x_i$ variables.
2. $t \leftarrow 0$
3. **while** ($t < 2mn^2$ **and** some clause is unsatisfied) **do**
4. Choose an *arbitrary* $C_h$ from all unsatisfied clauses;
5. Choose one of the 2 literals in $C_h$ *uniformly at random* and flip the value of its variable;
6. **if** (we end with a satisfying assignment) **then**
7. **return** this assignment to the $x_1, \ldots x_n$ **else**
8. **return** FAILED.

# 2-SAT Randomized Algorithm

Imagine Algorithm 2SATRANDOM running on our 2SAT example, with the initial assignment being $x_i = 0$ for all $i \in [n]$.

$$(x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee \bar{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \bar{x_3}) \wedge (x_4 \vee \bar{x_1}).$$

- Then $(x_1 \vee x_2)$ is the sole unsatisfied clause.
- Flipping the value of $x_2$ (say) from 0 to 1, will ensure that $(x_1 \vee x_2)$ now becomes satisfied.
- However, making this flip would *also* change the assignment for $(x_1 \vee \bar{x_2})$, making this clause now *unsatisfied*.This is a balanced consequence overall (number of satisfied clauses stays the same).
- However, there are examples where a flip might end up violating **many** clauses. So it's not so helpful for us to use "number of clauses satisfied" as our measure of progress.

# 2-SAT Randomized Algorithm - Analysis

Consider an (unknown so far) satisfying assignment $S \in \{0, 1\}^n$ that makes our 2SAT formula $\phi$ true (satisfies all the clauses).

Our "measure of progress" will be *the number of indices k such that* $x_k = S_k$, $(x_1, \ldots, x_n)$ being the current assignment.

We will analyse the *expected number of steps* before $(x_1, \ldots, x_n)$ becomes $S$.

- Note that if $\phi$ does not have any satisfying assignment, Algorithm 2SATRANDOM always returns FAILED (as it should).
- Let's first assume that the formula $\phi$ has some satisfying assignment.
- $(x_1^t, \ldots, x_n^t)$ is the assignment at time step $t$.

# 2-SAT Randomized Algorithm - Analysis (II)

To analyse the behaviour of Algorithm 2SATRANDOM when given a 2CNF formula $\phi$ that *is* satisfiable, we need some definitions.

## Definition

- Let $S$ be some satisfying assignment for $\phi$.
- $(x_1^t, \ldots, x_n^t)$ the logical variables after the $t$-th iteration.
- Let $X_t$ denote the number of variables of the assignment $(x_1^t, \ldots, x_n^t)$ having the same value as in $S$.

We work with the $X_t$ variable mainly, and bound the time before it reaches the value $n$.

# 2-SAT Randomized Algorithm - Analysis (III)

Some observations:

- If $X_t$ ever hits the value 0, and $\phi$ is not yet satisfied, we are guaranteed that at the next step, $X_{t+1} = 1$. $X_t = 0$ means all bits are different (Hamming weight $= n$).
- Alternatively, suppose $X_t = j$ for some value $j \in \{1, \ldots, n-1\}$ and that $\phi$ is unsatisfied.

1. On an unsatisfied clauses, current assignment $x^t$ must differ from $S$ on *at least one* of the two variables.
2. *Probability at least 1/2* we increase the value of $X_t$ by 1:
   - Case I: one is different, we get the right one with $p = 1/2$.
   - Case II: both are different, success with $p = 1$.
3. Probability at most $1/2$ decrease the value of $X_t$ by 1: chose the one that is equal.

$$\Pr[X_{t+1} = j + 1 \mid ((X_t = j) \,\&\, \phi \text{ not-sat})] \;\geq 1/2;$$
$$\Pr[X_{t+1} = j - 1 \mid ((X_t = j) \,\&\, \phi \text{ not-sat})] \;\leq 1/2.$$

# 2-SAT Randomized Algorithm - Analysis (IV)

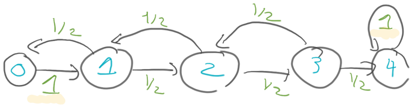
We want to imagine the *progress of* 2SATRANDOM as a Markov chain on the states $0, 1, \ldots, n$. Our concern is bounding the *expected number of steps for $X_t$ to hit the state $n$* (from an *arbitrary* starting point).

Markov chains should be *memoryless*, and this is problematic:

- The value for $\Pr[X_{t+1} = j + 1 \mid ((X_t = j) \ \& \ \phi \text{ not-sat})]$ could depend on whether $x_t$ and $S$ disagree on one or two variables i the unsatisfied clause chosen.
- This may depend on the clauses that have been consider in the past.

# 2-SAT Randomized Algorithm - Analysis

We choose to “tweak” the probabilities and study the process on $\{0, 1, 2, \ldots, n\}$ defined by the variable $Y_t$:




Consider the Markov chain $Y_0, Y_1, \ldots, Y_t, \ldots$ such that

$$
\begin{aligned}
Y_0 &= X_0; \\
\Pr[Y_{t+1} = 1 \mid ((Y_t = 0) \,\&\, \phi \text{ not-sat})] &= 1; \\
\Pr[Y_{t+1} = j+1 \mid ((Y_t = j) \,\&\, \phi \text{ not-sat})] &= 1/2; \\
\Pr[Y_{t+1} = j-1 \mid ((Y_t = j) \,\&\, \phi \text{ not-sat})] &= 1/2.
\end{aligned}
$$

Clearly the *expected number of steps for $X_t$ to hit $n$* is $\leq$ that for $Y_t$.

# 2-SAT Randomized Algorithm - Analysis

For any $j = 0, \ldots, n-1$, define $h_j$ to be the *expected number of steps to hit n starting from j*.

- Clearly, the expected number of steps for 2SATRANDOM to find a satisfying assignment is *at most* $\max_j h_j$ (may well be better).
- We will bound $h_j$ for every $j = 0, 1, \ldots, n$.

# 2-SAT Randomized Algorithm - Analysis

We have $h_n = 0$ and $h_0 = h_1 + 1$ for the "end cases".

- We will use $Z_j$, for $0, 1, \ldots, n-1$, to be the random variable for the "number of steps" to reach $n$ from $j$ ($h_j$ will be $\mathrm{E}[Z_j]$).
- For $j = 1, \ldots, n-1$, recalling the steps of the "random walk", and using linearity of expectation:

$$\begin{aligned} \mathrm{E}[Z_j] &= \frac{1}{2}(\mathrm{E}[Z_{j-1}] + 1) + \frac{1}{2}(\mathrm{E}[Z_{j+1}] + 1), \\ h_j &= \frac{1}{2}(h_{j+1} + 1 + h_{j-1} + 1) \end{aligned}$$

This gives us the following system of equations:

$$h_j = \frac{h_{j-1} + h_{j+1}}{2} + 1 \qquad \text{for } j = 1, \ldots, n-1 \tag{1}$$

Leads to: $h_j = h_{j+1} + 2j + 1$.

# 2-SAT Randomized Algorithm - Analysis

We show by induction that for $j = 0, \ldots, n-1$,

$$h_j = h_{j+1} + 2j + 1.$$

## Proof.

Base case: If $j = 0$ we have $h_0 = h_1 + 1$ that was shown to be true.
Inductive step: Suppose this was true for $j = k-1$ (we had $h_{k-1} = h_k + 2(k-1) + 1$, this is our (IH)). Now consider $j = k$.
By the “middle case” of our system of equations,

$$\begin{aligned} h_k &= \frac{h_{k-1} + h_{k+1}}{2} + 1 \\ &= \frac{h_k + 2(k-1) + 1}{2} + \frac{h_{k+1}}{2} + 1 \qquad \text{by our (IH)} \\ &= \frac{h_k}{2} + \frac{h_{k+1}}{2} + \frac{2k+1}{2} \end{aligned}$$

Subtracting $\frac{h_k}{2}$ from each side, this is equivalent to

$$h_k = h_{k+1} + 2k + 1,$$

as claimed.

# 2-SAT Randomized Algorithm - Analysis

## Lemma (Lemma 7.1)

*Assume that the given 2CNF formula has a satisfying assignment, and that* 2SATRANDOM *is allowed to carry out as many iterations as it wants to find a satisfying assignment. Then the expected number of iterations to find that assignment is at most $n^2$.*

## Proof.

We showed that the expected number of iterations is at most $\max_{j=0,\dots,n-1}\{h_j\}$. We now know the max is $h_0$.
Applying $h_k = h_{k+1} + 2k + 1$ iteratively, we have

$$
\begin{aligned}
h_0 &= \sum_{k=0}^{n-1}(2k+1) + h_n \\
&= 2\sum_{k=0}^{n-1} k + n + 0 \\
&= 2\frac{(n-1)n}{2} + n \;=\; n^2.
\end{aligned}
$$

# Probability of failure

## Theorem

*The algorithm* 2SATRANDOM *perform up to* $2mn^2$ *iterations of the while loop. Then, when there is a satisfying assignment for* $\phi$*, the probability that* 2SATRANDOM *does not discover one, is at most* $2^{-m}$*.*

## Proof.

1. Modify the algorithm to be run $m$ times in parallel over “blocks" of $2n^2$ size.
2. Markov inequality guarantees a failure of $1/2$ for $2\mathrm{E}[Z_0] = 2n^2$ iterations per block: $P(Z_0 > a) \leq \frac{\mathrm{E}[Z_0]}{2}$, choose $a = 2\mathrm{E}[Z_0]$.
3. If one of the $m$ repetition succeeds we find the solution. We get failure overall only if all the $m$ blocks fail, i.e., $P_f = (1/2)^m = 2^{-m}$.

□

The algorithm 2SATRANDOM run the $2mn^2$ in a single loop, but this can only reach the solution faster: instead of imputing $m$ independent input to each block, we can feed one block with the output of the previous one.