A randomized algorithm for Satisfiability (SAT)

Boolea formula in conjunctive normal form (CNF)

e.g. \( E = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor \overline{x_3}) \land (x_2 \lor \overline{x_3}) \)

\( \text{clause} \)

\( \text{literals} \)

\( x_i \text{-variables} \)

SAT: Given such a formula, \( n \) variables, \( m \) clauses

can we assign T/F to variables to satisfy the formula

i.e. make all clauses True.

e.g. for above \( x_1 = \text{T} \) \( x_3 = \text{F} \) works.

3-SAT — all clauses have 3 distinct literals — NP-complete

2-SAT — poly time, in fact \( O(n) \).

Applications

— AI

— quantified Boolean formulas

 SAT is case of one (implicit) \( \exists \) quantifier.

Techniques for SAT

— heuristics, "resolution".

— can reduce worst case run time from obvious \( O(2^n \text{poly}(n,m)) \)

to \( O(1.5^n) \) for 3-SAT (not obvious).

What about randomized algorithms?

unlikely to get randomized poly time (this would imply rand. poly. time for all problems in NP).

We will show we can beat best det. alg. for 3-SAT

(Actually we will mostly concentrate on 2-SAT)
SAT Alg. [Papadimitriou’91]
input: Boolean formula E in CNF
idea: local improvement (“hill climbing”) 
start with any T/F assignment A,
repeat t times (t to be chosen)
if A satisfies E return YES
pick an unsatisfied clause C
randomly pick a literal x in C
flip x’s value.
end
return NO (maybe)
We need to analyze error prob. and choose t.
If E is not satisfiable, the alg. outputs NO, so correct.
Bad situation is when E is satisfiable but we return NO.
So suppose A* is a truth-value assign. that satisfies E.
Let i = # variables with same value in A and A*.
If i reaches n then A=A* and alg. outputs YES.
How does i change? Goes up by 1 or down by 1.
We will analyze this (depending on 2-SAT vs 3-SAT vs…)
First just look at random walk on line

```
0 ↓ 1
\  /  \
H i H
```

Start at i. At each step, move right (to i+1) w/ Prob. $\frac{1}{2}$
move left (to i-1) w/ Prob. $\frac{1}{2}$
Except at 0, always go right.
What is expected # steps to get to n?
Equivalent to a Markov chain

finite automaton with probabilities

Expected number of steps to get from \( i \) to \( n - 1 \): \( t_i \)

\[
t_n = 0 \\
t_0 = 1 + t_1 \\
t_i = 1 + \frac{1}{2} t_{i-1} + \frac{1}{2} t_{i+1}, \quad 1 \leq i \leq n-1
\]

Awkward for induction.

But differences are nice

\[
\frac{t_i - t_{i+1}}{d_i} = 2 + \left( \frac{t_{i-1} - t_i}{d_{i-1}} \right)
\]

\[
d_0 = t_0 - t_1 = 1 \\
d_i = 2 + d_{i-1}
\]

so \( d_i = 1 + 2i \)

plug back in for \( t_i \)

\[
t_i = d_i + t_{i+1} \\
t_n = 0 \\
t_i = \sum_{j=i}^{n-1} d_j = \sum_{j=i}^{n-1} (1 + 2j) = (n-i) + 2 \sum_{j=i}^{n-1} j = (n-i) + n(n+1) - i(i+1)
\]

\[
= i(i) + n^2 - n - i^2 + 1
\]

\[
= n^2 - i^2
\]

Max is \( t_0 = n^2 \) and \( t_i \leq n^2 \)

end of random walk on line.
Back to Papadimitriou's Alg.

Claim: For 2-SAT we get random walk on line with Prob ≥ \( \frac{1}{2} \) of moving right (from \( i \) to \( i+1 \)).

Suppose clause \( C \) is not satisfied.

\[ C = (\alpha \lor \beta) \]

in \( A \), both \( \alpha \) and \( \beta \) are false.

in \( A^* \) at least one is true — say \( \alpha \).

Alg. picks \( \alpha \) to flip with prob. \( \frac{1}{2} \Rightarrow i \) increases.

Alg. picks \( \beta \) to flip with prob. \( \frac{1}{2} \Rightarrow i \) might go up or down.

So \( \Pr [i \text{ increases}] \geq \frac{1}{2} \)

Note: if \( \Pr [i \text{ increases}] > \frac{1}{2} \) we just get to \( n \) faster.

So expected # steps to reach \( A^* \) is \( \leq n^2 \).

Note: of course the alg. might find a satisfying truth-value assign. different from \( A^* \), and stop earlier.

So how many repeats? value of \( t \) ?

We can use Markov's inequality

If \( X \geq 0 \) and \( E[X] = \mu \)

then \( \Pr [X \geq c \cdot \mu] \leq \frac{1}{c} \). \( c \) constant.

In our case \( \mu = n^2 \). Choose \( c = 2 \)

\( \Pr [\# \text{ steps} > 2n^2] < \frac{1}{2} \)

So set \( t = 2n^2 \). Then \( \Pr [\text{error}] < \frac{1}{2} \)

# trials \( O(n^2) \). Runtime \( O(n^2 \cdot \text{poly}(n,m)) \)

time to keep track of T/F clauses and flip one literal.
Turning to 3-SAT.

For clause \( C = (\alpha \lor \beta \lor \gamma) \)

If \( A \) does not satisfy \( C \) but \( A^* \) does — say with \( \alpha = T \)

\[ \Pr[ \text{alg. flips } \alpha ] = \frac{1}{3} \]

So \( \Pr[ \text{i increases} ] \geq \frac{1}{3} \)

If we analyze random walk on line \( i = \# \text{ variables same in } A, A^* \)

\[ \Pr[ \text{i goes to } i+1 ] = \frac{1}{3} \]

\[ \Pr[ \text{i goes to } i-1 ] = \frac{2}{3} \]

expected # steps to reach \( n \) is \( \sim 2^n \) (we won’t do analysis)

Schöning ‘99 gave 2 ideas for improvements:

1. Start with random truth-value assignment \( A \)

2. Increasing # trials (t value) is not helpful.
   If we’ve taken many steps without reaching \( A^* \)
   we are likely stuck near \( 0 \) — so pick new random \( A \).
Schöning's alg.
repeat S times
  randomly pick A (truth value assign).
  repeat t = 3n times
    if A satisfies E output YES
    else pick unsatisfied clause C
      randomly pick variable in C
      flip its value.
end

Fact: in inner loop $\Pr[\text{error}] \leq 1 - \left(\frac{3}{4}\right)^n$

Set $S = c \cdot \left(\frac{4}{3}\right)^n$

$\Pr[\text{error}] \leq \left(1 - \left(\frac{3}{4}\right)^n\right)^c \cdot \left(\frac{4}{3}\right)^n$

From calculus $\left(1 - \frac{1}{x}\right)^{cx} \leq \frac{1}{e}$ (from $\frac{1}{ex} \geq 1 - xe$)

So $\Pr[\text{error}] \leq \frac{1}{ec}$

Bottom line: get $\Pr[\text{error}] \leq \frac{1}{2}$
with # steps $O\left(\left(\frac{4}{3}\right)^n, n\right)$

run time $O\left((\frac{4}{3})^n \text{ poly}(n, m^5)\right)$

$\approx O(1.33^n, n)$

best deterministic is $O(1.465^n)$

exponential but better than best known non-randomized alg.