Algorithms for 3-SAT

Exposition by William Gasarch

Credit Where Credit is Due

This talk is based on Chapters 4,5,6 of the AWESOME book

The Satisfiability Problem SAT, Algorithms and Analyzes
by
Uwe Schoning and Jacobo Torán

What is 3SAT?

Definition: A Boolean formula is in *3CNF* if it is of the form

$$C_1 \wedge C_2 \wedge \cdots \wedge C_k$$

where each C_i is an $\vee$ of three or less literals.

Definition: A Boolean formula is in *3SAT* if it in 3CNF form and is also SATisfiable.

BILL- Do examples and counterexamples on the board.

Why Do We Care About 3SAT?

1. 3SAT is NP-complete.
2. ALL NPC problems can be coded into SAT. (Some directly like 3COL.)

OUR GOAL

1. Will we show that 3SAT is in P?

OUR GOAL

1. Will we show that 3SAT is in P?

   NO.

OUR GOAL

1. Will we show that 3SAT is in P?

   NO.

   Too bad.

OUR GOAL

1. Will we show that 3SAT is in P?

   NO.

   Too bad.

   If we had \$1,000,000 then we wouldn't have to worry about whether the REU grant gets renewed.

OUR GOAL

1. Will we show that 3SAT is in P?

   NO.

   Too bad.

   If we had $1,000,000 then we wouldn't have to worry about whether the REU grant gets renewed.

2. We will show algorithms for 3SAT that
   2.1 Run in time $O(\alpha^n)$ for various $\alpha < 1$. Some will be randomized algorithms. NOTE: By $O(\alpha^n)$ we really mean $O(p(n)\alpha^n)$ where p is a poly. We ignore such factors.
   2.2 Quite likely run even better in practice.

2SAT

2SAT is in P:
We omit this but note that the algorithm is FAST and PRACTICAL.

Convention For All of our Algorithms

Definition:

1. A *Unit Clause* is a clause with only one literal in it.
2. A *Pure Literal* is a literal that only shows up as non negated or only shows up as negated.

BILL: Do EXAMPLES.

Conventions:

1. If have unit clause immediately assign its literal to TRUE.
2. If have pure literal immediately assign it to be TRUE.
3. If we have a partial assignment z.
   3.1 If $(\forall C)[C(z) = TRUE$ then output YES.
   3.2 If $(\exists C)[C(z) = FALSE]$ then output NO.

META CONVENTION: Abbreviate doing this STAND (for STANDARD).

DPLL ALGORITHM

DPLL (Davis-Putnam-Logemann-Loveland) ALGORITHM

DPLL ALGORITHM

ALG(F: 3CNF fml; z: Partial Assignment)

STAND
Pick a variable x (VERY CLEVERLY)
ALG$(F; z \cup \{x = T\})$
ALG$(F; z \cup \{x = F\})$

BILL: TELL CLASS TO DISCUSS CLEVER WAYS TO PICK x.

DPLL and Heuristics Functions

Choose literal L such that

1. L appears in the most clauses. Try $L = 1$ first.
2. L appears A LOT, $\overline{L}$ appears very little. Try $L = 1$ first.
3. L is an arbitrary literal in the shortest clause.
4. (Jeroslaw-Wang) L that maximizes

$$\sum_{k=2}^{\infty} (\text{number of times } L \text{ occurs in a clause of length } k)\ 2^{-k}.$$

5. Other functions that combine the two could be tried.
6. Variant: set several variables at a time.

Key Idea Behind Recursive 7-ALG

KEY1: If F is a 3CNF formula and z is a partial assignment either

1. $F(z) = TRUE$, or
2. there is a clause $C = (L_1 \vee L_2)$ or $(L_1 \vee L_2 \vee L_3)$ that is not satisfied. (We assume $C = (L_1 \vee L_2 \vee L_3)$.)

KEY2: In ANY extension of z to a satisfying assignment ONE of the 7 ways to make $(L_1 \vee L_2 \vee L_3)$ true must happen.

Recursive-7 ALG

ALG(F: 3CNF fml; z: Partial Assignment)

STAND
if $F(z)$ in 2CNF use 2SAT ALG
find $C = (L_1 \vee L_2 \vee L_3)$ a clause not satisfied
for all 7 ways to set (L_1, L_2, L_3) so that C=TRUE
    Let z' be z extended by that setting
    ALG($F; z'$)

VOTE: IS THIS BETTER THAN $O(2^n)$?

Recursive-7 ALG

ALG(F: 3CNF fml; z: Partial Assignment)

STAND
if $F(z)$ in 2CNF use 2SAT ALG
find $C = (L_1 \vee L_2 \vee L_3)$ a clause not satisfied
for all 7 ways to set (L_1, L_2, L_3) so that C=TRUE
    Let z' be z extended by that setting
    ALG($F; z'$)

VOTE: IS THIS BETTER THAN $O(2^n)$?
IT IS! Work it out in groups NOW.

The Analysis

$T(0) = O(1)$
$T(n) = 7T(n-3)$.
$T(n) = 7^2 T(n-3\times 2)$
$T(n) = 7^3 T(n-3\times 3)$
$T(n) = 7^4 T(n-3\times 4)$
$T(n) = 7^i T(n-3i)$
Plug in $i = n/3$.
$T(n) = 7^{n/3} O(1) = O(((7^{1/3})^n) = O((1.913)^n)$

1. Good News: BROKE the 2^n barrier. Hope for the future!
2. Bad News: Still not that good a bound.
3. Good News: Can Modify to work better in practice.
4. Bad News: Do not know modification to work better in theory.

Recursive-7 ALG MODIFIED

ALG(F: 3CNF fml; z: partial assignment)

STAND

```
if ∃C = (L1 ∨ L2) not satisfied then
      for all 3 ways to set (L1, L2) s.t. C=TRUE
            Let z' be z extended by that setting
            ALG(F; z')
if ∃C = (L1 ∨ L2 ∨ L3) not satisfied then
      for all 7 ways to set (L1, L2, L3) s.t. C=TRUE
            Let z' be z extended by that setting
            ALG(F; z')
```

Formally still have : $T(n) = 7T(n-3)$.

Intuitively will often have: $T(n) = 3T(n-3)$.

Generalize?

BILL: ASK CLASS TO TRY TO DO 4-SAT, 5-SAT, etc using this.

Monien-Speckenmeyer

MS (Monien-Speckenmeyer) ALGORITHM

Key Ideas Behind Recursive-3 ALG

KEY1: Given F and z either:

1. $F(z) = TRUE$, or
2. there is a clause $C = (L_1 \vee L_2)$ or $(L_1 \vee L_2 \vee L_3)$ that is not satisfied. (We assume $C = (L_1 \vee L_2 \vee L_3)$.)

KEY2: in ANY extension of z to a satisfying assignment either:

1. L_1 TRUE.
2. L_1 FALSE, L_2 TRUE.
3. L_1 FALSE, L_2 FALSE, L_3 TRUE.

Recursive-3 ALG

ALG(F: 3CNF fml; z: Partial Assignment)

STAND
```
if F(z) in 2CNF use 2SAT ALG
find C = (L1 ∨ L2 ∨ L3) a clause not satisfied
```
ALG($F; z \cup \{L_1 = T\}$)
ALG($F; z \cup \{L_1 = F, L_2 = T\}$)
ALG($F; z \cup \{L_1 = F, L_2 = F, L_3 = T\}$)

VOTE: IS THIS BETTER THAN $O((1.913)^n)$?

Recursive-3 ALG

ALG(F: 3CNF fml; z: Partial Assignment)

STAND
if $F(z)$ in 2CNF use 2SAT ALG
find $C = (L_1 \vee L_2 \vee L_3)$ a clause not satisfied
ALG$(F; z \cup \{L_1 = T\})$
ALG$(F; z \cup \{L_1 = F, L_2 = T\})$
ALG$(F; z \cup \{L_1 = F, L_2 = F, L_3 = T\})$

VOTE: IS THIS BETTER THAN $O((1.913)^n)$?
IT IS! Work it out in groups NOW.

The Analysis

$T(0) = O(1)$
$T(n) = T(n-1) + T(n-2) + T(n-3).$
Guess $T(n) = \alpha^n$
$\alpha^n = \alpha^{n-1} + \alpha^{n-2} + \alpha^{n-3}$
$\alpha^3 = \alpha^2 + \alpha + 1$
$\alpha^3 - \alpha^2 - \alpha - 1 = 0$
Root: $\alpha \sim 1.84.$
Answer: $T(n) = O((1.84)^n).$

So Where Are We Now?

1. Good News: BROKE the $(1.913)^n$ barrier. Hope for the future!
2. Bad News: $(1.84)^n$ Still not that good.
3. Good News: Can modify to work better in practice!
4. Good News: Can modify to work better in theory!!

Recursive-3 ALG MODIFIED

ALG(F: 3CNF fml, z: partial assignment)

STAND

```
if ∃C = (L1 ∨ L2) not satisfied then
        ALG(F; z ∪ {L1 = T})
        ALG(F; z ∪ {L1 = F, L2 = T})
if (∃C = (L1 ∨ L2 ∨ L3) not satisfied then
        ALG(F; z ∪ {L1 = T})
        ALG(F; z ∪ {L1 = F, L2 = T})
        ALG(F; z ∪ {L1 = F, L2 = F, L3 = T})
```

Formally still have : $T(n) = T(n-1) + T(n-2) + T(n-3)$.

Intuitively will often have: $T(n) = T(n-1) + T(n-2)$.

Generalize?

BILL: ASK CLASS TO TRY TO DO 4-SAT, 5-SAT, etc using this.
BILL: ASK CLASS FOR IDEAS TO IMPROVE 3SAT VERSION.

IDEAS

Definition: If F is a fml and z is a partial assignment then z is COOL if every clause that z affects is made TRUE.
BILL: Do examples and counterexamples.
Prove to yourself:
Lemma: Let F be a 3CNF fml and z be a partial assignment.

1. If z is COOL then $F \in 3SAT$ iff $F(z) \in 3SAT$.
2. If z is NOT COOL then $F(z)$ will have a clause of length 2.

Recursive-3 ALG MODIFIED MORE

ALG(F: 3CNF fml, z: partial assignment)

```
COMMENT: This slide is when a 2CNF clause not satis
STAND
if (∃C = (L1 ∨ L2) not satisfied then
        z1 = z ∪ {L1 = T})
        if z1 is COOL then ALG(F; z1)
           else
             z01 = z ∪ {L1 = F, L2 = T})
             if z01 is COOL then ALG(F; z01)
                 else
                    ALG(F; z1)
                    ALG(F; z01)
else (COMMENT: The ELSE is on next slide.)
```

Recursive-3 ALG MODIFIED MORE

```
(COMMENT: This slide is when a 3CNF clause not sati
if (∃C = (L1 ∨ L2 ∨ L3) not satisfied then
     z1 = z ∪ {L1 = T})
     if z1 is COOL then ALG(F; z1)
       else
         z01 = z ∪ {L1 = F, L2 = T})
         if z01 is COOL then ALG(F; z01)
           else
             z001 = z ∪ {L1 = F, L2 = F, L3 = T})
             if z001 is COOL then ALG(F; z001)
               else
                 ALG(F; z1)
                 ALG(F; z01)
                 ALG(F; z001)
```

IS IT BETTER?

VOTE: IS THIS BETTER THAN $O((1.84)^n)$?

IS IT BETTER?

VOTE: IS THIS BETTER THAN $O((1.84)^n)$?

IT IS! Work it out in groups NOW.

IT IS BETTER!

KEY1: If any of $z1$, $z01$, $z001$ are COOL then only ONE recursion: $T(n) = T(n-1) + O(1)$.
KEY2: If NONE of the $z0$, $z01$ $z001$ are COOL then ALL of the recurrences are on fml's with a 2CNF clause in it.
$T(n)$= Time alg takes on 3CNF formulas.
$T'(n)$= Time alg takes on 3CNF formulas that have a 2CNF in them.
$T(n) = \max\{T(n-1), T'(n-1) + T'(n-2) + T'(n-3)\}$.
$T'(n) = \max\{T(n-1), T'(n-1) + T'(n-2)\}$.
Can show that worst case is:
$T(n) = T'(n-1) + T'(n-2) + T'(n-3)$.
$T'(n) = T'(n-1) + T'(n-2)$.

The Analysis

$T'(0) = O(1)$
$T'(n) = T'(n-1) + T'(n-2)$.
Guess $T(n) = \alpha^n$
$\alpha^n = \alpha^{n-1} + \alpha^{n-2}$
$\alpha^2 = \alpha + 1$
$\alpha^2 - \alpha - 1 = 0$
Root: $\alpha = \frac{1+\sqrt{5}}{2} \sim 1.618$.
Answer: $T'(n) = O((1.618)^n)$.
Answer: $T(n) = O(T(n)) = O((1.618)^n)$.
VOTE: Is better known?
VOTE: Is there a proof that *these techniques* cannot do any better?

Hamming Distances

Definition If x, y are assignments then $d(x, y)$ is the number of bits they differ on.

BILL: DO EXAMPLES
KEY TO NEXT ALGORITHM: If F is a fml on n variables and F is satisfiable then either

1. F has a satisfying assignment z with $d(z, 0^n) \leq n/2$, or
2. F has a satisfying assignment z with $d(z, 1^n) \leq n/2$.

HAM ALG

HAMALG(F: 3CNF fml, z: full assignment, h: number) h bounds $d(z, s)$ where s is SATisfying assignment h is distance

STAND

```
if ∃C = (L1 ∨ L2) not satisfied then
        ALG(F; z ⊕ {L1 = T}; h − 1}
        ALG(F; z ⊕ {L1 = F, L2 = T}; h − 1)
if ∃C = (L1 ∨ L2 ∨ L3) not satisfied then
        ALG(F; z ⊕ {L1 = T}; h − 1)
        ALG(F; z ⊕ {L1 = F, L2 = T}; h − 1)
        ALG(F; z ⊕ {L1 = F, L2 = F, L3 = T}; h − 1)
```

REAL ALG

HAMALG(F; 0^n; $n/2$)
`If returned` NO `then` HAMALG(F; 1^n; $n/2$)

VOTE: IS THIS BETTER THAN $O((1.61)^n)$?

REAL ALG

HAMALG$(F; 0^n; n/2)$
If returned NO then HAMALG$(F; 1^n; n/2)$

VOTE: IS THIS BETTER THAN $O((1.61)^n)$?
IT IS NOT! Work it out in groups anyway NOW.

ANALYSIS

KEY: We don't care about how many vars are assigned since they all are. We care about h.

$T(0) = 1$.
$T(h) = 3T(h-1)$.
$T(h) = 3^i T(h-i)$.
$T(h) = 3^h$.
$T(n/2) = 3^{n/2} = O((1.73)^n)$.

BETTER IDEAS?

BILL: Ask Class for Ideas on how to use the HAM DISTANCE ideas to get a better algorithm.

KEY TO HAM

KEY TO HAM ALGORITHM: Every element of $\{0,1\}^n$ is within $n/2$ of either 0^n or 1^n

Definition: A *covering code of* $\{0,1\}^n$ *of SIZE* s *with RADIUS* h is a set $S \subseteq \{0,1\}^n$ of size s such that

$$(\forall x \in \{0,1\}^n)(\exists y \in S)[d(x,y) \le h].$$

Example: $\{0^n, 1^n\}$ is a covering code of SIZE 2 of RADIUS $n/2$.

ASSUME ALG

Assume we have a Covering code of $\{0,1\}^n$ of size s and radius h. Let Covering code be $S = \{v_1, \ldots, v_s\}$.

```
i = 1
FOUND=FALSE
while (FOUND=FALSE) and (i ≤ s)
    HAMALG(F; v_i; h)
    If returned YES then FOUND=TRUE
        else
          i = i + 1
end while
```

ANALYSIS OF ALG

Each iteration satisfies recurrence
$T(0) = 1$
$T(h) = 3T(h-1)$
$T(h) = 3^h$.
And we do this s times.
ANALYSIS: $O(s3^h)$.
Need covering codes with small value of $O(s3^h)$.

IN SEARCH OF A GOOD COVERING CODE

RECAP: Need covering codes of size s, radius h, with small value of $O(s3^h)$.

IN SEARCH OF A GOOD COVERING CODE

RECAP: Need covering codes of size s, radius h, with small value of $O(s3^h)$.
THATS NOT ENOUGH: We need to actually CONSTRUCT the covering code in good time.

IN SEARCH OF A GOOD COVERING CODE

RECAP: Need covering codes of size s, radius h, with small value of $O(s3^h)$.
THATS NOT ENOUGH: We need to actually CONSTRUCT the covering code in good time.
YOU"VE BEEN PUNKED: We'll just pick a RANDOM subset of $\{0,1\}^n$ and hope that it works.

IN SEARCH OF A GOOD COVERING CODE

RECAP: Need covering codes of size s, radius h, with small value of $O(s3^h)$.
THATS NOT ENOUGH: We need to actually CONSTRUCT the covering code in good time.
YOU"VE BEEN PUNKED: We'll just pick a RANDOM subset of $\{0,1\}^n$ and hope that it works.
SO CRAZY IT MIGHT JUST WORK!

IN SEARCH OF A GOOD COVERING CODE-RANDOM!

Let $A = \{\alpha_1, \ldots, \alpha_s\}$ be a RANDOM subset of $\{0,1\}^n$.
Let $h \in \mathrm{N}$. Let $\alpha_0 \in \{0,1\}^n$.
We want PROB that NONE of the elements of A are within h of α_0.
We consider just one $\alpha = \alpha_i$ first:

$$\begin{aligned} \Pr(d(\alpha,\alpha_0) > h) = 1 - \Pr(d(\alpha,\alpha_0) \le h) = & \quad 1 - \frac{\sum_{j=0}^{h} \binom{n}{j}}{2^n} \\ \le & \quad e^{-\frac{\sum_{j=0}^{h} \binom{n}{j}}{2^n}} \end{aligned}$$

IN SEARCH OF A GOOD COVERING CODE-RANDOM!

$\Pr(d(\alpha,\alpha_0) > h) \le e^{-\frac{\sum_{j=0}^{h}\binom{n}{j}}{2^n}}$
So Prob that NONE of the s elements of A are within h of α is bounded by

$$e^{-t\frac{\sum_{j=0}^{h}\binom{n}{j}}{2^n}}$$

Let

$$t = \frac{n^2 2^n}{\sum_{j=0}^{h}\binom{n}{j}}.$$

Prob that NONE of the s elements of A are within h of α is $\le e^{-n^2}$.

SETTING THE PARAMETERS

Want $t = \frac{n^2 2^n}{\sum_{j=0}^{h} \binom{n}{j}}$ to be small.
Set $h = \delta n$.

$$s = \frac{n^2 2^n}{\sum_{j=0}^{h} \binom{n}{j}} = \frac{n^2 2^n}{\sum_{j=0}^{\delta n} \binom{n}{j}} \sim \frac{n^2 2^n}{\binom{n}{\delta n}} \sim \frac{n^2 2^n}{2^{h(\delta)n}} = n^2 2^{n(1-h(\delta))}$$

Where $h(\delta) = -\delta \lg(\delta) - (1-\delta)\lg(1-\delta)$.

Recall: We want a small value of $O(s3^h) = O(n^2 2^{n(1-h(\delta))} 3^{\delta n})$

SETTING THE PARAMETERS

Recall: We want a small value of $O(s3^h) = O(n^2 2^{n(1-h(\delta))} 3^{\delta n})$

1. $\delta = 1/4$
2. $s = n^2 \times 2^{.188n} 3^{0.25n} \sim O((1.5)^n)$.

RANDOMIZED ALG

```
Pick  S ⊆ {0,1}^n ,  |S| = n^2(1.5)^n ,  RANDOMLY.
i = 1
FOUND=FALSE
while  (FOUND=FALSE)  and  (i ≤ s)
    HAMALG(F; v_i; n/2)
     If  returned  YES  then  FOUND=TRUE
         else
            i = i + 1
end  while
```

CAUTION: Prob of error is NONZERO! Its $\leq e^{-n^2}$.
TIME: $O((1.5)^n)$.

ALT VIEW

If you know you will be looking at MANY FMLS of n variables can pick an S, TEST IT, and if its find then use it. Expensive Preprocessing.

Faster in Practice

Speed up tips for ALL algorithms mentioned:
Which clause to pick?

1. Always pick shortest clause.
2. Find clause where all three literals in many other clauses.