Good but still Exp Algorithms for 3-SAT

Exposition by William Gasarch
This talk is based on parts of the following AWESOME books:

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

Exact Exponential Algorithms
by
Fedor Formin and Dieter Kratsch
We will show algorithms for 3SAT that

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. Quite likely run even better in practice, or modifications of them do.
2SAT is in P:
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.

Conventions:
1. If have unit clause assign its literal to TRUE.
2. If have POS-pure literal then assign it to be TRUE.
3. If have NEG-pure literal then assign it to be FALSE.
4. If we have a partial assignment $z$.
   4.1 If $(\forall C)[C(z) = TRUE]$ then output YES.
   4.2 If $(\exists C)[C(z) = FALSE]$ then output NO.

CONVENTION: Abbreviate this STAND (for STANDARD).
DPLL (Davis-Putnam-Logemann-Loveland) ALGORITHM

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

STAND
Pick a variable $x$ (VERY CLEVERLY)
ALG($F; z \cup \{x = T\}$)
ALG($F; z \cup \{x = F\}$)
KEY1: If $F$ is a 3CNF formula and $z$ is a partial assignment either

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

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

\[ \text{ALG}(F: 3\text{CNF }\text{fml}; z: \text{Partial Assignment}) \]

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

\[ T(n) = 7 T(n - 3) \text{ so } T(n) = O((1.913)^n) \]
1. Good News: BROKE the $2^n$ barrier. Hope for the future!
2. Bad News: Still not that good a bound.
3. Good News: Similar ideas gets time to $O((1.84)^n)$.
4. Bad News: Still not that good a bound.
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.

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

COMMENT: This slide is when a 2CNF clause not satisfied.

STAND

if ($\exists C = (L_1 \lor L_2)$) not satisfied then
    $z_1 = z \cup \{L_1 = T\}$
    if $z_1$ is COOL then ALG($F;z_1$)
else
    $z_{01} = z \cup \{L_1 = F, L_2 = T\}$
    if $z_{01}$ is COOL then ALG($F;z_{01}$)
    else
        ALG($F;z_1$)
        ALG($F;z_{01}$)
else (COMMENT: The ELSE is on next slide.)
(COMMENT: This slide is when a 3CNF clause not satisfied.

If \( \exists C = (L_1 \lor L_2 \lor L_3) \) not satisfied then

\[
z_1 = z \cup \{L_1 = T\}
\]

If \( z_1 \) is COOL then \( \text{ALG}(F; z_1) \)

else

\[
z_{01} = z \cup \{L_1 = F, L_2 = T\}
\]

If \( z_{01} \) is COOL then \( \text{ALG}(F; z_{01}) \)

else

\[
z_{001} = z \cup \{L_1 = F, L_2 = F, L_3 = T\}
\]

If \( z_{001} \) is COOL then \( \text{ALG}(F; z_{001}) \)

else

\[
\text{ALG}(F; z_1)
\]
\[
\text{ALG}(F; z_{01})
\]
\[
\text{ALG}(F; z_{001})
\]
VOTE: IS THIS BETTER THAN $O((1.84)^n)$?
VOTE: IS THIS BETTER THAN $O((1.84)^n)$?
IT IS!
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.

$T(n) = \text{Time alg takes on 3CNF formulas.}$
$T'(n) = \text{Time alg takes on 3CNF formulas that have a 2CNF in them.}$

$T(n) = \max\{T(n - 1), T'(n - 1) + T'(n - 2) + T'(n - 3)\}.$
$T'(n) = \max\{T(n - 1), T'(n - 1) + T'(n - 2)\}.$

Can show that worst case is:
$T(n) = T'(n - 1) + T'(n - 2) + T'(n - 3).$
$T'(n) = T'(n - 1) + T'(n - 2).$
\[ T'(0) = O(1) \]
\[ T'(n) = T'(n - 1) + T'(n - 2). \]

\[ T'(n) = O((1.618)^n). \]

So

\[ T(n) = O(T(n)) = O((1.618)^n). \]

**VOTE:** Is better known?

**VOTE:** Is there a proof that these techniques cannot do any better?
**Definition** If $x, y$ are assignments then $d(x, y)$ is the number of bits they differ on.

**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 $d(z, 1^n) \leq n/2$. 

Exposition by William Gasarch

Good but still Exp Algorithms for 3-SAT
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 $\exists C = (L_1 \lor L_2)$ not satisfied then
  ALG($F; z \oplus \{L_1 = T\}; h - 1$)
  ALG($F; z \oplus \{L_1 = F, L_2 = T\}; h - 1$)
if $\exists C = (L_1 \lor L_2 \lor L_3)$ not satisfied then
  ALG($F; z \oplus \{L_1 = T\}; h - 1$)
  ALG($F; z \oplus \{L_1 = F, L_2 = T\}; h - 1$)
  ALG($F; z \oplus \{L_1 = F, L_2 = F, L_3 = T\}; h - 1$)
HAMALG\( (F; 0^n; n/2) \)

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

**VOTE:** Is this better than \( O((1.61)^n) \)?
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! It is $O((1.73)^n)$.
KEY TO HAM ALGORITHM: Every element of \( \{0, 1\}^n \) is within \( n/2 \) of either \( 0^n \) or \( 1^n \)

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

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

Example: \( \{0^n, 1^n\} \) is a covering code of \textit{SIZE} 2 of \textit{RADIUS} \( n/2 \).
Assume we have a Covering code of \( \{0, 1\}^n \) of size \( s \) and radius \( h \). Let Covering code be \( S = \{v_1, \ldots, v_s\} \).

\[ i = 1 \]
\[ \text{FOUND} = \text{FALSE} \]
\[ \text{while } (\text{FOUND} = \text{FALSE}) \text{ and } (i \leq s) \]
\[ \text{HAMALG}(F; v_i; h) \]
\[ \text{If returned YES then } \text{FOUND} = \text{TRUE} \]
\[ \text{else} \]
\[ i = i + 1 \]
\[ \text{end while} \]
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 \( O(s3^h) \).
RECAP: Need covering codes of size $s$, radius $h$, with small value of $O(s3^h)$. 
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.
RECAP: Need covering codes of size $s$, radius $h$, with small value of $O(s^{3^h})$.
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.
CAN find with high prob a covering code with
- Size \( s = n^2 2^{.4063n} \)
- Distance \( h = 0.25n. \)

Can use to get SAT in \( O((1.5)^n) \).

Note: Best known: \( O((1.306)^n) \).
SATisfiable?

The AND of the following:

1. \( x_{11} \lor x_{12} \)
2. \( x_{21} \lor x_{22} \)
3. \( x_{31} \lor x_{32} \)
4. \( \neg x_{11} \lor \neg x_{21} \)
5. \( \neg x_{11} \lor \neg x_{31} \)
6. \( \neg x_{21} \lor \neg x_{31} \)
7. \( \neg x_{12} \lor \neg x_{22} \)
8. \( \neg x_{12} \lor \neg x_{32} \)
9. \( \neg x_{22} \lor \neg x_{32} \)

This is Pigeonhole Principle: \( x_{ij} \) is putting \( i \)th pigeon in \( j \) hole! Can't put 3 pigeons into 2 holes!
SATisfiable?

The AND of the following:

1. \( x_{11} \lor x_{12} \)
2. \( x_{21} \lor x_{22} \)
3. \( x_{31} \lor x_{32} \)
4. \( \neg x_{11} \lor \neg x_{21} \)
5. \( \neg x_{11} \lor \neg x_{31} \)
6. \( \neg x_{21} \lor \neg x_{31} \)
7. \( \neg x_{12} \lor \neg x_{22} \)
8. \( \neg x_{12} \lor \neg x_{32} \)
9. \( \neg x_{22} \lor \neg x_{32} \)

This is Pigeonhole Principle: \( x_{ij} \) is putting \( i \)th pigeon in \( j \) hole!
SATisfiable?

The AND of the following:

1. \( x_{11} \lor x_{12} \)
2. \( x_{21} \lor x_{22} \)
3. \( x_{31} \lor x_{32} \)
4. \( \neg x_{11} \lor \neg x_{21} \)
5. \( \neg x_{11} \lor \neg x_{31} \)
6. \( \neg x_{21} \lor \neg x_{31} \)
7. \( \neg x_{12} \lor \neg x_{22} \)
8. \( \neg x_{12} \lor \neg x_{32} \)
9. \( \neg x_{22} \lor \neg x_{32} \)

This is Pigeonhole Principle: \( x_{ij} \) is putting \( i \)th pigeon in \( j \) hole!
Can’t put 3 pigeons into 2 holes!