Random Search on 3SAT

Group 4

CS6234 - Advanced Algorithms

April 19, 2016

Contents

- Boolean Satisfiability Problem
- Schöning's Algorithm for 3SAT
- Analysis 1
- Analysis 2
- Analysis 3

Boolean Satisfiability Problem - By Sapumal

Boolean Satisfiability Problem

- Is the problem of determining if there exists an interpretation that satisfies a given Boolean formula.

Boolean Satisfiability Problem

- Is the problem of determining if there exists an interpretation that satisfies a given Boolean formula.
- It asks whether the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE.

Boolean Satisfiability Problem

- Is the problem of determining if there exists an interpretation that satisfies a given Boolean formula.
- It asks whether the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE.
  - If this is the case, the formula is called satisfiable.

Boolean Satisfiability Problem

- Is the problem of determining if there exists an interpretation that satisfies a given Boolean formula.
- It asks whether the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE.
    - If this is the case, the formula is called satisfiable.
    - On the other hand, if no such assignment exists for all possible variable assignments and the formula is unsatisfiable.

Boolean Satisfiability Problem

- Is the problem of determining if there exists an interpretation that satisfies a given Boolean formula.
- It asks whether the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE.
  - If this is the case, the formula is called satisfiable.
  - On the other hand, if no such assignment exists for all possible variable assignments and the formula is unsatisfiable.
- Referred to as SATISFIABILITY or SAT

Boolean Satisfiability Problem

- SAT formula usually take input in Conjunctive Normal Form (CNF): "an AND of ORs of literals".

Boolean Satisfiability Problem

- SAT formula usually take input in Conjunctive Normal Form (CNF): "an AND of ORs of literals".
  - Variable - a propositional variable: x_1, x_2, x_3

Boolean Satisfiability Problem

- SAT formula usually take input in Conjunctive Normal Form (CNF): "an AND of ORs of literals".
  - Variable - a propositional variable: x_1, x_2, x_3
  - Literal - an variable or its negation: $x_1, \neg x_1, x_2, \neg x_2$

Boolean Satisfiability Problem

- SAT formula usually take input in Conjunctive Normal Form (CNF): "an AND of ORs of literals".
    - Variable - a propositional variable: x_1, x_2, x_3
    - Literal - an variable or its negation: $x_1, \neg x_1, x_2, \neg x_2$
    - Clause - A disjunction of some literals: $(x_1 \vee x_2 \vee x_3)$

Boolean Satisfiability Problem

- SAT formula usually take input in Conjunctive Normal Form (CNF): "an AND of ORs of literals".
  - Variable - a propositional variable: x_1, x_2, x_3
  - Literal - an variable or its negation: $x_1, \neg x_1, x_2, \neg x_2$
  - Clause - A disjunction of some literals: $(x_1 \vee x_2 \vee x_3)$
  - CNF formula - A conjunction of some clauses: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2)$

Boolean Satisfiability Problem

- Simple example,
  - $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_4)$

Boolean Satisfiability Problem

- Simple example,
  - $(x_1 \lor x_2) \land (\neg x_1 \lor x_3) \land (x_3 \lor x_4) \land (\neg x_2 \lor \neg x_4)$
  - $x_1 = x_3 = TRUE$

Boolean Satisfiability Problem

- Simple example,
  - $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_4)$
  - $x_1 = x_3 = \mathit{TRUE}$ and $x_2, x_4 = \mathit{FALSE}$

Applications

- Combinational equivalence checking (CEC)
  - 2 combinational circuits, each with n inputs and m outputs.
  - Are the outputs same for all input values?

Applications

- Combinational equivalence checking (CEC)
  - 2 combinational circuits, each with n inputs and m outputs.
  - Are the outputs same for all input values?
- Automatic test pattern generation (ATPG)
  - Fabricated integrated circuits may be subject to defects, which may cause circuit failure
  - Computing input assignments that allow demonstrating the existence or absence of each target fault

Applications

- Combinational equivalence checking (CEC)
  - 2 combinational circuits, each with n inputs and m outputs.
  - Are the outputs same for all input values?
- Automatic test pattern generation (ATPG)
  - Fabricated integrated circuits may be subject to defects, which may cause circuit failure
  - Computing input assignments that allow demonstrating the existence or absence of each target fault
- Model checking
- Applications in Bioinformatics

- Ref: Marques-Silva, Joao. "Practical applications of boolean satisfiability." Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on. IEEE, 2008.

2SAT

- Collection $C = C_1, \ldots, C_m$ of clauses n Boolean variables such that $|C_i| \leq 2$ for $1 \leq i \leq m$

2SAT

- Collection $C = C_1, \ldots, C_m$ of clauses n Boolean variables such that $|C_i| \leq 2$ for $1 \leq i \leq m$
- 2SAT can be solved in polynomial time (in fact in linear time)

2SAT

- Collection $C = C_1, \ldots, C_m$ of clauses n Boolean variables such that $|C_i| \leq 2$ for $1 \leq i \leq m$
- 2SAT can be solved in polynomial time (in fact in linear time)
- 2SAT can be solved by formulating it as a implication graph

2SAT

- Collection $C = C_1, \ldots, C_m$ of clauses n Boolean variables such that $|C_i| \leq 2$ for $1 \leq i \leq m$
- 2SAT can be solved in polynomial time (in fact in linear time)
- 2SAT can be solved by formulating it as a implication graph
- $(x_1 \vee x_2)$ is logically equivalent to either of $\neg x_1 \Rightarrow x_2$ or $\neg x_2 \Rightarrow x_1$
- Thus a 2SAT formula may be viewed as a set of implications.
  - Construct a directed graph G such that vertices of G are the variables and their negations.
  - There is an arc (x_1, x_2) in G if and only if there is a clause $(\neg x_1 \vee x_2)$ or $(x_2 \vee \neg x_1)$ in the 2SAT instance.

2SAT

- If for some variable x_i, there is a string of implications,
  - $x_i \Rightarrow \cdots \Rightarrow \neg x_i$, and another string of implications.
  - $\neg x_i \Rightarrow \cdots \Rightarrow x_i$, then it is not satisfiable,
  - otherwise it is satisfiable.

2SAT

- If for some variable x_i, there is a string of implications,
  - $x_i \Rightarrow \cdots \Rightarrow \neg x_i$, and another string of implications.
  - $\neg x_i \Rightarrow \cdots \Rightarrow x_i$, then it is not satisfiable,
  - otherwise it is satisfiable.
- The 2SAT problem thus reduces to the graph problem of finding strongly connected components (SCC) in the implication graph

2SAT

- If for some variable x_i, there is a string of implications,
  - $x_i \Rightarrow \cdots \Rightarrow \neg x_i$, and another string of implications.
  - $\neg x_i \Rightarrow \cdots \Rightarrow x_i$, then it is not satisfiable,
  - otherwise it is satisfiable.
- The 2SAT problem thus reduces to the graph problem of finding strongly connected components (SCC) in the implication graph
- As computing SCC is known to have a linear-time solution
- It is clear that 2SAT may be decided under the same time bound.

3SAT

- In 3SAT every clause must have at most 3 literals.

3SAT

- In 3SAT every clause must have at most 3 literals.
- Unrestricted SAT problems can be reduced to 3SAT

3SAT

- In 3SAT every clause must have at most 3 literals.
- Unrestricted SAT problems can be reduced to 3SAT
- No known polynomial time reduction from SAT (or 3SAT) to 2SAT. If there was, then SAT and 3SAT would be solvable in polynomial time.

Cook Levin Theorem

- Decision problem: Is there a valid solution or not?

Cook Levin Theorem

- Decision problem: Is there a valid solution or not?
- Cook Levin Theorem states that the SAT decision problem is NP-complete

Cook Levin Theorem

- Decision problem: Is there a valid solution or not?
- Cook Levin Theorem states that the SAT decision problem is NP-complete
- Although any given solution to an NP-complete problem can be verified quickly (in polynomial time), no fast way of solving them is known.

The Algorithm - By Naheed

Outline

- Brute Force Search Algorithm for 3SAT

Outline

- Brute Force Search Algorithm for 3SAT
- Schöning's Algorithm for 3SAT

Outline

- Brute Force Search Algorithm for 3SAT
- Schöning's Algorithm for 3SAT
- Schöning's Algorithm: Illustrative Examples

Brute-Force Search for 3SAT

Let $E = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be the 3SAT formulae where C_i is the i-th Clause.

A Truth assignment, $\mathbf{a} = (x_1, x_2, \ldots, x_n)$

Let Ω be the set of all possible (2^n) truth assignments of **a**.

for all assignment $\mathbf{a} \in \Omega$ **do**
  if a satisfies E **then**
    return "satisfiable"
  end if
end for
return "unsatisfiable"

Complexity: $\mathcal{O}(2^n)$

Brute-Force Search for 3SAT

Let $E = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be the 3SAT formulae where C_i is the i-th Clause.
A Truth assignment, $\mathbf{a} = (x_1, x_2, \ldots, x_n)$
Let Ω be the set of all possible (2^n) truth assignments of $\mathbf{a}$.
for a

if a

end if
end for
return "unsatisfiable"

Question

Can We do Better?

Complexity: $\mathcal{O}(2^n)$

Schöning's Algorithm for 3SAT

Let $E = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be the 3SAT formulae where C_i is the i-th Clause.
Let Ω be the set of all possible (2^n) truth assignments.
repeat T times (or until a satisfying truth assignment is found)
  choose an initial truth assignment, $\mathbf{a}_0$ uniformly at random from Ω
  current assignment, $\mathbf{a} = \mathbf{a}_0$
  repeat n times (or until **a** satisfies E)
    Choose a clause C violated by the current assignment **a**.
    Choose one of the literals from C uniformly at random, and modify **a** by flipping the value of the corresponding variable.
if a satisfying assignment was found **then**
  return "satisfiable"
else
  return "unsatisfiable"
end if

Complexity: $\mathcal{O}(Tn)$

Example (Case 1: E unsatisfiable)

$n = 3 \ \{x_1, x_2, x_3\}$
$m = 7 \ \{C_1, C_2, \ldots, C_7\}$

- $E = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_3) \wedge (x_1 \vee \neg x_3) \wedge (x_3)$
- Set of Satisfiable Truth Assignment, $A^* = \{\}$

Example (Case 1: E unsatisfiable)

$n = 3\ \{x_1, x_2, x_3\}$
$m = 7\ \{C_1, C_2, \ldots, C_7\}$

- $E = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_3) \wedge (x_1 \vee \neg x_3) \wedge (x_3)$
- Set of Satisfiable Truth Assignment, $A^* = \{\}$
- Schöning's algorithm will always return unsatisfiable when E is unsatisfiable.

Example (Case 2: E satisfiable in 1st Trial)

$n = 3$
$m = 4$ $\{C_1, C_2, C_3, C_4\}$

- $E = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_3)$
- Set of Satisfiable Truth Assignment, $A^* = \{(True, True, False), (False, True, False)\}$
- If Truth assignment at the 1st Iteration, $\mathbf{a_0} = (True, True, False)$ (lucky!)

Example (Case 3: E satisfiable but Schöning Fails!)

$n = 3$
$m = 4$ $\{C_1, C_2, C_3, C_4\}$

- $E = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_3)$
- Set of Satisfiable Truth Assignment, $A^* = \{(True, True, False), (False, True, False)\}$
- Truth assignment at first iteration, $\mathbf{a_0} = (False, True, True)$, Violated Clause $= C_4$
  - Flip x_1: $\mathbf{a} = (True, True, True)$. Violated Clause $= C_2$.
  - Flip x_1: $\mathbf{a} = (False, True, True)$. Violated Clause $= C_4$.
  - Flip x_1: $\mathbf{a} = (True, True, True)$. Violated Clause $= C_2$.
- returns Unsatisfiable.

Example (Case 4: E satisfiable, Schöning Succeeds!)

$n = 3$

$m = 4 \; \{C_1, C_2, C_3, C_4\}$

- $E = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_3)$
- Set of Satisfiable Truth Assignment, $A^* = \{(True, True, False), (False, True, False)\}$
- **Iteration 1:**
- **Iteration 2:**
- ...
- **Iteration i:** Initial Truth assignment, $\mathbf{a_0} = (False, False, True)$, Violated Clause $= C_4$
  - Flip x_3: $\mathbf{a} = (False, False, False)$. Violated Clause $= C_1$
  - Flip x_2: $\mathbf{a} = (False, True, False)$. E is satisfied!
- returns Satisfiable.

Example (Case 4: E satisfiable, Schöning Succeeds!)

$n = 3$

$m = 4 \; \{C_1, C_2, C_3, C_4\}$

- $E = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_3)$
- Set of Satisfiable Truth Assignment, $A^* =$ $\{(True, True, False), (False, True, False)\}$
- **Ite**
- **Ite**
- ...

Question

How large **T** should be to find a satisfiable truth assignment with High Probability?

- **Iteration i:** Initial Truth assignment, $\mathbf{a_0} = (False, False, True)$, Violated Clause $= C_4$
  - Flip x_3: $\mathbf{a} = (False, False, False)$. Violated Clause $= C_1$
  - Flip x_2: $\mathbf{a} = (False, True, False)$. E is satisfied!
- returns Satisfiable.

Analysis Part 1 - By DME Manupa Karunaratne

The Analysis 1

- We only do the analysis on the satisfiable instance.

The Analysis 1

- We only do the analysis on the satisfiable instance.
- The set of assignments that satisfies all the clauses is $A^* = \{\mathbf{a_1^*}, \mathbf{a_2^*}, \cdots, \mathbf{a_p^*}\}$

The Analysis 1

- We only do the analysis on the satisfiable instance.
- The set of assignments that satisfies all the clauses is $A^* = \{\mathbf{a_1^*}, \mathbf{a_2^*}, \cdots, \mathbf{a_p^*}\}$
- We'll arbitrarily pick one assignment for the analysis $\mathbf{a}^*$.

The Analysis 1

- We only do the analysis on the satisfiable instance.
- The set of assignments that satisfies all the clauses is $A^* = \{\mathbf{a_1^*}, \mathbf{a_2^*}, \cdots, \mathbf{a_p^*}\}$
- We'll arbitrarily pick one assignment for the analysis $\mathbf{a}^*$.
- We want to analyze the distance of a particular assignment $\mathbf{a}$ and $\mathbf{a}^*$.

Hamming Distance

- This distance is called the "Hamming Distance".

Hamming Distance

- This distance is called the "Hamming Distance".
- Example :

$$Let\ the\ number\ of\ variables, n = 3$$

$$Let\ V = \{x_1, x_2, x_3\}$$

$$Let\ \mathbf{a}^* = (True, False, True)$$

$$Particular\ Assignment\ \mathbf{a} = (False, True, True)$$

Hamming Distance

- This distance is called the "Hamming Distance".
- Example :

$$Let\ \ the\ \ number\ \ of\ \ variables, n = 3$$

$$Let\ \ V = \{x_1, x_2, x_3\}$$

$$Let\ \ \mathbf{a}^* = (True, False, True)$$

$$Particular\ \ Assignment\ \ \mathbf{a} = (False, True, True)$$

- Since the difference is only at the first two locations and the third one is same as $\mathbf{a}^*$, the Hamming Distance is 2.

Claim 1

- Let the hamming distance between a given assignment a and the satisfying assignment $\mathbf{a}^*$ is k.

Claim 1

$$Pr(k \leq \frac{n}{2}) \geq \frac{1}{2}$$

.

The Proof of Claim 1 : The Symmetry

- There is a symmetry in the possible space of assignments along the k (Hamming Distance) axis.

.

The Proof of Claim 1 : The Symmetry

- There is a symmetry in the possible space of assignments along the k (Hamming Distance) axis.
- The vectors with $k = p$, are essentially the vectors which differ in k number of locations to $\mathbf{a}^*$.

The Proof of Claim 1 : The Symmetry

- There is a symmetry in the possible space of assignments along the k (Hamming Distance) axis.
- The vectors with $k = p$, are essentially the vectors which differ in k number of locations to $\mathbf{a}^*$.
- Therefore number of such vectors is $\binom{n}{p}$.

$$\binom{n}{p} = \binom{n}{n-p}$$

The Proof of Claim 1 : The Symmetry

- There is a symmetry in the possible space of assignments along the k (Hamming Distance) axis.
- The vectors with $k = p$, are essentially the vectors which differ in k number of locations to $\mathbf{a}^*$.
- Therefore number of such vectors is $\binom{n}{p}$.

$$\binom{n}{p} = \binom{n}{n-p}$$

- In other words,

$$assignments\ with\ \{k = p\} = assignments\ with\ \{k = n - p\}$$

The Proof of Claim 1 : The "n is odd" Case

I'll denote the number of assignments with $k = p$ as f_p.

Case : n is odd

$$Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n-1}{2}} f_k}{\sum_{k=0}^{n} f_k}$$

The Proof of Claim 1 : The "n is odd" Case

I'll denote the number of assignments with $k = p$ as f_p.

Case : n is odd

$$\Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n-1}{2}} f_k}{\sum_{k=0}^{n} f_k}$$

$$\Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n-1}{2}} f_k}{\sum_{k=0}^{\frac{n-1}{2}} f_k + \sum_{k=\frac{n+1}{2}}^{n} f_k}$$

The Proof of Claim 1 : The "n is odd" Case

I'll denote the number of assignments with $k = p$ as f_p.

Case : n is odd

$$Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n-1}{2}} f_k}{\sum_{k=0}^{n} f_k}$$

$$Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n-1}{2}} f_k}{\sum_{k=0}^{\frac{n-1}{2}} f_k + \sum_{k=\frac{n+1}{2}}^{n} f_k}$$

Since $f_p = f_{n-p}$,

$$Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n-1}{2}} f_k}{\sum_{k=0}^{\frac{n-1}{2}} f_k + \sum_{k=0}^{\frac{n-1}{2}} f_k}$$

The Proof of Claim 1 : The "n is odd" Case

The "n is odd" Case : Claim 1

$$Pr(k \leq \frac{n}{2}) = \frac{1}{2}$$

.

The Proof of Claim 1 : The "n is even" Case

Case : n is even

$$\Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n}{2}} f_k}{\sum_{k=0}^{n} f_k}$$

The Proof of Claim 1 : The "n is even" Case

Case : n is even

$$Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n}{2}} f_k}{\sum_{k=0}^{n} f_k}$$

$$Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n}{2}} f_k}{\sum_{k=0}^{\frac{n}{2}-1} f_k + f_{\frac{n}{2}} + \sum_{k=\frac{n}{2}+1}^{n} f_k}$$

The Proof of Claim 1 : The "n is even" Case

Case : n is even

$$\Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n}{2}} f_k}{\sum_{k=0}^{n} f_k}$$

$$\Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n}{2}} f_k}{\sum_{k=0}^{\frac{n}{2}-1} f_k + f_{\frac{n}{2}} + \sum_{k=\frac{n}{2}+1}^{n} f_k}$$

Since $f_p = f_{n-p}$,

$$\Pr(k \leq \frac{n}{2}) = \frac{\sum_{k=0}^{\frac{n}{2}} f_k}{\sum_{k=0}^{\frac{n}{2}-1} f_k + f_{\frac{n}{2}} + \sum_{k=0}^{\frac{n}{2}-1} f_k} > \frac{1}{2}$$

The Proof of Claim 1 : The General Case

The "n is even" Case : Claim 1

$$\Pr(k \leq \frac{n}{2}) > \frac{1}{2}$$

.

The Proof of Claim 1 : The General Case

The "n is even" Case : Claim 1

$$Pr(k \leq \frac{n}{2}) > \frac{1}{2}$$

.

The General Case : Claim 1

$$Pr(k \leq \frac{n}{2}) \geq \frac{1}{2}$$

.

"Good" and "Bad" variables

- *Def*^{n} : Good variable = a value of the variable of the assignment that differs from $\mathbf{a}^*$.

"Good" and "Bad" variables

- Def^n : Good variable = a value of the variable of the assignment that differs from $\mathbf{a}^*$.
- Def^n : Bad variable = a value of the variable of the assignment that is same of $\mathbf{a}^*$.

"Good" and "Bad" variables

- *Def^n* : Good variable = a value of the variable of the assignment that differs from $\mathbf{a}^*$.
- *Def^n* : Bad variable = a value of the variable of the assignment that is same of $\mathbf{a}^*$.
- If the clause is violated, there should be at least one "Good variable"

"Good" and "Bad" variables

- *Def^n* : Good variable = a value of the variable of the assignment that differs from $\mathbf{a}^*$.
- *Def^n* : Bad variable = a value of the variable of the assignment that is same of $\mathbf{a}^*$.
- If the clause is violated, there should be at least one "Good variable"
- Therefore if we choose to flip one variable uniformly random in a violated clause,
  - it would be a "Good variable" with at least the probability of $\frac{1}{3}$
  - it would be a "Bad variable" with at most the probability of $\frac{2}{3}$

Claim 2

Claim 2

$$Pr(\frac{n}{2} \textit{ flips to be "Good variables"}) \geq (\frac{1}{3})^{\frac{n}{2}}$$

.

Using Claim 1 and Claim 2

- Using first claim,

$$\Pr(\mathbf{a_0} \textit{ with } k \leq \frac{n}{2}) \geq \frac{1}{2}$$

Using Claim 1 and Claim 2

- Using first claim,

$$\Pr(\mathbf{a_0}\ \textit{with}\ k \leq \frac{n}{2}) \geq \frac{1}{2}$$

- We want to do $\frac{n}{2}$ consecutive flips for $\mathbf{a_0}$, to make it $\mathbf{a}^*$

Using Claim 1 and Claim 2

- Using first claim,

$$\Pr(\mathbf{a_0} \ \textit{with} \ k \leq \frac{n}{2}) \geq \frac{1}{2}$$

- We want to do $\frac{n}{2}$ consecutive flips for $\mathbf{a_0}$, to make it $\mathbf{a}^*$
- Using second claim,

$$\Pr(\textit{consecutive} \ \frac{n}{2} \ \textit{flips to be} \ \textit{"Good variables"}) \geq (\frac{1}{3})^{\frac{n}{2}}$$

Using Claim 1 and Claim 2

- Using first claim,

$$\Pr(\mathbf{a_0}\ with\ k \leq \frac{n}{2}) \geq \frac{1}{2}$$

- We want to do $\frac{n}{2}$ consecutive flips for $\mathbf{a_0}$, to make it $\mathbf{a}^*$
- Using second claim,

$$\Pr(consecutive\ \frac{n}{2}\ flips\ to\ be\ "Good\ variables") \geq (\frac{1}{3})^{\frac{n}{2}}$$

$$\Pr(finding\ a\ satisfying\ assignment\ in\ a\ single\ iteration) \geq \frac{1}{2 \cdot 3^{\frac{n}{2}}} = p$$

Failure Probability

Failure Probability

With T iterations, the failure probability is at most $\frac{1}{n^d}$.

Failure Probability

Failure Probability

With T iterations, the failure probability is at most $\frac{1}{n^d}$.

$$\Pr(\textit{not finding a satisfying assignment in } T \textit{ iterations}) \leq (1-p)^T$$

- (Using $1 + x \leq e^x$),

$$(1-p)^T \leq e^{-pT}$$

Failure Probability

Failure Probability

With T iterations, the failure probability is at most $\frac{1}{n^d}$.

$$\Pr(\textit{not finding a satisfying assignment in } T \textit{ iterations}) \leq (1-p)^T$$

- (Using $1 + x \leq e^x$),

$$(1-p)^T \leq e^{-pT}$$

- *Choose*, $T = \frac{d \ln n}{p}$

Failure Probability

Failure Probability

With T iterations, the failure probability is at most $\frac{1}{n^d}$.

$$\Pr(\textit{not finding a satisfying assignment in } T \textit{ iterations}) \leq (1-p)^T$$

- (Using $1 + x \leq e^x$),

$$(1-p)^T \leq e^{-pT}$$

- *Choose*, $T = \frac{d \ln n}{p}$
- $(1-p)^T \leq e^{-pT} = e^{-\ln(n^d)} = \frac{1}{n^d}$

Time Complexity

- The outer loop,

$$T = \frac{d\ln(n)}{p}$$

Substitute $p = \frac{1}{2.3^{\frac{n}{2}}}$,

Time Complexity

- The outer loop,

$$T = \frac{d \ln(n)}{p}$$

Substitute $p = \frac{1}{2.3^{\frac{n}{2}}}$,

Time Complexity

- The outer loop,

$$T = \frac{d\ln(n)}{p}$$

Substitute $p = \frac{1}{2.3^{\frac{n}{2}}}$,

$$T = \frac{d\ln(n)}{\frac{1}{2\cdot 3^{\frac{n}{2}}}} = 2d(\sqrt{3})^n \ln(n) = \Theta((\sqrt{3})^n \log(n))$$

Time Complexity

Conclusion

Taking $T = \Theta((1.74)^n \log n)$, the random search algorithm is correct with a high probability.

Analysis Part 2 – By Erick

Planning

- Keep the algorithm the same
  - Repeat T times

Planning

- Keep the algorithm the same
  - Repeat T times
- But prove better bound
  - Smaller T
  - Better analysis gives less iteration
  - Faster running time!

Observation on Version 1

Success probability of an iteration in Version 1

$$\Pr[\text{success}] \geq \frac{1}{2} \cdot \left(\frac{1}{3}\right)^{\frac{n}{2}}$$

- Only count initial assignments $\mathbf{a}_0$ where initial distance $k \leq \frac{n}{2}$

Observation on Version 1

Success probability of an iteration in Version 1

$$\Pr[\text{success}] \geq \frac{1}{2} \cdot \left(\frac{1}{3}\right)^{\frac{n}{2}}$$

- Only count initial assignments $\mathbf{a}_0$ where initial distance $k \leq \frac{n}{2}$
  - Ignore the ones with initial distance $k > \frac{n}{2}$
  - Even though inner loop repeat n times

Observation on Version 1

Success probability of an iteration in Version 1

$$\Pr[\text{success}] \geq \frac{1}{2} \cdot \left(\frac{1}{3}\right)^{\frac{n}{2}}$$

- Only count initial assignments $\mathbf{a}_0$ where initial distance $k \leq \frac{n}{2}$
  - Ignore the ones with initial distance $k > \frac{n}{2}$
  - Even though inner loop repeat n times
- Want to count all values of initial distance k
  - Let the success probability be a function of k

Initial Assignment Probability

- Probability an initial assignment $\mathbf{a}_0$ having initial distance k?
  - Flip a sequence of n coins and get k heads

$$\Pr[\,\mathrm{dist}(\mathbf{a}_0, \mathbf{a}^*) = k\,] = ?$$

Initial Assignment Probability

- Probability an initial assignment $\mathbf{a}_0$ having distance k :

$$\Pr[\,\mathrm{dist}(\mathbf{a}_0, \mathbf{a}^*) = k\,] = \binom{n}{k} 2^{-n}$$

Success Probability

- Probability an iteration succeeds:

$$\Pr[\text{success}] = \sum_{k=0}^{n} \Pr[\text{dist}(\mathbf{a}_0, \mathbf{a}^*) = k] \cdot \Pr[\text{success} \mid \text{dist}(\mathbf{a}_0, \mathbf{a}^*) = k]$$

$$\geq$$

Success Probability

- Probability an iteration succeeds:

$$
\begin{aligned}
\Pr[\text{success}] &= \sum_{k=0}^{n} \Pr[\text{dist}(\mathbf{a}_0, \mathbf{a}^*) = k] \cdot \Pr[\text{success} \mid \text{dist}(\mathbf{a}_0, \mathbf{a}^*) = k] \\
&\geq \sum_{k=0}^{n} \binom{n}{k} 2^{-n} \left(\frac{1}{3}\right)^k \\
&=
\end{aligned}
$$

Success Probability

- Probability an iteration succeeds :

$$
\begin{aligned}
\Pr[\,\text{success}\,] &= \sum_{k=0}^{n} \Pr\left[\,\text{dist}(\mathbf{a}_0, \mathbf{a}^*) = k\,\right] \cdot \Pr[\,\text{success} \mid \text{dist}(\mathbf{a}_0, \mathbf{a}^*) = k\,] \\
&\geq \sum_{k=0}^{n} \binom{n}{k} 2^{-n} \left(\frac{1}{3}\right)^{k} \\
&= 2^{-n} \left(1 + \frac{1}{3}\right)^{n} \\
&= \left(\frac{2}{3}\right)^{n}
\end{aligned}
$$

Outer Loop Iterations

By similar analysis in Version 1,

- A single outer loop iteration success probability at least $p = \left(\frac{2}{3}\right)^n$
- If we take $T = \frac{d \ln n}{p}$ for a constant $d > 0$, then the algorithm succeeds except with inverse polynomial probability $\frac{1}{n^d}$

Outer Loop Iterations

By similar analysis in Version 1,

- A single outer loop iteration success probability at least $p = \left(\frac{2}{3}\right)^n$
- If we take $T = \frac{d \ln n}{p}$ for a constant $d > 0$, then the algorithm succeeds except with inverse polynomial probability $\frac{1}{n^d}$
- Substituting for p, the number of outer loop iterations

$$T = \Theta\left(\left(\frac{3}{2}\right)^n \log n\right)$$

Schöning's Algorithm (Version 2)

Conclusion

Taking $T = \Theta\left((1.5)^n \log n \right)$, the random search algorithm is correct with high probability

Analysis Part 3 – By Dmitrii

Success Probability

$$\Pr[\text{success}] = \sum_{k=0}^{n} \Pr[\text{dist}(\mathbf{a}_0, \mathbf{a}^*) = k] \cdot \Pr[\text{success} \mid \text{dist}(\mathbf{a}_0, \mathbf{a}^*)]$$

Updated Schöning's Algorithm for 3SAT

Let $E = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be the Boolean Expression where C_i is the i-th Clause.
Let Ω be the set of all possible (2^n) truth assignments of E.
repeat T times (or until a satisfying truth assignment is found)
  choose a truth assignment **a** uniformly at random from Ω
  repeat **3n** times (or until **a** satisfies E)
    Choose a clause C violated by the current assignment **a**.
    Choose one of the literals from C uniformly at random, and modify **a** by flipping the value of the corresponding variable.
if a satisfying assignment was found **then**
  return "satisfiable"
else
  return "unsatisfiable"
end if

Complexity: $\mathcal{O}(T \cdot \mathbf{3n})$

Intuition

- Previously we counted only k consecutive "Good variables" from the start

Intuition

- Previously we counted only k consecutive "Good variables" from the start
- k "Bad variables" and $2k$ "Good variables" also lead to success

Updated probability of success

$$\Pr[\,\text{success}\,] = \sum_{k=0}^{n} \Pr\left[\,\text{dist}(\mathbf{a}_0, \mathbf{a}^*) = k\,\right] \cdot \Pr\left[\text{success} \,\middle|\, \text{dist}(\mathbf{a}_0, \mathbf{a}^*)\right]$$

$$\geq \sum_{k=0}^{n} 2^{-n} \binom{n}{k} \cdot \binom{3k}{k} \left(\frac{1}{3}\right)^{2k} \left(\frac{2}{3}\right)^{k}$$

Stirling's approximation

$$n! = \Theta\left(\sqrt{n}\left(\frac{n}{e}\right)^n\right)$$

Approximation of binomial coefficient

$$\binom{3k}{k} = \frac{(3k)!}{(2k)! \cdot k!} = \Theta\left(\frac{\sqrt{3k}}{\sqrt{2k} \cdot \sqrt{k}} \cdot \frac{(\frac{3k}{e})^{3k}}{(\frac{2k}{e})^{2k} \cdot (\frac{k}{e})^{k}}\right) = \Theta\left(\frac{1}{\sqrt{k}} \cdot \frac{3^{3k}}{2^{2k}}\right)$$

Approximation of binomial coefficient 2

$$\binom{3k}{k}\left(\frac{1}{3}\right)^{2k}\left(\frac{2}{3}\right)^{k} = \Theta\left(\frac{1}{\sqrt{k}} \cdot \frac{3^{3k}}{2^{2k}} \cdot 3^{-2k} \cdot \frac{2^k}{3^k}\right) = \Theta\left(\frac{2^{-k}}{\sqrt{k}}\right)$$

Approximation of success probability

$$\Pr[\text{success}] \geq \sum_{k=0}^{n} 2^{-n} \binom{n}{k} \binom{3k}{k} \left(\frac{1}{3}\right)^{2k} \left(\frac{2}{3}\right)^{k}$$

Approximation of success probability

$$\Pr[\text{success}] \geq \sum_{k=0}^{n} 2^{-n} \binom{n}{k} \binom{3k}{k} \left(\frac{1}{3}\right)^{2k} \left(\frac{2}{3}\right)^{k} \geq$$

$$c \cdot 2^{-n} \cdot \sum_{k=0}^{n} \binom{n}{k} \frac{2^{-k}}{\sqrt{k}}$$

Approximation of success probability

$$\Pr[\text{success}] \geq \sum_{k=0}^{n} 2^{-n} \binom{n}{k} \binom{3k}{k} \left(\frac{1}{3}\right)^{2k} \left(\frac{2}{3}\right)^{k} \geq$$

$$c \cdot 2^{-n} \cdot \sum_{k=0}^{n} \binom{n}{k} \frac{2^{-k}}{\sqrt{k}} \geq \frac{c}{\sqrt{n}} \cdot 2^{-n} \cdot \sum_{k=0}^{n} \binom{n}{k} 2^{-k}$$

Approximation of success probability

$$\Pr[\text{success}] \geq \sum_{k=0}^{n} 2^{-n} \binom{n}{k} \binom{3k}{k} \left(\frac{1}{3}\right)^{2k} \left(\frac{2}{3}\right)^{k} \geq$$

$$c \cdot 2^{-n} \cdot \sum_{k=0}^{n} \binom{n}{k} \frac{2^{-k}}{\sqrt{k}} \geq \frac{c}{\sqrt{n}} \cdot 2^{-n} \cdot \sum_{k=0}^{n} \binom{n}{k} 2^{-k} =$$

$$\frac{c}{\sqrt{n}} \cdot 2^{-n} \left(1 + \frac{1}{2}\right)^{n}$$

Approximation of success probability

$$\Pr[\text{success}] \geq \sum_{k=0}^{n} 2^{-n} \binom{n}{k} \binom{3k}{k} \left(\frac{1}{3}\right)^{2k} \left(\frac{2}{3}\right)^{k} \geq$$

$$c \cdot 2^{-n} \cdot \sum_{k=0}^{n} \binom{n}{k} \frac{2^{-k}}{\sqrt{k}} \geq \frac{c}{\sqrt{n}} \cdot 2^{-n} \cdot \sum_{k=0}^{n} \binom{n}{k} 2^{-k} =$$

$$\frac{c}{\sqrt{n}} \cdot 2^{-n} \left(1 + \frac{1}{2}\right)^{n} = \frac{c}{\sqrt{n}} \left(\frac{3}{4}\right)^{n}$$

Schöning's Algorithm (Version 3)

Conclusion

Taking $T = \Theta\left(1.33^n \cdot \sqrt{n} \log n\right)$, the random search algorithm is correct with high probability

Summary

- SAT problem
- Brute force for 3SAT : **Complexity:** $\mathcal{O}(2^n)$
- Schöning's Algorithm for 3SAT
  - Analysis 1 : **Complexity:** $\mathcal{O}(1.74^n \cdot n \log n)$
  - Analysis 2 : **Complexity:** $\mathcal{O}\left(1.5^n \cdot n \log n\right)$
  - Analysis 3 : **Complexity:** $\mathcal{O}(1.33^n \cdot 3n\sqrt{n} \log n)$