Solving Random Satisfiable 3CNF Formulas in Expected Polynomial Time

Michael Krivelevich and Dan Vilenchik

Tel-Aviv University

Outline

1. **Introduction**
   (a) Review of 3CNF and 3SAT
   (b) Random 3CNF
2. The Planted 3SAT Distribution
3. Expected Polynomial Time
4. An Expected Polytime Algorithm for 3SAT
   (a) Description
   (b) Notation
   (c) Write up
   (d) Correctness
5. Runtime Analysis

3CNF

- A formula over the variables $x_1, x_2, \ldots, x_n$ is a conjunction of clauses $C_1, C_2, \ldots, C_m$ where each clause is a disjunction of 3 literals
- Each literal is either a variable or its negation
- A 3CNF is *satisfiable* if there is a boolean assignment to the variables such that every clause contains at least one literal that evaluates to true
- **3SAT** is the language of all satisfiable 3CNF formulas
- Although 2SAT is known to be in P, 3SAT is NP-complete

Random 3SAT

- Uniformly random 3CNFs can be generated by selecting

$$m = m(n)$$

clauses over the variables

$$x_1, x_2, \ldots, x_n$$

- Such random 3CNFs have attracted much study recently
- Results show that random 3SAT has a sharp satisfiability threshold, in the clause-to-variable ratio

Random 3SAT Continued

- Below this ratio, a random 3CNF is satisfiable **whp**
- Above the ratio it is unsatisfiable **whp**
- However the threshold t is unknown, only that it is

$$3.42 \leq t \leq 4.5$$

- However it is unknown if t is even independent of n

Outline

1. Introduction
   (a) Review of 3CNF and 3SAT
   (b) Random 3CNF
2. **The Planted 3SAT Distribution**
3. Expected Polynomial Time
4. An Expected Polytime Algorithm for 3SAT
   (a) Description
   (b) Notation
   (c) Write up
   (d) Correctness
5. Runtime Analysis

Planted 3SAT

- This work concentrates on formulas with a clause-to-variable ratio $\geq t$.
- Most formulas are unsatisfiable, and the analysis of such a distribution seems hard.
- The focus is ultimately then on the *planted* 3SAT distribution, denoted:

$$\mathbb{P}_{p,n}$$

- A planted instance in $\mathbb{P}_{p,n}$ is generated by:
  1. Picking at random a truth assignment φ to the n variables
  2. Each clause satisfied by φ is included with probability p

Planted 3SAT Continued

- This work considers $p \geq d/n^2$, where d is a sufficiently large constant
- The analog of this distribution has been studied in planted clique, planted bisection, and planted coloring
- The planted 3SAT has been studied with greedy variable assignment and proven to work **whp** when p is dense $p = \Theta(1/n)$ with indications it will work with sparser inputs.

Outline

1. Introduction
   (a) Review of 3CNF and 3SAT
   (b) Random 3CNF
2. The Planted 3SAT Distribution
3. **Expected Polynomial Time**
4. An Expected Polytime Algorithm for 3SAT
   (a) Description
   (b) Notation
   (c) Write up
   (d) Correctness
5. Runtime Analysis

Expected Polynomial Time

- An algorithm that works **whp** works well on typical instances
- However, on rare instances, it may not produce a solution at all
- An algorithm may be required to provide an answer, *and* perform well on average.
- An algorithm $\mathcal{A}$ with running time $t_{\mathcal{A}}(I)$ on an input instance I, has an *expected polynomial running time* over a distribution $\mathcal{D}$ if

$$\sum_{I} t_{\mathcal{A}}(I) \cdot Pr_{\mathcal{D}}[I]$$

is polynomial

Expected Polytime Continued

- To have an expected polytime algorithm, a distinction needs to be made between:
  - Typical instances that are “easy” and run in polynomial time
  - Atypical instances that are rare and run in superpolynomial time
- The amount of work done by the algorithm should be inversely proportional to the likelihood that instance occurs
- To understand the frequency of occurrence of the various instances requires a deep understanding of the problem domain, and can lead to more robust and efficient algorithms

Results

This work sets out to prove the following theorem:

> There exists an algorithm SAT that solves 3SAT, and runs in expected polynomial time over the planted SAT distribution with $p \geq d/n^2$, d a sufficiently large constant.

Outline

1. Introduction
   (a) Review of 3CNF and 3SAT
   (b) Random 3CNF
2. The Planted 3SAT Distribution
3. Expected Polynomial Time
4. **An Expected Polytime Algorithm for 3SAT**
   (a) Description
   (b) Notation
   (c) Write up
   (d) Correctness
5. Runtime Analysis

An Expected Polytime Algorithm

- The algorithm searches for the planted assignment (assuming an input sampled from $\mathbb{P}_{d/n^2,n}$) but may find another satisfying assignment along the way
- The first step is to count the number of occurrences of a literal and its negation and assign the variable according to the majority
- **whp** this is already a satisfying assignment reconstructed from the planted assignment when $p \geq d \log n/n^2$
- However when $p = d/n^2$, this assignment is wrong on a fraction of the variables $\exp\{-\Theta(n^2 p)\}$ (constant in this work)

An Expected Polytime Algorithm

- Next, variables suspected to be incorrectly assigned are peeled off leaving a partial assignment
- Finally, exhaustive search is done over the $O(\log n)$ variables remaining at this point

Notation

$V = \{x_1, x_2, \ldots, x_3\}$ the set of all variables

$G(U, E)$ the residual graph with $U \subseteq V$ vertices whose edges are induced by a clause in I containing both endpoints

ψ**, partial truth assignment** a vector of length n over $\{0, 1, *\}$ where $*$ means not assigned

x **supports** a clause C when it is the only variable to satisfy C and the other 2 variables in the clause are also assigned in ψ

$I|_\psi$ the 3CNF I simplified according to ψ

$\pi(Y)$ is all possible assignments of Y

Algorithm 1: SAT

In the first part of the algorithm, the majority vote is taken, and assignments that are suspected of being wrong are removed.

1. MAJ $\leftarrow$ the majority vote assignment
2. Set i = 0 and $\psi_0 \leftarrow$ MAJ
3. **while** there exists a variable x whose support is less than $(1-\epsilon)d/2$ w.r.t. ψ_i **do**
4. $\psi_{i+1} \leftarrow \psi_i$ with the variable x unassigned
5. $i \leftarrow i+1$
6. **end while**

ϵ is a small constant independent of d, say $\epsilon = 10^{-5}$

Algorithm 1: SAT Continued

7. Let ψ be the final partial assignment, and U be the set of unassigned variables in ψ
8. Construct the residual graph $G(U, E)$ and find the set of connected components $\{\Gamma_i\}$
9. **for** $y = 0$ to $|V \backslash U|$ **do**
10. **for all** $Y \subseteq V \backslash U$, $|Y| = y$ and every assignment $\pi(Y)$ to the variables of Y **do**
11. Let $\psi \circ \pi(Y)$ be $\pi(x)$ if $x \in Y$, and $\psi(x)$ otherwise.
12. **if** $I|_{\psi \circ \pi(Y)}$ doesn't induce an empty clause **then**
13. Independently in every Γ_i, using exhaustive search try to satisfy $I|_{\psi \circ \pi(Y)}[\Gamma_i]$
14. **if** the search succeeded **return** the satisfying assignment **end if**
15. **end if**
16. **end for**
17. **end for**
18. **return** I is not satisfiable.

Correctness

- If the algorithm returns an assignment (line 14), it satisfies I
- If I is satisfiable, in the worst case exhaustive search is done over all possible assignments of the variables
- If I is unsatisfiable, one of the conditions on lines 12 and 14 will always fail, and *not satisfiable* is returned by line 18

Outline

1. Introduction
   (a) Review of 3CNF and 3SAT
   (b) Random 3CNF
2. The Planted 3SAT Distribution
3. Expected Polynomial Time
4. An Expected Polytime Algorithm for 3SAT
   (a) Description
   (b) Notation
   (c) Write up
   (d) Correctness
5. **Runtime Analysis**

Properties of Random Instances

- A truth assignment φ is picked uniformly at random from $\mathbb{P}_{d/n^2,n}$
- Every clause satisfied by φ is included in the formula with probability $p = d/n^2$, with I clearly being satisfied
- There are $(2^3 - 1) \cdot \binom{n}{3}$ clauses satisfied by φ
- $\mathbf{E}[I] = p \cdot 7 \cdot \binom{n}{3} = 7dn/6 + o(n)$
- Every variable x supports $\binom{n-1}{2}$ clauses w.r.t. φ
- $\mathbf{E}[\text{support of } x \text{ in } I] = p \cdot \binom{n-1}{2} = d/2 + o(1)$ which explains the choice in line 3

Majority Vote

- We want to bound the number of variables in I that the majority vote disagrees with the planted assignment
- We can apply the Chernoff bound if we treat the number of clauses x appears in positively and negatively as two binomially distributed **rv**
- Thus the probability that the majority vote on x disagrees with the planted assignment (F_x) is:

$$Pr[F_x] \leq exp\{-\Theta(d)\}$$

Majority Vote

- Applying the union bound we get the probability of the number of disagreed upon variable assignments being greater than αn is:

$$\leq \binom{n}{\alpha n} Pr[F_{x_1} \wedge F_{x_2} \wedge \ldots F_{x_{\alpha n}}]$$

- However this is difficult to estimate since the F_x's are not independent
- The proof involves showing that the influence of a certain assignment does not affect overly many clauses, and can be bounded by counting all such variables. The complete proof is shown in the paper in Appendix A

Support and Dense Subformulas

- The bounds for the number of supported clauses of a variable x is computed similarly to Majority Vote, but each event here is in fact independent.
- This yields the bound:

$$\leq exp\{-\epsilon^2 d/16\}$$

- We also need to prove **whp** a random graph does not contain a small induced subgraph with a large average degree.
- This bound is straightforward when we count the number of clauses containing at least 2 variables from a subset of the possible variables

Expanding Set of Variables

- Let H be a subset of the variables such that every variable in H supports at least $(1-\epsilon)d/2$ clauses in $I[H]$ w.r.t. both the planted assignment and the majority vote
- The probability that $\bar{H}$ is large ($\geq \alpha n$) can be bounded by:

$$\leq exp\{-\epsilon^2 d/100 \cdot \alpha n\}$$

Construction of H

- Let W be the set of variables whose majority vote disagrees with φ
- Let B be the set of variables whose support w.r.t. φ is less than $(1-\epsilon/2)d/2$
- Let $H \subset V$ be constructed as follows:
  1. Let $H_0 = V \backslash (B \cup W)$
  2. While there exists a variable $a_i \in H_i$ which supports less than $(1-\epsilon)d/2$ clauses in $I[H_i]$, define $H_{i+1} = H_i \backslash \{a_i\}$
  3. Let a_m be the last variable removed at step 2. Define $H = H_{m+1}$

Analysis of the Expected Running Time

- The majority vote sets H correctly due to the construction of H
- H survives the reassignment step due to its expansion properties w.r.t. MAJ
- Therefore, by step 7, $I[H]$ is satisfied by ϕ
- It remains to satisfy (if not satisfied so far) $I \backslash I[H]$
- Therefore **whp** $I \backslash I[H]$ breaks down to small-size mutually-disjoint subformulas allowing exhaustive search to run in expected polynomial time

Analysis of the Expected Running Time

- The expected running time of the algorithm $\mathcal{A}$ is bounded by:

  $$\mathbf{E}[R_{MAIN}] + \mathbf{E}[R_{REC} \cdot R_{EXH}] + \mathbf{E}[R_{REC}] \cdot \mathbf{E}[R_{SIMPLIFY}]$$

- $\mathbf{E}[R_{MAIN}]$ and $\mathbf{E}[R_{SIMPLIFY}]$ are $O(dn)$
- $\mathbf{E}[R_{REC}] \leq O(1)$
- $\mathbf{E}[R_{REC} \cdot R_{EXH}] = n^{1+\Theta(1/)}$
- Therefore, $\mathbf{E}[\mathrm{R}(\mathcal{A})] \leq n^{1+\Theta(1/d)}$

Conclusion

- A general expected polytime algorithm for 3SAT obviously is unreasonable to expect
- However over the planted assignment $\mathbb{P}_{d/n^2,n}$ it is possible to limit the amount of exhaustive search that must be done so that on average the algorithm runs in polynomial time