New Upper Bounds for Formula SAT

Rahul Santhanam
University of Edinburgh

Plan of the Talk

- Introduction
- A New Upper Bound for Formula SAT
  - The Algorithm
  - The Analysis
- Extending the Technique to Other Contexts
- Future Directions

Plan of the Talk

- *Introduction*
- A New Upper Bound for Formula SAT
  - The Algorithm
  - The Analysis
- Extending the Technique to Other Contexts
- Future Directions

Fighting Perebor

- When can we beat brute-force search for NP-hard problems?
- In practice, we do need to solve SAT instances in various contexts (planning, verification etc.)
  - What can we say formally about algorithms solving SAT?

Satisfiability Variants

- k-SAT: Satisfiability of k-CNFs
- CNF SAT: Satisfiability of formulae in conjunctive normal form
- Formula SAT: Satisfiability of arbitrary Boolean formulae
- Circuit SAT: Satisfiability of arbitrary Boolean circuits
  - Constant-depth Circuit SAT: Satisfiability of constant-depth circuits

Satisfiability Variants

Circuit SAT

Formula SAT

Constant-depth Circuit SAT

CNF SAT

k-SAT

Algorithms for Satisfiability

- m: input size; n: number of variables
- Brute-force algorithm runs in time $2^n \text{poly}(m)$
- We are interested in algorithms running in time $2^{n-f(n)}\text{poly}(m)$, for f(n) asymptotically as large as possible
  - We call f the *savings* of the algorithm

Main Algorithmic Paradigms

- DLL: Search for a solution by iteratively setting variables, and backtracking if a full assignment does not yield a solution
- Random Walk: Start with an arbitrary assignment, and iteratively modify it to satisfy random unsatisfied clauses

Algorithms for Satisfiability: State of the Art

- 3-SAT: ~ 1.3^n [R06]
- k-SAT: Savings $\Omega(n/k)$ [PPSZ98, S99]
- CNF SAT: Savings $\Omega(n/\log(m/n))$ [S05]
- Constant-depth Circuit SAT: Savings $\Omega(n)$ for $m = O(n)$ [CIP09]
- Formula SAT: ?
- Circuit SAT: ?

Our Results

- Theorem 1: Formula SAT can be solved in time $2^{n-\Omega(n)}$ on formulae of linear size
- Theorem 2: The number of satisfying assignments of linear-size formulae can be computed exactly in time $2^{n-\Omega(n)}$
- Theorem 3: Any linear size formula fails to compute Parity on at least a $1/2 - 1/2^{\Omega(n)}$ fraction of inputs of length n

Barriers to Improved SAT Algorithms

- Exponential-Time Hypothesis (ETH) formulated by [IP99,IPZ01]: 3-SAT cannot be solved in time $2^{o(n)}$
  - Under ETH, we cannot achieve savings n-o(n) on k-SAT or Formula SAT/Circuit SAT on linear size
- [W09] shows that savings of ω(log(n)) for Circuit SAT for superpoly m would imply NEXP not in SIZE(poly) (and analogously for Formula SAT)

Plan of the Talk

- Introduction
- *A New Upper Bound for Formula SAT*
  - The Algorithm
  - The Analysis
- Extending the Technique to Other Contexts
- Future Directions

The Upper Bound for Formula SAT

- Theorem 1 (restated): There is a constant $k > 2$ for which there is a deterministic algorithm solving Formula SAT with savings n/c^k on formulae of size cn
- Note: if we could achieve $k = 0.1$, then by [W09] we would have new formula size lower bounds for E^{NP}

Plan of the Talk

- Introduction
- A New Upper Bound for Formula SAT
  - *The Algorithm*
  - The Analysis
- Extending the Technique to Other Contexts
- Future Directions

The Algorithm

- Search(ϕ)
  - Simplify ϕ according to simplification rules
  - If $\phi \leftrightarrow 1$, return “yes” and halt
  - If $\phi \leftrightarrow 0$, return “no”
  - Let x be the variable with max no of occurrences
  - Search ($\phi|_{x=0}$)
  - Search ($\phi|_{x=1}$)
  - Return “no”

Simplification Rules

- $1 \wedge \phi \rightarrow \phi$
- $1 \vee \phi \rightarrow 1$

1-simplification rules

- $0 \wedge \phi \rightarrow 0$
- $0 \vee \phi \rightarrow \phi$

0-simplification rules

- $x \vee \phi \rightarrow x \vee \phi|_{x=0}$
- $x \wedge \phi \rightarrow x \wedge \phi|_{x=1}$

Variable simplification rules

An Example

$(x \vee y) \wedge (x \vee (x \wedge y' \wedge z))$

An Example

$(x \vee y) \wedge (x \vee (0 \wedge y' \wedge z))$
(applying variable simpl. rule)

An Example

(x V y) Λ (x V 0)
(applying 0-simpl. rule)

An Example

$(x \vee y) \wedge x$
(applying 0-simpl. rule)

An Example

$(x \vee y) \wedge x$

$(0 \vee y) \wedge 0$

1st recursive call

An Example

$(x \vee y) \wedge x$

$y \wedge 0$

An Example

$(x \vee y) \wedge x$

0

An Example

$(x \vee y) \wedge x$

0 $(1 \vee y) \wedge 1$

2nd recursive call

An Example

$(x \vee y) \wedge x$

0 $1 \wedge 1$

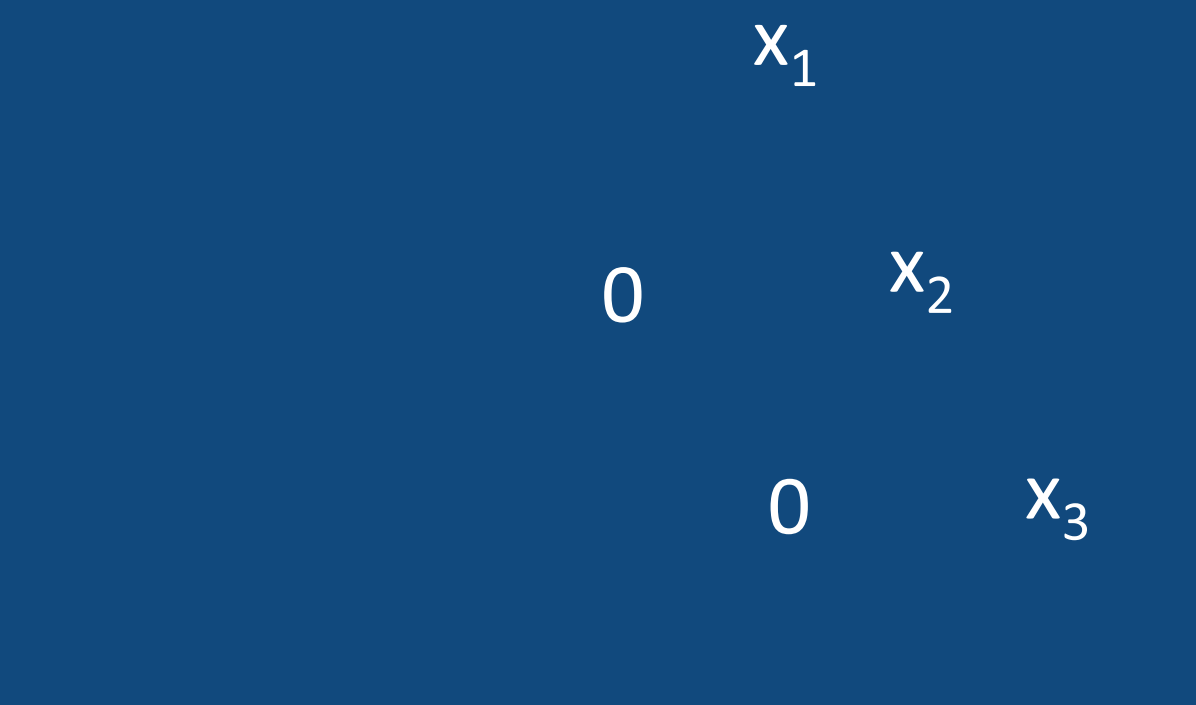
An Example

$(x \vee y) \wedge x$

0 1

Success!

Decision Trees



$\Phi = x_1 \Lambda x_2 \Lambda x_3$

Recursion Tree → Decision Tree

- Running the algorithm on a formula yields a *decision tree* for the Boolean function computed by the formula
- Analyzing the running time reduces to bounding the *size* (no. of leaves) of this decision tree

Plan of the Talk

- Introduction
- A New Upper Bound for Formula SAT
  - The Algorithm
  - *The Analysis*
- Extending the Technique to Other Contexts
- Future Directions

Analyzing Decision Tree Size

- Typically done by solving a recurrence on m and n
- Instead, we derive inspiration from the method of *random restrictions* (though our algorithm itself is deterministic)

Pure Random Restrictions

- Let $0 < p < 1$ be a parameter. Given n input variables, choose each one independently to be 1 w.p. (1-p)/2, 0 w.p. (1-p)/2 and free w.p p
- Note that the choice of *which* variables to set is uniform, as well as the choice of *which* value to set a given variable to

Formula Size Lower Bounds via Pure Random Restrictions

- [S61] proved that when a formula of size m is hit by a random restriction with parameter p, expected size of simplified formula is $O(p^{1.5}m)$
  - Implies Parity requires formulae of size $\Omega(n^{1.5})$, by choosing $p = O(1/n)$
- [H98] proved optimal result: expected size of simplified formula is $O(p^2m)$
  - Implies best known formula size lower bound of $n^{3-o(1)}$ for a function in P

Formula Size Lower Bounds via Pure Random Restrictions

- [S61] proved that when a formula of size m is hit by a random restriction with parameter p, expected size of simplified formula is $O(p^{1.5}m)$
- [H98] proved optimal result: expected size of simplified formula is $O(p^2m)$
- Note that for either result, if $m = O(n)$, there is constant p such that expected size of simplified formula $<< pn$

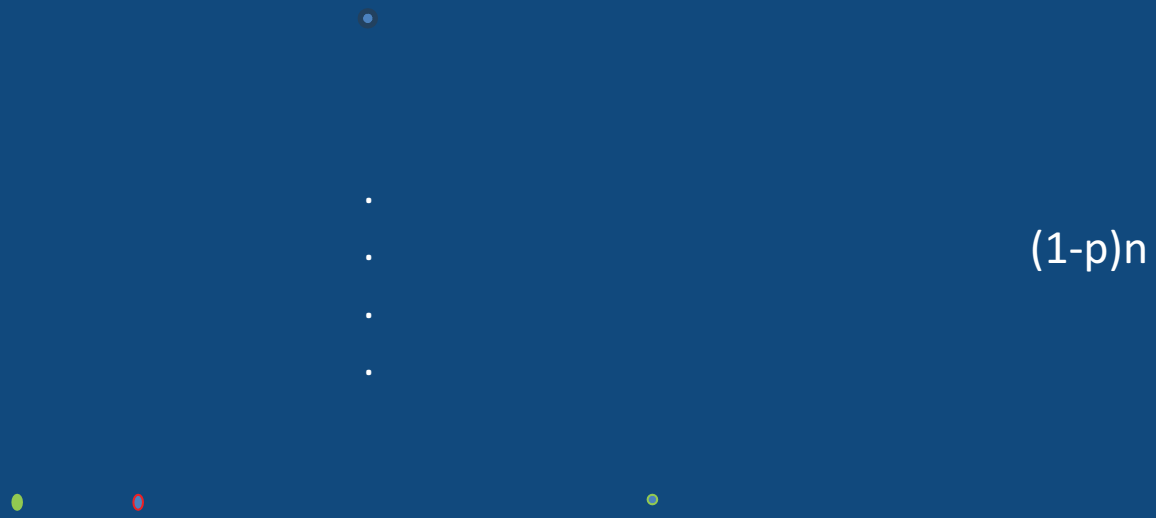
Adaptively Random Restrictions

- Choice of which variable to set next is not uniform
  - Indeed, in our algorithm, setting of variables is *deterministic*, according to number of occurrences
- Choice of value, however, is uniformly random
- Greedy a.r.r: Variables are set sequentially in decreasing order of no. of occurrences
- [S61] and [H98] results hold also for (1-p)n -step greedy a.r.r

Random Restrictions and Decision Tree Size: Basic Idea

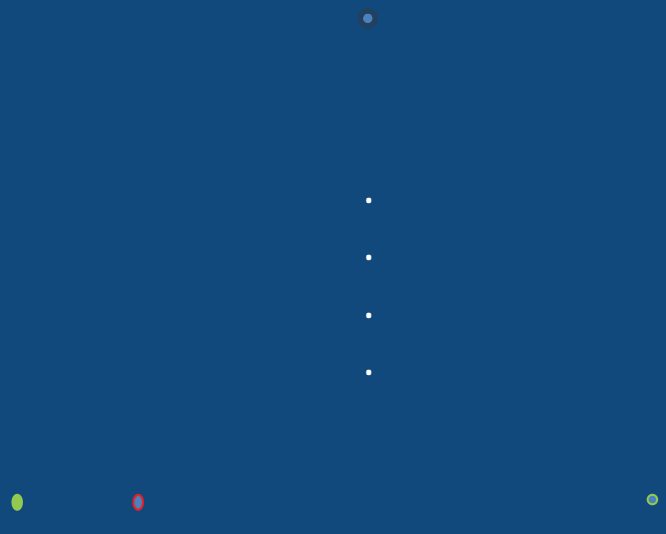
- The simplified formulae at depth d of the decision tree correspond to d-step greedy a.r.r
- Our goal: Prove that size of simplified formula after (1-p)n steps of greedy a.r.r << pn with prob. $1-2^{-\Omega(n)}$ (strong concentration version of Subbotovskaya's result)
- This will give us non-trivial bound on size of decision tree

Why a Concentration Bound Helps



Good node: Simplified formula at node has size < pn/2
Bad node: Simplified formula has size >= pn/2

Why a Concentration Bound Helps



(1-p)n

Say we could show that fraction of bad nodes at depth (1-p)n is at most q. Then size of decision tree is at most $2^{n-pn/2} + q2^n$, which is $2^{n-\Omega(n)}$ if $q=2^{-\Omega(n)}$

Proof Idea for the Concentration Bound

- Basic idea: Ratio m/n of size of formula to no. of variables tends to decrease as we go down decision tree since we are setting variables according to no. of occurrences
- Suppose variable x we are setting occurs in K OR gates and L AND gates (wlog positively)
- $x \rightarrow 1$ means m decreases by at least $2K + L$
- Dually, $x \rightarrow 0$ means decrease of $2L + K$
- So, with prob >= 1/2, m/n decreases

Proof Idea for the Concentration Bound (Contd.)

- Basic idea: Ratio m/n of size of formula to no. of variables tends to decrease as we go down decision tree since we are setting variables according to no. of occurrences
- So, with prob >= 1/2, m/n decreases after one step of greedy a.r.r
- Intuition: After (1-p)n steps, w.h.p. enough decreases occur so that m << pn

Pure vs Adaptive

- Unclear how to use pure restrictions to analyze deterministic algorithms
- More importantly, strong concentration *does not hold* in general for pure restrictions

Beating Brute Force Search for Exact Count (Theorem 2)

- Count(ϕ;n)
  - Simplify ϕ according to simplification rules
  - If $\phi \leftrightarrow 1$, return 2^n
  - If $\phi \leftrightarrow 0$, return 0
  - Let x be the variable with max no of occurrences
  - Return Count($\phi|_{x=0}$;n-1) + Count($\phi|_{x=1}$;n-1)
- Analysis same as before, giving same runtime

Average Case Lower Bounds for Formula Size (Theorem 3)

- Proof of Theorem 1 shows that formulae of linear size have decision trees of size $2^{n-\Omega(n)}$
- Let *advantage* of a decision tree T on a function f be $\Pr(T=f) - \Pr(T\neq f)$
- Lemma: Any decision tree of size s has advantage at most $s/2^n$ on Parity
- Corollary : Any formula of linear size has advantage $2^{-\Omega(n)}$ on Parity

Plan of the Talk

- Introduction
- A New Upper Bound for Formula SAT
  - The Algorithm
  - The Analysis
- *Extending the Technique to Other Contexts*
- Future Directions

Analysis First, Algorithm Afterwards

- Could other random restriction results be used to get new upper bounds?
- Hastad's (pure) random restriction results give strong lower bounds for constant-depth circuits
- Does this imply anything for Constant-depth Circuit SAT?

An Upper Bound for Constant-depth Circuit SAT

- Pure random restrictions correspond to random restriction order of variables
- Consider the following simple algorithm
  - Choose random ordering of variables
  - Search the assignment tree according to this ordering
- Essentially k-SAT algorithm of [PPZ97]
- Hastad's switching lemma gives savings of $n^{1/(d+1)}$ for this algorithm, where d is depth
- Can we do better?

An Anachronistic Conjecture

- Conjecture: Any function computed by poly-size circuits of depth d has decision trees of size $2^{n-n/\text{polylog}(n)}$
- Hope is that proof of this conjecture will also lead to a satisfiability algorithm with the same running time

An Anachronistic Conjecture

- Conjecture: Any function computed by poly-size circuits of depth d has decision trees of size $2^{n-n/\text{polylog}(n)}$
- Why anachronistic? Well, Ajtai might already have solved this conjecture 30 years ago...
- Needed: A proof which is comprehensible and constructive, whose techniques can be used in other contexts

Satisfiability for Linear-Size Constant-Depth Circuits

- [CIP09] give algorithm with $\Omega(n)$ savings, however constant in the Ω is reciprocal of double exponential in d (the depth)
- [LW10] prove that any linear-size circuit of depth d has advantage at most $2^{-\Omega(n)}$ in computing Parity
- Good news: Their proof is based on adaptively random restrictions

Satisfiability for Linear-Size Constant-Depth Circuits

- [CIP09] give algorithm with $\Omega(n)$ savings, however constant in the Ω depends doubly exponentially on d (the depth)
- [LW10] prove that any linear-size circuit of depth d has advantage at most $2^{-\Omega(n)}$ in computing Parity
- Bad news: The constant in *their* Ω term goes down as a *tower* of height d

Plan of the Talk

- Introduction
- A New Upper Bound for Formula SAT
  - The Algorithm
  - The Analysis
- Extending the Technique to Other Contexts
- *Future Directions*

Future Directions

- Using the method of random restrictions in other settings or to get better parameters
- A comprehensible (and constructive) proof of the Anachronistic Conjecture
- More connections between upper bounds and lower bounds

Thank You!