3-SAT Faster and Simpler - Unique-SAT Bounds for PPSZ Hold in General

Timon Hertli

Institute for Theoretical Computer Science
Department of Computer Science
ETH Zürich, Switzerland

October 10, 2011

China Theory Week 2011

The 3-SAT problem

- Given a logical formula in 3-CNF on n variables
  $F := \underbrace{(x \vee y \vee \overline{z})}_{clause} \wedge (\overline{x} \vee a \vee \overline{y}) \dots$
- Does there exist an assignment
  $\alpha := \{x \mapsto \alpha(x), y \mapsto \alpha(y), \dots\}$ s.t. F evaluates to true?
  - We call such an assignment *satisfying*

Goal: Moderately Exponential Algorithm

Randomized algorithm for 3-SAT running in time $O(b^n)$ for $b < 2$

History of Randomized 3-SAT algorithms

- At 1998, Paturi, Pudlák, Saks, Zane invented the PPSZ algorithm solving 3-SAT in time $\mathcal{O}(1.364^n)$
- PPSZ finds a **unique** satisfying assignment in $\mathcal{O}(1.308^n)$

History of Randomized 3-SAT algorithms

- At 1998, Paturi, Pudlák, Saks, Zane invented the PPSZ algorithm solving 3-SAT in time $\mathcal{O}(1.364^n)$
- PPSZ finds a **unique** satisfying assignment in $\mathcal{O}(1.308^n)$
  - Schöning's algorithm (1999): $\mathcal{O}(1.334^n)$
  - Combination by Iwama, Tamaki (2004): $\mathcal{O}(1.324^n)$
  - Improved to 1.323^n, 1.322^n, 1.321^n

History of Randomized 3-SAT algorithms

- At 1998, Paturi, Pudlák, Saks, Zane invented the PPSZ algorithm solving 3-SAT in time $\mathcal{O}(1.364^n)$
- PPSZ finds a **unique** satisfying assignment in $\mathcal{O}(1.308^n)$
  - Schöning's algorithm (1999): $\mathcal{O}(1.334^n)$
  - Combination by Iwama, Tamaki (2004): $\mathcal{O}(1.324^n)$
  - Improved to 1.323^n, 1.322^n, 1.321^n
- We show: PPSZ finds a satisfying assignment in $\mathcal{O}(1.308^n)$ in the general case

PPSZ

- One PPSZ-run tries to build a satisfying assignment by fixing variables in F, repeating the following:

A PPSZ-step

- Set all variables we "know"
- Guess a random variable (uniformly at random)

- We "know" x if constantly many clauses imply $x \mapsto 0$ or $x \mapsto 1$.

PPSZ

- One PPSZ-run tries to build a satisfying assignment by fixing variables in F, repeating the following:

A PPSZ-step

- Set all variables we "know"
- Guess a random variable (uniformly at random)

- We "know" x if constantly many clauses imply $x \mapsto 0$ or $x \mapsto 1$.

Theorem (PPSZ)

If x has the same value in all satisfying assignments of F, then x is guessed with probability at most 0.387.

- In the unique case, this holds for all variables. The satisfying assignment is found with probability $\left(\frac{1}{2}\right)^{0.387n} \approx 1.308^{-n}$.

General 3-SAT: Original Analysis

- Original analysis: partition the set of 2^n assignments; unique satisfying assignment in each part
- Worse running time and very complicated analysis
- Our approach: Consider each variable separately, do some bookkeeping

General 3-SAT: Frozen Variables

- If x has has the same value in all satisfying assignments of F, we call x **frozen**
- Otherwise we call it **non-frozen**

General 3-SAT: Frozen Variables

- If x has has the same value in all satisfying assignments of F, we call x **frozen**
- Otherwise we call it **non-frozen**

Observation

If x is non-frozen, it's ok to assign any value

- What's the problem?

General 3-SAT: Frozen Variables

- If x has has the same value in all satisfying assignments of F, we call x **frozen**
- Otherwise we call it **non-frozen**

Observation

If x is non-frozen, it's ok to assign any value

- What's the problem?
- OK: x started frozen or x is non-frozen when assigned
- What if x starts non-frozen, but becomes frozen and is set afterwards?

Handling non-frozen variables

Q: What if x starts non-frozen, but becomes frozen and is set afterwards?

- As soon as x becomes frozen, it will be guessed with probability at most 0.387 in the **remainder**
- Suppose y starts frozen. If it is not set, we expect it's guessing probability to be lower
- Balance x being guessed more with having x around non-frozen

Approach: Quantify this, do the math and hope it works out

Cost Function

Give each intermediate (satisfiable) formula a **cost** $c(F)$

- Cost measures how hard F is for PPSZ
- The cost is the sum of contributions of individual variables
  - Frozen variables contribute their probability to be guessed to the cost
  - Non-frozen variables contribute 0.387 to the cost
- Hence $c(F) \leq 0.387n$
  - In unique case: $c(F)$ is the expected number of variables we have to guess
- We show: PPSZ finds a sat. assignment with probability $2^{-c(F)}$

Analysis in the Unique Case

Remember $c(F) \leq 0.387n$. Suppose F has a unique satisfying assignment, so all variables are frozen

- $c(F)$ decreases by 1 each guessing step on average
  - We expect $c(F)$ variables to be guessed now. We guess a variable. How many variables do we expect to be guessed afterwards?
- Hence there are only $0.387n$ guessing steps on average
- Each guessing step succeeds with probability $1/2$
- The total success probability is $(1/2)^{0.387n} \approx 1.308^{-n}$

Analysis in the General Case

Remember $c(F) \leq 0.387n$. Now consider the general case, where some variables are non-frozen

- Non-frozen variables do not become "less guessed", hence $c(F)$ decreases by less than 1.
- So we have more guessing steps, however the failure probabilty is smaller
- Doing the calculation shows that the total success probability is still at least $2^{-0.387n} \approx 1.308^{-n}$.
  - Because $4 - 4\log 2 \approx 1.23 < 1.44 \approx 1/\log 2$
  - If PPSZ would run in 1.2^n, it wouldn't work anymore

Conclusion

- Using the cost function, we can look at individual PPSZ-steps
- This gives us the flexibility to accomodate non-frozen variables

Open Problems:

- Does PPSZ get even **better** with more assignments?
- Is UNIQUE k-SAT **always** the worst case? (conjectured by Calabro et al.)
- PPSZ derandomized for Unique k-SAT[Rolf, 2005]
- Derandomization for general k-SAT?

Questions?