Improved Upper Bounds for 3-SAT

Electronic Colloquium on Computational Complexity, Report No. 53 (2003)

Kazuo Iwama and Suguru Tamaki
School of Informatics, Kyoto University, Kyoto 606-8501, Japan
{iwama, tamak}@kuis.kyoto-u.ac.jp

1 CNF Satisfiability

The CNF Satisfiability problem is to determine, given a CNF formula F, whether or not there exists a satisfying assignment for F. If each clause of F contains at most k literals, then F is called a k-CNF formula and the problem is called k-SAT. For small k's, especially for $k = 3$, there have been a lot of algorithms which run significantly faster than the trivial 2^n bound. The following list summarizes those algorithms where a constant c means that the algorithm runs in time $O(c^n)$. Roughly speaking most algorithms are based on Davis-Putnam. [Sch99] is the first local search algorithm which gives a guaranteed performance for general instances and [DGH+02], [HSSW02] and [BS03] follow up this Schöning's approach.

3-SAT	4-SAT	5-SAT	6-SAT	type	ref.
1.782	1.835	1.867	1.888	det.	[PPZ97]
1.618	1.839	1.928	1.966	det.	[MS85]
1.588	1.682	1.742	1.782	prob.	[PPZ97]
1.579	-	-	-	det.	[Sch92]
1.505	-	-	-	det.	[Kul99]
1.481	1.6	1.667	1.75	det.	[DGH+02]
1.362	1.476	1.569	1.637	prob.	[PPSZ98]
1.334	1.5	1.6	1.667	prob.	[Sch99]
1.3302	-	-	-	prob.	[HSSW02]
1.3290	-	-	-	prob.	[BS03]
1.324	1.474	-	-	prob.	[*]

2 Our Contribution

Our new bounds are denoted by [*] in the above list, namely we prove:

Theorem 1 *For any satisfiable n-variable 3-CNF (4-CNF) formula F, there exists a randomized algorithm that finds a satisfying assignment of F in expected running time $O(1.324^n)$ $(O(1.474^n))$.*

The basic idea is to combine two existing algorithms, the one by Paturi, Pudlák, Saks and Zane [PPSZ98] and the other by Schöning [Sch99]. It should be noted, however, that simply running the two algorithms independently does not seem to work. Also, our approach can escape one of the most complicated portions in the analysis of [PPSZ98]. In this paper we focus on the 3-SAT case; the 4-SAT case is very similar and may be omitted. The same approach does not improve the bounds for 5-SAT or more.

3 Basic Ideas

The algorithm of [PPSZ98] is called ResolveSat, which is based on a randomized Davis-Putnam combined with bounded resolution. This algorithm has the unique feature that it achieves a quite nice performance, $O(1.3071^n)$, for a unique 3-CNF formurla, i.e., a formula which has only one

ISSN 1433-8092

satisfying assignment. As the number m of satisfying assignments grows, the bound, denoted by $T_{\text{PPSZ}}(m)$, degenerates, i.e., $T_{\text{PPSZ}}(m)$ is an increasing function. [PPSZ98] needed a lot of effort to stop this degeneration by formalizing the intuition that if the formula has many satisfying assignments, then finding one should be easy.

In contrast, the algorithm of [Sch99] is based on the standard local search for which the above intuition is obviously true. Namely its running time $T_{\text{SCH}}(m)$ is the worst when $m = 1$ and then decreases. Recall that $T_{\text{PPSZ}}(1) < T_{\text{SCH}}(1) = O(1.334^n)$. So, if we run the two algorithm in parallel, then its running time is bounded by $\min\{T_{\text{PPSZ}}(m), T_{\text{SCH}}(m)\}$ which becomes maximum $(= T_{\text{PPSZ}}(m_0) = T_{\text{SCH}}(m_0))$ at $m = m_0$. Obviously $T_{\text{SCH}}(m_0) < T_{\text{SCH}}(1)$. Although $T_{\text{SCH}}(1)$ is not the currently best, there is a lot of hope of breaking it since $T_{\text{PPSZ}}(1)$ is much better than the current best.

Unfortunately, this approach has an obstacle. We know the value of $T_{\text{PPSZ}}(m)$ but we do not know that of $T_{\text{SCH}}(m)$ for the following reason. To obtain $T_{\text{SCH}}(m)$, it appears that we need to know the Hamming distance between the (randomly chosen) initial assignment and its closest satisfying assignment. However, there is no obvious way of doing so, since it is quite hard to analyze how (multi) satisfying assignments of a 3-CNF formula can distribute in the whole space of 2^n assignments.

4 Our solution

Both [PPSZ98] and [Sch99] repeat an exponential number of *tries*. Each try of [Sch99], denoted by **SCH**, looks like:

(1)Generate a random initial assignment y.

(2)Execute a local search $3n$ steps starting from y.

Each try of [PPSZ98], denoted by **PPSZ**, has a similar structure, namely:

(1)Generate a random initial assignment y.

(2)Generate a random initial permutation π of $[1, n]$.

(3)Execute Davis-Putnam based on y and π, which takes at most n steps.

As mentioned previously, a simple repetition of **SCH** and **PPSZ** does not work. (It probably works but we cannot analyze.) Our solution is to use the same random assignment for each execution of **SCH** and **PPSZ**. Namely, our algorithm is:

repeat I times

(1)Generate a random initial assignment y.

(2)Only (2) of **SCH**; if a satisfying assignment is found, then answer YES.

(3)Only (2) and (3) of **PPSZ**; if a satisfying assignment is found, then answer YES.

end Answer NO.

Now let p_0 be the probability that the above single try (a single execution of (1)-(3)) finds a satisfying assignment if the given formula is satisfiable. To obtain p_0, there are two key lemmas:

Lemma 1 ([PPSZ98]) *For any satisfiable 3-CNF formula F, the probability that a single try of* **PPSZ** *finds a satisfying assignment, denoted by $\tau(F, z|B_z)$, is bounded as follows:*

$$\tau(F, z|B_z) \geq 2^{-((1-\gamma)-(1-\gamma-\beta)\Delta_z)n-o(n)},$$

where $\gamma = 2 - 2\ln 2$ and $\beta = 1.115$.

We need to know the formal definition of $\tau(F, z|B_z)$ and Δ_z. Let $S(F)$ be the set of satisfying assignments of the formula F. A set of assignments, $B \subseteq \{0, 1\}^n$, is called a *subcube*, if B is determined by fixing a certain number of variables. For example, $\{0000, 0001, 0010, 0011\}$ is a subcube obtained by fixing $x_1 = x_2 = 0$. Now it is not hard to see that the whole space, $\{0,1\}^n$, can always be partitioned into a family $\{B_z \mid z \in S(F)\}$ of disjoint subcubes so that B_z contains $z \in S(F)$ but no other $z' \in S(F) - \{z\}$. Now, given the formula F and the subcube B_z, $\tau(F, z|B_z)$ is defined as the probability (averaged over y) that a single execution of (2) and (3) of **PPSZ** finds the assignment z under the condition that the initial assignment $y \in B_z$. Also Δ_z is defined as

$$\Delta_z = \frac{\log_2(2^n/|B_z|)}{n},$$

namely, it is the ratio of the number of the variables fixed to determine B_z.

Lemma 2 ([Sch99]) *Suppose that* **SCH** *starts from the initial assignment y and there is a satisfying assignment z. Let $d(y,z)$ be the Hamming distance between y and z. Then the probability that a single try of* **SCH** *starting from y finds z is at least* $(1/2)^{d(y,z)}$.

Recall that our new try uses the same y for both **SCH** and **PPSZ**. If $y \in B_z$, then apparently $\Delta_z n$ variables are assigned correct values, i.e., $d(y,z) \le (1 - \Delta_z)n$. Also, by averaging over y:

Lemma 3 *For any satisfiable 3-CNF formula F, the probability that a single try of* **SCH** *finds a satisfying assignment, denoted by $\sigma(F, z|B_z)$, is bounded as follows:*

$$\sigma(F, z|B_z) \ge \left(\frac{3}{4}\right)^{(1-\Delta)n},$$

where, given F and B_z, $\sigma(F, z|B_z)$ is defined as the probability that a single execution of (2) *of* **SCH** *finds z supposing that the initial assignment $y \in B_z$.* (Proof is similar to [Sch99].)

Now our success probability of a single try is at least the probability of Lemmas 1 and 3, i.e.,

$$p_0 \ge \max\{\tau(F, z|B_z), \sigma(F, z|B_z)\},$$

which becomes minimum ($= \Omega(1.3238^{-n})$) when $\Delta_z = 0.02513$. Now the standard probabilistic argument allows us to claim that our algorithm finds a satisfying assignment with high probability for $I = O(1.324^n)$. Recall that I is the number of repetitions.

References

[BS03] S. Baumer and R. Schuler. Improving a probabilistic 3-SAT Algorithm by Dynamic Search and Independent Clause Pairs. ECCC Report No. 10, 2003. Also presented at SAT 2003.

[DGH+02] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $2-\frac{2}{k+1}$ algorithm for k-SAT based on local search. *Theoretical Computer Science*, 289(1):69–83, 2002.

[HSSW02] T. Hofmeister, U. Schöning, R. Schuler and O. Watanabe. Probabilistic 3-SAT Algorithm Further Improved. *Proceedings 19th Symposium on Theoretical Aspects of Computer Science*, LNCS 2285:193–202, 2002.

[Kul99] O. Kullmann, New methods for 3-SAT decision and worst-case analysis. *Theoretical Computer Science*, 223(1-2):1–72, 1999.

[MS85] B. Monien and E. Speckenmeyer. Solving satisfiability less than 2^n steps. *Discrete Applied Mathematics*, 10:287–295, 1985.

[PPSZ98] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. *Proceedings 39th Annual Symposium on Foundations of Computer Science*, 628–637, 1998.

[PPZ97] R. Paturi, P. Pudlák, and F. Zane. Satisfiability coding lemma. *Proceedings 38th Annual Symposium on Foundations of Computer Science*, 566–574, 1997. (full version is available at http://www.math.cas.cz/ pudlak/ppsz.ps.gz)

[Sch92] I. Schiermeyer. Solving 3-Satisfiability in less than 1.579^n steps. In *Selected papers from Computer Science Logic 92*, LNCS 702:193–20, 1992.

[Sch99] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. *Proceedings 40th Annual Symposium on Foundations of Computer Science*, 410–414, 1999.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'