Journal on Satisfiability, Boolean Modeling and Computation 1 (2006) 111-122

Improved Bound for the PPSZ/Schöning-Algorithm for 3-SAT

Daniel Rolf rolf@informatik.hu-berlin.de
Humboldt-University Berlin
Department of Computer Science
Logic in Computer Science
Unter den Linden 6
10099 Berlin, GERMANY

Abstract

The PPSZ Algorithm presented by Paturi, Pudlak, Saks, and Zane in 1998 has the nice feature that the only satisfying solution of a uniquely satisfiable 3-SAT formula can be found in expected running time at most $\mathcal{O}(1.3071^n)$. Its bound degenerates when the number of solutions increases. In 1999, Schöning proved an $\mathcal{O}(1.3334^n)$ bound for 3-SAT. In 2003, Iwama and Tamaki combined both algorithms to yield an $\mathcal{O}(1.3238^n)$ bound. We tweak the PPSZ-Bound to get a slightly better contribution to the combined algorithm and prove an $\mathcal{O}(1.32216^n)$ bound.

Keywords: *3-SAT, worst case bound, bounded resolution, randomized local search*

Submitted January 2006; revised October 2006; published November 2006

1. Introduction

The problem of deciding whether a k-CNF G has a satisfying assignment is well known as the k-SAT problem, which is NP-complete for $k > 2$. Hence, if $NP \neq P$ holds (which is widely assumed), there is no hope to find a polynomial time algorithm for the k-SAT problem for $k > 2$.

For a CNF G on n variables, a naive approach is to enumerate all possible assignments and to check for each one whether it satisfies G. This algorithm has $\mathcal{O}\left(poly(|G|) \cdot 2^n\right)$ running time at most. There are significantly more sophisticated algorithms known, and the evolution of expected running time bounds for 3-SAT, which are somewhat below the deterministic ones, is given as [1, 2, 3, 4, 5, 6] with bounds of $\mathcal{O}(1.334^n)$, $\mathcal{O}(1.3302^n)$, $\mathcal{O}(1.32971^n)$, $\mathcal{O}(1.3290^n)$, $\mathcal{O}(1.32793^n)$, and $\mathcal{O}(1.3238^n)$.

Using an elegant simple random walk algorithm, Schöning showed in 1999 that a satisfying assignment for a satisfiable 3-SAT formula can be found in $\mathcal{O}(1.3334^n)$ expected running time, cf. [1].

In [7], Paturi, Pudlak, Saks, and Zane proved that for a uniquely satisfiable 3-CNF, the solution can be found in $\mathcal{O}(1.3071^n)$ expected running time at most. We refer to their algorithm as the PPSZ Algorithm. This is the best randomized bound known for Unique-3-SAT and it is possible to derandomize it, essentially yielding the same bound deterministically, cf. [8]. But paradoxically, the bound gets worse when the number of solutions increases.

The best known randomized bounds for 3-SAT is $\mathcal{O}(1.3238^n)$, established in [6] by Iwama and Tamaki. Their bound automatically improves to $\mathcal{O}(1.32266^n)$ by modifying their analysis to use the latest bound for the PPSZ Algorithm that was presented in Corollary 14 in [9]. However, we tune the bound to improve their result to $\mathcal{O}(1.32216^n)$.

2. Preliminaries

Firstly, we make some common definitions. A *literal* is a variable or its negation. An assignment β to a set of variables X maps each variable in X to 0 or 1. A literal l is satisfied by β if $X(l) = 1$ if l is not negated resp. $X(\bar{l}) = 0$ if l is negated. A *clause* is a set of literals based on different variables. A clause is satisfied by some assignment β if at least one literal is satisfied by β. A *formula* is a set of clauses. A formula is satisfied by β if each clause is satisfied by β. For a formula G, we define $sat(G)$ to be the set of satisfying assignments of G. A *k-clause* is a clause of size k and a *k-CNF* is a set of clauses of size at most k. Finally, a 1-clause is commonly known as *unit clause*. For a set of clauses G, let $vars(G)$ be the set of variables occurring in G. Moreover, for G, we denote with n_G the number of variables in G.

We will not consider polynomial factors in complexity calculations because we always expect an exponential expression which outweighs all polynomials for large problems, and because the number of clauses is $\mathcal{O}(|vars(G)|^k)$, polynomials that depend on the number of clauses can also be replaced by some polynomial in $|vars(G)|$. Note that there could be an exponential number of clauses in the formula if we allow the same clause to occur more than once. However, in our setting, we do not allow repeated clauses.

For a CNF G and a literal l, we denote with $G|_l$ the formula obtained by making l true in G, i.e. we remove all clauses that contain l and remove $\bar{l}$ from all clauses that contain it.

A clause pair (C_1, C_2) is a *resolvent pair* if they have only one variable v in common whereby $v \in C_1$ and $\overline{v} \in C_2$. Their *resolvent* $R(C_1, C_2)$ is the clause $(C_1 - v) \cup (C_2 - \overline{v})$. Because any satisfying assignment of C_1 and C_2 must also satisfy $R(C_1, C_2)$, adding $R(C_1, C_2)$ to a CNF does not change its set of satisfying assignments.

s-bounded resolution means to add to G all resolvent pairs of clauses in G where the size of the resolvent is at most s, over and over again until there is nothing more to do. Note that, if s is a constant, this has polynomial time and space complexity in $|vars(G)|$.

3. The Algorithm

As proposed by Iwama and Tamaki in [6], we combine both PPSZ and Schöning to a new algorithm.

Algorithm 1. *$PPSZ$*(**k-CNF G, integer d, assignment β**)

```
1   $G :=$ do $k^d$-bounded resolution on $G$
2   $\pi :=$ permutation of $vars(G)$ uniformly at random
3   **for each** variable $v \in vars(G)$ ordered by $\pi$ {
4       **if** $G$ contains a unit clause $v$ resp. $\overline{v}$
5       **then** Set the value of $v$ in $\beta$ to satisfy that unit clause
6       Choose $G := G|_v$ or $G := G|_{\overline{v}}$ depending on $\beta(v) = 1$ or $\beta(v) = 0$.
7   }
8   **return** $\beta$
```

Algorithm 2. *SCH*(**k-CNF G, assignment β**)

```
1   **repeat** $3|vars(G)|$ times {
2       **if** $\beta$ satisfies $G$ **then break**
3       Select an arbitrary clause $C \in G$ that is not satisfied by $\beta$
4       Choose a variable in $C$ uniformly at random and flip its value in $\beta$
5   }
6   **return** $\beta$
```

Algorithm 3. *$COMB$*(**k-CNF G,integer d**)

```
1   $\beta :=$ assignment to $G$ drawn uniformly at random
2   $\beta' :=$ PPSZ$(G, d, \beta)$
3   **if** $\beta'$ satisfies $G$ **then return** $\beta'$
4   $\beta' :=$ SCH$(G, \beta)$
5   **if** $\beta'$ satisfies $G$ **then return** $\beta'$
6   **return** *null*
```

4. The Analysis

4.1 Main Result

Iwama and Tamaki proved in [6] that the expected number of repetitions of COMB(G, d) is $\mathcal{O}(1.3238^{n_G})$ (resp. $\mathcal{O}(1.32266^{n_G})$ as noted in the introduction) for a satisfiable 3-CNF formula G and some large but fixed d. We improve that result to:

Proposition 4.1. *For a satisfiable 3-CNF formula G and some large but fixed d, the expected number of repetitions of COMB(G, d) is $\mathcal{O}(1.32216^{n_G})$.*

In Section 4.2, we show how to disassemble the analysis for the combined algorithm into two separate ones. We provide bounds for both algorithms in Section 4.3 resp. Section 4.4. After that, we combine the bounds for both algorithm to prove the main result in Section 4.5. Finally, we consider some technical details in Section 5 and 6 that were left out in Section 4.4.

4.2 Disassembling *COMB*

For a set of variables $D \subseteq vars(G)$ and some assignment β of G, we define the set $B(D, \beta)$ to be the set of all assignments that agree with β on at least the variables in D, i.e. the subcube of the solution space where the variables in D are fixed to their values according to β and the others take all possible combinations.

For example, assume $vars(G) = \{a, b, c, d\}$, $D = \{a, b\}$ and let β assign 0 to all variables in $vars(G)$. Then $B(D, \beta)$ contains the following assignments:

a	b	c	d
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1

From [9], we know:

Lemma 4.2. *For a satisfiable k-CNF formula G, there exists a family of sets of variables $(D_\beta : \beta \in sat(G))$ so that the family of the corresponding subcubes $(B(D_\beta, \beta) : \beta \in sat(G))$ partitions the solution space (i.e. covering completely while being pairwise distinct). Moreover, it is true that*

$$\sum_{\beta \in sat(G)} 2^{-|D_\beta|} = 1.$$

So, throughout the rest of this chapter, fix (D_β) to be one of the families defined in the preceding lemma, and let (B_β) be the corresponding subcubes.

For some $\beta^* \in sat(G)$, let β be drawn uniformly at random from $\beta \in B_{\beta^*}$. Then the success probability of Algorithm COMB is at least

$$\max\{\mathbb{P}[PPSZ : \beta \in B_{\beta^*}], \mathbb{P}[SCH : \beta \in B_{\beta^*}]\}$$

where $PPSZ$ and SCH denote the events that Algorithm PPSZ(G, d, β) resp. SCH(G, β) return some satisfying assignment. For a random β, the probability that $\beta \in B_{\beta^*}$ holds is equal to $2^{-|D_{\beta^*}|}$. Observe that β is still distributed uniformly on B_{β^*}. To get the success probability, we just sum up the success probabilities over all subcubes. Hence Algorithm COMB succeeds with probability at least

$$\sum_{\beta^* \in sat(G)} 2^{-|D_{\beta^*}|} \cdot \max\{\mathbb{P}[PPSZ : \beta \in B_{\beta^*}], \mathbb{P}[SCH : \beta \in B_{\beta^*}]\}$$
$$\geq \min_{\beta^* \in sat(G)} \max\{\mathbb{P}[PPSZ : \beta \in B_{\beta^*}], \mathbb{P}[SCH : \beta \in B_{\beta^*}]\}.$$

The inequality follows because we know that $\sum_{\beta^* \in sat(G)} 2^{-|D_{\beta^*}|} = 1$.

Therefore, to have a lower bound on the success probability, we can focus on computing a lower bound for the success probability given a single satisfying assignment β^* and its subcube. Hence, fix some $\beta^* \in sat(G)$, $B = B_{\beta^*}$, $D = D_{\beta^*}$, and $N = vars(G) \setminus D$ to the end of this chapter.

4.3 Bound for SCH

To bound the running time of his algorithm, Schöning proved the following theorem, which bounds the success probability of Algorithm SCH in terms of the hamming distance $dist(\beta, \beta^*)$ of some initial assignment β and some satisfying assignment β^*:

Theorem 4.3. *Let G be a satisfiable k-CNF on n variables and β^* be a satisfying assignment for F. For each initial assignment β, the probability that Algorithm SCH(G, β) finds a satisfying assignment is at least $(k-1)^{-dist(\beta,\beta^*)-o(n)}$.*

Conditioning on $\beta \in B_{\beta^*}$, we know that β agrees with β^* on D, whereby the assignment to N is uniformly distributed. So we have that

$$\begin{aligned}
&\mathbb{P}\left[SCH : \beta \in B_{\beta^*}\right] \\
&\geq \mathbb{E}\left[(k-1)^{-dist(\beta,\beta^*)-o(n_G)} : \beta \in B_{\beta^*}\right] \\
&= 2^{-o(n_G)} \prod_{v \in N} \left(\mathbb{P}[\beta(v) = \beta^*(v)] \cdot (k-1)^0 + \mathbb{P}[\beta(v) \neq \beta^*(v)] \cdot (k-1)^{-1}\right) \\
&= 2^{-o(n_G)} \prod_{v \in N} \left(1/2 \cdot (k-1)^0 + 1/2 \cdot (k-1)^{-1}\right) \\
&= (2 - 2/k)^{-|N|-o(n_G)} \\
&= (2 - 2/k)^{-n_G+|D|-o(n_G)} \\
&= 2^{-\sigma_k(n_G-|D|)-o(n_G)}
\end{aligned}$$

where $\sigma_k = \log_2(2 - 2/k)$.

Obviously, the success probability of Algorithm SCH increases with increasing $|D|$.

4.4 Bound for $PPSZ$

Let us define a *nice distribution* H. H is a nondecreasing, continuous mapping from $[0, 1]$ to $[0, 1]$ with $H(0) = 0$ and $H(1) = 1$. Moreover, it must be differentiable in all but at most a finite number of points. Finally, its derivate h must be uniformly bounded on $[0, 1]$. We set

$$\beta_H = \int_0^1 h(r) \log_2(h(r))\, dr$$

$$\gamma_H = \int_0^1 \min\{H(r)^{k-1}, R_k(r)\}\, dr$$

where $R_k(r)$ is the smallest non-negative x that satisfies $f_k(x, r) = x$ with $f_k(x, r) = (r + (1-r)x)^{k-1}$.

In Section 5, we will prove:

Lemma 4.4. *The probability that Algorithm PPSZ finds a satisfying assignment given $\beta \in B_{\beta^*}$ is at least*

$$2^{-\beta_H|D|-(1-\gamma_H)(n_G-|D|)-\epsilon n_G-o(n_G)}$$

where ϵ can be made arbitrary small positive by choosing d large enough.

Of course, H is only a parameter in the analysis, it actually does not change the success probability of Algorithm PPSZ. However, β_H and γ_H are subject to H. Hence choosing H affects the upper bound on the number of repetitions needed for Algorithm PPSZ in terms of $|D|/n_G$.

For $k = 3$, we are not able to find H in such a way that the success probability does not decrease for small $|D|$. But, we will see that we can tweak H so that the bound does not decrease too much until Schöning's can take over.

4.5 Reassembling *COMB*

We saw that the bound for SCH and the bound for $PPSZ$ depend on $|D|$, where the first increases with increasing $|D|$ and the second decreases with increasing $|D|$ if H is chosen appropriately. Hence we have to find the 'worst' $|D|$. Clearly, the worst case $|D|$ is attained when $\max\{\mathbb{P}[PPSZ : \beta \in B_{\beta^*}], \mathbb{P}[SCH : \beta \in B_{\beta^*}]\}$ is minimized.

Assuming that we have a distribution H so that the success probability of PPSZ decreases with increasing $|D|$, we can compute the worst $|D|$ since the success probability of SCH increases with increasing $|D|$. Thus

$$\max\{\mathbb{P}[PPSZ : \beta \in B_{\beta^*}],\mathbb{P}[SCH : \beta \in B_{\beta^*}]\}$$

is minimized if

$$\begin{aligned}\sigma_k(n_G - |D|) + o(n_G) &= \beta_H|D| + (1-\gamma_H)(n_G - |D|) + o(n_G)\\ |D| &= n_G\frac{\sigma_k - 1 + \gamma_H}{\sigma_k - 1 + \gamma_H + \beta_H} + o(n_G)\end{aligned}$$

holds.

We have proved:

Proposition 4.5. *Let H be a nice distribution so that the bound for $PPSZ$ decreases with increasing $|D|$, and let*

$$\delta = \frac{\sigma_k - 1 + \gamma_H}{\sigma_k - 1 + \gamma_H + \beta_H}$$

be well defined with $0 \leq \delta \leq 1$. For a satisfiable k-CNF formula G, the success probability of Algorithm COMB(G, d) is at least

$$2^{-\sigma_k(1-\delta)n_G - \epsilon n_G - o(n_G)}$$

where ϵ can be made arbitrary small positive by choosing d large enough.

In Section 6, we will provide some H_3 with $\beta_{H_3} \leq 0.90625$, $\gamma_{H_3} \geq 0.61229$, and thus $\delta_3 \geq 0.02927$. Therefore, we have a lower bound of $\Omega\left(1.32216^{-n_G}\right)$ for the success probability of COMB for a satisfiable 3-CNF formula G. This finishes the proof of the main result, Proposition 4.1.

5. Proof of the $PPSZ$ Bound

At first, we have to recapitulate some technical features around the PPSZ algorithm, some of which have been dismissed from the latest version of [9] because they are not necessary anymore by their analysis, but we need them for this one.[1.]

For some permutation π, let $F(\pi)$ denote the set of variables in N that have been reduced to unit clauses during a run of Algorithm PPSZ. When β agrees with β^* on the variables in $vars(G) \setminus F(\pi)$, the algorithm will find β^*. Given that $\beta \in B_{\beta*}$, we know that β and β^* already agree on D. Thus we have:

$$\mathbb{P}[PPSZ : \beta \in B_{\beta^*}] \geq 2^{-N} \cdot \mathbb{E}\left[2^{F(\pi)}\right]$$

In order to have a good bound on the expectation, we will choose some subset Γ of the permutation space and compute instead:

$$\mathbb{P}[PPSZ : \beta \in B_{\beta^*}] \geq 2^{-N} \cdot \mathbb{P}[\pi \in \Gamma] \cdot \mathbb{E}\left[2^{|F(\pi)|} : \pi \in \Gamma\right]$$

A placement α is a function that maps each variable to a real value in $[0, 1]$. With $\pi(\alpha)$, we denote the permutation obtained by ranking the variables of G due to the values α takes on them with some arbitrary rule for breaking ties. Hence a uniform distribution of $\alpha(.)$ yields a uniform distribution of $\pi(\alpha(.))$.

Let v be a variable in N. For a set of placements Γ, we define $Q_\Gamma(r)$ to be the probability that v is in $F(\pi(\alpha))$ where α is a random placement from Γ having $\alpha(v) = r$. Then we have:

$$\mathbb{P}[v \in F(\pi(\alpha)) : \alpha \in \Gamma] \geq Q_\Gamma = \int_0^1 Q_\Gamma(r)\, dr$$

For every $\lambda \in [0, 1)$, we consider the set of placements $\Gamma_{H,\lambda,D}$ to be the set of all placements where for each $r \in [\lambda, 1]$, at least $H(r)|D|$ variables $v \in D$ have $\alpha(v) < r$.

From Lemma 26 in the old version of [9], we know that:

Lemma 5.1. *Define the recursive function $Q_k^d(r)$ by $Q_k^0(r) = 0$ and $Q_k^d(r) = f_k(Q_k^{d-1}(r), r)$ for $d > 0$. For $\Gamma = \Gamma_{H,\lambda,D}$ and $r \in [\lambda, 1]$, it is true that*

$$Q_\Gamma(r) \geq \min\{H(r)^{k-1}, Q_k^d(r)\} - \rho(H(r))$$

where $\rho(x) = 0$ for $x \in \{0, 1\}$ and $\rho(x) = \min\left\{\frac{k^{2d}}{|A|}\left(\frac{1}{x(1-x)}\right), 1\right\}$ for $x \in (0, 1)$.

1. An older version of [9] is still available in the citeseer-cache at http://citeseer.ist.psu.edu/paturi98improved.html.

We compute Q_Γ for $\Gamma = \Gamma_{H,\lambda,D}$:

$$\begin{aligned} Q_\Gamma &= \int_0^1 Q_\Gamma(r)\,dr \\ &\geq \int_\lambda^1 Q_\Gamma(r)\,dr \\ &\geq \int_0^1 (\min\{H(r)^{k-1}, Q_k^d(r)\} - \rho(H(r)))\,dr - \lambda \\ &\geq \int_0^1 \min\{H(r)^{k-1}, Q_k^d(r)\}\,dr - \int_0^1 \rho(H(r))\,dr - \lambda \end{aligned}$$

Paturi et al evaluated $\int_0^1 \rho(H(r))\,dr$ to be $o(1)$ when $|D| \geq \sqrt{n_G}$ as n_G tends to infinity. We omit the analysis for $|D| \leq \sqrt{n_G}$ here since it is very likely for less than $\sqrt{n_G}$ variables to appear at the very beginning of the permutation π before all variable in N. For those, Paturi et al showed that $Q_\Gamma \geq \int_0^1 Q_k^d(r)$. We conclude:

$$Q_\Gamma \geq \int_0^1 \min\{H(r)^{k-1}, Q_k^d(r)\}\,dr - o(1) - \lambda$$

In Proposition 3 in [7], they also show that $Q_k^d(r)$ converges to $R_k(r)$ for every $r \in [0,1]$. Hence for every small positive ϵ, there exists a large d_ϵ so that for every $d \geq d_\epsilon$, $Q_k^d(r) \geq R_k(r) - \epsilon$ is true for all $r \in [0,1]$. We conclude that

$$\begin{aligned} Q_\Gamma &\geq \int_0^1 \min\{H(r)^{k-1}, R_k(r) - \epsilon\}\,dr - \lambda - o(1) \\ &\geq \int_0^1 \min\{H(r)^{k-1}, R_k(r)\}\,dr - \epsilon - \lambda - o(1) \end{aligned}$$

is true.

For the reader familiar with the details of [9], it is noticable that we have just proved a generalized version of Lemma 24 in the old version of [9]. In that lemma, they restricted $H(r)$ to be at most $R_k(r)^{1/(k-1)}$. The corresponding lemma in the latest version, Lemma 13 in [9], does not make use of any function H at all. Nevertheless, comparing the details, the new lemma looks like using $H(r) = \min\{r \cdot (k-1)/(k-2), 1\}$ in the old one. But, that H violates the (unnecessary) restriction $H(r) \leq R_k(r)^{1/(k-1)}$. Therefore, in the proof above, we only unified both approaches.

Since we have computed Q_Γ, we can consider the expected number of variables that will be in $F(\pi(\alpha))$:

$$\begin{aligned} &\mathbb{E}\left[2^{|F(\pi(\alpha))|} : \alpha \in \Gamma_{H,\lambda,D}\right] \\ &\geq 2^{\mathbb{E}[|F(\pi(\alpha))| : \alpha\in\Gamma_{H,\lambda,D}]} \\ &\geq 2^{\gamma_H|N| - \epsilon|N| - \lambda|N| - o(|N|)} \end{aligned}$$

Thus we conlude:

$$\mathbb{P}[PPSZ : \beta \in B_{\beta^*}] \geq \mathbb{P}[\alpha \in \Gamma_{H,\lambda,D}] \cdot 2^{-(1-\gamma_H)|N| - \epsilon|N| - \lambda|N| - o(|N|)}$$

For $\mathbb{P}[\alpha \in \Gamma_{H,\lambda,D}]$, Paturi et al proved a nice lower bound, cf. Lemma 23 in the old version of [9]:

Lemma 5.2. *For $\lambda > 0$, it is true that*

$$\mathbb{P}[\alpha \in \Gamma_{H,\lambda,D}] \geq 2^{-\beta_H |D| - o(|D|)}.$$

Because ϵ and λ are both arbitrary small positive values, we have

$$\mathbb{P}[PPSZ : \beta \in B_{\beta^*}] \geq 2^{-\beta_H |D| - (1-\gamma_H)(n_G - |D|) - \epsilon' n_G - o(n_G)},$$

where ϵ' is some arbitrary small positive real. This finishes the proof of Lemma 4.4.

6. Optimized Nice Distributions for 3-SAT

By Proposition 4.5, the running time bound depends on the choice of some H which produces a large δ. Experiments showed that we should consider functions H where there is some $r_0 \leq 1/2$ with $H(r)^2 \geq R_3(r)$ for $r \leq r_0$ and $H(r)^2 \leq R_3(r)$ for $r \geq r_0$. In this case, we have:

$$\gamma_H = \int_0^{r_0} R_3(r)\, dr + \int_{r_0}^1 H(r)^2\, dr$$

For $r \in [0, 1/2]$, we have:

$$R_3(r) = \frac{r^2}{(1-r)^2}$$
$$\int_0^r R_3(r')\, dr' = 2\ln(1-r) - 1 + \frac{r^2 - r - 1}{r - 1}$$

As a simple example, we consider the function $H_\theta(r) = \min\{1, r/\theta\}$ for some $\theta \in [1/2, 1]$. Firstly, for $r \in [0, 1-\theta]$, we have $H_\theta(r)^2 \geq R_3(r)$. Secondly, for $r \in [1-\theta, \theta]$, we have $H_\theta(r)^2 \leq R_3(r)$, and finally, for $r \in [\theta, 1]$, we have $H_\theta(r)^2 = R_3(r) = 1$. Hence the following holds:

$$\begin{aligned}
\gamma_{H_\theta} &= \int_0^{1-\theta} R_3(r)\, dr + \int_{1-\theta}^{\theta} H_\theta(r)^2\, dr + 1 - \theta \\
&= \frac{6\ln(\theta)\theta^2 + 6\,\theta - 4\,\theta^3 - 1}{3\,\theta^2} \\
\beta_{H_\theta} &= \int_0^{\theta} \frac{1}{\theta} \log_2\left(\frac{1}{\theta}\right) dr \\
&= -\log_2(\theta)
\end{aligned}$$

We insert this into the formula for δ in Proposition 4.5 and compute the root of the derivate with respect to θ to get the optimal $\theta = 0.5109968782$. For this θ, we get $\beta_{H_\theta} \leq 0.9686136176$, $\gamma_{H_\theta} \geq 0.613242472$, and thus $\delta \geq 0.28368$. This yields an upper bound of $\mathcal{O}(1.3225^{n_G})$ for the expected number of repetitions of Algorithm COMB.

But, we can do better. In order to find an optimal H, we can set up a continuous function H consisting of linear pieces and try to optimize it until we hit the best result. Experiments showed that the resulting curve is perfectly resembled by the following function, with some appropriate parameters a and b:

$$H(r) = \begin{cases} r/\theta & \text{if } r \in [0, 1-\theta) \\ 1 - (-a \ \ln(r))^b & \text{if } r \in [1-\theta, 1] \end{cases}$$

$$h(r) = \frac{dH}{dr} = \begin{cases} 1/\theta & \text{if } r \in [0, 1-\theta) \\ -b \, \frac{(-a \ \ln(r))^b}{r \ \ln(r)} & \text{if } r \in [1-\theta, 1) \end{cases}$$

$H(r)$ must be continuous, and naturally, it should also be differentiable completely. Moreover, we propose that $H(r)$ should hit $R_3(r)^{1/2}$ exactly when the linear part finishes, i.e. at $1-\theta$ since $R_3(r)^{1/2} = r/(1-r)$ for $r \in [0, 1/2]$. Using these constraints, i.e.

$$\begin{aligned} H(1-\theta) &= R_3(1-\theta)^{1/2} \qquad \text{and} \\ h(1-\theta) &= 1/\theta, \end{aligned}$$

we can eliminate a and b:

$$\begin{aligned} a &= -\left(\frac{2\,\theta - 1}{\theta}\right)^{\frac{2\,\theta-1}{\ln(1-\theta)(\theta-1)}} (\ln(1-\theta))^{-1} \\ b &= \frac{\ln(1-\theta)\,(\theta-1)}{2\,\theta - 1} \end{aligned}$$

For the antiderivative of $h(r) \log h(r)$, we have

$$\beta_1(r) = -\frac{r \log \theta}{\theta} + C$$

for $r \in [0, 1-\theta)$ and

$$\beta_2(r) = \frac{-(-a \ln r)^b \left(b \ln r - b^2 + 1 + (b + b^2) \ln\left(-\frac{(-a \ln r)^b b}{r \ln r}\right)\right)}{(b + b^2) \ln 2} + C$$

for $r \in [1-\theta, 1)$. Observe that $\beta_2(r)$ is not defined for $r = 1$. However, when r approaches 1^-, then $\beta_2(r)$ tends to C. We have:

$$\begin{aligned} \beta_H &= \int_0^1 h(r) \log h(r) \, dr \\ &= \beta_1(1-\theta) - \beta_1(0) + \lim_{r \to 1^-} \beta_2(r) - \beta_2(1-\theta) \\ &= \beta_1(1-\theta) - \beta_1(0) - \beta_2(1-\theta) \end{aligned}$$

For the antiderivative of $H(r)^2$, we have

$$\gamma_2(r) = r - 2\Gamma(1 + b, -\ln r) \cdot a^b + \Gamma(1 + 2b, -\ln r) \cdot a^{2b}$$

for $r \in [1-\theta, 1]$ where $\Gamma(a, x)$ is the (upper) incomplete gamma function. For $r \in [0, 1-\theta)$, we need the antiderivative of $R_3(r)$, which is

$$\gamma_1(r) = 2\ln(1-r) + \frac{r^2 - r - 1}{r - 1} + C.$$

Since $H(r)^2 \geq R_3(r)$ for $r \in [0, 1-\theta]$ and $H(r)^2 \leq R_3(r)$ for $r \in [\theta, 1]$, we conclude:

$$\begin{aligned} \gamma_H &= \int_0^{1-\theta} R_3(r)\,dr + \int_{1-\theta}^{1} H(r)^2\,dr \\ &= \gamma_1(1-\theta) - \gamma_1(0) + \gamma_2(1) - \gamma_2(1-\theta) \end{aligned}$$

To find θ so that δ in Proposition 4.5 is maximized, we just insert the terms for γ_H and β_H in the formula for δ and find the optimum with respect to θ. Numerical optimization yields that δ is maximized using:

$$\begin{aligned} \theta &= 0.5111885981... \\ a &= 1.1437170697... \\ b &= 15.635592073... \\ \beta_H &\leq 0.9062404894 \\ \gamma_H &\geq 0.6122939734 \\ \delta_H &\geq 0.0292762355 \\ 2^{\sigma_3(1-\delta_H)} &\leq 1.3221508262 \end{aligned}$$

This yields an upper bound of $\mathcal{O}(1.32216^{n_G})$ for the expected number of repetitions of Algorithm COMB.

Acknowledgements

The author would like to thank Mario Szegedy and the unknown referees for valuable remarks.

References

[1] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In: *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 410–414, 1999.

[2] R. Schuler, U. Schöning, and O. Watanabe. A probabilistic 3-SAT algorithm further improved. In: *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science* (STACS), 192–202, 2002.

[3] D. Rolf. 3-SAT $\in RTIME(1.32971^n)$. Diploma thesis, Department Of Computer Science, Humboldt University Berlin, Germany, 2003.

[4] S. Baumer and R. Schuler. Improving a probabilistic 3-SAT algorithm by dynamic search and independent clause pairs. In: *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing* (SAT), 150–161, 2003.

[5] D. Rolf. 3-SAT $\in RTIME(O(1.32793^n))$ - improving randomized local search by initializing strings of 3-clauses. *Electronic Colloquium on Computational Complexity* (ECCC), 2003.

[6] K. Iwama and S. Tamaki. Improved upper bounds for 3-SAT. In: *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), 328–328, 2004.

[7] R. Paturi, P. Pudlak, M.E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. In: *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 628–637, 1998.

[8] D. Rolf. Derandomization of PPSZ for Unique-k-SAT. In: *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing* (SAT), 216–225, 2005.

[9] R. Paturi, P. Pudlak, M.E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. *Journal of the Association for Computing Machinery* **52**(3): 337–364, 2006.