Derandomization of PPSZ for Unique-k-SAT

Daniel Rolf

Humboldt-Universität zu Berlin,
Institut für Informatik,
Lehrstuhl für Logik in der Informatik,
Unter den Linden 6, 10099 Berlin, Germany
rolf@informatik.hu-berlin.de

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 $\mathcal{O}(1.3071^n)$. Using the technique of limited independence, we can derandomize this algorithm yielding $\mathcal{O}(1.3071^n)$ deterministic running time at most.

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 $\mathcal{O}(1.3238^n)$.

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. This is the best randomized bound known for Unique-3-SAT. We refer to their algorithm as the PPSZ Algorithm. But paradoxically, the bound gets worse when the number of solutions increases. This is even more curious since Unique-k-SAT has been proven to be the hardest case of k-SAT for k tending to infinity (cf. [8]). Alas, for the general 3-SAT resp. 4-SAT case, this algorithm achieves expected running time bounds of $\mathcal{O}(1.362^n)$ resp. $\mathcal{O}(1.476^n)$ only, which is worse than the best known randomized bounds of $\mathcal{O}(1.3238^n)$ resp. $\mathcal{O}(1.474^n)$, established in [6] by Iwama and Tamaki.

The best bounds for k-SAT make excessive usage of random bits so that enumerating the entire probability space would yield useless bounds, i.e. much more than $\mathcal{O}(2^n)$. But, do random bounds really compete with deterministic bounds when the existence of true randomness is not provable? At least, randomized algorithms often supply a good starting point to develop fast deterministic algorithms. For example, for k-SAT, the algorithm of Schöning in [1], based on randomized local search and restart, yields a bound of $\mathcal{O}((2 - 2/k + \epsilon)^n)$ expected running time at most, which has been derandomized in [9] to the best known deterministic bound of $\mathcal{O}(1.481^n)$ for $k = 3$ and $\mathcal{O}((2 - 2/(k+1) + \epsilon)^n)$ for $k > 3$, based on limited local search and covering codes. Alas, like so often, the deterministic bound is much worse than the original randomized one. However, in this paper, we derandomize the PPSZ Algorithm, already mentioned in the paragraph before, for the uniquely satisfiable case yielding (almost) the same bound as the randomized version making it the best known deterministic bound for Unique-k-SAT. We use the technique of limited independence (cf. [10]) to prove that the algorithm can be adapted to enumerate some small probability space yielding a deterministic running time which equals to the former expected running time up to a subexponential factor. Moreover, this means that the best bound for Unique-3-SAT is not only a deterministic one, but also better than the best known randomized bound for (general) 3-SAT.

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. 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.

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 $|vars(G)|$.

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

Now, we present our algorithm, which is a derandomized form of the PPSZ Algorithm. Note that π denotes a permutation of the variables of G computed using a polynomial time function $\pi(\alpha)$ where α is a member of some set $\Omega(n, w, L)$. The definition of both objects and the role of the parameters is deferred to Section 4.

Algorithm 1. ***dPPSZ*(*k*-CNF *G*, integer *d*, integer *L*, integer *t*)**

```
1  $G :=$ do $k^d$-bounded resolution on $G$
2  for each $\pi = \pi(\alpha)$ with $\alpha \in \Omega(|vars(G)|, (k-1)^{d+1} - 1, L)$ and for each
   bit string $b$ of size $t$ do {
3      $G' := G$
4      repeat as long as there is an unused bit in $b$ {
5          $v :=$ next unused variable in $\pi$
6          if $G'$ contains a unit clause $v$ resp. $\overline{v}$
7          then $G' := G'|_v$ resp. $G' := G'|_{\overline{v}}$
8          else Choose $G' := G'|_v$ or $G' := G'|_{\overline{v}}$ depending on the next unused
           bit of $b$ being 1 or 0
9      }
10     If $G'$ is the empty formula then return $true$
11 }
12 return $false$
```

The only difference between the PPSZ Algorithm and this one is that PPSZ choose a permutation π of $vars(G)$ and a bit string b of length $|vars(G)|$ uniformly at random.

4 The Analysis

Without loss of generality, we denote the variables of G with the integers in $[n]$. Moreover, let β denote the one and only satisfying assignment of G.

4.1 Deterministic Bounds for Unique-*k*-SAT

In the algorithm, we use a set $\Omega(n, w, L)$ with $w = (k-1)^{d+1} - 1$, the set will be defined in Section 4.2. But for now, let us use it as a black box probability space that can be used to draw permutations π of $[n]$ at random so that the following lemma is satisfied, which is proved in Section 4.4:

Lemma 2. *Let d and L be integers and let G be a uniquely satisfiable k-CNF G with more than d variables. Fix some variable v of G. Assume that Algorithm 1* *reaches variable v and all variables before v in π were set according to β. At this step, there will be a unit clause for v with probability at least $\lambda_{k,d,L}$ with*

$$\lambda_{k,d,L} = \frac{\mu_k}{k-1} - \epsilon_{k,d,L} \text{ and}$$
$$\mu_k = \sum_{j=1}^{\infty} \frac{1}{j\left(j+\frac{1}{k-1}\right)}$$

where $\epsilon_{k,d,L}$ can be made arbitrary small positive by choosing L and d large enough.

Using linearity of expectation, we can expect to have $\lambda_{k,d,L}n$ unit clauses on the average. Because we try all elements of $\Omega(n,w,L)$, we must encounter at least on permutation π, where the number of unit clauses is at least $\lambda_{k,d,L}n$. Now, assume that the bit string b is chosen so that all bits used for variables agree with β. But, because at least $\lambda_{k,d,L}n$ variables are determined using unit clauses, we only need at most $n - \lambda_{k,d,L}n$ bits from b. So, if we set $t = \lceil n - \lambda_{k,d,L}n \rceil$, we will face that *good* bit string.

Enumerating all bit strings of length t takes time at most $\mathcal{O}(2^t)$. In Section 4.2, we will prove that $\Omega(n,w,L)$ can be constructed and enumerated in polynomial time in $\mathcal{O}(n^{Lw/2})$ which is a polynomial in n for constant k, d, and L. Formulas which do not satisfy the precondition of Lemma 2, i.e. which have at most d variables, can be solved in polynomial time since d is a constant. Finally, we can state:

Proposition 3. *For a uniquely satisfiable k-CNF on n variables, integers $d > 0$, $L > 0$, and $t = \lceil n - \lambda_{k,d,L}n \rceil$, Algorithm 1 finds the satisfying assignment in deterministic running time at most*

$$\mathcal{O}\left(2^{\left(1-\frac{\mu_k}{k-1}\right)n+\epsilon_{k,d,L}n}\right)$$

where $\epsilon_{k,d,L}$ can be made arbitrary small positive by choosing L and d large enough.

Corollary 4. *For a uniquely satisfiable 3-CNF resp. 4-CNF on n variables, the satisfying assignment can be found in deterministic running time at most $\mathcal{O}(1.3071^n)$ resp. $\mathcal{O}(1.4699^n)$.*

4.2 A w-Wise Independent Probability Space for n Reals with Precision L

The (original) PPSZ Algorithm chooses a permutation π uniformly at random, but in Section 4.3, we will see that we need only randomness with respect to a subset of the variables which has size bounded by a constant w for the Unique-k-SAT case. So, we could just draw n integers from a finite pool (to have a finite probability space) of w-wise independent integers and order π according to the rank of these. But, what do we do if we draw the same integer for two variables? Fortunately, the bigger the pool is, the less likely it is for two variables to clash. Guided by this idea, we will discuss a handy construction of π and show some useful properties. For that, our basic tool is Theorem 2.1 from [10–Chapter 15]:

Theorem 5. *For every $n \geq w \geq 1$ there exists a probability space $\Omega(n, w)$ of size $\mathcal{O}(n^{w/2})$ and w-wise independent random variables $y_1, ..., y_n$ over $\Omega(n, w)$ each of which takes 0 or 1 with probability 1/2. $\Omega(n, w)$ can be constructed in polynomial time.*

Let us start with a mapping α which assigns each integer in $[n]$ a real value in $[0, 1)$. We construct a random α in the following way. Define integers $w > 0$, $L > 0$. For each $l \in [L]$ and independently from each other, draw n w-wise independent random variables $y_{1,l}, ..., y_{n,l}$ as stated in the theorem using $\Omega(n, w)$. Define

$$\alpha(v) = \sum_{l \in [L]} 2^{-l} y_{v,l},$$

i.e. $y_{v,.}$ is seen as a binary encoding for $\alpha(v)$ with length L. Let $A^{(L)}$ be the set of all possible real values $\alpha(.)$ can take. For fixed v, the random variables $y_{v,.}$ are fully independent since they are drawn from independent probability spaces. Hence, each value in $A^{(L)}$ has equal probability to be chosen for $\alpha(v)$. On the other hand, for fixed l, the random variables $y_{.,l}$ are w-wise independent since they are drawn using $\Omega(n, w)$. Because this holds for every l independently, the values of α are w-wise independent.

Therefore, the construction above yields w-wise independent α values where each of them takes a value from $A^{(L)}$ uniformly at random. We call this probability space a *w-wise independent probability space for n reals with precision L*, denoted by $\Omega(n, w, L)$. We have:

Lemma 6. *$\Omega(n, w, L)$ can be constructed with size $\mathcal{O}(n^{Lw/2})$ and in polynomial time in its size.*

Given α, we construct a permutation $\pi = \pi(\alpha)$ of $[n]$ so that $\alpha(u) < \alpha(v)$ implies that u occurs before v in π. Such a permutation can clearly be constructed in a deterministic way by ordering $[n]$ due to the values α takes on them with some arbitrary deterministic rule if two take the same value.

Fix some arbitrary $v \in [n]$ and fix some arbitrary $V \subseteq [n] - v$ with $|V| < w$. We want to have a lower bound for the probability that a variable u in V occurs before v in π. The fact that $\alpha(u) = \alpha(v)$ could hold, makes the analysis a little bit complicated. Fortunately, this is not very likely. So, we call v *unique* with respect to V if $\alpha(u) \neq \alpha(v)$ holds for all $u \in V$. Clearly, the probability that v is unique with respect to V is $(1 - 2^{-L})^{|V|}$.

Now, assume that we already know that v is unique with respect to V. All $\alpha(u)$ for $u \in V$ can still be seen as being drawn independently at random from $A^{(L)} - \alpha(v)$. Again, fix a variable u in V. Under the condition that v is unique with respect to V, the probability that $\alpha(u) < \alpha(v)$, i.e. that u occurs before v in π, is equal to $\alpha(v) \cdot 2^L/(2^L - 1)$. This comes from the fact that we have $\alpha(v) \cdot 2^L$ elements in $A^{(L)}$ which are strict less than $\alpha(v)$ and because the condition allows all $2^L - 1$ elements of $A^{(L)} - \alpha(v)$ to be chosen for $\alpha(u)$ uniformly at random. Let us sum up:

Lemma 7. *Let $v \in [n]$ be a variable and $V \subseteq [n]$ be a set of variables with $|V| < w$ and $v \notin V$. Then the following are true:*

1. *The probability that v is unique is $(1 - 2^{-L})^{|V|}$.*
2. *Given that v is unique, all $\alpha(u)$ with $u \in V$ are independent, and for each $u \in V$, the probability that $\alpha(u) < \alpha(v)$ holds is equal to $\alpha(v) \cdot 2^L/(2^L - 1)$.*

4.3 Admissible Trees

Before we can go back to Unique-k-SAT, we need the notion of an admissible tree and have to prove some important properties.

Let T be a tree where the root is labeled by v. Each node of the tree can have a label in $[n]$ or it is unlabeled. Moreover, for each path from a leaf to the root, no integer occurs more than once as a label. Then T is called an *admissible tree.* The depth of T is the maximum distance from any leaf to the root, e.g. a tree containing only one node has depth 0. We limit the depth of an admissible tree to d and we limit the number of children of each node to $k - 1$. Then T has at most $(k-1)^{d+1} - 1$ nodes. A *cut* A is a set of nodes that does not include the root, and every path from the root to a leaf includes a node in A.

Let $\pi = \pi(\alpha)$ where α is drawn from $\Omega(n, (k-1)^{d+1} - 1, L)$ at random. We say a cut A happens if all variables corresponding to labeled nodes of A occur before v in π.

Given that v is unique and a subtree T_0 of T, we denote with $Q_{T_0}(r)$ the probability that at least one of the possible cuts of T_0 happens and conveniently, where we use r to stand for $\alpha(v) \cdot 2^L/(2^L - 1)$. We will establish a lower bound for $Q_{T_0}(r)$:

Lemma 8. *Given an admissible tree T with root labeled by v, let T_0 be some subtree of T with more than one node and let $T_1, ..., T_t$ be the subtrees rooted at the labeled children of the root of T. Let $u_1, ..., u_t$ be the labels of their roots. Then it is true that*

$$Q_{T_0}(r) \geq \prod_{i=1}^{t} (r + (1-r) Q_{T_i}(r))$$

holds where the empty product is interpreted as 1.

Proof. [1] Consider the case that $t = 0$. Since T_0 has at least one child, there is a cut in the tree. But because, no child is labeled, the cut is empty, which

[1] Note that this proof is almost the same like the one for Lemma 4 in [7], which can be found in [11].

corresponds to an empty event, which occurs with probability 1. So, assume $t \geq 1$.

Let U be the set of variables occurring as labels in T. Since $|U| \leq (k-1)^{d+1}-1$ and since we have chosen α from $\Omega(n, (k-1)^{d+1}-1, L)$, we can apply Lemma 7 using $U - v$ for V.

Consider the event that $\alpha(u_i) < \alpha(v)$, which has probability r, and the event that a cut in T_i occurs. Because a subtree of an admissible tree is also admissible, u_i does not occur anywhere else in T_i. Thus, both events are independent, causing their union, denoted with K_i, to have probability $r + (1-r)Q_{T_i}(r)$.

To finish the proof, we have to show that $\mathbb{P}[\bigcap_{i=1}^{t} K_i] \geq \prod_{i=1}^{t} \mathbb{P}[K_i]$. At first, let us recall some standard correlation inequality, which is a special case of the FKG-inequality (cf. Theorem 3.2 in [10–Chapter 6]):

Lemma 9. *Let N be a finite set and let $\mathcal{A}$ and $\mathcal{B}$ be two monotone increasing families of subsets of N, i.e. each super-set of a set in $\mathcal{A}$ resp. $\mathcal{B}$ is also contained in $\mathcal{A}$ resp. $\mathcal{B}$. Draw a random set $M \subseteq N$ by choosing each u in N independently with probability p. Then it is true that*

$$\mathbb{P}[M \in \mathcal{A} \cap \mathcal{B}] \geq \mathbb{P}[M \in \mathcal{A}]\mathbb{P}[M \in \mathcal{B}].$$

We set N to be the set of all variables occurring as labels in T_0, but we exclude v. Moreover, we determine M as follows. For all $u \in N$, we include u in M if it occurs before v in π. These events occur independently each with probability r. Let $\mathcal{W}_i$ denote the family of all subsets of N that imply K_i, i.e. all sets of variables $W \subseteq [n]$ for which holds that K_i happens when all $u \in W$ occur before v in π. Because K_i only depends on variables in N, M is a member of $\mathcal{W}_i$ if and only if the event K_i happens. Clearly, $\mathcal{W}_i$ is monotone increasing since all supersets of a set that implies K_i also imply K_i, i.e. more variables than necessary before v in π is not bad. The set $\mathcal{V}_i = \bigcap_{j=1}^{i-1} \mathcal{W}_i$ is also monotone increasing. We plug $\mathcal{V}_i$ and $\mathcal{W}_i$ as $\mathcal{A}$ and $\mathcal{B}$ in the Lemma and obtain

$$\begin{aligned}\mathbb{P}[M \in \mathcal{V}_{i+1}] &\geq \mathbb{P}[M \in \mathcal{V}_i]\mathbb{P}[M \in \mathcal{W}_i] \\ &\geq \prod_{j \leq i} \mathbb{P}[M \in \mathcal{W}_j].\end{aligned}$$

Because $M \in \mathcal{V}_{t+1}$ means that the event $\bigcap_{i=1}^{t} K_i$ happens, we can conclude that

$$\mathbb{P}\left[\bigcap_{i=1}^{t} K_i\right] \geq \prod_{i=1}^{t} \mathbb{P}[K_i]$$

is true, which completes the proof. □

Let us multiply the probability for the event that v is unique with $Q_T(r)$ to obtain:

Corollary 10. *Given an admissible tree T of depth d with root labeled by v and given that $\alpha(v) = r'$ is true, the probability that a cut of T occurs is at least $Q'_T(r)$ with*

$$Q'_T(r') = Q_T\left(r' \cdot 2^L / \left(2^L - 1\right)\right) \cdot \left(1 - 2^{-L}\right)^{(k-1)^{d+1}-1}.$$

To calculate the probability that at least one cut occurs, we have to average $Q'_T(r')$ with respect to all possible values $r' \in A^{(L)}$. This is given by:

$$2^{-L} \sum_{l=0}^{2^L-1} Q'_T\left(l/2^L\right) = 2^{-L} \sum_{l=0}^{2^L-1} Q_T\left(l/(2^L-1)\right) \cdot \left(1-2^{-L}\right)^{(k-1)^{d+1}-1}$$

Some simple calculations show

$$\lim_{L\to\infty} 2^{-L} \sum_{l=0}^{2^L-1} Q_T\left(l/\left(2^L-1\right)\right) \cdot (1-2^{-L})^{(k-1)^{d+1}-1} = \int_0^1 Q_T(r')dr'.$$

Clearly, the difference between the integral and the sum can be made arbitrary small positive by chosing L large enough. From Section 3.4 in [7], we have that

$$\lim_{d\to\infty} \int_0^1 Q_{T(d)}(r')dr' = \frac{\mu_k}{k-1}$$

where $T(d)$ is an admissible tree of depth d, i.e. we can come arbitrarily close to the right hand side by increasing the depth of T. Combining this with the equation before yields the following result:

Lemma 11. *Given an admissible tree T of depth d, the probability that at least one cut of T occurs is at least $\lambda_{k,d,L}$ with*

$$\lambda_{k,d,L} = \frac{\mu_k}{k-1} - \epsilon_{k,d,L}$$

where $\epsilon_{k,d,L}$ can be made arbitrary small positive by choosing L and d large enough.

4.4 Critical Clause Trees

So, let us draw the connection between Unique-k-SAT and our abstract admissible trees.

We call a clause $C \in G$ a *critical clause* for v if the only true literal in C with respect to β is the one corresponding to v, i.e. flipping the value assigned to v in β would make C instantly false.

Algorithm 1 applies k^d-bounded resolution to G and then steps through the variables ordered by a permutation π. Assume that the bit string b is chosen so that all bits used for variables agree with β. When the algorithm reaches a variable v and there is a critical clause C for v so that the variables $vars(C) - v$ occur before v in π, C has been reduced to a unit clause for v so that the algorithm can immediately determine the right assignment to v. But, when is there a clause C meeting this condition? We need the notion of a critical clause tree.

We call an admissible tree T with root labeled by v a *critical clause tree* for v if for each cut A in T, there exists a critical clause for v in G where $vars(C) - v$ contains only variables which occur as labels in A. The existence of a critical clause tree is given by Lemma 4 in [7]:

Lemma 12. *Let G be a uniquely satisfiable k-CNF with more than d variables. Apply k^d-bounded resolution to G. For each $v \in vars(G)$, there exists a critical clause tree for v with depth d.*

So, if a cut of a critical clause tree of v happens with respect to π, there must be a critical clause C corresponding to that cut meeting the condition. By Lemma 11, this has probability at least $\lambda_{k,d,L}$, and we have proved what was claimed in Lemma 2.

5 Conclusion

We derandomized the uniquely satisfiability case of the PPSZ Algorithm using an approximation of the uniform distribution on $[0, 1]$ using a discrete subset of $[0, 1]$ and showed that we can come arbitrary close to the randomized bound by making the discrete subset large enough.

We can also conclude that a sufficient pseudo-random number generator can be used for the PPSZ Algorithm instead of true randomness for the unique satisfiability case.

Acknowledgements

The author would like to thank Martin Grohe and the unknown referees for valuable remarks and helpful comments.

References

1. Schöning, U.: A probabilistic algorithm for k-SAT and constraint satisfaction problems. In: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS). (1999) 410–414
2. Schuler, R., Schöning, U., Watanabe, O.: A probabilistic 3-SAT algorithm further improved. In: Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS). (2002) 192–202
3. Rolf, D.: 3-SAT $\in RTIME(1.32971^n)$. Diploma thesis, Department Of Computer Science, Humboldt University Berlin, Germany (2003)
4. Baumer, S., Schuler, R.: 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). (2003) 150–161
5. Rolf, D.: 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. Iwama, K., Tamaki, S.: Improved upper bounds for 3-SAT. In: Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). (2004) 328–328
7. Paturi, R., Pudlak, P., Saks, M.E., Zane, F.: An improved exponential-time algorithm for k-SAT. In: Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS). (1998) 628–637

8. Calabro, C., Impagliazzo, R., Kabanets, V., Paturi, R.: The complexity of unique k-SAT: An isolation lemma for k-CNFs. In: Proceedings of the 18th Annual IEEE Conference on Computational Complexity (CCC). (2003) 135–141
9. Dantsin, E., Goerdt, A., Hirsch, E.A., Kannan, R., Kleinberg, J., Papadimitriou, C., Raghavan, P., Schöning, U.: A deterministic $(2 - 2/(k+1))^n$ algorithm for k-SAT based on local search. Theoretical Computer Science **289** (2002) 69–83
10. Alon, N., Spencer, J.: The Probabilistic Method. John Wiley (1992)
11. Paturi, R., Pudlak, P., Saks, M.E., Zane, F.: An improved exponential-time algorithm for k-SAT. Journal of the Association for Computing Machinery (JACM) (to appear)
12. Paturi, R., Pudlak, P., Zane, F.: Satisfiability coding lemma. Chicago Journal of Theoretical Computer Science (1999)