Algorithmica (2002) 32: 615–623
DOI: 10.1007/s00453-001-0094-7

Algorithmica
© 2002 Springer-Verlag New York Inc.

A Probabilistic Algorithm for *k*-SAT Based on Limited Local Search and Restart

U. Schöning[1]

Abstract. A simple probabilistic algorithm for solving the NP-complete problem k-SAT is reconsidered. This algorithm follows a well-known local-search paradigm: randomly guess an initial assignment and then, guided by those clauses that are not satisfied, by successively choosing a random literal from such a clause and changing the corresponding truth value, try to find a satisfying assignment. Papadimitriou [11] introduced this random approach and applied it to the case of 2-SAT, obtaining an expected $O(n^2)$ time bound. The novelty here is to restart the algorithm after $3n$ unsuccessful steps of local search. The analysis shows that for any satisfiable k-CNF formula with n variables the expected number of repetitions until a satisfying assignment is found this way is $(2 \cdot (k-1)/k)^n$. Thus, for 3-SAT the algorithm presented here has a complexity which is within a polynomial factor of $(\frac{4}{3})^n$. This is the fastest and also the simplest among those algorithms known up to date for 3-SAT achieving an $o(2^n)$ time bound. Also, the analysis is quite simple compared with other such algorithms considered before.

Key Words. Satisfiability problem, NP-completeness.

1. Preliminaries. The decision problem k-SAT consists of the set of satisfiable formulas in conjunctive normal form (CNF) where each clause has at most k literals (a literal being a variable or a negated variable). By n we denote the number of variables that occur in a given formula. For convenience we assume that in a k-SAT formula each clause has *exactly* k literals. This can be achieved by doubling some of the literals. The "naive algorithm" for k-SAT which tries out all 2^n truth value assignments to the n variables has a complexity which is within a polynomial factor of 2^n. By the fact that k-SAT is NP-complete [2], [6] for every $k \geq 3$, it would follow that $\mathrm{P} = \mathrm{NP}$ if a polynomial-time algorithm could be devised for this problem (which seems very difficult if not impossible). However, it is still interesting and desirable for practical purposes to find algorithms which are better than the naive 2^n algorithm. A milestone paper in this respect is by Monien and Speckenmeyer [9] where a deterministic algorithm for k-SAT is presented. For 3-SAT their bound is 1.618^n. The best bounds so far have been obtained by a probabilistic algorithm (see [10]) which started in the paper [13] and was further improved by Paturi et al. in [12]. Their algorithm is based on a probabilistic version of the Davis–Putnam procedure. In the case of 3-SAT the bound given in [12] is 1.362^n. Here we present a different probabilistic algorithm for k-SAT based on local search that achieves the bound $(2(k-1)/k)^n$. In the case of 3-SAT the complexity is therefore $(\frac{4}{3})^n$. This is the fastest known algorithm for 3-SAT up to date (but see the remark at the

[1] Abteilung Theoretische Informatik, Universität Ulm, James-Franck-Ring, D-89069 Ulm, Germany. schoenin@informatik.uni-ulm.de.

Received May 21, 2000; revised November 24, 2001. Communicated by K. Mehlhorn.
Online publication January 7, 2002.

end of this paper). Also, the algorithm and its analysis is quite simple compared with its predecessors. Comparing our bounds in the cases $k \geq 4$, these bounds are slightly beaten by the probabilistic algorithm developed by Paturi et al. [12]. They obtain 1.476^n, 1.569^n, and 1.637^n for the cases $k = 4, 5, 6$.

Since we are dealing with exponential complexity bounds here, asymptotically, it is convenient to ignore polynomial factors. The following notation turns out to be very useful. Say that the functions $f, g\colon \mathbb{N} \longrightarrow \mathbb{R}$ are *polynomially related*, if there is a polynomial p such that for all n,

$$f(n) \leq p(n) \cdot g(n) \quad \text{and} \quad g(n) \leq p(n) \cdot f(n).$$

Symbolically we write $f \asymp g$ in this case.

2. The Algorithm and Its Analysis. In the following we describe and analyze our algorithm. First consider the following probabilistic procedure:

Procedure *try* (F: a formula in k-CNF with n variables): Boolean;
Guess an initial assignment $a \in \{0, 1\}^n$, uniformly at random;
Repeat $3n$ times:
  If the formula is satisfied by the actual assignment then return 1;
  Let C be some clause not being satisfied by the actual assignment;
  Pick one of the k literals in the clause uniformly at random and flip its value in the current assignment;
Return 0;

If F is a formula which is unsatisfiable, the result of *try* (F) will always be 0. However, if F is satisfiable, suppose the probability of obtaining the result 1 is p (where p depends on n). Then it is clear that the expected number of repetitions of the procedure *try* until we find a satisfying assignment (i.e. *try* $(F) = 1$) is $1/p$. The probability that we do not find a satisfying assignment after t repetitions with independent random bits is $(1-p)^t \leq e^{-pt}$. Therefore, to achieve an acceptable error probability of, say, e^{-20} one needs to choose $t = 20/p$ independent repetitions of *try*. It is shown below that $p \geq \frac{2}{3} \cdot (k/2(k-1))^n$. Therefore, the following algorithm

For $i := 1$ to $30 \cdot (\frac{2(k-1)}{k})^n$ Do If *try*(F) $= 1$ Then Write ("F is satisfiable");
Write ("No satisfying assignment found")

has complexity which is within a polynomial factor of $(2(k-1)/k)^n$ and achieves a (one-sided) error probability of no more than e^{-20}. A potential error occurs only in the case when the formula is satisfiable, and the algorithm does not find a satisfying assignment.

Now we calculate p. Suppose F is satisfiable. Fix some satisfying assignment a^*. Under a^*, in each clause of F at least one literal is set to 1. In each clause we fix *exactly one* literal which is set to 1 under a^*. Call this literal the *special literal* of the respective clause. Since each clause has exactly k literals, in each step of the procedure *try* the probability of selecting the special literal is *exactly* $1/k$. Let $X_t \in \{0, 1, \ldots, n\}$ ($t = 0, 1, 2, \ldots$) be the random variable which counts the number of bits in which the actual assignment a in the procedure *try* differs from our fixed satisfying assignment a^*,

i.e. the Hamming distance $d(a, a^*)$ between a and a^*. The index t refers to the number of repetitions performed within the procedure *try*. Since the initial assignment a is chosen uniformly at random, X_0 follows a symmetric binomial distribution,

$$\mathbf{Pr}(X_0 = j) = 2^{-n}\binom{n}{j} \quad \text{for} \quad j = 0, 1, \ldots, n.$$

Each time a literal is randomly selected in the procedure *try* and its value is flipped we either decrease the Hamming distance $d(a, a^*)$ by one or we increase it by one, i.e. $X_{t+1} = X_t + 1$ or $X_{t+1} = X_t - 1$. Decreasing the Hamming distance means that we pick one of those literals in the clause which are satisfied under a^*. Notice that it might be the case that the procedure *try* finds at a certain step t a satisfying assignment different from a^*. In this case the procedure returns 1 and the stochastic process stops. In this case (and also in the case that the procedure hits on a^*) we define $X_t, X_{t+1}, X_{t+2}, \ldots$ to be 0.

The actual stochastic process $X_0, X_1, X_2, \ldots$ is a Markov chain with a reflecting barrier at state n, and has varying time- and state-dependent transfer probabilities such as $1/k$, $2/k$, and so on. Also note that the apparent worst-case of reaching state n is not bad at all, since the complementary assignment $\bar{a}$ is a satisfying assignment in this case. Therefore, one might modify the algorithm such that it always checks whether the complement of the actual assignment is satisfying. Instead of analyzing this somewhat complicated stochastic process we choose to analyze closely another process $Y_0, Y_1, Y_2, \ldots$, which is a Markov chain with infinitely many states $0, 1, 2, \ldots$. Let Y_t denote the random variable which takes as its value the state number of this Markov chain after t steps. Initially, this Markov chain is started like the stochastic process above, i.e. $Y_0 = X_0$. As long as the procedure *try* is operating we let $Y_{t+1} = Y_t - 1$ if the procedure selects the special literal for flipping, otherwise we set $Y_{t+1} = Y_t + 1$ (even if the selected literal is satisfied under a^*). After the procedure *try* has stopped we continue with the same transfer probabilities, namely

$$\mathbf{Pr}(Y_{t+1} = j - 1 \mid Y_t = j) = \frac{1}{k} \quad \text{and} \quad \mathbf{Pr}(Y_{t+1} = j + 1 \mid Y_t = j) = \frac{k-1}{k}.$$

By induction on t, it is clear that for each t, $X_t \leq Y_t$. Therefore we can lower bound the above-mentioned probability p as follows:

$$p \;=\; \mathbf{Pr}(\exists t \leq 3n : X_t = 0) \geq \mathbf{Pr}(\exists t \leq 3n : Y_t = 0)$$

since $3n$ is the chosen repetition number within the procedure *try*.

If the Markov chain starts in some state j (i.e. $Y_0 = j$), then it can reach the state 0 in j steps by transfering through the states $j - 1, j - 2, \ldots, 1, 0$. The probability of this happening is $(1/k)^j$. Also, for $i = 1, 2, 3, \ldots$ the state 0 can be reached after $2i + j$ steps where there are i steps which increase the state number and $i + j$ steps which decrease the state number. Let $q(i, j)$ be the probability that $Y_{2i+j} = 0$, such that the state 0 is not reached in any earlier step—under the condition that the Markov chain started in state j, i.e. $Y_0 = j$. More formally,

$$q(i, j) := \mathbf{Pr}(Y_{2i+j} = 0 \text{ and } Y_k > 0 \text{ for all } k < 2i + j \mid Y_0 = j).$$

Clearly, $q(0, j) = (1/k)^j$. In the general case, $q(i, j)$ is $((k-1)/k)^i \cdot (1/k)^{i+j}$ times the number of ways of arranging i increasing steps and $i + j$ decreasing steps such that the

whole sequence starts in state j, ends in state 0 and does not reach 0 before the last step. By the *ballot theorem* (see 3.10 (6), page 77, of [8] or page 73 of [5]), this number is

$$\binom{2i+j}{i} \cdot \frac{j}{2i+j}.$$

Therefore, we have

$$q(i,j) = \binom{2i+j}{i} \cdot \frac{j}{2i+j} \cdot \left(\frac{k-1}{k}\right)^i \cdot \left(\frac{1}{k}\right)^{i+j}.$$

The expression is not defined in the case $i = j = 0$. In this case, $q(0,0) = 1$. Thus we get

$$\begin{aligned}
p &\geq \mathbf{Pr}(\exists t \leq 3n : Y_t = 0) \\
&= \sum_{j=0}^{n} 2^{-n} \binom{n}{j} \sum_{2i+j \leq 3n} q(i,j) \\
&\geq 2^{-n} \sum_{j=0}^{n} \binom{n}{j} \sum_{i=0}^{j} q(i,j) \\
&= 2^{-n} \sum_{j=0}^{n} \binom{n}{j} \sum_{i=0}^{j} \binom{2i+j}{i} \cdot \frac{j}{2i+j} \cdot \left(\frac{k-1}{k}\right)^i \cdot \left(\frac{1}{k}\right)^{i+j} \\
&\asymp 2^{-n} \sum_{j=0}^{n} \binom{n}{j} \sum_{i=0}^{j} \binom{2i+j}{i} \cdot \left(\frac{k-1}{k}\right)^i \cdot \left(\frac{1}{k}\right)^{i+j} \\
&\asymp 2^{-n} \sum_{j=0}^{n} \binom{n}{j} \left(\frac{1}{k-1}\right)^j \\
&= \left(\frac{k}{2(k-1)}\right)^n \qquad \text{by the binomial theorem.}
\end{aligned}$$

Here, the asymptotic estimation

$$\sum_{i=0}^{j} \binom{2i+j}{i} \cdot \left(\frac{k-1}{k}\right)^i \cdot \left(\frac{1}{k}\right)^{i+j} \asymp \left(\frac{1}{k-1}\right)^j$$

is justified as follows. We set $i = \alpha j$ and estimate the summand

$$\binom{2i+j}{i} \cdot \left(\frac{k-1}{k}\right)^i \cdot \left(\frac{1}{k}\right)^{i+j}$$

by

$$\left[\left(\frac{1+2\alpha}{\alpha}\right)^{\alpha} \cdot \left(\frac{1+2\alpha}{1+\alpha}\right)^{1+\alpha} \cdot \left(\frac{k-1}{k}\right)^{\alpha} \cdot \left(\frac{1}{k}\right)^{1+\alpha}\right]^j;$$

this holds since by Stirling's inequality $n! \asymp (n/e)^n$, and then

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} \asymp \frac{(n/e)^n}{(k/e)^k \cdot ((n-k)/e)^{n-k}} = \left(\frac{n}{k}\right)^k \cdot \left(\frac{n}{n-k}\right)^{n-k}.$$

Therefore, for $0 \le \alpha \le 1$,

$$\binom{n}{\alpha n} \asymp \left[\left(\frac{1}{\alpha}\right)^{\alpha} \cdot \left(\frac{1}{1-\alpha}\right)^{1-\alpha}\right]^n,$$

where αn is assumed to be an integer.

Since there are just polynomially many summands the value of the sum is polynomially related to its greatest summand. The greatest summand can be determined by setting the derivative of the above expression in brackets to zero. It turns out that the greatest summand is obtained for $\alpha = 1/(k-2)$. Inserting this value for α yields $(1/(k-1))^j$ as claimed.

We have proved the following theorem.

THEOREM. *For every $k \ge 2$ there is a probabilistic algorithm which solves the k-SAT problem in time which is within a polynomial factor of $(2(k-1)/k)^n$ where n is the number of variables in the input formula.*

Symbolically,

$$k\text{-SAT} \in \text{RTIME}\left(\text{poly}(n) \cdot \left(\frac{2(k-1)}{k}\right)^n\right).$$

The complexity class RTIME$(t(n))$ denotes those decision problems that can be solved by probabilistic algorithms with expected running time $t(n)$ having just one-sided errors (with probability less than 1/2), denoting a generalization of the class RP, see [1].

3. Final Remarks. This paper is based on the conference presentation [14].

Very recently, the algorithm presented here has been further extended and improved for the special case of 3-SAT [15]. Instead of guessing the initial assignments unifomly at random, a different probability distribution is used which depends on the formula F. The improvement is from $(\frac{4}{3})^n$ to 1.3303^n.

The deterministic algorithms presented in [3] and [4] can be considered as derandomized versions of our probabilistic algorithm here. The obtained bound in the case of 3-SAT is 1.481^n.

Acknowledgments. For valuable remarks and discussions I thank S. Baumer, B. Hollas, O. Kullmann, P. Pudlák, R. Schuler, T. Thierauf, and E. Welzl. The unknown referee has done an excellent job by giving several very useful hints for improving and simplifying the presentation of the paper.

Appendix. In this appendix we give an alternative analysis of the algorithm using calculations involving power series. These calculations have been generously supplied to the author by Emo Welzl. The advantage here is that no polynomial "slack" terms occur.

Again, consider the Markov chain given by the random variables $Y_0, Y_1, Y_2, \ldots$. We want to estimate $\mathbf{Pr}(\exists t \leq 3n : Y_t = 0)$. Let $q = 1/k$ be the probability of decreasing the state number on the Markov chain by 1. Let N_j be the random variable that counts the number of steps until the first encounter of state 0, assuming that the process starts in state j, i.e. $Y_0 = j$. (Notice that it is possible that the state 0 will never be reached.)

LEMMA 1. *For $q < \frac{1}{2}$ and $j \in \mathbb{N}_0$ it holds that*

$$\mathbf{Pr}(N_j < \infty) = \left(\frac{q}{1-q}\right)^j.$$

PROOF. By the ballot theorem the number of walks of length $2i + j$ from j to 0 where the first encounter of 0 happens in the last step is $\binom{2i+j}{i}(j/(2i+j))$. Hence,

$$\begin{aligned}\mathbf{Pr}(N_j < \infty) &= \sum_{i=0}^{\infty} \binom{2i+j}{i} \frac{j}{2i+j} (1-q)^i q^{i+j} \\ &= q^j \sum_{i=0}^{\infty} \binom{2i+j}{i} \frac{j}{2i+j} (q(1-q))^i \\ &= q^j (\mathcal{B}_2(q(1-q)))^j,\end{aligned}$$

for $\mathcal{B}_2(z)$ being the generalized Binomial series defined by

$$\mathcal{B}_2(z) = \sum_i \binom{2i+1}{i} \frac{z^i}{2i+1} = \frac{1-\sqrt{1-4z}}{2z}$$

for which

$$(\mathcal{B}_2(z))^r = \sum_i \binom{2i+r}{i} \frac{r}{2i+r} z^i$$

for all $r \in \mathbb{N}_0$, see [7]. So

$$\mathbf{Pr}(N_j < \infty) = q^j \left(\frac{1-\sqrt{1-4q+4q^2}}{2(1-q)q}\right)^j = q^j \left(\frac{1}{1-q}\right)^j. \qquad \square$$

LEMMA 2. *For $q < \frac{1}{2}$ and $j \in \mathbb{N}_0$ it holds that*

$$\mathbf{E}(N_j \mid N_j < \infty) = \frac{j}{1-2q}.$$

PROOF.

$$\begin{aligned}
\mathbf{E}(N_j \mid N_j < \infty) &= \frac{1}{\mathbf{Pr}(N_j < \infty)} \sum_{i=0}^{\infty} (2i+j) \cdot \binom{2i+j}{i} \cdot \frac{j}{2i+j} \cdot (1-q)^i q^{i+j} \\
&= j(1-q)^j \cdot \sum_{i=0}^{\infty} \binom{2i+j}{i} (q(1-q))^i \qquad \text{(by Lemma 1)} \\
&= j(1-q)^j \cdot \frac{(\mathcal{B}_2(q(1-q)))^j}{\sqrt{1-4q(1-q)}} \\
&= \frac{j}{1-2q}. \qquad \square
\end{aligned}$$

Notice that it is not really necessary to resort to power series to prove Lemmas 1 and 2. The standard approach of putting up a difference equation works as well (for similar examples see [8]).

Let N be the random variable that counts the number of steps until state 0 is encountered for the first time. Here the initial distribution Y_0 of the Markov chain is taken into account.

LEMMA 3. *For $q < \frac{1}{2}$ it holds that*

$$\mathbf{Pr}(N < \infty) = \left(\frac{1}{2(1-q)}\right)^n.$$

PROOF.

$$\begin{aligned}
\mathbf{Pr}(N < \infty) &= \sum_{j=0}^{n} \binom{n}{j} 2^{-n} \cdot \mathbf{Pr}(N_j < \infty) \\
&= \sum_{j=0}^{n} \binom{n}{j} 2^{-n} \cdot \left(\frac{q}{1-q}\right)^j \qquad \text{(by Lemma 1)} \\
&= \left(\frac{1}{2(1-q)}\right)^n \qquad \text{(by the binomial theorem).} \qquad \square
\end{aligned}$$

LEMMA 4. *For $q < \frac{1}{2}$ it holds that*

$$\mathbf{E}(N \mid N < \infty) = \frac{qn}{1-2q}.$$

PROOF.

$$\begin{aligned}
\mathbf{E}(N \mid N < \infty) &= \sum_i i \cdot \mathbf{Pr}(N = i \mid N < \infty) \\
&= \sum_i i \cdot \sum_{j=0}^{n} \binom{n}{j} 2^{-n} \cdot \mathbf{Pr}(N_j = i \mid N < \infty)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{2^{-n}}{\mathbf{Pr}(N < \infty)} \cdot \sum_{j=0}^{n} \binom{n}{j} \cdot \sum_{i} i \cdot \mathbf{Pr}(N_j = i) \\
&= \frac{2^{-n}}{\mathbf{Pr}(N < \infty)} \cdot \sum_{j=0}^{n} \binom{n}{j} \cdot \mathbf{E}(N_j \mid N_j < \infty) \cdot \mathbf{Pr}(N_j < \infty) \\
&= (1-q)^n \cdot \sum_{j=0}^{n} \binom{n}{j} \cdot \frac{j}{1-2q} \cdot \left(\frac{q}{1-q}\right)^j \quad \text{(by Lemmas 1–3)} \\
&= \frac{n(1-q)^n}{1-2q} \cdot \sum_{j=0}^{n} \binom{n-1}{j-1} \cdot \left(\frac{q}{1-q}\right)^j \\
&= \frac{n(1-q)^n}{1-2q} \cdot \frac{q}{1-q} \cdot \left(1 + \frac{q}{1-q}\right)^{n-1} = \frac{nq}{1-2q}. \qquad \square
\end{aligned}
$$

LEMMA 5. *For $q < \frac{1}{2}$ and $\lambda \geq 1$ it holds that*

$$\mathbf{Pr}\left(N \leq \frac{\lambda q n}{1-2q}\right) > \left(1 - \frac{1}{\lambda}\right)\left(\frac{1}{2(1-q)}\right)^n.$$

PROOF. Write μ for $\mathbf{E}(N \mid N < \infty)$. Observe that

$$\mathbf{Pr}(N > \lambda\mu \mid N < \infty) < \frac{1}{\lambda}$$

by Markov's inequality, and

$$\mathbf{Pr}(N \leq \lambda\mu) = \mathbf{Pr}(N \leq \lambda\mu \mid N < \infty) \cdot \mathbf{Pr}(N < \infty)$$

since $(N \leq \lambda\mu \wedge N < \infty) \Leftrightarrow (N \leq \lambda\mu)$. $\square$

Now using $q = 1/k$, $k \geq 3$, and $\lambda = 3$, we obtain

$$\mathbf{Pr}(\exists t \leq 3n : Y_t = 0) = \mathbf{Pr}(N \leq 3n) > \frac{2}{3} \cdot \left(\frac{k}{2(k-1)}\right)^n.$$

Thus the number of repetitions necessary to obtain an error probability which is less than e^{-20} is

$$20 \cdot \frac{3}{2} \cdot \left(\frac{2(k-1)}{k}\right)^n,$$

whereas, for a satisfiable formula, the *expected* number of repetitions of procedure *try* until a satisfying assignment is found is at most

$$\frac{3}{2} \cdot \left(\frac{2(k-1)}{k}\right)^n.$$

It is interesting to note here that we did not work with a reflecting barrier at state n. In this case the expected number of steps until zero is reached for the first time is on the

order of 2^n, which forbids a direct application of Markov's inequality. The (apparent) detour via omitting this barrier works here, because then, conditional on the event that zero is reached at all, the expected number of steps until this happens is on the order of n, which makes the tail estimate an easy consequence of Markov's inequality.

Finally, we remark that it is possible to use Lemma 2 directly to prove the estimate

$$\mathbf{E}(N \mid N < \infty) \leq \frac{n}{1-2q}$$

which is somewhat weaker than Lemma 4. This estimate implies

$$\mathbf{Pr}(N \leq 4n) > \frac{1}{4} \cdot \left(\frac{k}{2(k-1)}\right)^n .$$

This would be good enough if the search in *try* was be for $4n$ steps instead of $3n$.

References

[1] J. L. Balcázar, J. Díaz, J. Gabarro: *Structural Complexity I*. Springer-Verlag, New York, 1995.

[2] S. A. Cook: The complexity of theorem-proving procedures. *Proceedings of the* 3*rd Annual ACM Symposium on the Theory of Computing*, 1971, pages 151–158.

[3] E. Dantsin, A. Goerdt, E. A. Hirsch, U. Schöning: Deterministic algorithms for k-SAT based on covering codes and local search. *Proceedings of the* 27*th International Colloquium on Automata, Languages and Programming*. Lecture Notes in Computer Science, Vol. 1853, pages 236–247. Springer-Verlag, Berlin, 2000.

[4] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, U. Schöning: A deterministic $(2-2/(k+1))^n$ algorithm for k-SAT based on local search. *Theoretical Computer Science*, to appear.

[5] W. Feller: *An Introduction to Probability Theory and Its Applications*. Wiley, New York, 1968.

[6] M. R. Garey, D. S. Johnson: *Computers and Intractability - A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[7] R. L. Graham, D. E. Knuth, O. Patashnik: *Concrete Mathematics*. Addison-Wesley, Reading, MA, 1989.

[8] G. R. Grimmett, D. R. Stirzaker: *Probability and Random Processes*, 2nd Edition. Oxford University Press, Oxford, 1992.

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

[10] R. Motwani, P. Raghavan: *Randomized Algorithms*. Cambridge University Press, Cambridge, 1995.

[11] C. H. Papadimitriou: On selecting a satisfying truth assignment. *Proceedings of the* 32*nd Annual IEEE Symposium on the Foundations of Computer Science*, 1991, pages 163–169.

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

[13] R. Paturi, P. Pudlák, F. Zane: Satisfiability coding lemma. *Proceedings of the* 38*th IEEE Symposium on the Foundations of Computer Science*, 1997, pages 566–574.

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

[15] R. Schuler, U. Schöning, O. Watanabe: An improved randomized algorithm for 3-SAT. Technical Report, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2001. Submitted for publication.