Journal of Computer and System Sciences **62**, 367–375 (2001)
doi:10.1006/jcss.2000.1727, available online at http://www.idealibrary.com on IDEAL®

On the Complexity of k-SAT

Russell Impagliazzo[1] and Ramamohan Paturi[2]

University of California–San Diego, La Jolla, California

Received June 22, 1999; revised June 4, 2000;
published online January 21, 2001

The *k-SAT* problem is to determine if a given k-CNF has a satisfying assignment. It is a celebrated open question as to whether it requires exponential time to solve k-SAT for $k \geqslant 3$. Here exponential time means $2^{\delta n}$ for some $\delta > 0$. In this paper, assuming that, for $k \geqslant 3$, k-SAT requires exponential time complexity, we show that the complexity of k-SAT increases as k increases. More precisely, for $k \geqslant 3$, define $s_k = \inf\{\delta$: there exists $2^{\delta n}$ algorithm for solving k-SAT$\}$. Define **ETH** (Exponential-Time Hypothesis) for k-SAT as follows: for $k \geqslant 3$, $s_k > 0$. In this paper, we show that s_k is increasing infinitely often assuming **ETH** for k-SAT. Let s_∞ be the limit of s_k. We will in fact show that $s_k \leqslant (1 - d/k)\, s_\infty$ for some constant $d > 0$. We prove this result by bringing together the ideas of *critical clauses* and the *Sparsification Lemma* to reduce the satisfiability of a k-CNF to the satisfiability of a disjunction of $2^{\varepsilon n}$ k'-CNFs in fewer variables for some $k' \geqslant k$ and arbitrarily small $\varepsilon > 0$. We also show that such a disjunction can be computed in time $2^{\varepsilon n}$ for arbitrarily small $\varepsilon > 0$.

Although all **NP**-complete problems are equivalent as far as the existence of polynomial-time algorithm is concerned, there is wide variation in the worst-case complexity of known algorithms for these problems. For example, there have been several algorithms for maximum independent set [6, 12, 17, 18], and the best of these takes time 1.2108^n in the worst-case [12]. Recently, a 3-coloring algorithm with 1.3446^n worst-case time complexity is presented [2] and it is known that k-coloring can be solved in 2.442^n time [8]. However, it is not known what, if any, relationships exist among the worst-case complexities of various problems. In this paper, we examine the complexity of k-SAT, and derive a relationship that governs

[1] This research is supported by NSF Grant CCR-9734911 from the Theory of Computing Program and INT-9600919/ME103 of the NSF and Czech Republic of Education.

[2] This research is supported by NSF Grant CCR-9734911 from the Theory of Computing Program.

0022-0000/01 $35.00

the complexity of k-SAT for various k under the assumption that k-SAT does not have subexponential algorithms for $k \geqslant 3$.

When we consider algorithms for k-SAT, we find detailed information on the variation in the worst-case complexity: Experimental evidence suggests that variants of classical Davis-Putnam heuristic scales as $2^{n/19.5}$ for the hardest instances of 3-SAT [3]. Furthermore, it has been observed [16] that the Davis-Putnam heuristic scales much worse for $k \geqslant 4$ due to reduced number of unit clauses and the noneffectiveness of the shortest clause heuristic. Also, all the recent results that show improved exponential-time algorithms for k-SAT [4, 7, 9–11, 13–15] exhibit increasing complexity as k increases. In particular, [11] exhibits a randomized algorithm for solving k-SAT with time complexity $O(2^{(1-(\mu_k/k-1))n})$ where $\mu_k > 1$ is an increasing function of k and approaches $\pi^2/6 \approx 1.644$. More recently, using a very simple analysis, Schöning [15] also obtained upper bounds of the form $2^{(1-\gamma_k/k)n}$. This is the best known upper bound for 3-SAT. However, for $k \geqslant 4$, the bounds in [11] are better.

To support the claim that the complexity of k-SAT increases with increasing k, we provide the first rigorous evidence. We make this claim more precise as follows: For $k \geqslant 3$, define $s_k = \inf\{\delta$: there exists $O(2^{\delta n})$ algorithm for solving k-SAT$\}$. Define **ETH** (Exponential-Time Hypothesis) for k-SAT as follows: for $k \geqslant 3$, $s_k > 0$. In other words, for $k \geqslant 3$, k-SAT does not have a subexponential-time algorithm. In this paper, we show that s_k is increasing infinitely often assuming **ETH** for k-SAT. Although many non-trivial algorithms for k-SAT exists, all are strictly exponential ($2^{\Omega(n)}$) in the worst-case, and it is an important open question whether subexponential algorithms exist. The plausibility of such a subexponential time algorithm for k-SAT was investigated in [5], using subexponential time reductions. It is shown there that linear size 3-SAT is complete for the class **SNP** with respect to such reductions, where **SNP** is the class of properties expressible by a series of second order existential quantifiers, followed by a series of first order universal quantifiers, followed by a basic formula (a boolean combination of input and quantified relations applied to the quantified element variables.) This result implies the following equivalent formulations of **ETH**.

THEOREM 1. *The following statements are equivalent* [5]:

1. **ETH**: *For all* $k \geqslant 3$, $s_k > 0$.
2. *For some* k, $s_k > 0$.
3. $s_3 > 0$.
4. **SNP** $\not\subseteq$ **SUBEXP**.
5. *Satisfiability of linear-sized circuits cannot be solved in subexponential time.*

We feel that this provides at least intuitive evidence that such an algorithm is unlikely to exist.

If k-SAT does not have a subexponential time algorithm (**ETH** for k-SAT), it is interesting to have a more precise idea regarding the constant s_k in the exponent. For instance, even under **ETH** for k-SAT, it still probably is a very challenging question to prove any lower bounds on $s_\infty = \lim_{k \to \infty} s_k$. Can we at least show that s_k is an increasing sequence? How are s_k related? Uncovering the relationships

among s_k will enable us to bound s_k in terms of s_∞ and k thus giving some evidence as to the optimality or nonoptimality (under the assumption **ETH** for k-SAT) of the recent exponential-time algorithms for k-SAT.

In this paper, we will show that $s_k \leqslant (1-d/k)\, s_\infty$ where the constant $d \approx s_\infty / (2e \log(2/s_\infty))$. More precisely, given any integer $k \geqslant 3$ and $\varepsilon > 0$, we find a k' so that the following type of reduction is possible: Let F be a k-CNF in the variables $\{x_1, ..., x_n\}$. For some $m \leqslant 2^{\varepsilon n}$, we will construct k'-CNF's $F_1, ..., F_m$ in at most $n(1-d/k)$ variables in time $\text{poly}(n)\, 2^{\varepsilon n}$ such that F is satisfiable iff $\bigvee_{i=1}^m F_i$ is satisfiable. Then we bound s_k as follows: By solving each F_i using an algorithm running in $2^{(s_{k'}+\varepsilon)(1-d/k)\, n}$ time, we can determine whether F is satisfiable in time

$$\text{poly}(n)\, 2^{s_{k'}(1-d/k)\, n + 2\varepsilon n} \leqslant 2^{s_\infty (1-d/k)\, n + 2\varepsilon n}.$$

Thus $s_k \leqslant s_\infty (1-d/k) + 2\varepsilon$. Since ε is arbitrarily small, we get the desired bound for s_k. It then follows that, assuming **ETH**, every $s_k < s_{k'}$, for some $k' > k$. Thus, the nondecreasing sequence s_k is, in fact, strictly increasing infinitely often.

1. INTUITION

1.1. Tools from Previous Work

Our proof relies on earlier ideas regarding critical clauses [10, 11] and the *decomposition* of an arbitrary k-CNF into linear size k-CNFs [5]. We will first develop these ideas.

The *Sparsification Lemma* [5] essentially says that an arbitrary k-CNF can be expressed (in subexponential time) as the disjunction of a subexponential number of linear size k-CNFs. More precisely, the Sparsification Lemma states the following:

> For all $\varepsilon > 0$, k-CNF F can be written as the disjunction of at most $2^{\varepsilon n}$ k-CNF F_i such that F_i contains each variable in at most $c(k, \varepsilon)$ clauses for some function c. Moreover, this reduction takes at most $\text{poly}(n)\, 2^{\varepsilon n}$ time.

Let F be a k-CNF. We say that a satisfying assignment $\vec{x} = (x_1, ..., x_n)$ of F is *isolated* with respect to a variable x if $\vec{x}$ is no longer a satisfying assignment when the bit x is flipped. The crucial observation [10] is that if a satisfying assignment $\vec{x}$ is isolated with respect to a variable x (a *critical variable* for $\vec{x}$), there must exist a clause C (a *critical clause* for x at $\vec{x}$) in F such that the only true literal in C at the assignment $\vec{x}$ is the one corresponding to the variable x. We say that a variable x is *forced* by an assignment α to a subset of the variables if x or $\bar{x}$ appears in a clause and all the other variables in the clauses are set to false by the assignment α.

The ideas of critical clauses and forcing are used to analyze a variant of Davis-Putnam procedure in [10] to obtain better exponential-time upper bounds for k-SAT. The analysis relies on accounting for the number of variables forced due to unit clauses. The key idea of the analysis is that a k-CNF either has a sufficiently isolated satisfying assignment and thus a satisfying assignment which has *critical clauses* for many

variables or has a large number of satisfying assignments. If a satisfying assignment has critical clauses for l variables, it is argued that on average at least l/k variables are forced. On the other hand, if the k-CNF has sufficiently many satisfying assignments, then it is easier to randomly find one. In either case, it is shown that the probability of finding a satisfying assignment is $2^{-n(1-1/k)}$ which implies a time bound of $\text{poly}(n)\, 2^{n(1-1/k)}$. A more intricate analysis of critical clauses [11] yields the better upper bound mentioned earlier.

1.2. New Ideas

Another way of looking at the analysis of [10] is that a sufficiently isolated satisfying assignment of k-CNF F can be succinctly represented: if a satisfying assignment x is isolated in δn directions, then x can be *coded* using $(1-\delta/k)\, n$ bits. Thus, one need only search in a smaller space to find a satisfying solution. Alternately, we can say that the satisfiability of the k-CNF F can be reduced to the satisfiability of a polynomial size circuit C on at most $(1-\delta/k)\, n$ variables. The circuit C views its inputs as the code of a satisfying assignment, decodes it, and checks whether it is in fact a satisfying assignment of F. Unfortunately, for a general formula F and for the coding method used in [10], the complexity of such a circuit C is high and it is hard to reduce C to a k'-CNF. However, we observe that by first applying the Sparsification Lemma, it suffices to handle the case when each variable occurs in a constant number of clauses. Then by combining some nondeterminism with a simpler coding of satisfying assignments, we make the decoding function local in the sense that each output of the decoded satisfying assignment depends on only a constant number of bits of the coded assignment. This allows us to reduce a k-CNF to a disjunction of a small exponential number of k'-CNFs in fewer variables.

For the special case when the formula is uniquely satisfiable, we follow the above argument directly. To prove our result for general k-CNF, we require a further modification: We argue that if a general k-CNF has a satisfying assignment with at most δn (for an appropriately chosen $\delta > 0$) 1's, then such a satisfying assignment can be found in time $2^{h(\delta)\, n}$ using exhaustive search, where $h(\delta)$ is the binary entropy function. In the other case, we are guaranteed that if the k-CNF is satisfiable, then it has a satisfying assignment that is critical with respect to at least δn variables.

In the following sections, we provide the details of the proof for the uniquely satisfiable k-CNF and then extend the proof to the general case.

Unique k-SAT

Let F be a k-CNF with at most one satisfying assignment and each variable appearing in at most c clauses. If F is satisfiable, then there is at least one critical clause for each variable at the unique satisfying assignment α. If we apply the standard Davis-Putnam procedure with a random ordering of the variables, then we expect that at least n/k of the variables appear as unit clauses and thus are forced if all the other nonforced variables are set according to α. Our goal is to

eliminate the forced variables by rewriting them in terms of other variables. In this trade-off, we increase the clause width for a reduced number of variables.

Although the implicit dependencies among the variables do not seem obvious, we show that we need only search a relatively small space to uncover these dependencies. We first show that by a simple random selection we can *concentrate* the forced variables. Let the variables be partitioned into sets A and B. With respect to the partition (A, B), we say that the variable x is *forced* by an assignment α_A to the variables in A if $x \in B$ and F contains a clause containing x or its complement such that all the other literals in the clause are from A and are set to False by the assignment α_A.

LEMMA 1. *Let F be a uniquely satisfiable k-CNF and α be the unique satisfying assignment. Let A and B be random sets of variables created by the following process: For each variable x, $x \in B$ with probability $1/k$, otherwise $x \in A$. Then B contains at least $n/(ek)$ forced variables on average with respect to the assignment α_A, the restriction of α to A.*

Proof. Since α is the unique satisfying assignment of F, all variables are critical. Let x be a variable and C_x be a critical clause for x at the unique satisfying assignment α. Since α makes all the other literals in the clause False, the probability that x is forced by α_A is the same as the probability $x \in B$ and all other variables in C_x are in A, which is at least $\frac{1}{k}(1-1/k)^{k-1} \geqslant 1/(ek)$. Hence B contains at least $n/(ek)$ forced variables on average with respect to α_A. ∎

To obtain the above lower bound on the probability of the event "x is forced", it is sufficient that the events "$y \in B$" for the variables y appearing in the clause C_x are independent. Thus, we can eliminate the randomness by using a k-wise independent distribution. We can indeed construct a size $O(n^{3k})$ k-wise independent probability space in polynomial time [11]. We will try all possible selections of A and B from such a space.

In the rest of the discussion, we will assume A and B are a partition for which the conclusion of the lemma is true. We will also assume that α is the unique satisfying assignment of F and α_A is its restriction to A. We also assume that each variable in F appears in at most c clauses. Our goal is to eliminate the forced variables by rewriting them in terms of other variables. More precisely, we want to find a formula $F(A, B)$ (in subexponential time) which only depend on the variables in A and a small number of new variables (these are the renamed unforced variables in B) such that $F(A, B)$ is satisfiable iff F is satisfiable.

For each variable $x \in B$, the proposition "x is forced by α" can be expressed by the formula G_x where G_x is a DNF with at most c terms and with each term containing at most $(k-1)$ literals. Call a clause C of F an (x, A) *clause* if x or $\bar{x}$ occurs in C and all other variables in C are from A. Call a clause C of F a *positive* (x, A) *clause* if x occurs in C and all other variables in C are from A. If an (x, A) clause were a critical clause for x at α, then all the other literals in the clause will assume the value False at α_A. G_x is precisely the disjunction of terms where each term is the product of the negations of all literals except x or $\bar{x}$ of a (x, A) clause of F. Similarly, we define G'_x as the disjunction of terms where each term is the product of the negations of all literals except x of a positive (x, A) clause of F. G'_x expresses

the statement, "x is forced to be true". Observe that G_x and G'_x depend only on at most $c(k-1)$ variables in A.

Let l be an integer (to be chosen later). To identify the forced variables in B further, we partition B arbitrary into sets $B_1, ..., B_p$ of size l. Our goal is to rewrite F by eliminating the forced variables in each B_i and to rename the remaining variables using new variables names. For this purpose, we want to keep track of the number of forced variables. Let f_i be the number of forced variables in B_i. We will now rewrite F as a k'-CNF $\Gamma_{\vec{f}}$ over the variables in A and the renamed unforced variables in B.

Let Φ_i be the f_ith slice function in the variables G_x for $x \in B_i$. (A Boolean function $g(x_1, ..., x_n)$ is a j'th slice function if g is true iff exactly j of its inputs are true.) Φ_i depends only on the variables in A and furthermore only on at most $cl(k-1)$ of them.

Having sufficiently identified the forced variables, we will express each variable in B_i in terms of variables in A and a smaller number, $l-f_i$, of new variables. Let $Y_i = \{y_{i_1}, ..., y_{i_{l-f_i}}\}$ be a set of new variables. We will rename the unforced variables in B_i with the variables in Y_i. Let $B_i = \{x_1, ..., x_l\}$ without loss of generality. For $x_j \in B_i$, observe that the following proposition is satisfiable:

> F and [x_j is true iff
> either x_j is not forced to be false
> or x_j is the j' th unforced variable renamed as $y_{i_{j'}} \in Y_i$ and $y_{i_{j'}}$ is true].

To figure out which new variable $y_{j'}$ is to be assigned to x_j in case x_j is not forced, we consider all the variables $x_1, x_2, ..., x_{j-1}$ in B_i that occur before x_j and check how many of them are forced. Let $\beta_j = \beta_j(G_{x_1}, ..., G_{x_{j-1}}, y_{i_1}, ..., y_{i_j})$ be a Boolean expression that evaluates to y_{i_q} if and only if $q-1$ of the G are true. Let Ψ_{i,x_j} be the proposition $G'_{x_j} \vee (\bar{G}_{x_j} \wedge \beta_j)$. Intuitively, Ψ_{i,x_j} expresses the following:

> Either x_j is not forced to be false
> or x_j is the j'th unforced variable renamed as $y_{i_{j'}}$ and $y_{i_{j'}}$ is true.

Ψ_{i,x_j} depends on at most $lc(k-1)$ variables in A and on the variables in Y_i, and thus on a total of lck variables.

Substitute Ψ_{i,x_j} for x_j in F. After the substitution, each clause in F depends on at most lck^2 variables from $Y = A \cup \bigcup_{i=1}^{p} Y_i$. Call the new formula F'. Define $\Gamma_{\vec{f}} = F' \wedge \bigwedge_{i=1}^{p} \Phi_i$. Define $k' = clk^2$. Thus $\Gamma_{\vec{f}}$ is a k'-CNF in the variables in Y. Intuitively, $\Gamma_{\vec{f}}$ expresses that for the satisfying assignment α of F, f_i variables in B_i are forced by α_A and the remaining variables in B are renamed as y_{i_j}'s.

We will now define $\Gamma = \bigvee_{\vec{f}} \Gamma_{\vec{f}}$ where $\vec{f}$ ranges over all vectors $(f_1, ..., f_p)$ such that $\sum_{i=1} f_i \geqslant n/(ke)$.

Since $|B| \leqslant n$, it then follows that the number of vectors $\vec{f}$ under consideration is at most $(l+1)^{n/l}$. By selecting l such that l satisfies $\log(l+1)/l \leqslant \varepsilon$, we have Γ as the disjunction of at most $2^{\varepsilon n}$ k'-CNFs on at most $n(1-1/(ek))$ variables, where $k' = clk^2$.

It is clear that if F is uniquely satisfiable, then there is exactly one $\vec{f}$ such that $\Gamma_{\vec{f}}$ is uniquely satisfiable and all other $\Gamma_{\vec{f}}$ are unsatisfiable. Moreover if F is not satisfiable, then Γ is also not satisfiable.

To eliminate the randomness in the selection of the partition A and B, we try all the partitions (A, B) from an appropriate k-wise independent probability space and construct Γ_{AB} as above. Define $\Gamma = \bigvee \Gamma_{AB}$ where the disjunction ranges over all partitions from the probability space of size $n^{O(k)}$.

So far we have assumed that each variable in F appears in at most c clauses. If this is not true, we first use the Sparsification Lemma to write $F = \bigvee_i F_i$ where each F_i contains at most $c(k, \varepsilon)$ occurrences of each variable and there are at most $2^{\varepsilon n}$ formulas F_i. Then for each F_i, we construct Γ_i as above and let $\Gamma = \bigvee_i \Gamma_i$. Since each Γ_i is a disjunction of at most $2^{\varepsilon n}$ k'-CNFs, Γ is a disjunction of at most $2^{2\varepsilon n}$ k'-CNFs. Thus we obtain the following theorem.

THEOREM 2. *For any $\varepsilon > 0$, there is a k' such that the following holds: If F is a k-CNF with at most one satisfying assignment, then the satisfiability of F is equivalent to the satisfiability of $\hat{F}$ where $\hat{F}$ is a disjunction of at most poly$(n)\, 2^{2\varepsilon n}$ k'-CNF on at most $n(1 - 1/(ek))$ variables. Moreover, $\hat{F}$ can be computed from F in time poly$(n)\, 2^{2\varepsilon n}$.*

Let $\sigma_k = \inf\{\sigma \mid \text{Unique } k\text{-SAT is solvable in time } 2^{\sigma n}\}$. Let $\sigma_\infty = \lim_{k \to \infty} \sigma_k$. By Theorem 2, we can reduce the satisfiability of a k-CNF F with at most one satisfying assignment to the satisfiability of a disjunction of at most $2^{2\varepsilon n}$ k'-CNF F_i on at most $n(1 - 1/(ek))$ variables in time poly$(n)\, 2^{2\varepsilon n}$ for arbitrary $\varepsilon > 0$. By solving each F_i using an algorithm running in $2^{(\sigma_{k'} + \varepsilon)(1 - 1/(ek))\, n}$ time, we can determine whether F is satisfiable in time

$$\text{poly}(n)\, 2^{\sigma_{k'}(1 - 1/(ek))\, n + 2\varepsilon n} \leqslant 2^{\sigma_\infty (1 - 1/(ek))\, n + 2\varepsilon n}.$$

Thus $\sigma_k \leqslant \sigma_\infty (1 - 1/(ek)) + 2\varepsilon$. Since ε is arbitrarily small, we get

COROLLARY 1. $\sigma_k \leqslant (1 - 1/(ek))\, \sigma_\infty$

General k-SAT

For general k-SAT, we consider two cases. Let $\delta > 0$ (to be selected later). For a k-CNF F, if F is satisfiable, either there is a satisfying assignment which is isolated with respect to at least δn variables or not. If there is such a δn-isolated satisfying assignment, we will use a similar analysis as in the unique k-SAT case to obtain the following lemma.

LEMMA 2. *Let F be a k-CNF such that F is not satisfiable by any assignment that contains fewer than δn 1's. For any $\varepsilon > 0$, there exists k' such that the following holds: The satisfiability of F is equivalent to the satisfiability of $\hat{F}$ where $\hat{F}$ is a disjunction of at most $2^{2\varepsilon n}$ k'-CNFs on at most $n(1 - \delta/(ek))$ variables. Moreover, $\hat{F}$ can be computed from F in time poly$(n)\, 2^{2\varepsilon n}$.*

Sketch of the proof. Assume that F has a satisfying assignment. Let α be a minimal assignment. By hypothesis, α has at least δn l's and by minimality α is isolated with respect to each of these δn variables. As before, for any partition A and B of variables, we define our notion of "forcing" with respect to the assignment α_A. A random (or k-wise independent) partition A and B will on average force at least $\delta n/(ek)$ variables in B with respect to α_A. The rest of the proof follows a very similar line to that of the unique satisfiability case.

From this, we derive the following theorem.

THEOREM 3. *$s_k \leqslant s_\infty(1-d/k)$ where $d \approx s_\infty/(2e \log(2/s_\infty))$.*

Sketch of the proof. Set δ so that $h(\delta) \leqslant s_\infty/2$ and let $\varepsilon > 0$. The following algorithm solves k-SAT problem:

On input k-CNF F on n variables:

1. By exhaustive search, check if any assignment with at most δn 1's satisfies F.
2. If so, accept and halt.
3. Otherwise, using the reduction in Lemma 2 construct $\hat{F}$, a disjunction of at most $2^{\varepsilon n}$ k'-CNF F_i on at most $n(1-\delta/(ek))$ variables each.
4. To check the satisfiability of each F_i, use any algorithm that runs in time at most $2^{(s_{k'}+\varepsilon)(1-\delta/(ek))n}$.
5. Accept iff one of the F_i is satisfiable.

The correctness of this algorithm follows directly from Lemma 2. This algorithm runs in time

$$2^{h(\delta)n} + 2^{\varepsilon n} + 2^{\varepsilon n} 2^{(s_{k'}+\varepsilon)(1-\delta/(ek))n} \leqslant 2^{(s_\infty(1-d/k)+2\varepsilon)n}.$$

The theorem follows since ε is arbitrarily small.

Open Problems

While our results are a first step towards understanding why some **NP**-complete problems seem more difficult than others, much more work is needed to clarify the relationships between the **NP**-complete problems. A few concrete problems that explore such relationships are:

1. Prove a converse of Theorem 3. Namely, efficiently reduce k-CNF to k'-CNF so that the clause width is reduced ($k' < k$) by increasing the number of variables as little as possible. One such reduction can be obtained by the Sparsification Lemma. However, the increase in the number of variables is too large to be useful.
2. It is reasonable to believe that $s_\infty = 1$ based on known exponential time algorithms for k-SAT. Assuming **ETH** for k-SAT or another reasonable hypothesis, prove $s_\infty = 1$.
3. Obtain a similar relationship for the worst-case complexity for various k-colorability problems.

ACKNOWLEDGMENTS

We thank the reviewers for their careful review and useful comments.

REFERENCES

1. N. Alon, J. Spencer, and P. Erdős, "The Probabilistic Method," Wiley, New York, 1992.
2. R. Beigel and R. Eppstein, 3-Coloring in time $O(1.3446^n)$ time: a no-MIS algorithm, *in* "Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science," pp. 444–453, 1995.
3. J. M. Crawford and L. D. Auton, Experimental results on the crossover point in random 3SAT, *Artificial Intelligence* **81** (1996).
4. E. A. Hirsch, Two new upper bounds for SAT, *in* "SIAM Conference on Discrete Algorithms, 1997."
5. R. Impagliazzo, R. Paturi, and F. Zane, Which problems have strongly exponential complexity, *in* "1998 Annual IEEE Symposium on Foundations of Computer Science," pp. 653–662.
6. T. Jian, An $O(2^{0.304n})$ algorithm for solving maximum independent set problem, *IEEE Trans. Comput.* **35** (1986), 847–851.
7. O. Kullmann and H. Luckhardt, Deciding propositional tautologies: algorithms and their complexity, submitted.
8. E. Lawler, A note on the complexity of the chromatic number problem, *Inform. Process. Lett.* **5** (1976), 66–67.
9. B. Monien and E. Speckenmeyer, Solving satisfiability in less than 2^n steps, *Discrete Appl. Math.* **10** (1985), 287–295.
10. R. Paturi, P. Pudlák, and F. Zane, Satisfiability coding lemma, *in* "Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science," pp. 566–574, 1997.
11. R. Paturi, P. Pudlák, M. Saks, and F. Zane, An improved exponential-time algorithm for k-SAT, *in* "1998 Annual IEEE Symposium on Foundations of Computer Science," pp. 628–637, 1998.
12. J. Robson, Algorithms for maximum independent sets, *J. Algorithms* **7** (1986), 425–440.
13. I. Schiermeyer, Solving 3-satisfiability in less than 1.579^n steps, *in* "Selected Papers from CSL '92," Lecture Notes in Computer Science, Vol. 702, pp. 379–394, Springer-Verlag, Berlin/New York, 1993.
14. I. Schiermeyer, Pure literal look ahead: an $O(1.497^n)$ 3-satisfiability algorithm, *in* "Workshop on the Satisfiability Problem, Siena, April 29–May 3, 1996," Technical Report, University of Köln, Report 96-230, 1996.
15. U. Schöning, A probabilistic algorithm for k-SAT and constraint satisfaction problems, *in* "1999 Annual IEEE Symposium on Foundations of Computer Science," pp. 410–414, 1999.
16. B. Selman, Personal communication, 1999.
17. M. Shindo and E. Tomita, A simple algorithm for finding a maximum clique and its worst-case time complexity, *Systems Comput. Japan* **21** (1990), 1–13.
18. R. Tarjan and A. Trojanowski, Finding a maximum independent set, *SIAM J. Comput.* **6** (1977), 537–546.