Complexity of k-SAT

Russell Impagliazzo*, Ramamohan Paturi†
University of California, San Diego

Abstract

The problem of k-SAT is to determine if the given k-CNF has a satisfying solution. It is a celebrated open question as to whether it requires exponential time to solve k-SAT for $k \geq 3$. Define s_k (for $k \geq 3$) to be the infimum of $\{\delta :$ there exists an $O(2^{\delta n})$ algorithm for solving k-SAT$\}$. Define **ETH** (Exponential-Time Hypothesis) for k-SAT as follows: for $k \geq 3$, $s_k > 0$. In other words, for $k \geq 3$, k-SAT does not have a subexponential-time algorithm. In this paper, we show that s_k is an increasing sequence assuming **ETH** for k-SAT. Let s_∞ be the limit of s_k. We will in fact show that $s_k \leq (1 - d/(ek))s_\infty$ for some constant $d > 0$.

Although all **NP**-complete are equivalent as far as the existence of polynomial-time algorithms is concerned, there is wide variation in the worst-case complexity of known algorithms for these problems. Consider the the complexity of k-SAT algorithms: It has been rumored that the classical Davis-Putnam heuristic does not work well as k increases. Also, all the recent results that show improved exponential-time algorithms for k-SAT [5, 4, 2, 8, 9, 6, 7, 10] exhibit increasing complexity as k increases. In particular, the best of these results [7] exhibits a randomized algorithm for solving k-SAT with time complexity $O(2^{(1-\frac{\mu_k}{k-1})n})$ where $\mu_k > 1$ is an increasing function of k and approaches $\pi^2/6 \approx 1.644$. The algorithm is a variant of Davis-Putnam procedure and its analysis relies on accounting for the number of variables *forced* due to unit clauses. The key idea of the analysis [6] is that the k-SAT either has a sufficiently isolated solution and thus a solution which has several *critical clauses* for many variables or has a large number of satisfying solutions. If a variable has l critical clauses at a satisfying solution, it is argued that the probability that the variable is forced is at least l/k. On the other hand, if the k-SAT has sufficiently many solutions, then it is easier to randomly

*This research is supported by NSF grant CCR-9734911 from Theory of Computing Program and INT-9600919/ME103 of the NSF and Czech Republic of Education

†This research is supported by NSF grant CCR-9734911 from Theory of Computing Program

find one. In either case, it is shown that the probability of finding a satisfying solution is $2^{-n(1-1/k)}$ which implies a time bound of poly$(n)2^{n(1-1/k)}$. A more intricate analysis of critical clauses [7] yields the better lower bounds mentioned earlier. More recently, using a very simple analysis, Schöning [10] also obtained upper bounds of the form $2^{(1-\frac{\gamma_k}{k})n}$ although the constant γ_k is smaller than the constant μ_k in [7].

Despite the accumulated evidence regarding the complexity of k-SAT, we do not have any results that rigorously support the claim that the complexity of k-SAT increases with increasing k. In this paper, we prove that the complexity of k-SAT has to necessarily increase with increasing k provided k-SAT does not have subexponential time algorithms for $k \geq 3$. Although many non-trivial algorithms for k-SAT exist, all are strictly exponential, $2^{\Omega(n)}$, in the worst-case, and it is an important open question whether subponential algorithms exist. The plausibility of such a subexponential time algorithm for k-SAT was investigated in [3], 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 that the **ETH** mentioned earlier can be equivalently stated as "For some $k \geq 3$, $s_k > 0$" or "$\mathbf{SNP} \not\subseteq \mathbf{SUBEXP}$", or "Satisfiability of linear-sized circuits can be solved in sub-exponential 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_∞. 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 of therecent better exponential-time algorithsm for k-SAT under the assumption **ETH** for k-SAT.

In this paper, we will show that $s_k \leq (1 - d/(ek))s_\infty$ where the constant $d \approx s_\infty/(2\log(2/s_\infty))$. More precisely, given any integer $k \geq 3$ and any $\epsilon > 0$, we find a k' so that the following type of reduction is possible: Let F be a k-CNF in the variables $\{x_1, \ldots, x_n\}$. For some $m \leq 2^{\epsilon n}$, we will construct k'-CNF's $F_1, \ldots, F_m$ in at most $n(1 - d/(ek))$ variables in time poly$(n)2^{\epsilon n}$ 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 $2^{(s_{k'}+\epsilon)(1-d/(ek))n}$ time, we can determine whether F is satisfiable in time poly$(n)2^{s_{k'}(1-d/(ek))n+2\epsilon n} \leq 2^{s_\infty(1-d/(ek))n+2\epsilon n}$.

Thus $s_k \leq s_\infty(1 - d/(ek)) + 2\epsilon$. Since ϵ is arbitrarily small, we get the desired bound for s_k.

Our proof relies on earlier ideas regarding critical clauses [6, 7] and the *decomposition* of an arbitrary k-CNF into linear size k-CNFs [3]. The new key idea in the paper is a reduction by which we can decrease the number of variables by increasing the width of the clauses. In the following we develop the ideas we need to prove our result.

Let F be a k-CNF. We say that a satisfying solution $\vec{x} = (x_1, \ldots, x_n)$ of F is *isolated* with respect to a variable x if $\vec{x}$ is no longer a satisfying solution when the bit x is flipped. The crucial observation [6] is that if a satisfying solution $\vec{x}$ is isolated with respect to 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. One of our key ideas is to use critical clauses to express the critical variables in terms of other variables. In order to efficiently identify the critical clauses, at the outset we express an arbitrary k-CNF as a subexponential disjunction of linear size k-CNFs. Such an expression is possible following the result from the *Sparsification Lemma* [3]: For all $\epsilon > 0$, k-CNF F can be written as the disjunction of at most $2^{\epsilon n}$ k-CNF F_i such that F_i contains each variable in at most $c(k, \epsilon)$ clauses. Moreover, this reduction takes at most $O(\text{poly}(n)2^{\epsilon n})$ time.

Hence, it is sufficient to deal with k-CNF F where each variable occurs at most c times for some $c > 0$. To prove our result, we first consider the case of k-CNF containing exactly one solution. Since the unique solution is isolated in all directions, the k-CNF must containat least one critical clause for each variable. We will prove our result in two cases. We first consider the case that the k-CNF F has at most one satisfying solution. We then extend the proof to the general k-CNF.

Unique k-SAT

Let F be a k-CNF with at most one satisfying solution with each variable appearing at most c cluases. If F is uniquely satisfiable, then there is at least one critical clause for each variable at the unique solution. 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 to appear as unit clauses and thus are forced if all the other nonforced variables are set according to the unique solution. We will make the dependencies among the variables explicit by trading up 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 smaller search 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. We say that the variable x is *forced* (according to the partition A and B) if $x \in B$ and F contains a critical clause for x such that all the other variables in the critical clause are in A. From now on, we use the notion "x is forced" or "forced variable" with respect to a partition and we usually suppress the reference to the partition when no ambiguity results.

Lemma 1 *Let F be a uniquely satisfiable k-CNF. Let A and B be random sets of variables created by the process: variable $x \in B$ with probability $1/k$, otherwise $x \in A$. Then B contains at least $n/(ek)$ forced variables on average.*

Proof: Let x be a variable and C_x be a critical clause for it at the unique solution. The probability $x \in B$ and all other variables in C_x are in A is at least $\frac{1}{k}(1 - 1/k)^{k-1} \geq 1/(ek)$. Hence B contains at least $n/(ek)$ forced variables on average. ■

In fact, we can eliminate the randomness by using a k-wise independent distribution. We can construct k-wise random space of size $O(n^{3k})$ in polynomial time [1]. We will try all possible selections of A and B from such a space.

Let l be an integer (to be chosen later). To identify the forced variables in B further, we partition B into sets $B_1, \dots, B_p$ of size l. For each B_i, we nondeterministically select the number $f_i \in \{0, 1, \dots, l\}$ of forced variables in B_i such that $\sum_i f_i \geq n/(ek)$. We will eliminate the nondeterminism by trying out atmost $2^{n \log(l+1)/l}$ such choices. For each variable $x \in B_i$, the proposition "x is forced" can be expressed by the formula $F \wedge G_{i,x}$ where $G_{i,x}$ is a DNF with at most c terms and with each term containing at most $(k-1)$ literals. Consider a clause in F containing x or its negation such that all the other variables in the clause are in A. If the clause were a critical clause for x at the unique solution, then all the other literals in the clause will assume the value False at the unique solution and the literal corresponding to x will be True. Hence, consider the product of the negation of the literals in the clause except for the literal corresponding to the variable x. $G_{i,x}$ is precisely the disjunction of all such products obtained by considering all the clauses in F containing x or its negation with all the other variables in the clause in A. Observe that $G_{i,x}$ depends only on the variables in A. It is now easy to see that $F \wedge G_{i,x}$ is satisfiable iff x is forced. If x is forced, then there is a critical clause C for x at the unique solution such that $x \in B$ and all other variables in C are in A. $G_{i,x}$ then contains a term which is the conjunction of the negation of all the other variables in C. Hence, the unique solution satisfies both F and $G_{i,x}$. If, on the other hand, $F \wedge G_{i,x}$ is satisfiable, then it is satisfiable by the unique solution of F. Since $G_{i,x}$ is satisfiable at the unique solution, there must exist a clause C in F which is critical for x at the unique solution and such that $x \in B$ and all the other variables in C are in A.

Let Φ_i be the f_i-th slice function in the variables $G_{i,x}$ for $x \in B_i$. It is now easy to see that $F \wedge \Phi_i$ is satisfiable iff and exactly f_i of the x's in B_i are forced. 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 a fewer number $l - f_i$ of variables. Let $Y_i = \{y_{i_1}, \ldots, y_{i_{l-f_i}}\}$. Let $B_i = \{x_1, \ldots, x_l\}$ without loss of generality. For $x_j \in B_i$, observe that the following proposition is consistent: F **and** x_j is true iff either x_j is forced to be true or if x_j is not forced, then x_j is the j'-th unforced variable $y_{i_{j'}}$ in Y_i and $y_{i_{j'}}$ is true. The proposition "x_i is forced to be true" can be expressed by $F \wedge G'_{i,x_j}$ where G'_{i,x_j} is the disjunction of terms with each term corresponding to a clause in which x_j appears positively and all other variables in it belong to A. Each term is precisely the conjunction of the negations of all the other variables in the such a clause. 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 $x_1, x_2, \ldots, x_{j-1}$ in B_i that occur before x_j and check how many of them are forced. Let $\beta_j = \beta(G_{i,x_1}, \ldots, G_{i,x_{j-1}}, y_{i_1}, \ldots, y_{i_{j-1}})$ 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 condition $G'_{i,x_j} \vee (\bar{G}_{i,x_j} \wedge \beta_j)$. Hence, we can express our *consistency* statement as $F \wedge (x_j \equiv \Psi_{i,x_j})$. Ψ_{i,x_j} only 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 k'-CNF in the variables in Y.

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

Since $|B| \leq n$, 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 $\log(l+1)/l \leq \epsilon$, we have Γ as the disjunction of at most $2^{\epsilon n}$ k'-CNF 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.

Combining all the ideas thus far, we obtain the following theorem.

Theorem 1 *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 $2^{2\epsilon n}$ k'-CNF on at most $n(1-1/(ek))$ variables. Moreover, $\hat{F}$ can be computed from F in time $O(poly(n)2^{2\epsilon n})$.*

General k-SAT

Once again, consider a satisfiable k-CNF where each variable appears at most c times. Let $\delta > 0$ (to be selected later). We will consider all inputs $\vec{x} = (x_1, \ldots, x_n)$ with at most δn 1's. If one such $\vec{x}$ is a satisfying solution, we can find a satisfying solution in time $\text{poly}(n)2^{h(\delta)n}$ where $h(\delta)$ is the binary entropy function. In the other case, we are guaranteed that there exists a satisfying solution of F which is isolated with respect to at least δn variables. Since we have a sufficiently isolated solution, we will do a similar analysis as in the unique k-SAT case with $\log(l+1)/l \leq \epsilon$ to conclude

Theorem 2 *Let F be a k-CNF. The satisfiability of F is equivalent to the satisfiability of $\hat{F}$ where $\hat{F}$ is a disjunction of at most $2^{2\epsilon n}$ k'-CNF on at most $n(1-\delta/(ek))$ variables. Moreover, $\hat{F}$ can be computed from F in time $O(poly(n)2^{(2\epsilon+h(\delta))n})$.*

By selecting δ so that $h(\delta) \leq s_\infty/2$, we get the following corollary (assuming **ETH**).

Corollary 1 $s_k \leq s_\infty(1 - d/(ek))$ *where* $d \approx s_\infty/(2\log(2/s_\infty))$.

Open Problems

Efficiently reduce k-CNF to k'-CNF so that the clause width is reduced ($k' < k$) by increasing the number of variables.

Obtain similar relationships for various k-colorability problems.

References

[1] Alon, N., Spencer, J., and Erdös, P., (1992), "The Probabilistic Method", John Wiley & Sons, Inc.

[2] E. A. Hirsch, Two New Upper Bounds for SAT, SIAM conference on Discrete Algorithms, 1997.

[3] R. Impagliazzo, R. Paturi and F. Zane, Which Problems have Strongly Exponential Complexity?, *1998 Annual IEEE Symposium on Foundations of Computer Science*.

[4] O. Kullmann and H. Luckhardt (1997), Deciding Propositional Tautologies: Algorithms and their Complexity, *submitted to Information and Computation*.

[5] Monien, B. and Speckenmeyer, E., (1985), Solving Satisfiability In Less Than 2^n Steps, Discrete Applied Mathematics **10**, pp. 287–295.

[6] Paturi, R., Pudlák, P., and Zane, F., (1997), Statisfiability Coding Lemma, *in* Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, October 1997.

[7] Paturi, R., Pudlák, P., Saks, M., and Zane F., (1998), An Improved Exponential-time Algorithm for k-SAT, *1998 Annual IEEE Symposium on Foundations of Computer Science*.

[8] Schiermeyer, I. (1993), Solving 3-Satisfiability in less than 1.579^n Steps, *in* Selected papers from CSL '92, LNCS Vol. 702, pp. 379-394.

[9] Schiermeyer, I. (1996), Pure Literal Look Ahead: An $O(1.497^n)$ 3-Satisfiability Algorithm, Workshop on the Satisfiability Problem, Technical Report, Siena, April 29 - May 3, 1996; University of Köln, Report No. 96-230.

[10] Schöning, U. (1999), An $O((2k/(k+1))^n.\mathrm{poly}(n))$ algorithm for k-satisfiability, manuscript