Which Problems Have Strongly Exponential Complexity?

Russell Impagliazzo*, Ramamohan Paturi*
University of California, San Diego
Francis Zane
Bell Laboratories, Lucent Technologies

Abstract

For several **NP***-complete problems, there have been a progression of better but still exponential algorithms. In this paper, we address the relative likelihood of sub-exponential algorithms for these problems. We introduce a generalized reduction which we call Sub-Exponential Reduction Family (SERF) that preserves sub-exponential complexity. We show that Circuit-SAT is SERF-complete for all* **NP***-search problems, and that for any fixed k, k-SAT, k-Colorability, k-Set Cover, Independent Set, Clique, Vertex Cover, are SERF–complete for the class* **SNP** *of search problems expressible by second order existential formulas whose first order part is universal. In particular, sub-exponential complexity for any one of the above problems implies the same for all others.*

We also look at the issue of proving strongly exponential lower bounds for $\mathbf{AC}^0$*; that is, bounds of the form $2^{\Omega(n)}$. This problem is even open for depth-3 circuits. In fact, such a bound for depth-3 circuits with even limited (at most n^ϵ) fan-in for bottom-level gates would imply a nonlinear size lower bound for logarithmic depth circuits. We show that with high probability even degree 2 random GF(2) polynomials require strongly exponential size for Σ_3^k circuits for $k = o(\log\log n)$. We thus exhibit a much smaller space of $2^{O(n^2)}$ functions such that almost every function in this class requires strongly exponential size Σ_3^k circuits. As a corollary, we derive a pseudorandom generator (requiring $O(n^2)$ bits of advice) that maps n bits into a larger number of bits so that computing parity on the range is hard for Σ_3^k circuits.*

Our main technical lemma is an algorithm that, for any fixed $\epsilon > 0$, represents an arbitrary k-CNF formula as a disjunction of $2^{\epsilon n}$ k-CNF formulas that are sparse*, that is, each has $O(n)$ clauses.*

1 Introduction

In two areas of complexity, **NP**-completeness and lower bounds for constant depth circuits, the complexity of specific, natural problems have been shown to be weakly exponential ($2^{n^{\Omega(1)}}$), either absolutely or under a reasonable complexity assumption. However, the complexity of many such problems is believed to be strongly exponential, $2^{\Omega(n)}$. In this paper, we make progress towards closing the gap between weakly and strongly exponential bounds.

Surprisingly, the same technical lemma is useful in both situations. The *Sparsification Lemma* implies that an arbitrary k-CNF formula can be written as a (relatively small) disjunction of sparse k-CNF formulas, More precisely, the Sparsification Lemma shows that 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, for some function c. Moreover, this representation can be computed in $O(\text{poly}(n)2^{\epsilon n})$ time. We apply the Sparsification Lemma to show that many natural problems are *complete* for a large sub-class of NP under reductions preserving strongly exponential complexity. We also apply the Sparsification Lemma to construct sparse distributions on instances for the parity function which are almost as hard for depth-3 circuits as arbitrary instances. In the rest of the section, we elaborate on these two applications.

1.1 Reductions Preserving Strongly Exponential Complexity

Motivation: The theory of **NP**-completeness identifies a natural class of natural combinatorial problems that are all equivalent with respect to polynomial-time solvability. However, some **NP**-complete problems have "better" worst-case algorithms

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

than others. More recently, a series of papers have obtained improved algorithms for Formula-SAT, CNF-SAT, and k-SAT [20, 27, 28, 18, 19, 21, 22, 29]. Similar sequences exist for other problems, such as graph 3-colorability [7, 11]. Is there any complexity-theoretic reason to believe that these improvements could not extend to arbitrarily small constants in the exponent? Is progress on the different problems connected? Can we classify **NP**-complete problems by their "likely complexity" to see how close we are to the best algorithms?

We give formalizations and some preliminary answers to these questions.

1.1.1 Related Work

Our formalization differs from the previous approach to exact time complexity by Hunt and Stearns ([31]) in that we look at the complexity of solving NP problems in terms of the lengths of *solutions* rather than of *instances*. We feel this is more natural, since the obvious exhaustive search algorithm is strongly exponential in the solution size, but may be only weakly exponential in the instance size. Also, for many of these problems, instance size is representation dependent, but solution size is robust. Another difference is that we consider mainly whether the complexity is $2^{\Omega(n)}$ or $2^{o(n)}$. They instead considered the *power index*, which is defined to be the infimum of all x for which the problem is in $DTIME(2^{n^x})$.

Despite these differences, the results in [31] are interesting both for what they say and the limitations of the techniques used. It is hypothesized in [31] that the *Satisfiability* problem has power index 1 and as a consequence of the hypothesis, it is shown that Clique and Partition problems have power index $1/2$ (in terms of the problem size in matrix representation.) They get their results mainly by analyzing how reductions change the input lengths. The less a reduction blows up the input size, the tighter the connection between the exact complexities of the problems. To show stronger connections between problems, we will need to introduce a new notion of reduction, that uses a small exponential amount of time, but only makes queries to instances that are linearly related in size to the input.

Another related result was obtained via fixed parameter complexity. It is shown that the tractability of the class of fixed parameter problems is equivalent to the existence of subexponential-time algorithm for *Circuit Satisfiability* [1].

1.1.2 Complexity parameters

As mentioned before, we feel that *solution size* is the most relevant parameter when discussing exact complexities of NP problems. There are, however, a number of other possible parameters, and relating results which use different parameters can be difficult. For 3-SAT, natural choices could be n, the number of variables, m, the number of clauses, or $\Theta((n+m)\log n)$, the total length of its binary description. Our results also relate the likelihood of being sub-exponential in some of the other parameters in the literature. For example, it follows from our results that k-SAT has a $2^{o(n)}$ algorithm if and only if it has a $2^{o(m)}$ algorithm. To discuss such results, we define a notion of a general complexity parameter, which need not be the instance or solution size. However, we will insist that the trivial search algorithm be bounded by an exponential in this complexity parameter.

More precisely, a complexity parameter $m(x)$ for $L \in NP$ must be polynomial time computable and polynomially bounded in $|x|$. Also, there must be a polynomial time relation R called the *constraint*, so that the search version of L can be written in the form "Given x, find a y so that $|y| \leq m(x)$ and $R(x, y)$ (if such a y exists)". We define **SE** (sub-exponentially solvable search problems) as the class of those problems $L \in NP$ and complexity parameters m that can be solved in time poly$(|x|)2^{\epsilon m(x)}$ for every fixed $\epsilon > 0$.

We view the relevant complexity parameter as being included in the problem specification. If we do not explicitly state which parameter we are discussing, then we mean the length of the (naive) solution. For example, if a result applies to k-SAT, then unless we give another parameter, we mean k-SAT with parameter the number of input variables. Confusion might arise because some problems also have parameters which are more definitional than they are measures of complexity. For example, the parameter k in k-SAT belongs to this category. We view these parameters as being fixed in the problem definition, not as complexity parameters of the instance.

1.1.3 Our results

All of the algorithms referred to earlier, while (for at least some relationships between complexity parameters) much faster than exhaustive search, are of the form 2^{cp} for $c > 0$ a constant and p the relevant parameter. Later papers in the sequence get better constants c. We ask "for which problems and choices of parameters is there a non-trivial (> 0) limit to this sequence of improvements"? To resolve this question completely, one would either need to give a sub-exponential algorithm or prove strong lower bounds. However, we show that the answer to this question is the same for many commonly studied **NP**-complete problems and for all the commonly studied choices of parameters. To be more precise, we show that problems such as k-SAT and k-colorability are complete for the class **SNP** [24] under a notion of reduction that preserves subexponential complexity.

This extends the theory of **NP**-completeness to give a stratification of problems based on their likely complexity. Other recent work has divided **NP**-complete problems by their extent of approximability. The extension of **NP**-completeness theory to a wide variety of approximation problems (See, e.g., [24, 10, 5, 15, 16] and see [6] for a survey) has allowed researchers to compute sharp bounds on how well one should be able to approximate solutions to NP-complete problems. This work has spurred the discovery of new approximation algorithms for some problems [4]. While not claiming the same depth or significance for our work, we hope that this paper will further distinguish "easy" from "hard" problems within the ranks of **NP**-complete problems, and lead to a similar systematic study of the exact complexity of **NP**-complete problems.

1.1.4 Reductions that preserve sub-exponential complexity

Since the standard reductions between NP-problems increase the problem instance size, the question of whether the reductions preserve sub-exponential complexity is delicate and depends on the choice of parameters. For example, an instance of 3-SAT with n variables and m clauses maps to a graph with $O(n+m)$ nodes and edges in the standard reduction to independent set. So this reduction would be sub-exponential time preserving if we measure in terms of edges and clauses, but not in the more natural measure of variables and nodes.

SERF Reducibility: Note that for a reduction to preserve sub-exponential time, it is vital that the relevant parameters not increase more than linearly, but the time complexity of the reduction is less important. Let A_1 be a problem with complexity parameter m_1 and constraint R_1 and A_2 be a problem with complexity parameter m_2 and constraint R_2. For a many-one reduction f from A_1 to A_2 to preserve sub-exponential complexity, we would want $m_2(f(x)) \in O(m_1(x))$. We will call such a reduction a *strong many-one reduction*. Many of the standard reductions between **NP**-complete problems are strong, at least for some choices of parameters, and it is almost always a trivial review to verify that a reduction is strong. For example, most of the reductions from k-SAT are strong if we use the number of clauses as our parameter, but not if we use the number of variables.

To get a more complete web of reductions between natural problems and the most natural parameters, we will need to consider significantly more complicated forms of reduction. First, we will need Turing reductions rather than many-one reductions. Secondly, we will trade off sparseness for time: we note that it is vital to keep the condition that the complexity of the instances of A_2 be linear in the complexity of the instances of A_1, but not that the reduction is polynomial-time.

We define a sub-exponential reduction family SERF as a collection of Turing reductions $M_\epsilon^{A_2}$ from A_1 to A_2 for each $\epsilon > 0$, so that we have for each $\epsilon > 0$:

- $M_\epsilon^{A_2}(x)$ runs in time at most poly$(|x|)2^{\epsilon m_1(x)}$
- If $M_\epsilon^{A_2}(x)$ queries A_2 with the input x', then $m_2(x') \in O(m_1(x))$ and $|x'| = |x|^{O(1)}$.

If such a reduction family exists, we say A_1 is SERF-reducible to A_2. Strong many-one reducibility is a special case of SERF-reducibility. SERF-reducibility is transitive, and, if A_1 is SERF-reducible to A_2 and if $A_2 \in$ **SE** then $A_1 \in$ **SE** . We note that the definition of SERF reducibility can be simplified if the length of the input is polynomially bounded in the complexity parameter.

Let A be a search problem with $R(x,y)$ as the relation and $m(x)$ as the complexity parameter. If $m'(x) \geq m(x)$, the search problem with complexity parameter m' and relation $R'(x,y) \iff R(x,y) \wedge |y| \leq m(x)$ is an alternate formulation of A. So we can talk about the same problem with different complexity parameters. If $m'(x) > m(x)$, then (A, m') is always strong many-one reducible to (A, m) by the identity function, but we may or may not also have a SERF-reduction in the opposite direction.

Some standard reductions are seen to be strongly-preserving for solution size, even if they are not for instance size. For example, it follows easily from the definition of complexity measure and strong reductions that:

Proposition 1 *Circuit-SAT with the number of variables as the complexity measure is complete for* **NP***-search problems under strong many-one reductions.*

Proof: Let L be a language in NP and m a complexity parameter for L. Then, by the definition of complexity parameter, we can find a polynomial-time computable relation $R(x,y)$ so that $x \in L$ iff there is a $y, |y| \leq m(x)$ and so that $R(x,y)$. Then the (standard) reduction on input z computes a circuit $C(x,y)$ that simulates $R(x,y)$ on inputs of size n and $m(n)$, and then sets $x = z$. The number of inputs for the resulting Circuit-SAT instance is $m(x)$, so the reduction is strong. ■

Our main technical contribution is that k-SAT with the number of variables as the complexity parameter is SERF-reducible to k-SAT with the number of clauses as the complexity parameter. It then easily follows by standard strong many-one reductions that problems like Independent Set, k-Colorability, k-Set Cover, and Vertex Cover are all SERF-equivalent under several natural complexity parameters.

SERF-Completeness for a logically defined class: We can make the complexity implications of sub-exponential algorithms for the above problems more precise using descriptive complexity, which specifies complexity classes in terms of the quantifier structure in the problem definition.

In descriptive complexity, an input to a problem is viewed as a finite set of relations on a universe of n elements. Then the problem is whether a given logical formula is true or false in the input model. Second order quantifiers quantify over relations, first order quantifiers over elements of the universe.

Define **SNP** to be 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 class is considered by Papadimitriou and Yannakakis [24] for studying approximability of optimization problems. The associated search problem looks for instantiations of the quantified relations for which the resulting first-order formula is true. If the quantified relations are $R_1, \ldots, R_q$ and R_i has arity α_i, the complexity parameter for the formula is $\sum_{i=1}^{q} n^{\alpha_i}$, the number of bits to describe all relations. (In many cases of interest, the relations in question are all monadic, i.e., take a single input, in which case the complexity parameter is just $O(n)$.)

To see that k-SAT $\in$ **SNP** , view an input as 2^k k-ary relations, $R_{s_1,\ldots,s_k}$ where $s_i \in \{+,-\}$. $R_{s_1,\ldots,s_k}(y_1,\ldots,y_k)$ is true if the clause $\vee_{i=1}^{k} y_i^{s_i}$ is present among the input clauses where $y_i^+ = y_i$ and $y_i^- = \overline{y}_i$. Define $I_+(S,y)$ as the formula $S(y)$ and $I_-(S,y)$ as the formula $\neg S(y)$. Then we can express satisfiability of the formula by:

$$\exists S[\forall(y_1,\ldots,y_k)\forall(s_1,\ldots,s_k)[R_{s_1,\ldots,s_k}(y_1,\ldots,y_k) \rightarrow \bigvee_{1\le i\le k} I_{s_i}(S,y)],$$

where S is a unary predicate (interpreted as "the variable y is set to true.").

We show that k-SAT and k-Colorability are SERF-complete for **SNP** . Problems like Independent Set and k-Set Cover are SERF-equivalent to these, but are technically not in **SNP** because their statements involve minimization.

1.2 A Sparse Distribution for Parity which is Hard for Depth 3 Circuits

Considerable progress has been made in understanding the limitations of unbounded fan-in Boolean circuits of bounded depth. The results of Ajtái, Furst, Saxe, Sipser, Yao, Håstad, Razborov, and Smolensky [2, 12, 36, 14, 25, 30], among others, show that if the size of the circuit is not too large, then any function computed by such circuit must be constant on a large subcube or can be approximated by a small degree polynomial. Such limitations of small size bounded depth circuits can be used to show that certain explicit functions such as parity and majority require a large number of gates. More precisely, a result of Håstad [14] says that computing the parity function in depth d requires $\Omega(2^{\varepsilon n^{1/(d-1)}})$ gates for some $\varepsilon < 1$. More recently, Paturi, Pudlák and Zane obtained the tight bound $\Theta(n^{1/4}2^{\sqrt{n}})$ for computing parity on depth-3 circuits. However, these and other techniques seem incapable of proving a lower bound on depth-3 circuits of $2^{\omega(\sqrt{n})}$ for any explicit Boolean function in **NP**.

To clarify the situation, it is useful to parameterize the lower bound in terms of the maximum fan-in of the bottom gates. Define Σ_d^k to be the set of depth d circuits with top gate OR such that each bottom gate has fan-in at most k. Then it follows from [21] that any Σ_3^k circuit for the parity function or the majority function requires $\Omega(2^{n/k})$ gates at level 2, and such bounds are tight for $k = O(\sqrt{n})$. For the special case $k = 2$, [23] proved strong exponential lower bounds of the form $\Omega(2^{n-o(n)})$ for recognizing the codewords of a good error-correcting code. They show that any 2-CNF accepting a large number of inputs must include a certain generalized subcube of a large dimension and then diagonalize over such objects. However, it does not seem that similar techniques will extend to the case $k \ge 3$.

We would like to prove strongly exponential lower bounds on depth–3 circuits that go beyond the above trade-off between bottom fan-in and size. The best such trade-off is from [22], proving a lower bounds of the form $\Omega(2^{\delta_k n/k})$ with $\delta_k > 1$ for recognizing the codewords of a good error-correcting code. (δ_k approaches 1.644 for large k). It is still open to prove a $2^{n/2}$ lower bound for Σ_3^k circuits even when k is bounded. One compelling motivation for studying the depth–3 model with limited fan-in is that Valiant [33] showed that linear–size logarithmic–depth Boolean circuits with bounded fan–in can be computed by depth–3 unbounded fan-in circuits of size $O(2^{n/\log\log n})$ and bottom fan-in limited by n^{ε} for arbitrarily small ε. If we consider linear–size logarithmic–depth circuits with the additional restriction that the graph of the connections is series–parallel, then such circuits can be computed by depth–3 unbounded fan-in circuits of size $2^{n/2}$ with bounded bottom fan–in. Thus, strong exponential lower bounds on depth-3 circuits would imply nonlinear lower bounds on the size of fan-in 2 Boolean circuits with logarithmic–depth, an open problem proposed some twenty years ago [33]. For the depth–3 circuits constructed from such series-parallel circuits, we could obtain similar results using simpler arguments, because the construction ensures that the depth–2 subcircuits are sparse. Nonetheless, these methods yield interesting new results as well as providing more general results.

Our approach to proving strongly exponential lower bounds is rooted in the following observation: Computing parity even on much smaller subsets of the input space requires a very large size for Σ_3^k circuits. More precisely, if A is a set of random input instances of size $2^{O(n/k)}$, then with high probability computing parity on A requires $2^{\Omega(n/k)}$ gates for any Σ_3^k circuit. Our approach is to try to find an explicit subset where this is true. We would like to find a "pseudo-randomly" generated set, i.e., the range of a function that maps n bits into more than n bits such that computing parity on the range requires strongly exponential bounds for Σ_3^k circuits in the parameter n. ¿From this, one can hope to prove the composition of the map with the parity function requires strongly exponential lower bounds.

We give a probabilistic construction of such a function from maps $\{0,1\}^n$ to $\{0,1\}^{O(n^2)}$ bits just using a family of randomly chosen monomials of degree 2 (AND of two variables). Parity composed with such a function can be thought of as a degree 2 polynomial over GF(2). Although the intuition is that computing parity on the range is hard and that therefore the composition is also hard to compute, our proof actually goes in the reverse direction. We directly prove that random degree 2 polynomials require strongly exponential size for Σ_3^k circuits for $k = o(\log\log n)$. We then show as a corollary that parity is hard on the corresponding subset of inputs. Our proof relies on our ability to represent k-CNF as a subexponential union of linear size k-CNFs (Sparsification Lemma) and an algebraic counting technique. Although one can use the algebraic counting technique for showing that almost every degree d GF(2) polynomial requires strongly exponential lower bounds for Σ_3^k circuits as long as $d > k$, one needs the Sparsification Lemma to show that even degree 2 polynomials suffice to obtain strongly exponential lower bounds for Σ_3^k circuits for all $k = o(\log\log n)$.

Furthermore, our result implies that computing parity on the range of the map defined by randomly selected degree 2 monomials is hard for Σ_3^k circuits for $k = o(\log\log n)$. We thus exhibit a pseudorandom generator (which relies on about $O(n^2)$ bits of advice) which amplifies its input bit string so that computing parity on the range is hard for Σ_3^k circuits.

As in earlier papers [21, 22], we observe that *certain characterizations* of k-CNF have applications to lower bounds on depth-3 circuits as well as upper bounds for the satisfiability problem. The key to further constructivization of our arguments is a better understanding of k-CNFs and their solution spaces. Such understanding may have dividends for us in the form of improved worst-case algorithms for the satisfiability problem as well as strongly exponential lower bounds for depth-3 circuits.

2 Sparsification Lemma

Call an instance of k-SAT *sparse* if it has $m = O(n)$ clauses, where n is the number of variables. Our main technical lemma is the following algorithm that reduces a general k-SAT formula to the disjunction of a collection of sparse instances. This gives a SERF-reduction from k-SAT with parameter n to k-SAT with parameter m. It also gives a normal form for depth-3 circuits with bottom fan-in k.

One of the basic techniques for solving search problems is back-tracking. In back-tracking one narrows in on a solution meeting certain constraints through case analysis and deductions. Typically the case analysis for satisfiability involves a single variable. One searches for a solution where that variable is TRUE, and then if that search fails, backs up and searches given the variable is FALSE. To narrow the search, we look for other facts about the solution that are forced by the other constraints. Intuitively, back-tracking is more effective in highly constrained searches where each constraint is simple; for example, in dense instances of k-SAT. Our proof is a formalization of that intuition: we show that back-tracking works well until the formula becomes sparse. We argue that each time we branch while the formula is dense, we can create many simpler constraints, i.e., smaller clauses. Eventually, the process must create many clauses of size 1, i.e., force many variables.

Our back-tracking algorithm sometimes branches on an OR of variables rather than a single variable. This is to avoid the situation where we have too many constraints involving a single variable. Although such a variable is in itself good, because we get much information by setting it, the creation of those constraints might have required more branching increasing the number of leaves in the tree. If it required several branches to obtain many clauses of the form $x \vee y_i$, the new clauses tell us strictly less than if we had branched on x being true immediately. By trying to simplify small clauses first, and by branching on OR's of variables as soon as they appear in many clauses, we can avoid excessive branching which creates redundant constraints. (If the problem originally involved redundant constraints, that does not hurt us.)

It is somewhat simpler and general to state and prove the lemma for k-Set Cover rather than k-SAT, and derive the version for k-SAT as a Corollary. k-Set Cover has as an instance a universe of n elements $x_1, \dots, x_n$ and a collection $\mathcal{S}$ of subsets $S \subseteq \{x_1, \dots, x_n\}$ with $|S| \leq k$ for each $S \in \mathcal{S}$. A *set cover* for $\mathcal{S}$ is a set $C \subseteq \{x_1, \dots, x_n\}$ so that $C \cap S \neq \emptyset$ for each $S \in \mathcal{S}$. Let $\sigma(\mathcal{S})$ be the collection of all set covers of $\mathcal{S}$. The k-Set Cover problem is to find a minimal size set cover.

$\mathcal{T}$ is a *restriction* of $\mathcal{S}$ if for each $S \in \mathcal{S}$ there is a $T \in \mathcal{T}$ with $T \subseteq S$. Obviously, if $\mathcal{T}$ is a restriction of $\mathcal{S}$, then $\sigma(\mathcal{T}) \subseteq \sigma(\mathcal{S})$.

We can represent k-SAT as a k-Set Cover problem in a canonical way. The universe is the set of $2n$ literals. We first define $\mathcal{U}$ to be the family of sets $\{x_i, \overline{x}_i\}$. It is obvious that the coverings of $\mathcal{U}$ of size n are exactly all the input instances. Then we add, for each clause, the set of its literals. $\mathcal{S}$ is the collection of all such sets. Clearly any cover of $\mathcal{S}$ of size n is a satisfying solution of the underlying formula. Conversely, any family of sets over the universe of $2n$ literals can be interpreted as a formula given by a conjunction of disjunctions, and moreover if the family includes all the sets in the family $\mathcal{U}$, then any cover of size n is a satisfying instance of the formula.

When we view k-SAT as a special case of k-Set Cover, the underlying sets are sets of literals. Thus, we never explicitly have to require that exactly one of the literals $\{x_i, \overline{x}_i\}$ has to be in the cover. This makes the analysis easier although it seems somewhat artificial for k-SAT.

Call a formula Φ a restriction of Ψ if each clause of Ψ contains some clause of Φ. Then a restriction of the k-Set Cover instance corresponds to a restriction of the k-SAT formula.

Theorem 1 *For all $\epsilon > 0$ and positive k, there is a constant C and an algorithm that, given an instance $\mathcal{S}$ of k-Set Cover on a universe of size n, produces a list of $t \leq 2^{\epsilon n}$ restrictions $\mathcal{T}_1, \ldots, \mathcal{T}_t$ of $\mathcal{S}$ so that $\sigma(\mathcal{S}) = \cup_{i=1}^{t} \sigma(\mathcal{T}_i)$ and so that for each $\mathcal{T}_i$, $|\mathcal{T}_i| \leq Cn$. Furthermore, the algorithm runs in time poly$(n) 2^{\epsilon n}$.*

Proof: Choose integer α so that $\alpha / \log(4\alpha) > 4k2^k\epsilon^{-1}$. Let $\theta_0 = 2$, $\beta_i = (4\alpha)^{2^{i-1}-1}$ and $\theta_i = \alpha\beta_i$ for $i = 1, \ldots, k-1$. We say that a collection, $S_1, \ldots, S_c$, of sets of size j is a *weak sunflower* if $H = \cap_{i=1}^{c} S_i \neq \emptyset$. We call H the *heart* of the sunflower, and the collection $S_1 - H, \ldots, S_c - H$ the *petals*. Each petal has the same size, $j - |H|$, called the *petal size*. The size of the sunflower is the number of sets in the collection. By a j-set, we mean a set of size j. For a family of sets $\mathcal{S}$, let $\pi(\mathcal{S}) \subseteq \mathcal{S}$ be the family of all sets S in $\mathcal{S}$ such that no other set S' in $\mathcal{S}$ is a proper subset of S. Thus $\pi(\mathcal{S})$ is an antichain.

We start with the collection $\mathcal{S}$ and apply the following algorithm, Reduce, which outputs the families $\mathcal{T}_i$.

```
Reduce($\mathcal{S}$:a collection of sets of size at most $k$)

1. $\mathcal{S} \leftarrow \pi(\mathcal{S})$
2. FOR $j$=2 TO $k$ DO
3.     FOR $i = 1$ to $j-1$ DO
4.          IF there is a weak sunflower of $j$-sets and petal size $i$ where the number of petals is at least $\theta_i$, that is, if there are $S_1, \ldots, S_c \in \mathcal{S}$, $|S_d| = j$ for $d = 1, \ldots, c$, and with $|H| = j - i$ where $H = \bigcap_{d=1}^{c} S_d$,
5.          THEN Reduce($\mathcal{S} \cup \{H\}$); Reduce($\mathcal{S} \cup \{S_1 - H, \ldots, S_c - H\}$); HALT.
6. Return $\mathcal{S}$; HALT.
```

As usual, we can think of the execution of a recursive procedure as described by a tree. Each node is naturally associated with a family $\mathcal{S}'$ of sets. The π operator is applied to $\mathcal{S}'$ resulting in the elimination of any supersets. If $\pi(\mathcal{S}')$ contains a sunflower, then the node branches into two children. The family associated with one child is created by the adding the petals of the sunflower to $\mathcal{S}'$ and the other family is created by adding the heart to $\mathcal{S}'$. Otherwise, the node is a leaf. We consider the collection $\mathcal{S}$ as it changes along a path from the root to a leaf. New sets can be *added* to the collection $\mathcal{S}$ at a node due to a sunflower. Existing sets can be *eliminated* from $\mathcal{S}$ by the π operator. Note that a set S' can only be eliminated if some proper subset S is in $\mathcal{S}$; we then say that S eliminated S'.

We first observe some basic properties of the algorithm Reduce.

Consider a collection $\mathcal{T}$ associated with a node other than the root and the collection $\mathcal{T}'$ associated with its parent node. If the set $S \in \mathcal{T}$ eliminated the set $S' \in \mathcal{T}$, then S must be either the heart or a petal of the sunflower in $\mathcal{T}'$ that caused the branching. After the π operator is applied to $\mathcal{T}$, no proper superset of S would ever be present in any of the families associated with the descendants of $\mathcal{T}$. All the sets eliminated by a set S are eliminated in the next application of the π operator immediately after the set S is added to the collection.

The next observation states that T is a covering of $\mathcal{S}$ if and only if it is a covering of one of the families output by the algorithm.

Lemma 1 *Reduce($\mathcal{S}$) returns a set of restrictions of $\mathcal{S}$, $\mathcal{T}_1, \ldots, \mathcal{T}_t$, with $\sigma(\mathcal{S}) = \bigcup_{l=1}^{t} \sigma(\mathcal{T}_l)$.*

Proof: Clearly, if $S \subset S'$, then covers $\sigma(\mathcal{S}) = \sigma(\mathcal{S} - \{S'\})$. If $S_1, \ldots, S_c \in \mathcal{S}$ form a weak sunflower with heart H, and T is a cover of $\mathcal{S}$, then either $T \cap H \neq \emptyset$ or $T \cap (S_l - H) \neq \emptyset$ for each $1 \leq l \leq c$. So $\sigma(\mathcal{S}) = \sigma(\mathcal{S} \cup \{H\}) \cup \sigma(\mathcal{S} \cup \{S_l - H | 1 \leq l \leq c\})$. The lemma then follows by a simple induction on the depth of recursion. ■

Lemma 2 *For $1 \leq i \leq k$, every family $\mathcal{T}$ output by Reduce($\mathcal{S}$) has no more than $(\theta_{i-1} - 1)n$ sets of size i.*

Proof: If there were more than $(\theta_{i-1} - 1)n$ sets of size i in $\mathcal{T}$, then there would be an element x in at least θ_{i-1} of them by the pigeonhole principle. The sets containing x would form a weak sunflower with petal size at most $i-1$, so the program would branch rather than halt. ■

As a consequence, it follows that every element of Reduce($\mathcal{S}$) has no more than Cn sets where C is set to be $\sum_{i=1}^{k}(\theta_{i-1} - 1)$. Thus, the sparseness property is satisfied.

The most difficult part is to show $t \leq 2^{\epsilon n}$, where t is the number of families output, which is the same as the number of leaves in the recursion tree. We do this by showing that the maximum length of any path in the tree is $O(n)$, whereas the maximum number of nodes created by adding petals rather than the heart is at most kn/α. To prove this, for $1 \leq i \leq k$, we show that the following invariant, $J_i(\mathcal{T})$, holds for any family $\mathcal{T}$ in any path as long as a set of size i has been added to one of the predecessor families in the path. $J_i(\mathcal{T})$ is the condition that no element x appears in more than $2\theta_{i-1} - 1$ sets of size i in $\mathcal{T}$. As a consequence of the invariant, we show that any set added to such a family eliminates at most $2\theta_{i-1} - 1$ sets of size i during the next application of the operator π. Since we can have at most n sets of size one in any family, we show by induction that the total number of sets of size at most i that are ever added to all the families along a path is at most $\beta_i n$. Since at least θ_i petals are added to any family created by adding petals of size i, it follows that at most kn/α families are created due to petals. We now prove our assertions.

Lemma 3 *For any family $\mathcal{T}$ such that a set of size i has been added to it or any of its predecessors in the path, the invariant $J_i(\mathcal{T})$ holds: no element x appears in more than $2\theta_{i-1}$ sets of size i in $\mathcal{T}$.*

Proof: It suffices to prove the lemma for the families created by adding sets of size i. Let $\mathcal{T}$ be such a family obtained by adding some sets of size i to the family $\mathcal{T}'$. Consider the sunflower in the family $\mathcal{T}'$ that created these sets. The sunflower must be a family of sets of size $j > i$. Furthermore there is no sunflower of i-sets in $\mathcal{T}'$ with θ_{i-1} petals of petal size $i-1$ so we can conclude that no element x appears in more than $\theta_{i-1} - 1$ sets of size i in $\mathcal{T}'$. Either $\mathcal{T}$ is created by adding the petals of the sunflower or the heart. If the petals are added to $\mathcal{T}$, no element x is contained in more than $\theta_{i-1} - 1$ petals since otherwise $\mathcal{T}'$ would contain a sunflower of j-sets with θ_{i-1} petals of size $i-1$. Thus, no element x is contained in more than $2\theta_{i-1} - 1$ i-sets of the family $\mathcal{T}$. If the heart is added to $\mathcal{T}$, only one set of size i is added to the family $\mathcal{T}$ and since $\theta_{i-1} \geq 1$, the conclusion holds as well in this case. ■

Lemma 4 *At most $2\theta_{i-1}$ of the added sets of size i can be eliminated by a single set.*

Proof: If a set S of size i has been added to a family $\mathcal{T}$ associated with a node, then after an application of π, $\mathcal{T}$ and all its successor families in the tree, would satisfy the invariant J_i by Lemma 3. Hence, for such families, no element x appears in more than $2\theta_{i-1} - 1$ sets of size i. Let S be a set added to a family. Since all the sets S eliminates are eliminated during the next application of π, we conclude that S can eliminate at most $2\theta_{i-1} - 1$ sets of size i. ■

Lemma 5 *The total number of sets of size $\leq i$ that are added to the families along any path is at most $\beta_i n$.*

Proof: By induction on i. First, for $i = 1$, each added set is a new element of the universe, so there are at most $\beta_1 n = n$ such sets. Assume at most $\beta_{i-1}n$ sets of size $\leq i-1$ are added to the families along a path. Each added set of size i is either eliminated subsequently by a set of size at most $i-1$ or is in the final collection of sets output by the algorithm. By Lemma 2, there are at most $\theta_{i-1}n$ sets of size i in the final collection. By the induction hypothesis and Lemma 4, each of at most $\beta_{i-1}n$ of the added smaller sets eliminates at most $2\theta_{i-1}$ sets of size i. Thus, the total number of added sets of size $\leq i$ is at most $2\beta_{i-1}\theta_{i-1}n + \theta_{i-1}n \leq 4\beta_{i-1}\theta_{i-1}n = (4\alpha\beta_{i-1}^2)n = \beta_i n$. ■

Lemma 6 *Along any path, at most kn/α families are created by adding petals.*

Proof: There are at most $\beta_i n$ sets of size i that are ever added to all the families along any path. If a family is created by adding petals of size i, then at least θ_i such petals are added. Thus, there are at most $\beta_i n/\theta_i = n/\alpha$ such families. Summing over all sizes i gives the claimed bound. ■

Lemma 7 *The algorithm outputs at most $2^{\epsilon n}$ families.*

Proof: It is sufficient to upper bound the number of paths from the root to a leaf. A path is completely specified by the sequence of decisions taken at each of the nodes along the path. Each decision corresponds to whether the heart or the petals are added to the family. Since a set of size at most $k-1$ is added to the family at every step, there are a total of at most $\beta_{k-1}n$ sets ever added to the families along the path by Lemma 5. At most nk/α of these families are created by adding petals, by Lemma 6. Thus, there are at most $\sum_{r=0}^{\lfloor nk/\alpha \rfloor} \binom{\beta_{k-1}n}{r}$ paths. Using the approximation $\binom{\ell}{\delta\ell} \leq 2^{h(\delta)\ell}$ where $h(\delta) = -\delta\log_2\delta - (1-\delta)\log_2(1-\delta)$ is the binary entropy function, we see that this is at most $2^{\beta_{k-1}nh(k/(\alpha\beta_{k-1}))} \leq 2^{((2k/\alpha)\log(\alpha\beta_{k-1}/k))n} \leq 2^{((2k/\alpha)\log(\alpha(4\alpha)^{2^{k-2}-1}))n} \leq 2^{2k2^{k-2}(\log(4\alpha)/\alpha)n} \leq 2^{\epsilon n}$, the first inequality follows from $h(k/(\alpha\beta_{k-1}) \leq 2k/(\alpha\beta_{k-1})\log(k/(\alpha\beta_{k-1}))$ since $k/(\alpha\beta_{k-1}) \leq 1/2$ and the last inequality by our choice of α. ■

The Theorem then follows from Lemmas 1, 2 (setting $C = \sum_{i=1}^{k-1}\theta_i$), and 7, and the observation that the algorithm Reduce takes polynomial time per path. ■

Using the formulation of k-SAT as a Set Cover problem, we then obtain:

Corollary 1 *For all $\epsilon > 0$ and positive k, there is a constant C so that any k-SAT formula Φ with n variables, can be expressed as $\Phi = \bigvee_{i=1}^{t}\Psi_i$, where $t \leq 2^{\epsilon n}$ and each Ψ_i is a k-SAT formula with at most Cn clauses. Moreover, this disjunction can be computed by an algorithm running in time poly$(n)2^{\epsilon n}$.*

The algorithm Reduce gives us a family of reductions from k-SAT to sparse k-SAT:

Corollary 2 *k-SAT with complexity measure n, the number of variables, is SERF-reducible to k-SAT with measure m, the number of clauses.*

3 Complete Problems for SNP

In this section, we combine standard reductions with Corollary 2 to show that many of the standard **NP**-complete problems are equivalent as far as having sub-exponential time complexity. We show that the question of whether these problems are sub-exponentially solvable can be rephrased as whether **SNP** $\subseteq$ **SE** . We also look at complete problems for more general classes. We show that SERF-reductions can be composed and preserve membership in **SE** .

Lemma 8 *If (A_1, m_1) SERF-reduces to (A_2, m_2), and (A_2, m_2) SERF-reduces to (A_3, m_3), then (A_1, m_1) SERF-reduces to (A_3, m_3).*

Proof: Let $\epsilon > 0$. Let $M_{\epsilon/2}^{A_2}$ be the member of the SERF-reduction from (A_1, m_1) to (A_2, m_2). Let C be such that $m_2(x') \leq Cm_1(x)$ for every query x' made by $M_{\epsilon/2}^{A_2}$ on x. Let $N_{\epsilon/(2C)}^{A_3}$ be the member of the SERF-reduction from (A_2, m_2) to (A_3, m_3).

Simulate M^{A_2} except that for each query x' simulate $N_{\epsilon/(2C)}^{A_3}$ on x' . Any query y made to A_3 in the simulation was part of the run of $N_{\epsilon/(2C)}^{A_3}$ on some x' queried by $M_{\epsilon/2}^{A_2}$ on x. Then $m_3(y) = O(m_2(x') = O(m_1(x))$. The total time taken by the simulation is at most the product of the time taken by $M_{\epsilon/2}^{A_2}$ on x and the maximum time taken by $N_{\epsilon/(2C)}^{A_3}$ on some query x', which is at most poly$(|x|)2^{(\epsilon/2)m_1(x)} *$ poly$(|x'|)2^{(\epsilon/2C)m_2(x')} \leq$ poly$(|x|)2^{(\epsilon/2)m_1(x)} *$ poly(poly$(|x|))2^{(\epsilon/2C)Cm_1(x)} \leq$ poly$(|x|)2^{\epsilon m_1(x)}$. ■

Lemma 9 *If (A_1, m_1) SERF-reduces to (A_2, m_2), and $(A_2, m_2) \in$* **SE** *, then $(A_1, m_1) \in$* **SE** .

Proof: Let $\epsilon > 0$. Let $M_{\epsilon/2}^{A_2}$ be the member of the $SERF$ reduction from A_1 to A_2. Let C be such that $m_2(x') \leq Cm_1(x)$ for any query x' that $M_{\epsilon/2}^{A_2}$ makes on x. Let N solve A_2 on x' in time poly$(|x'|)2^{(\epsilon/2C)m_2(x')}$. Then using N to simulate queries in $M_{\epsilon/2}^{A_2}$ solves A_1 in time poly$(|x|)2^{\epsilon/2m_1(x)} *$ poly$(|x'|)2^{(\epsilon/2C)m_2(x')} \leq$ poly$(|x|)2^{\epsilon/2m_1(x)} *$ poly(poly$(|x|))2^{(\epsilon/2C)Cm_1(x)} \leq$ poly$(|x|)2^{\epsilon m_1(x)}$. ■

We can now show that many of the standard NP-complete problems have sub-exponential complexity if and only if the same is true for any problem in **SNP** . First, we show that each problem in **SNP** has a strong many-one reduction to k-SAT for some k. Then, we use Corollary 2 to reduce k-SAT to sparse instances of itself, or more precisely, from treating n as the parameter to treating m as the parameter. Then we show the standard reduction from k-SAT to 3-SAT is strong in m. This shows that 3-SAT, and hence k-SAT for any $k > 3$, is SERF-complete for **SNP** .

Theorem 2 *Let $A \in$* **SNP** *. Then there is a k so that A has a strong many-one reduction to k-SAT with parameter n.*

Proof: Let

$$\exists R_1 \cdots \exists R_q \forall z_1 \cdots \forall z_r \Phi(I, R_1, \ldots, R_q, z_1, \ldots, z_r)$$

be the formula representing A where I is the structure representing the input. Let α_i be the arity of the relation R_i. Let k be the number of occurrences of the relation symbols $R_1, \ldots, R_q$ in Φ. For each α_i-tuple, $\vec{y}$ of elements of the universe, introduce a variable $x_{i,\vec{y}}$ representing whether $R_i(\vec{y})$ is true. Note that we have exactly $\sum_i n^{\alpha_i}$ variables, the complexity parameter for A. For each $\vec{z} \in \{1, \ldots, n\}^r$, we can look at $\Phi(\vec{z})$ as a boolean formula where we replace any occurrence of the relation $R_i(z_{j_1}, \ldots, z_{j_{\alpha_i}})$ with the corresponding Boolean variable. Since each $\Phi(\vec{z})$ depends on at most k variables, we can write each such formula in conjunctive normal form to get an equivalent k-CNF. ■

Next, we can apply Corollary 2 to conclude :

Corollary 3 *Any problem in* **SNP** *SERF-reduces to k-SAT with parameter m for some k.*

The next link is to reduce sparse k-SAT to 3-SAT:

Lemma 10 *For any $k \geq 3$, k-SAT with parameter m strongly reduces to 3-SAT with parameter m (and hence also with parameter n).*

Proof: Let Φ be a k-SAT formula. For every clause $x_1 \ldots, \vee x_j$ with $j \leq k$, introduce $j-1$ new variables $y_1, ..y_{j-1}$ and create the j clauses $x_1 \vee y_1$, $\overline{y_{i-1}} \vee x_i \vee y_i$, for $2 \leq i \leq j-1$ and $\overline{y_{j-1}} \vee x_j$. Thus the number of clauses in the resulting 3-CNF formula will be at most k times that of the original formula. ■

Summarizing:

Theorem 1 *For any $k \geq 3$, k-SAT is* **SNP** *-complete under SERF-reductions, under either clauses or variables as the parameter. Consequently, k-SAT $\in$* **SE** *if and only if* **SNP** $\subseteq$ **SE** .

We can then combine these with known reductions:

Theorem 2 *k-Colorability is* **SNP** *-complete under SERF reductions for any $k \geq 3$*

Proof: k-Colorability is in **SNP** . 3-SAT with parameter m has a strong many-one reduction to 3-colorability. See [9], p. 962, ex. 36-2. 3-colorability has a strong many one reduction to k-colorability for $k \geq 3$. ■

Many natural problems like Independent Set are not obviously in **SNP** . However, they are in a small generalization that is equivalent as far as sub-exponential time. A size-constrained existential quantifier is one of the form $\exists S, |S| \oplus s$, where $|S|$ is the number of inputs where relation S holds, and $\oplus \in \{=, <, >\}$. Define Size-Constrained **SNP** as the class of properties of relations and numbers that are expressible by a series of possibly size-constrained existential second-order quantifiers followed by a universal first-order formula.

Theorem 3 *The following problems are Size-Constrained* **SNP** *Complete under $SERF$ reductions: k-SAT for $k \geq 3$, k-Colorability, Independent Set, Vertex Cover, Clique, and k-Set Cover for $k \geq 2$*

Proof: (Sketch) All of the above are in Size-Constrained **SNP** . Like before, any problem in Size-Constrained strong many-one reduces to k-SAT for some k. The new part is to realize that the size constraints can be computed by a linear sized circuit, and hence using the usual reduction from circuit-SAT to 3-SAT only introduces a linear number of new variables, one per gate. It follows that since any **SNP** -Complete problem is equivalent to k-SAT, that any **SNP** -Complete problem is also Size-Constrained **SNP** Complete.

The usual reduction from 3-SAT with parameter m to Independent Set is strong [9]. We introduce 3 nodes per clause, and connect all nodes from the same clause and all inconsistent nodes. (Before doing this reduction, we can, for each variable that appears more than 8 times, introduce a new variable, setting it to be equal to the old one, and using it in half of the clauses, Then the reduction is also strong in the number of edges.) Independent Set, Vertex Cover and Clique are easily seen to be equivalent under strong reductions (in the number of nodes.)

The reduction from Vertex Cover to 2-Set Cover that simply makes each edge a set is strong. ■

Also, we can find other problems that are **SNP** hard. For example, the usual reduction from 3-SAT with parameter m to Hamiltonian Circuit with parameter $|E|$ is strong [9]. Thus, if we could improve arbitrarily the hidden constant on the $2^{O(n)}$ algorithm for Hamiltonian Circuit, even for sparse graphs, then **SNP** $\subseteq$ **SE** .

4 Lower Bounds for Depth-3 Circuits

In this section, we prove that almost all degree 2 GF(2) polynomials require strongly exponential lower bounds for Σ_3^k circuits for $k = o(\log\log n)$. The main problem is to show that very few degree-2 polynomials can be computed by depth-3 subexponential size circuits of fan-in k.

Theorem 3 *Almost every degree 2 GF(2) polynomial requires $\Omega(2^{n-o(n)})$ size Σ_3^k circuits for $k = o(\log\log n)$.*

As a corollary, we derive a pseudorandom generator (requiring $O(n^2)$ bits of advice) that maps n bits into a larger number of bits so that computing parity on the range is hard for Σ_3^k circuits.

Corollary 4 *There is a pseudorandom map defined by a family of degree 2 monomials (requiring $O(n^2)$ bits of advice) that maps $\{0,1\}^n$ bits into $\{0,1\}^{O(n^2)}$ bits such that computing parity on the range of the map requires $\Omega(2^{n-o(n)})$ size Σ_3^k circuits for $k = o(\log\log n)$.*

Proof: Let $f(x_1, \ldots, x_n)$ be a degree 2 polynomial that requires $\Omega(2^{n-o(n)})$ size Σ_3^k circuits for $k = o(\log\log n)$ as specified in Theorem 3. Order the monomials that occur in f and let $g = (y_1, \cdots, y_m)$ denote the sequence of the monomials where each y_i is the product of two variables x_{i_1} and x_{i_2}. Since the monomials have degree 2, $m = O(n^2)$. Let C be a Σ_3^k circuit of size s with m inputs, $y_1, \ldots, y_m$, that computes parity on the range of the map g. We compute each $y_i = x_{i_1} \wedge x_{i_2}$ and $\overline{y}_i = \overline{x}_{i_1} \vee \overline{x}_{i_2}$ from the inputs $x_1, \ldots, x_n$. The resulting circuit $\mathcal{C}'$with inputs $x_1, \ldots, x_n$ can be represented as a Σ_3^{2k} circuit of size $O(s)$. Since $\mathcal{C}'$ computes the polynomial f, the composition of g with parity function, we have $s = \Omega(2^{n-o(n)})$. ■

We now prepare to prove Theorem 3. Let C be a Σ_3^k circuit of size s computing a Boolean function on n inputs for $k = o(\log\log n)$. Let $0 < \epsilon < 1/2$ be any constant. We want to prove a lower bound on s of the form $2^{(1-2\epsilon)n}$. Define $c(k, \epsilon) = (2/\epsilon)^{O(k2^k)}$. Rewrite each depth-2 subcircuit of C, a k-CNF, as the union of $2^{\epsilon n}$ of k-CNFs with at most $c(k, \epsilon)n$ clauses as permitted by Corollary 1. We thus have an equivalent circuit C (by a slight abuse of notation) with at most $s2^{\epsilon n}$ depth-2 subcircuits.

Assume for the purpose of contradiction that $s < 2^{(1-2\epsilon)n}$. The circuit C computes f, a degree 2 polynomial over GF(2). Unless f is trivial, $|f^{-1}(0)| = O(|f^{-1}(1)|)$. Since the top gate of C is an OR, it follows that C contains a k-CNF F with $c(k, \epsilon)n$ clauses such that F accepts $O(2^{\epsilon n})$ inputs and $F^{-1}(1) \subseteq f^{-1}(1)$. Our goal now is to show that for most degree 2 GF(2) polynomials f, no such CNF F exists; this implies that $s \geq 2^{(1-2\epsilon n)}$ and the theorem follows.

Our proof strategy is similar to that of [23]. We show that among the inputs accepted by F, there is a subcollection of 2^l inputs with certain special structure. The existence of such a structure is proved based solely on the density of F. We then prove that almost all degree 2 GF(2) polynomials cannot be constant on any such structure, contradicting the fact that F accepts a large set of inputs where the polynomial is 1.

Let $A = F^{-1}(1)$, and let l be the least integer such that $\Sigma_{i=1}^{l} \binom{n}{i} \geq |A| \geq 2^{\epsilon n}$. ¿From the work of [26, 35], we know that there is a set of l variables which are *shattered* by A. That is, A contains a subset B of size 2^l such that B restricted to those l variables still has size 2^l, so all possible assignments to those variables appear in B. If more than one such subset exists, fix one in a canonical fashion. Without loss of generality, we assume that $x_1, \ldots, x_l$ represent the l shattered variables. We call $x_1, \ldots, x_l$ free variables since, for any instantiation of these variables, we can find an element of B that agrees with the instantiation. $l \geq \delta n$ for some constant $\delta = \delta(\epsilon) > 0$. We call the other $n - l$ variables, $x_{l+1}, \ldots, x_n$, nonfree variables. Each such nonfree variable can be functionally expressed in terms of the free variables. Using the fact that every Boolean function is represented by a unique GF(2) polynomial with degree at most the number of input variables, we define $x_j = f_j(x_1, \ldots, x_l)$ for $l + 1 \leq j \leq n$. Thus B is the set of 2^l solutions of a system of $n - l$ polynomial equations, $x_j = f_j(x_1, \ldots, x_l)$, in l free variables.

We now argue that not too many degree 2 GF(2) polynomials can be constant when restricted to such a set B.

Lemma 11 *Let B be a set of size 2^l given by the solutions of a system of polynomial equations, $x_j = f_j(x_1, \ldots, x_l)$ for $l + 1 \leq j \leq n$, where f_j are degree l GF(2) polynomials. Then there are at most $2^{1+(n-l)(n+l+1)/2}$ polynomials of degree at most 2 which are constant when restricted to the set B.*

Proof: Let $G_2(n)$ be the group of all GF(2) polynomials of degree at most 2 in n variables. The restriction of a polynomial $g \in G_2(n)$ to the set B can be obtained by substituting the nonfree variables by the corresponding polynomial in the l free variables. Consider the monomials of degree at most 2 in the resulting polynomial. Call the GF(2) sum of these monomials p'. Note that p' is a polynomial of degree at most 2 in l variables. Observe that the map from $G_2(n) \to G_2(l)$ that maps p to p' is a surjective group homomorphism. For the restriction of p to B to be constant, it is necessary that p' be a constant polynomial. By the unique representation of GF(2) polynomials, there are precisely two constant polynomials, 0 and 1. Hence, at most $2^{1+(n-l)(n+l+1)/2}$ polynomials of degree at most 2 are constant when restricted to B. ■

Now, since $B \subseteq A = F^{-1}(1)$, the polynomial f must be identically 1 on B since C computes f. Every circuit which computes a degree 2 polynomial f has an associated k-CNF F accepting many inputs, but by the lemma above, each such F is consistent with only a few polynomials f. Furthermore, F has at most $c(k, \epsilon)n$ clauses, and there are at most $\binom{(2n)^k}{c(k,\epsilon)n}$ such formulae. Thus, the number of polynomials which have some sparse k-CNF F consistent with them is at most

$$2^{1+(n-l)(n+l+1)/2} 2^{(1+\log n)kc(k,\epsilon)n}$$

Recall that $l \geq \delta(\epsilon)n$ and $\delta(\epsilon)$ is a constant and thus $\omega(\sqrt{\log n/n})$. Thus, only an exponentially small (in n) fraction of all degree 2 GF(2) polynomials can have Σ_3^k circuits of size $2^{(1-2\epsilon)n}$.

Acknowledgments: The authors thank Sanjeev Arora, Will Evans, Satya Lokam, Pavel Pudlák, Toni Pitassi, Mike Saks, Janos Simon, and Avi Wigderson for useful discussions. We also thank the referees for the useful comments.

References

[1] Abrahamson, K.A., Downey, R.G. and Fellows, M.R. (1995), Fixed Parameter Tractability and Completeness IV: On Completeness for $W[P]$ and $PSPACE$ Analogues, in *Annals of Pure and Applied Logic*, vol. 73, 235-276, 1995.

[2] Ajtái M., Σ^1_1-Formulae on Finite Structures, *Annals of Pure and Applied Logic*, **24**, pp. 1–48, 1983.

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

[4] S. Arora, Polynomial Time Approximation Schemes for Euclidean TSP and Other Geometric Problems, In *Proc. 37th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1996, pp. 2-11.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Hardness of Approximation Problems. In *Proc. 33rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 14–23, 1992.

[6] Bellare, M., Proof Checking and Approximation: Towards Tight Results, Complexity Theory Column of Sigact News, Vol. 27, No. 1, March 1996.

[7] R. Beigel and R. Eppstein, 3-Coloring in $O(1.3446^n)$ time: a no-MIS algorithm, *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pp. 444-453, 1995.

[8] Boppana, R. and Sipser, M. (1990), The Complexity of Finite Functions, *in* "The Handbook of Theoretical Computer science", Vol. A, Elsevier Science Publishers.

[9] Cormen, T., Leiserson, C., and Rivest, R. *Introduction to Algorithms*, MIT Press, Cambridge, Mass., 1990.

[10] Feige, U., Goldwasser, S., Lovász, L., Safra, S.,Szegedy, M., Approximating Clique is Almost NP-complete, In *Proc. 32nd IEEE Symposium on Foundations of Computer Science* , pages 2-12, 1991.

[11] T. Feder and R. Motwani, Worst-case Time Bounds for Coloring and Satisfiability Problems, manuscript.

[12] Furst, M., Saxe, J.B. and Sipser, M. (1984), Parity, Circuits, and the Polynomial Time Hierarchy, *Mathematical Systems Theory*, **17**, pp. 13–28.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, San Francisco, CA, 1979.

[14] Håstad, J., (1986), Almost Optimal Lower Bounds for Small Depth Circuits, *in* "Proceedings of the 18th ACM Symposium on Theory of Computing", pp. 6–20.

[15] Håstad, J., Clique is hard to approximate within $n^{1-\epsilon}$. In *37th Annual Symposium on Foundations of Computer Science*, 627-636, October 1996.

[16] Håstad J., Some optimal inapproximability results. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 1-10, El Paso, Texas, 4-6 May 1997.

[17] Håstad, J., Jukna, S., and Pudlák, P., (1993), Top–Down Lower Bounds for Depth 3 Circuits, Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science", pp. 124–129.

[18] E. A. Hirsch, Two New Upper Bounds for SAT, ACM-SIAM Symposium on Discrete Algorithms, 1998.

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

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

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

[22] R.Paturi, P. Pudlák, M.E. Saks, and F. Zane., (1998), An Improved Exponential-time Algorithm for k-SAT, In *1998 Annual IEEE Symposium on Foundations of Computer Science*, pp. 628-637.

[23] Paturi, R., Saks, M.E., and Zane F., (1997), Exponential Lower Bounds on Depth 3 Boolean Circuits, *in* Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pp. 86-91, May 1997.

[24] C. Papadimitriou and M. Yannakakis, Optimization, approximation and complexity classes. In *Journal of Computer and System Sciences*, 43, pages 425–440, 1991.

[25] Razborov, A.A. (1986), Lower Bounds on the Size of Bounded Depth Networks over a Complete Basis with Logical Addition, *Matematicheskie Zametki* **41** pp. 598–607 (in Russian). English Translation in *Mathematical Notes of the Academy of Sciences of the USSR* **41**, pp. 333–338.

[26] Sauer, N. (1972), On the Density of Families of Sets, *Journal of Combinatorial Theory, series A*, vol. **13**, pp. 145–147.

[27] 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.

[28] 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.

[29] Uwe Schöning, (1999), A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems, Preprint.

[30] Smolensky, R. (1987), Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity, *in* "Proceedings of the 19th ACM symposium on Theory of Computing", pp. 77–82.

[31] Stearns, R.E. and Hunt, H.B., III (1990), Power Indices and Easier Hard Problems, *Mathematical Systems Theory*, vol. 23, 209-225.

[32] D. S. Johnson, and M. Szegedy (1999), What are the least tractable instances of Max Clique?, ACM-SIAM Symposium on Discrete Algorithms, 1999.

[33] Valiant, L.G., (1977), Graph–theoretic arguments in low–level complexity, in *Proceedings of the 6th Symposium on Mathematical Foundations of Computer Science*, Springer–Verlag, Lecture Notes in Computer Science, vol. 53, pp. 162–176.

[34] Van Lint, J.H., Introduction to Coding Theory, 2nd Edition, Springer–Verlag, 1992.

[35] Vapnik, V.N., and Chervonenkis, A. Ya, (1971), On the Uniform Convergence of Relative Frequencies of Events to their Probabilities, *Theory of Probability Applications*, vol. **16**, pp. 264–280.

[36] Yao, A. C–C. (1985), Separating the Polynomial Hierarchy by Oracles, *in* "Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science", pp. 1–10.