Chicago Journal of Theoretical Computer Science

The MIT Press

Volume 1997, Article 1

12 March 1997

ISSN 1073–0486. MIT Press Journals, 55 Hayward St., Cambridge, MA 02142 USA; (617)253-2889; *journals-orders@mit.edu, journals-info@mit.edu*. Published one article at a time in LaTeX source form on the Internet. Pagination varies from copy to copy. For more information and other articles see:

- *http://www-mitpress.mit.edu/jrnls-catalog/chicago.html*
- *http://www.cs.uchicago.edu/publications/cjtcs/*
- *ftp://mitpress.mit.edu/pub/CJTCS*
- *ftp://cs.uchicago.edu/pub/publications/cjtcs*

The *Chicago Journal of Theoretical Computer Science* is abstracted or indexed in *Research Alert,*® *SciSearch,*® *Current Contents*® */Engineering Computing & Technology*, and *CompuMath Citation Index.*®

The *Chicago Journal of Theoretical Computer Science* is a peer-reviewed scholarly journal in theoretical computer science. The journal is committed to providing a forum for significant results on theoretical aspects of all topics in computer science.

On Limited versus Polynomial Nondeterminism

Uriel Feige Joe Kilian

12 March, 1997

Abstract

Abstract-1 In this paper, we show that efficient algorithms for some problems that require limited nondeterminism imply the subexponential simulation of nondeterministic computation by deterministic computation. In particular, if cliques of size $O(\log n)$ can be found in polynomial time, then nondeterministic time $f(n)$ is contained in deterministic time $2^{O(\sqrt{f(n) \operatorname{polylog} f(n)})}$.

1 Introduction

1-1 A major open question in computational complexity is whether $P = NP$. In other words, is it true that if a language $\mathcal{L}$ can be recognized in time $f(n)$ by a nondeterministic Turing machine (where n denotes the input length), then $\mathcal{L}$ can be recognized in time $(f(n))^c$ by a deterministic Turing machine (for some constant c)? Let $DTIME(f(n))$ denote the class of languages accepted by a deterministic Turing machine in time $O(f(n))$, and let $NTIME(f(n))$ denote the class of languages accepted by a nondeterministic Turing machine in time $O(f(n))$. The trivial relations between these classes are $DTIME(f(n)) \subset NTIME(f(n))$ (by definition), and $NTIME(f(n)) \subset DTIME(2^{O(f(n))})$ (by exhaustive search). Perhaps the only nontrivial relation known is that $DTIME(n) \neq NTIME(n)$ [PPST83].

1-2 A natural question to ask is whether there is a general methodology that improves over exhaustive search, and applies to a wide range of NP problems. For some specific NP problems, improvements over exhaustive search that involve the constant in the exponent were obtained [TT77, SS79, BE95].

However, we seek a more dramatic improvement, with more general applicability. This leads to the following question:

Question 1 *Is there a subexponential deterministic simulation of nondeterministic computation? More explicitly, is there some constant* $\delta < 1$ *such that* $NTIME(f(n)) \subset DTIME(2^{O(f(n)^{\delta})})$*?*

1-3 Certain NP problems require only a limited amount of nondeterminism, in the sense that their (natural) NP-witness is at most polylogarithmic in the input length. For such problems, improvements over exhaustive search are sometimes dramatic. Examples of problems that require limited nondeterminism can be constructed by considering parameterized versions of NP optimization problems, in which the parameter k is restricted to be small (typically constant, or $O(\log n)$). In the problem definitions below, k is a positive-integer input parameter.

Path

INSTANCE: A graph G of order n.

QUESTION: Does G contain a simple path of length at least k?

Vertex Cover

INSTANCE: A graph G of order n.

QUESTION: Does G contain a set of vertices of cardinality at most k that is incident with every edge of G?

Clique

INSTANCE: A graph G of order n.

QUESTION: Does G contain a complete subgraph of order at least k?

Monotone Circuit Satisfiability

INSTANCE: A monotone circuit C on n Boolean inputs.

QUESTION: Does C accept an input vector of Hamming weight at most k?

Tournament Dominating Set

INSTANCE: A digraph $G = (V, E)$ of order n, in which for every pair of vertices $u, v \in V$, either $(u, v) \in E$ or $(v, u) \in E$.

QUESTION: Find a set $D \subset V$ of minimum cardinality such that for each vertex $v \in V$, there is an edge $(u, v) \in E$ for some vertex $u \in D$.

VC Dimension

INSTANCE: A set U of cardinality n and a family $\mathcal{F}$ of n subsets of U.

QUESTION: Find a set $S \subset U$ of maximum cardinality that is *shattered* by $\mathcal{F}$. Set S is shattered by $\mathcal{F}$ if for every subset $T \subset \mathcal{F}$ there is a set $A \in \mathcal{F}$ such that $A \cap S = T$.

We remark that the last two problems above are search problems, rather than decision problems, and also that there is no explicit parameter k involved. However, a parameter of $k \simeq \log n$ is implicit, since the cardinality of the optimal solution is at most $\log n + 1$.

1-4 For the problems above, with parameter $k = \log n$, an exhaustive search requires time of roughly $n^{\log n}$. However, the first two of these problems have algorithms of time complexity $O(n^c 2^k)$, for some constant c, which is polynomial even for $k = \log n$ [PY96, DF95, AYZ95]. (Note that these two problems are *NP*-hard for a general k, since the Vertex Cover and Hamiltonian Path problems are.) Is this phenomenon of dramatically improving over exhaustive search a general one when problems that require a limited amount of nondeterminism are concerned? Does the same apply for the k-Clique problem when $k = O(\log n)$? These types of questions are treated by the theory of *fixed parameter intractability* [DF92], and motivate the introduction of the complexity class *LOGSNP* [PY96]. We shall return to discuss these notions in Section 4.2.1. For now, we shall concentrate on the log-Clique problem in which one must find a clique of size $k \simeq \log n$ in the input graph.

Question 2 *Is the* log-*Clique problem solvable in polynomial time?*

1-5 Our main result is a connection between Question 2, that deals with a case of limited nondeterminism, and Question 1, that deals with polynomial nondeterminism. In Section 2 we show that if the answer to Question 2 is positive, then so is the answer to Question 1. In Section 3 we present extensions that deal with the complexity of finding approximate solutions to the k-Clique problem, for small values of k. In Section 4 we discuss our results and their implications. In particular, we discuss related work showing (for the k-Monotone Circuit Satisfiability problem [ADF95]) or implying in conjunction with our work (for the Tournament Dominating Set and VC-Dimension problems [PY96]) additional problems of limited nondeterminism whose complexity is related with subexponential simulation of NP.

2 On the Complexity of the log-Clique Problem

2-1 The results of this section were first proven as corollaries of the results of Section 3. The elementary proofs given here were noticed by Noam Nisan, who permitted us to use them.

2-2

Lemma 1 *Suppose there is a polynomial time algorithm for the k-Clique problem when $k \leq \log n$. Then the k-Clique problem is solvable in time $O(n^{O(\sqrt{n})})$, for every k.*

Prove Lemma 1-1 **Proof of Lemma 1** Let $\mathcal{A}$ be an algorithm that in time $O(n^c)$ solves the k-Clique problem for $k \leq \log n$. Let (G, k) be an input instance for the k-Clique problem (with arbitrarily large k). Reduce (G, k) to a new instance $(\hat{G}, \hat{k})$ of the k-Clique problem as follows. Without loss of generality, assume that k is a perfect square. (Otherwise, let k' be the least perfect square greater than k. Add $k' - k$ vertices to G, connected to every other vertex.) Graph $\hat{G}$ contains $N = \binom{n}{\sqrt{k}}$ vertices, where each vertex of $\hat{G}$ corresponds to a $\sqrt{k}$-subset of vertices of G. Two vertices of $\hat{G}$ are connected by an edge if the $2\sqrt{k}$ vertices that they correspond to are distinct and form a clique of size $2\sqrt{k}$ in G. Set $\hat{k} = \sqrt{k}$.

Proposition 1 *$\hat{G}$ has a $\hat{k}$-clique if and only if G has a k-clique.*

Prove Prop 1-1 **Proof of Proposition 1** If G has a k-clique, then partition the vertices of this clique into $\sqrt{k}$ sets of size $\sqrt{k}$. These sets make up $\sqrt{k} = \hat{k}$ vertices of $\hat{G}$ that together form a clique in $\hat{G}$.

Prove Prop 1-2 If $\hat{G}$ has a $\hat{k}$-clique, then the vertices of G comprising each vertex of this $\hat{k}$-clique are all distinct, and form a clique of size $\sqrt{k} \cdot \sqrt{k} = k$ in G.

Proof of Proposition 1 □

Prove Lemma 1-2 The number of vertices of $\hat{G}$ is $N = \binom{n}{\sqrt{k}} < n^{\sqrt{n}}$. The construction of the edges of $\hat{G}$ takes time $O(N^2)$, times some polynomial in n reflecting the time it takes to check whether $2\sqrt{k}$ vertices form a clique in G. Hence the reduction from (G, k) to $(\hat{G}, \hat{k})$ takes time $n^{O(\sqrt{n})}$. The value of $\hat{k}$ satisfies $\hat{k} \leq \log N$. Therefore, the k-Clique problem on $(\hat{G}, \hat{k})$ can be solved in time $O(n^{c\sqrt{n}})$, using algorithm $\mathcal{A}$. The proof of the lemma follows from Proposition 1.

Proof of Lemma 1 □

2-3 Lemma 1 shows that if the log-Clique problem is in P, then a specific NP problem has subexponential complexity. The conclusion can be extended to some other NP problems—those from which there is a linear (or slightly superlinear) reduction to the k-Clique problem. In particular, let $\mathcal{L}$ be a language in $NTIME((f(n))$, where $f(n)$ is a proper complexity function (as defined in [Pap95], for example). Then the reductions of Pippenger and Fischer [PF79] show that it has a uniform family of nondeterministic circuits $C_L(n)$ of bounded fan-in and size $f'(n) = O(f(n) \log f(n))$. To check whether $x \in \mathcal{L}$, one determines whether input x is accepted by the nondeterministic circuit $C_L(n)$, where n is the length of x. This problem can be reduced to the k-Clique problem on a graph with $O(f'(n))$ vertices. Hence we obtain the following.

Theorem 1 *Suppose there is a polynomial time algorithm for the* log-*Clique problem. Then there is a deterministic subexponential simulation of nondeterministic computations. Specifically, for each proper complexity function $f(n) > n$, $NTIME(f(n))$ is contained in $DTIME((f'(n))^{\sqrt{f'(n)}})$, where $f'(n) = O(f(n) \log f(n))$.*

Prove Theorem 1-1 **Proof of Theorem 1** The theorem follows trivially from the discussion preceding it. For thoroughness, we explain the reduction from the Nondeterministic Circuit Value problem to Clique. Recall that a family of circuits for language $\mathcal{L}$ has a different circuit $C_L(n)$ for each input size n. The circuit has *and*, *or*, and *not* gates of fan-in 2 (or 1, for *not* gates) and unbounded fan-out, n input wires that encode the input x, and one output wire. A circuit is nondeterministic if, in addition, there are nondeterministic input wires (that encode the witness w). If $x \in \mathcal{L}$, there is a setting to these wires so that the output is 1, and if $x \notin \mathcal{L}$, then for every setting to these wires, the output is 0. The size of circuit C, denoted by $|C|$, is the number of gates in the circuit. The nondeterministic time for language $\mathcal{L}$ is a function of the input size n, and is defined as $NT_L(n) = \min_{C_n}[|C_n|]$ (where C_n correctly recognizes $\mathcal{L} \cap \{0,1\}^n$). This is the usual correspondence between circuit size and time. If we require that the circuit family is efficiently constructible, then the time-complexity measure that we consider is a *uniform* measure.

Prove Theorem 1-2 To check whether a given nondeterministic circuit C of size m accepts input x, we may use the following reduction to the k-Clique problem. Construct a graph G on at most $4m$ vertices that has a clique of size $k = m$ if and only if the circuit is accepting. Each input to a gate is labeled with a fresh variable. A two-input gate can be represented by four vertices, one for each combination of values to the two variables that are the input of the gate. (If one of the variables is an input variable, then include only vertices that are consistent with its value.) Each vertex also implies a value for its respective output. For the output gate of the circuit, include only the vertices that imply an output of 1 (*accept*). Every pair of vertices in the graph is connected by an edge, unless the vertices are contradictory (they imply two different values for some variable). The graph has a clique of size m precisely when there is an assignment to the nondeterministic inputs of the circuit that forces an output of 1.

Proof of Theorem 1 □

2-4 To put Lemma 1 and Theorem 1 in perspective, observe what would happen if we had a polynomial time algorithm that finds cliques of size $(\log n)^2$ in graphs. Then clearly, the k-Clique problem would have a subexponential algorithm for every value of k, simply by padding the input graph by $2^{\sqrt{k}}$ isolated vertices. A similar observation applies to parameterized versions of many other *NP* problems (such as the Path and the Vertex Cover problems). Hence the new element in our connection between limited and polynomial nondeterminism is a quantitative one, the level of limited nondeterminism in which such a connection can be demonstrated.

3 On Finding Small Cliques

3-1 As we have seen in Section 2, a polynomial time algorithm for finding cliques of size $k = \log n$ implies subexponential algorithms for *NP* problems. In this section, we investigate weaker conditions that imply subexponential algorithms for *NP*. The condition $k = \log n$ can be relaxed to $k = (\log n)^\alpha$, for every $\alpha > 0$. Moreover, the running time of the algorithm that finds small cliques can be slightly superpolynomial, to an extent that depends on α. We can further relax our conditions if we are satisfied with a weaker implication, that of the existence of *randomized* subexponential algorithms for NP problems. We present the relaxed implication in Section 3.1, the relaxed conditions in Section 3.2, and the reduction between the two in Section 3.3. This reduction is more sophisticated than that of Section 2, and it uses recent results from the theory of interactive proofs.

3.1 The Goal: Subexponential Algorithms for *NP*

3.1-1 It is most convenient to present our goal in the circuit model. Recall that (nonuniform) computation time is associated with circuit size, and let $T_L(n)$ ($NT_L(n)$) denote the (non)deterministic circuit complexity of language L. Clearly, $T_L(n) \leq 2^{O(NT_L(n))}$, since every function has exponential-size circuits. Interpreting Question 1 in the nonuniform circuit model, we obtain our first goal.

Goal 1 *For some $\epsilon > 0$ and for every language L, if time complexity is measured as size of nonuniform circuit families, then*

$$T_L(n) = O(2^{(NT_L(n))^{1-\epsilon}})$$

3.1-2 There is also a uniform version of our goal. However, it involves randomized algorithms. Since we will be dealing with superpolynomial randomized algorithms, we allow ourselves a relaxed notion of uniformity for nondeterministic circuits. It has the additional advantage of considering the complexity of individual instances, rather than clustering all size-n instances together.

Definition 1 *A uniform randomized algorithm A is a* weakly efficient generator of nondeterministic circuits for language L *if it has the following properties:*

1. *On input x, $A(x)$ outputs a nondeterministic circuit of size at most $S_{A,x}$, drawn at random from a distribution $C_{A,x}$ of nondeterministic circuits.*

2. *$A(x)$ runs in time subexponential in $S_{A,x}$, that is in time $O(2^{(S_{A,x})^{1-\epsilon}})$.*

3. *With probability at least 2/3 (over the coin tosses of A), the output circuit correctly decides (nondeterministically) whether $x \in L$.*

The weak uniform nondeterministic complexity of x relative to A is $S_{A,x}$.

3.1-3 The traditional notion of uniform circuit complexity is a special case of Definition 1, when A runs in deterministic polynomial time with unary input $n = |x|$. Then A generates a uniform nondeterministic circuit family for L with complexity $S_{A,1^n}$.

Goal 2 *For some $\epsilon > 0$ and for every language L that has a corresponding generator of nondeterministic circuits A, there exists a uniform randomized algorithm B that decides L, and on input x runs in time*

$$RT(x) = O(2^{(S_{A,x})^{1-\epsilon}})$$

3.1-4 In Section 4.1, we further discuss Goals 1 and 2.

3.2 The Challenge: Finding Small Cliques

3.2-1 We present our challenge, which is an algorithmic problem that, if solved, would lead to our goals.

Challenge 1 *Design a deterministic algorithm that for n-vertex graphs finds a clique of size k in time $O(n^{k^{1-\epsilon}})$ (if indeed the graph has such a clique), where $\epsilon > 0$, and $k = \Theta((\log n)^c)$ for some $c > 0$.*

3.2-2 Observe that using exhaustive search, $O(n^k)$ time is achievable. Observe also that we are not asking for a polynomial time algorithm for finding small cliques, but just for a substantial improvement over exhaustive search.

3.2-3 Our challenge can be relaxed, while still serving its purpose:

Challenge 2 *Design a randomized algorithm that distinguishes (with probability at least 2/3) between graphs with maximum cliques of size at most k and graphs with maximum cliques of size at least $2k$. The running time of the algorithm is required to be $O(n^{k^{1-\epsilon}})$, where $k = \Theta((\log n)^c)$, for some $c > 0$, $\epsilon > 0$. The algorithm is not required to output a clique, just a decision.*

3.2-4 In Section 4.2, we discuss the plausibility of our challenges.

3.3 The Reduction

3.3-1 Playing with the parameters of Lemma 1, one can verify that Challenge 1, if met, implies a positive answer to Question 1. In this section, we describe the reduction from Goals 1 and 2 to Challenge 2. It is based on interactive proofs, and in particular on proof systems developed by Polishchuk and Spielman [PS94]. They show that the verification of whether an input x of length n is accepted by a nondeterministic circuit C of size $m > n$ can be reduced to the verification of a corresponding *holographic proof* with the following properties.

1. The holographic proof is just a string of bits. It is an NP-witness to the fact that x is accepted bycircuit C, which can be verified with high confidence by examining at random only a few bits of the witness.

2. There is a uniform random polynomial-time verification algorithm V associated with the holographic proof. The number of random bits used by algorithm V is r, where r is bounded by some polynomial in m.

3. Algorithm V queries the holographic proof at $q = O(m^{o(1)})$ bit locations. These locations are selected at random, based on the r random bits.

4. If x is accepted by circuit C, then there exists a holographic proof h of length $l = O(m^{1+o(1)})$ that always causes $V(x, C, h)$ to accept (regardless of the value of the random bits used by V).

5. If x is not accepted by circuit C, then for every string h' of length l, the probability that $V(x, C, h')$ accepts is at most 1/2 (known as the *error probability*).

3.3-2 No familiarity with the concept of holographic proofs except for the properties listed above is required of the reader. We remark that the bounds that we require on l (the size of the holographic proof), on q (the number of bits queried), and on r (the number of random bits used), are weaker than those actually achieved by Polischchuk and Spielman [PS94].

Theorem 2 *Challenge 2 implies Goals 1 and 2.*

Prove Theorem 2-1

Proof of Theorem 2 We reduce the existence of the holographic proof described above to the k-Clique problem. The reduction is based on the reduction described by Feige et al. [FGL+96], and on subsequent extensions of Zuckerman [Zuc96]. We optimize the parameters of the reduction to best serve our goal.

Prove Theorem 2-2

The reduction proceeds in two phases. The goal of the first phase is to reduce the number of random bits needed by the verifier to check the holographic proof. In Polishchuk and Spielman's [PS94] proof system, the verifier uses r random bits, where $r \geq \log m$. For our construction, we need to reduce the number of random bits used by the verifier while maintaining a low error probability. We do this by considering two different types of random bits that the verifier can use. One type is *private* random bits that the verifier gets after the prover writes down the proposed holographic proof, as is the case in ordinary holographic proofs. The other type is a table T of *public* random bits, to which the prover also has access before writing down the proposed holographic proof. It turns out that for our purpose, we only need to reduce the number of private random bits. This reduction is made possible by the introduction of the public random bits, and by an increase in the number of queried bits.

Lemma 2 *Assume that there is a holographic proof system with verifier V for checking whether circuit C accepts input x, with parameters r (number of private random bits used by the verifier), l (proof length), and q (number of query bits), as described above. Let $s > 6$ be a parameter to be chosen later. Then there is a holographic proof system with public table T and verifier V' with the following properties.*

1. *The proof length is l, the number of private random bits is $\log(3l/s)$, the number of queried bits is qs, and the table T contains $3lr$ public random bits.*

2. *If x is accepted by circuit C, then there exists a holographic proof h of length l that always causes $V'(x, C, h, T)$ to accept (regardless of the value of the public and private random bits).*

3. *If x is not accepted by circuit C, then most possible tables T are* typical, *where T is typical if for every string h' of length l, the probability (over the private randomness) that $V(x, C, h', T)$ accepts is at most $1/2$.*

Prove Lemma 2-1 **Proof of Lemma 2** We describe the holographic proof system with public table T and verifier V'. When x is accepted by circuit C, the same holographic proof h as in Polishchuk and Spielman's proof system [PS94] serves as a holographic proof in our new proof system, regardless of the contents of the table T. Verifier V' proceeds as follows. Partition the table T of size $3lr$ into $k = 3l/s$ rows, where each row contains s strings of length r. Let t_{ij} denote the jth string of row i in table T. Using $\log k$ private random bits, V' selects a row i at random, and simulates the verifier V of [PS94] s times, each time using a fresh string t_{ij} (for $1 \leq j \leq s$) as private random bits for V. If V accepts in all s simulations, then V' accepts as well. Otherwise, V' rejects.

Prove Lemma 2-2 Properties 1 and 2 of Lemma 2 clearly hold. It remains to consider the case where x is not accepted by circuit C and show that most T are typical. In this case, fix a (false) holographic proof h'. The probability that it is not exposed by s simulations of V, each time with random and independent private coins, is at most 2^{-s}. Hence the probability (over the choice of T) that h' is not exposed by at least half of the k random strings in the table T is at most $2^k 2^{-ks/2} = 2^{3l/s-3l/2} \ll 2^{-l}$ (for $s > 6$). Since there are only 2^l possible false holographic proofs, it follows that with high probability every such holographic proof is exposed by at least $k/2$ of the random strings in T. Hence for most choices of T, V' accepts with probability at most 1/2.

Proof of Lemma 2 □

Prove Theorem 2-3 The first phase of our reduction corresponds to the selection of a table T with $k = 3l/s$ rows of rs random bits. The second phase of our reduction is similar to the reduction of Feige et al. [FGL⁺96]. We construct a k-partite graph G. Each part of the graph corresponds to one of the k rows of T. The vertices in each part are all strings of length qs, which represent all possible answer sequences to the qs queried bits. A vertex is removed if the corresponding answer sequence leads V' to reject. Two vertices (in different parts of G) are connected by an edge if there exists some (possibly false) holographic proof that is consistent with both vertices. Graph G contains at most $k2^{qs}$ vertices, and can be costructed in time $O((lk2^{qs})^c)$, for some constant c. If x is accepted by circuit C, then the largest clique is of size k. If x is not accepted by circuit C, then T is likely to be typical, in which case the largest clique is of size $k/2$.

Prove Theorem 2-4 Assume now that Challenge 2 is true. That is, there is an algorithm that distinguishes between the existence of cliques of size $(\log n)^c$ and the inexistence of cliques of half this size, in time $n^{(\log n)^{c(1-\epsilon)}}$. To check whether the nondeterministic circuit C of size m accepts input x, apply the reduction above with $s = m^{1/(c+1)}$. Recall that $l = m^{1+o(1)}$, that $k = 3l/s$, and that $q = n^{o(1)}$. Ignoring $o(1)$ factors in the exponent, we obtain that $k \simeq m^{c/(c+1)}$ and that the number of vertices in G is $N \simeq k2^{qs} \simeq 2^{m^{1/(c+1)}}$. Hence $k \simeq (\log N)^c$ (the graph G can be padded with dummy vertices to achieve exact equality, if desired). Now the question of whether C accepts x can be decided in randomized time

$$N^{(\log N)^{c(1-\epsilon)}} = 2^{m^{1/(c+1)+o(1)} m^{c(1-\epsilon)/(c+1)+o(1)}} = 2^{m^{1-c\epsilon/(c+1)}+o(1)}$$

or in time $2^{m^{1-\epsilon'}}$, where $\epsilon' = c\epsilon/(c+1) + o(1)$. This proves Goal 2.

Prove Theorem 2-5 To see that Goal 1 follows as well, we use the well known reductions from randomized algorithms to randomized circuits and from randomized circuits to nonuniform circuits. These reductions result in a circuit whose size is bounded by a polynomial in the running time of the original randomized algorithm, and this polynomial overhead is negligible relative to the super-polynomial complexities involved in our goals.

Proof of Theorem 2 □

4 Discussion

4-1 We have seen that if the log-Clique problem is in P (or if the weaker Challenge 2 is met), then there is subexponential deterministic (or randomized) simulation of nondeterministic computation. We do not wish to take a position regarding whether our results indicate that finding small cliques is hard, or that there are subexponential simulations. However, we do wish to point out that the connection between limited nondeterminism and polynomial nondeterminism suggests that the complexity of problems such as the log-Clique problem is a question worthy of further research. Hence we discuss our challenges in more detail in Section 4.2. Before that, in Section 4.1, we make some observations regarding the motivating goal.

4.1 The Significance of our Goal

4.1-1 The goal that we consider is that of finding subexponential simulation for nondeterministic computation. This goal came in three different versions, the uniform one (Question 1), the nonuniform one (Goal 1), and the randomized one (Goal 2). The uniform version is perhaps the most appealing. The other two versions were introduced so that we can draw consequences from a challenge that postulates the existence of an efficient algorithm that (in graphs with small cliques) approximates the size of the maximum clique within a constant factor, rather than give its exact size. Our reduction in this case is randomized (specifically, the choice of table T in Theorem 2), and we do not know if it can be replaced by a deterministic reduction.

4.1-2 In Goal 1, nondeterministic time is measured as the size of the nonuniform, nondeterministic circuit family. This can be replaced by other measures of nondeterministic time, provided that nondeterministic circuit size is at most slightly superlinear in these measures. Pippenger and Fischer [PF79] have shown that there is only a logarithmic blowup when circuits simulate multitape Turing machines. Similarly, the results of Goldreich and Ostrovsky [GO96], when transformed to a nondeterministic setting (details omitted), imply that nondeterministic RAMs can be simulated with polylogarithmic overhead by nondeterministic circuits. (Possibly there is a known simpler proof for this last fact.) Hence, both these measures for nondeterministic time can replace nondeterministic circuit complexity in Goal 1.

4.1-3 There is a large class of *NP*-complete problems for which even if subexponential simulation of *NP* is achieved (in the sense of this paper), no improvement in the known deterministic running time will follow. These problems are those for which the witness length is much shorter than the input length. Typical examples arise in many graph-theoretic problems, in which the input length depends on the number of edges, whereas the witness length depends on the number of vertices. For these problems (e.g., the Hamiltonicity, Vertex Cover, Chromatic Number, and Independent Set problems, all on dense graphs) it is easy to come up with deterministic algorithms that are subexponential in the size of the input. This reveals two weaknesses in our notion of subexponential simulation of nondeterminism:

1. we do not allow for the possibility of sublinear nondeterministic running times (at least not for functions that depend on all their inputs), and
2. we do not distinguish between deterministic and nondeterministic steps of a nondeterministic algorithm, a distinction that exhaustive search may well take advantage of.

4.1-4 Subexponential simulation of nondeterministic computation has cryptographic significance. A persistent trend in cryptography is to design encryption schemes that are as efficient as possible to apply (in nearly linear time), and as hard as possible to break (exponential in the length of the private key). The near-linearity requirement is important if one must encrypt large volumes of communication online, using only weak processors. The exponentiality requirement offers security against adversaries who try to break the encryption scheme offline, using very powerful computers. Goals 1 and 2, if achieved, imply that encryption schemes with properties as described above do not exist (at least not in an asymptotic sense, if private keys are long enough). The complexity of breaking nearly linear encryption schemes would be subexponential, comparable in nature to the complexity of known factorization algorithms (see, e.g., [BLP94]).

4.2 The Plausibility of Our Challenges

4.2-1 Challenge 1 calls for efficient algorithms for finding the maximum cliques in graphs that have small cliques. Challenge 2 calls for approximating the size of the maximum clique within a factor of two. Straightforward probabilistic arguments show that the size of the maximum clique in almost all graphs is roughly $2\log n$, and that simple greedy algorithms find cliques of size roughly $\log n$ in almost all graphs. Hence for most graphs, Challenge 2 can be met. We do not know if the same holds for Challenge 1. This partly motivates the introduction of Challenge 2.

4.2.1 Related Work

4.2.1-1 Finding small cliques in graphs is a special case of the question of finding (edge-induced) subgraphs that are isomorphic to other small subgraphs. For some of these subgraphs (e.g., finding a simple path of length $\log n$), Alon, Yuster, and Zwick [AYZ95] give a polynomial time algorithm, which is much more than we are asking for. The fastest known algorithms for finding subgraphs isomorphic to a graph H [PV90, AYZ95] require time $\Omega(n^t)$, where t is the *tree width* of H (a notion introduced by Robertson and Seymour [RS86]). Unfortunately, the tree width of k-cliques is $k-1$.

4.2.1-2 It is possible to find cliques of size $k = n - O(\log n)$ in polynomial time (when they exist). This follows from the fact that k-Vertex-Cover has complexity $O(n^c 2^k)$, and hence vertex covers of size $O(\log n)$ can be found in polynomial time [PY96, DF95, BG93]. To reduce an instance of the k-Clique problem to the k'-Vertex-Cover problem, simply consider the complement of the input graph, and set $k' = n - k$.

4.2.1-3 For decision problems that involve a parameter (such as in our case, where the size k of the clique is a parameter), one can sometimes design algorithms that have running time $O(n^c f(k))$, for some constant c, and arbitrary function f. In this case, the problem is *fixed-parameter tractable* [AEFM89, DF92], since for every fixed k, it can be solved in time $O(n^c)$, where c is independent of k. Downey and Fellows [DF92] introduced a hierarchy of fixed-parameter problems, and showed that the Clique problem is complete for $W[1]$, the lowest level of this hierarchy. Thus if the Clique problem is not fixed-parameter tractable, then no problem that is hard for some level of the hierarchy is fixed-parameter tractable. Our techniques allow us to draw some consequences from the assumption that the Clique problem is fixed-parameter tractable. More specifically, assume that the k-Clique problem can be decided in time $O(n^c f(k))$, and for simplicity, assume that $f(k)$ is a proper complexity function. Then for $k(n) = f^{-1}(n)$, the $k(n)$-Clique problem can be decided in polynomial time. Since $k(n)$ is not bounded by any constant, a proof similar to that of Lemma 1 implies that for every $\epsilon > 0$, the Clique problem can be decided in time $O(2^{\epsilon n})$. This implies similar savings when simulating nondeterministic circuits by deterministic ones (as in the proof of Theorem 1). We obtain the following.

Proposition 2 *If the k-Clique problem (or any other problem that is hard for the lowest level of the fixed-parameter hierarchy) is fixed-parameter tractable, then nondeterministic circuits of size NT can be simulated by deterministic circuits of size $2^{o(NT)}$.*

4.2.1-4 We remark that the VC-Dimension problem is contained in $W[1]$, and that the Tournament Dominating Set problem is complete for $W[2]$, the second level of the fixed-parameter hierarchy. The Monotone Circuit Satisfiability problem is complete for $W[P]$, the highest level of this hierarchy. In [ADF95] it is shown that the class $W[P]$ is fixed-parameter tractable if and only if satisfiability of a Boolean expression of size m on n variables can be solved in time $\text{poly}(m)2^{o(n)}$. This result is of a similar nature to our Proposition 2. It has the advantage of being an "if and only if" connection, and of being able to distinguish between nondeterministic and deterministic steps in efficiently simulating nondeterministic computation. Proposition 2 has the advantage of dealing with the lowest level of the fixed-parameter hierarchy.

4.2.1-5 A graph H is a *minor* of graph G if a graph isomorphic to H can be obtained by contracting edges of a subgraph of G. By the theory of Robertson and Seymour, for every fixed H, testing whether H is a minor of G can be done in time $O(|V(G)|^3)$ [RS95]. In particular, testing whether a graph G has a k-clique *as a minor* is fixed-parameter tractable.

4.2.1-6 Papadimitriou and Yannakakis [PY96] introduced the class *LOGSNP*. The problems in this class can be solved in time $O(n^{\log n})$. The log-Clique problem belongs to this class, but is not known to be complete for this class (see also [CC93]). Some natural questions, such as the VC-Dimension and Tournament Dominating Set problems, are *LOGSNP*-hard. A polynomial time algorithm for any of these problems implies a polynomial time algorithm for the log-Clique problem; hence, these problems can replace the log-Clique problem in Theorem 1.

4.2.2 Beating Exhaustive Search

4.2.2-1 We survey an approach developed by Itai and Rodeh [IR78] and Nesetril and Poljak [NP85] that shows that exhaustive search is not the quickest way to find cliques. Cliques of size 3 can be found quickly by computing G^3 (where G is the adjacency matrix of the graph) and searching for a nonzero diagonal entry. This requires $O(n^\omega)$ time, where $\omega < 3$ is the exponent for matrix multiplication (which currently stands at $\omega = 2.376$ [CW90]). Cliques of size k can be found by finding all $k/3$-cliques in the graph (for simplicity, assume that k is divisible by 3), then considering each such $k/3$-clique as a vertex in a new larger graph G', and connecting two vertices in G' if together their corresponding vertices in G constitute a $2k/3$-clique. Now matrix multiplication can be used to find 3-cycles in G', leading to an $O(n^{k\omega/3})$ algorithm.

4.2.2-2 The natural idea to improve upon the above algorithm is to use recursion. As a first step (one level of recursion), one may try to find cliques of size 9 in $O(n^{\omega^2})$ time, or even just improve upon $O(n^{3\omega})$. The problem that one encounters is that in order to employ the recursion, one needs not only to determine whether the graph has 3-cycles (which requires $O(n^\omega)$ time), but also to represent all such cycles. An explicit representation of all these cycles requires $\Omega(n^3)$ time and space, since the number of 3-cycles might be as large as $\binom{n}{3}$. An implicit representation may be much shorter (indeed, the graph itself serves as an implicit representation of all its cycles), but then it is not known how to perform matrix multiplication on this implicit representation. Thus, finding cliques of size 9 efficiently may be the key to the whole approach.

4.2.2-3 We would like to add here a philosophical remark (the last one). There is the known association of problems in P (or in BPP) as tractable, and problems not in BPP as intractable. When asked why an algorithm of complexity n^{13} is considered tractable, the usual answer is that n^{13} is intractable, but that one almost never comes up with a polynomial time problem that has this complexity. Is the question of finding cliques of size 20 an exception to this folklore rule?

5 Acknowledgments

5-1 We thank Noam Nisan for pointing out the simple proof of Section 2, and for allowing us to use it. We thank Mike Fellows for useful references on fixed-parameter tractability, Mihalis Yannakakis for directing us to [PY96], and Uri Zwick for helpful discussions.

Acknowledgement of support: Some of the research in this article was accomplished while Uriel Feige was visiting the NEC Research Institute.

References

[ADF95] K. Abrahamson, R. Downey, and M. Fellows. Fixed-parameter tractability and completeness IV: On completeness for $W[P]$ and *PSPACE* analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, June 1995.

[AEFM89] K. Abrahamson, J. Ellis, M. Fellows, and M. Mata. On the complexity of fixed-parameter problems. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 210–215, Washington, DC, 1989. IEEE Computer Society Press.

[AYZ95] N. Alon, R. Yuster, and U. Zwick. Color coding. *Journal of the ACM*, 42(4):844–856, July 1995.

[BE95] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3446^n)$: a no-MIS algorithm. In *Proceedings of the 36th IEEE Symposium on*

Foundations of Computer Science, pages 444–452, Washington, DC, 1995. IEEE Computer Society Press.

[BG93] J. Buss and J. Goldsmith. Nondeterminism within *P*. *SIAM Journal on Computing*, 22(3):560–572, June 1993.

[BLP94] J. P. Buhler, H. W. Lenstra, and Carl Pomerance. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1994.

[CC93] L. Cai and J. Chen. On the amount of nondeterminism and the power of verifying (extended abstract). In Andrzej M. Borzyszkowski and Stefan Sokolowski, editors, *Mathematical Foundations of Computer Science 1993, 18th International Symposium*, volume 711 of *Lecture Notes in Computer Science*, pages 311–320, Berlin, 1993. Springer-Verlag.

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progression. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[DF92] R. Downey and M. Fellows. Fixed-parameter intractability. In *Proceedings of the 7th Annual Symposium on Structure in Complexity Theory*, pages 36–49, 1992.

[DF95] R. Downey and M. Fellows. Parameterized computational feasibility. In P. Clote and J. B. Remmel, editors, *Feasible Mathematics II*, pages 219–244. Birkhauser, 1995.

[FGL+96] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.

[GO96] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.

[IR78] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978.

[NP85] J. Nesetril and S. Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

[Pap95] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1995.

[PF79] N. Pippenger and M. Fischer. Relation among complexity measures. *Journal of the ACM*, 26(2):361–381, 1979.

[PPST83] W. Paul, N. Pippenger, E. Szemeredi, and W. Trotter. On determinism versus nondeterminism and related problems. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 429–438, Washington, DC, 1983. IEEE Computer Society Press.

[PS94] A. Polishchuk and D. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 194–203, New York, 1994. ACM Press.

[PV90] J. Plehn and B. Voigt. Finding minimally weighted subgraphs. In *Proceedings of the 16th Workshop on Graph Theoretic Concepts in Computer Science*, pages 18–29, Berlin, 1990. Springer-Verlag.

[PY96] C. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the VC Dimension. *Journal of Computer and System Sciences*, 53(2):161–170, October 1996.

[RS86] N. Robertson and P. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[RS95] N. Robertson and P. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63:65–110, 1995.

[SS79] R. Schroeppel and A. Shamir. $T \cdot S^2 = O(2^n)$ time/space tradeoff for certain *NP*-complete problems. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 328–336, Washington, DC, 1979. IEEE Computer Society Press.

[TT77] R. Tarjan and A. Trojanowski. Finding a maximum independent set. *SIAM Journal on Computing*, 6(3):537–546, 1977.

[Zuc96] D. Zuckerman. On unapproximable versions of *NP*-complete problems. *SIAM Journal on Computing*, 25(6):1293–1304, December 1996.