Probabilistically Checkable Debate Systems and Approximation Algorithms for PSPACE-Hard Functions

Anne Condon*
University of Wisconsin
Computer Sciences Department
1210 West Dayton Street
Madison, WI 57306 USA
condon@cs.wisc.edu

Joan Feigenbaum
AT&T Bell Laboratories
Room 2C473
600 Mountain Avenue
Murray Hill, NJ 07974 USA
jf@research.att.com

Carsten Lund
AT&T Bell Laboratories
Room 2C324
600 Mountain Avenue
Murray Hill, NJ 07974 USA
lund@research.att.com

Peter Shor
AT&T Bell Laboratories
Room 2D149
600 Mountain Avenue
Murray Hill, NJ 07974 USA
shor@research.att.com

*Supported in part by NSF grants CCR-9100886 and CCR-9257241.

Abstract

We initiate an investigation of *probabilistically checkable debate systems* (PCDS's), a natural generalization of the probabilistically checkable proof systems studied in, e.g., [Babai et al., Proc. 23rd ACM Symp. Theory of Computing, 1991, pp. 21–31], [Feige et al., Proc. 32nd IEEE Symp. Foundations of Computer Science, 1991, pp. 2–12], [Arora et al., Proc. 33rd IEEE Symp. Foundations of Computer Science, 1992, pp. 2–13], [Arora et al., Proc. 33rd IEEE Symp. Foundations of Computer Science, 1992, pp. 14–23]. A PCDS for a language L consists of a probabilistic polynomial-time verifier V and a debate between player 1, who claims that the input x is in L, and player 0, who claims that the input x is not in L. We show that there is a PCDS for L in which V flips $O(\log n)$ random coins and reads $O(1)$ bits of the debate if and only if L is in PSPACE. This characterization of PSPACE is used to show that certain PSPACE-hard functions are as hard to approximate as they are to compute exactly.

These results were presented in preliminary form at the 25th ACM Symposium on Theory of Computing, San Diego CA, May 1993.

1 Introduction

Suppose that two candidates, B and C, agree to a debate format. Voter V is too busy to catch more than a very small number of bits of the debate. How does V decide which of B or C won the debate? In this paper, we show that if B and C choose the right debate format, V's problem is solved. By listening to a few, randomly chosen, sounds bites of the debate, V can with near certainty figure out who won.

Similarly, suppose that B or C is giving a speech to a set of voters $V_1, \cdots, V_n$, represented by finite automata. He would like to give the speech that results in acceptance (votes) by the greatest number of V_i's. We show that not only can he not compute this maximum exactly, but he cannot come within an arbitrary constant factor, unless he has access to an oracle (political consultant) with the full power of PSPACE.

Our work builds on the recent progress that has been made in the theory of *probabilistically checkable proof systems* (PCPS's). Results about the language-recognition power of PCPS's have led to lower bounds on the difficulty of approximating NP-hard functions. In this paper, we define *probabilistically checkable debate systems* (PCDS's). We prove several results about the language-recognition power of PCDS's and then use them to obtain lower bounds on the difficulty of approximating PSPACE-hard functions.

Let us describe the background for this work in more detail. Loosely speaking, a language L has a PCPS if, for every $x \in L$, there is a string π such that a probabilistic verifier V can be convinced with high probability that $x \in L$. The class PCP$(r(n), q(n))$ consists of those languages recognizable by PCPS's in which the verifier uses $O(r(n))$ coin flips and looks at $O(q(n))$ bits. It was recently shown that PCP$(\log n, 1)$ = NP (cf. [1, 2]).

Results on the power of classes PCP$(r(n), q(n))$ can be used to show that many important approximation problems are hard, unless there is some unexpected collapse of complexity classes. The first result along these lines showed that MAX-CLIQUE was difficult to approximate [8]. The seminal result of [8] has been improved several times, and it is now known that there is an ϵ such that approximating MAX-CLIQUE within a factor of n^ϵ is as difficult as solving NP-complete problems exactly [1]. Furthermore, there is a large class of natural optimization problems, those complete for the class MAX-SNP defined in [15], that do not have *polynomial-time approximation schemes* unless P

= NP; that is, for each of these problems, there is an ϵ such that approximating the optimal solution within ratio ϵ is as hard as solving NP-complete problems exactly [1]. This result on MAX-SNP shows that many well-known optimization problems are hard to approximate closely, including Traveling Salesman with Triangle Inequality, MAX-SAT, and MAX-CUT.

A PCDS is a generalization of a PCPS. In a PCDS for L, there are two computationally powerful players, 1 and 0 (called B and C at the beginning of this section) and a probabilistic polynomial-time verifier V. Players 1 and 0 play a game in which they alternate writing out strings on a debate tape π. Player 1's goal is to convince V that an input $x \in L$, and player 0's goal is to convince V that $x \notin L$. When the debate is over, V looks at x and π and decides whether $x \in L$ (player 1 wins the debate) or $x \notin L$ (player 0 wins the debate). Suppose V flips $O(r(n))$ random coins and reads $O(q(n))$ bits of π. If, under the best strategies of players 1 and 0, V's decision is correct with high probability, then we say that L is in PCD$(r(n), q(n))$.

Specifically, we say that a language L is in PCD$(r(n), q(n))$ if it has a nonadaptive PCDS with one-sided error in which players 1 and 0 write on a debate tape, and then V makes $O(r(n))$ coin flips and queries $O(q(n))$ bits based on these coin flips. By nonadaptive, we mean that the choice of bits queried by V is based solely on the input and the coin flips. By one-sided error, we mean that whenever $x \in L$, V must correctly decide that $x \in L$, no matter which sequence of $O(r(n))$ coins are flipped (assuming correct play on the part of player 1). When $x \notin L$, V is allowed to conclude incorrectly that $x \in L$ with probability at most ϵ, for some fixed $\epsilon < 1$.

With the above definition in hand, we can state our main results about the language-recognition power of PCDS's.

Theorem: PSPACE $\subseteq$ PCD$(\log n, 1)$. This result is best possible, because we can show that PCD$(\log n, q(n))$ is contained in PSPACE, for any function q.

The following is a technical building block, interesting in its own right, that is used in the proof that PSPACE = PCD$(\log n, 1)$: If $r(n) = \Omega(\log n)$, then PCD$(r(n), q(n))$ contains the same languages if the verifier reads $O(q(n))$ *rounds* of the debate as it does if the verifier reads $O(q(n))$ *bits* of the debate.

We then use our main result about the language-recognition power of PCDS's to prove lower bounds on the difficulty of approximating PSPACE-hard functions. Let MAX Q3SAT be the following natural optimization version of the canonical PSPACE-complete language QBF. Suppose $\Phi = Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \; \phi(x_1, x_2, \ldots, x_n)$ is a quantified boolean formula, with $Q_i \in \{\exists, \forall\}$, and ϕ in 3CNF. Suppose that the variables of the formula are chosen, in order of quantification, by two players 0 and 1, where player 0 chooses the universally quantified variables and player 1 chooses the existentially quantified variables. If player 1 can guarantee that k clauses of ϕ will be satisfied by the resulting assignment, regardless of what player 0 chooses, we say that k clauses of Φ are *simultaneously satisfiable*. We let MAX Q3SAT be the function that maps a quantified 3CNF formula Φ to the maximum number of simultaneously satisfiable clauses.

Theorem: There is a constant $0 < \epsilon < 1$ such that approximating MAX Q3SAT within ratio ϵ is PSPACE-hard. Thus MAX Q3SAT is as hard to approximate as it is to compute exactly.

We use reductions to prove that certain other PSPACE-hard functions are PSPACE-hard to approximate in a stronger sense. These include maximization versions of the Finite Automata Intersection problem, shown PSPACE-complete by Kozen [12], and the Generalized Geography problem, shown PSPACE-complete by Schaefer [17]. We show that there is a constant ϵ such that approximating these problems within ratio n^ϵ is PSPACE-hard.

The rest of this paper is organized as follows. We define PCDS's, and all of our other terms, precisely in Section 2. Our results on the language-recognition power of PCDS's are given in Section 3. Those on approximation of PSPACE-hard functions are given in Section 4. Section 5 contains open questions and a discussion of subsequent related results.

2 Preliminaries

We first review the definition of a PCPS. A *verifier* is a probabilistic polynomial-time Turing machine that takes as input a pair x, π, where $\pi \in \{0,1\}^*$, and either accepts or rejects. A language L has a *probabilistically checkable proof system*, or PCPS, with error probability ϵ if there is a verifier V with the following properties.

- For all x in L, there is a string π such that V accepts with probability 1 on input x, π.
- For all x not in L, on all strings π, V accepts with probability at most ϵ on input x, π.

We say that the verifier makes $q(n)$ queries if the number of bits of π read by the verifier is at most $q(n)$, when the input is of size n. PCP($r(n)$, $q(n)$) is the class of languages that have probabilistically checkable proof systems with error probability 1/3 in which the verifier uses $O(r(n))$ random bits and $O(q(n))$ queries.

We next extend this to define PCDS's. A *probabilistically checkable debate system*, or PCDS, consists of a *verifier* V and a *debate format* D. As before, the verifier is a probablistic polynomial-time Turing machine that takes as input a pair x, π, where $\pi \in \{0,1\}^*$, and outputs 1 or 0. We interpret these outputs to mean "player 1 won the debate" and "player 0 won the debate," respectively.

A debate format is a pair of functions $f(n), g(n)$. Informally, for a fixed n, a debate between two players, 0 and 1, consistent with format $f(n), g(n)$, contains $g(n)$ rounds. At round $i \geq 1$, player i mod 2 chooses a string of length $f(n)$.

For each n, corresponding to the debate format D is a *debate tree*. This is a complete binary tree of depth $f(n)g(n)$ such that, from any node, one edge is labeled 0 and the other is labeled 1. A *debate* is any string of length $f(n)g(n)$. Thus, there is a one-to-one correspondence between debates and the paths in the debate tree. Moreover, a debate is the concatenation of $g(n)$ substrings of length $f(n)$. Each substring is called a *round* of the debate, and each debate of this debate tree has $g(n)$ rounds.

Again for a fixed n, a *debate subtree* is a subtree of the debate tree of depth $f(n)g(n)$ such that each node at level i (the root is at level 0) has 1 child if i div $f(n)$ is even, and it has two children if i div $f(n)$ is odd. Informally, the debate subtree corresponds to a list of "responses" of player 1, against *all possible* "arguments" of player 0 in the debate.

A language L has a PCDS *with error probability* ϵ if there is a pair $(D = (f(n), g(n)), V)$ with the following properties.

- For all x in L, there is a debate subtree such that, for all debates π labeling a path of this subtree, V outputs 1 with probability 1 on input x, π. In this case, we say that x is accepted by (D, V).
- For all x not in L, on all debate subtrees, there exists a debate π labeling some path of the subtree such that V outputs 1 with probability at most ϵ on input x, π. In this case, we say that x is rejected by (D, V).

This definition allows "one-sided error," analogous to the type of errors that are allowed in the complexity class coRP. All our results also hold for a "zero-sided error" definition, with three possible outputs, 1, 0, and Λ, for "player 1 won," "player 0 won," and "I don't know who won," respectively. In this case, the verifier must never declare the losing player to be a winner, but it may, both in the case that $x \in L$ and in the case that $x \notin L$, say that it doesn't know who won.

As in the theory of PCPS's, we say that the verifier makes $q(n)$ queries if the number of bits of π read by the verifier is at most $q(n)$ when the input is of size n. The verifier V in a PCDS is required to be *nonadaptive*, by which we mean that the bits of π read by V depend solely on the input and the coin flips. If L has a PCDS with error probability 1/3 in which V flips $O(r(n))$ coins and reads $O(q(n))$ bits of π, we say that $L \in \mathrm{PCD}(r(n), q(n))$.

It will be convenient in later proofs to reason about a generalized class that we refer to as GPCD($r(n)$,$q(n)$). The definition of GPCD($r(n)$,$q(n)$) is the same as that of PCD($r(n)$,$q(n)$), except that the verifier of a GPCDS nonadaptively queries $O(q(n))$ *rounds* of the debate π (rather than $O(q(n))$ *bits* of the debate). Thus, in a GPCDS, there is no restriction on the number of bits queried by the verifier in each round.

Next, we give some definitions relating to approximability of PSPACE-hard functions. Let f be any real-valued function with domain $D \subseteq \{0,1\}^*$. Let A be an algorithm that, on input $x \in \{0,1\}^*$, produces an output $A(x)$. We say that A *approximates* f *within ratio* $\epsilon(n)$, $0 < \epsilon(n) < 1$, if for all $x \in D$, $\epsilon(|x|) \leq A(x)/f(x) \leq 1/\epsilon(|x|)$. If $\epsilon(n) > 1$, then "A approximates f within ratio $\epsilon(n)$" means that $1/\epsilon(|x|) \leq A(x)/f(x) \leq \epsilon(|x|)$. If algorithm A computes the function g, we also say that g approximates f within ratio ϵ.

The function f has a *polynomial-time approximation scheme*, or PTAS, if for each $\epsilon, 0 < \epsilon < 1$, there is a polynomial-time algorithm A that approximates f within ratio ϵ.

We say that a function g is *PSPACE-hard* if PSPACE $\subseteq$ P^g, i.e., if every language in PSPACE is polynomial-time reducible to g. By "approximating f within ratio $\epsilon(n)$ is PSPACE-hard," we mean that, if g approximates f within ratio $\epsilon(n)$, then g is PSPACE-hard.

Finally, we review some facts about algebraic techniques for encoding strings. We will use them to prove that PCD$(\log n, q(n))$ = GPCD$(\log n, q(n))$, which is Theorem 3.1 below. Let x be an element of $\{0,1\}^n$. The *robust encoding* $E_R(x)$ of x is an element of $\{0,1\}^{2^n}$, indexed by elements v of $\{0,1\}^n$, such that the v^{th} bit of $E_R(x)$ is $\sum_{i=1}^n v_i x_i \bmod 2$. Let l be $\lceil \log n / \log\log n \rceil$ and p be a prime in $[\log^c n,\ 2\log^c n]$, where c is a constant. Let $I = \{1, 2, \ldots \lceil \log n \rceil\} \subset Z_p$. Since $|I^l| \geq n$, we can fix an injective map from $\{0,1\}^n$ to the set of functions that map I^l to $\{0,1\}$. Regard x as one of these functions $I^l \rightarrow \{0,1\}$. There exists a unique l-variable polynomial X over Z_p, of degree at most $|I|-1$ in each variable, that agrees with x on all $\alpha \in I^l$. The *low-degree encoding* $E_P(x)$ is this function $X : Z_p^l \rightarrow Z_p$. Let $(y_1, y_2, \ldots, y_{p^l})$ denote $E_P(x)$.

The encoding E that is used in the proof of Theorem 3.1 is given by the formula

$$E(x) \equiv (E_R(y_1), E_R(y_2), \ldots, E_R(y_{p^l})).$$

Note that $|E(x)| = poly(|x|)$.

The following expression $\Delta_{E'}$ is defined for any function $E' : \Sigma^n \rightarrow \Sigma^{n'}$ and any alphabet Σ. Typically, E' will be an error-correcting code, and $\Delta_{E'}(y)$ measures the fraction of symbols of y that have to be changed in order to transform y into a codeword. Let y be an element of $\Sigma^{n'}$. Then

$$\Delta_{E'}(y) \equiv \frac{\min_{x \in \Sigma^n}(\mathrm{Ham}(y, E'(x)))}{n'},$$

where, if $y_i = \sigma_{i1} \cdots \sigma_{in'}$, $1 \leq i \leq 2$, $\sigma_{ij} \in \Sigma$, then Ham(y_1, y_2) is the number of j for which $\sigma_{1j} \neq \sigma_{2j}$.

Rubinfeld and Sudan [16] derive a *low-degree test* for programs that purport to compute polynomials. Their test is based on the following fact. Let $g : Z_p^d \mapsto Z_p$ be a function and g_{lines} be another function that, given $\alpha, \beta \in Z_p^d$, determines a polynomial

$q_{\alpha,\beta}$ of degree at most k such that $p = O((kd)^c)$, where c is some universal constant, and

$$Prob_{\alpha,\beta,t}(g(t\alpha + (1-t)\beta) \neq q_{\alpha,\beta}(t)) \leq \delta.$$

Then there exists a multivariate polynomial f of total degree at most k such that $Prob_\alpha(g(\alpha) \neq f(\alpha)) \leq 2\delta$.

3 Complexity-Theoretic Results

Our first theorem on the language-recognition power of PCDS's addresses the question of whether verifiers that read $O(q(n))$ *rounds* of the debate tape have more power than verifiers that read $O(q(n))$ *bits* of the debate tape. Surprisingly, for $r(n) = \log n$, the answer is no.

Theorem 3.1 *For every* $q(n)$*,* $\mathrm{PCD}(\log n, q(n)) = \mathrm{GPCD}(\log n, q(n))$*.*

Proof: Given a GPCDS D with players 1 and 0 and verifier V, we construct an equivalent PCDS D' with players $1'$ and $0'$ and verifier V'.

The debate system D' has one more round than D. In the rounds that correspond to rounds in D, players $1'$ and $0'$ play according to the strategies of players 1 and 0, except that they encode their moves using the error-correcting encoding E defined in Section 2.

The last round of D' is more involved. In this round, $1'$ proves to V' that, for each random string R of V, one of the following two statements holds: (1) V outputs 1 if it is given this random string and this debate tape, or (2) $0'$ is cheating. This task is complicated by the fact that $0'$ may play a move that is not encoded properly. Fix some $\epsilon > 0$. For any R, there are three cases:

1. All the moves y that $V(R)$ reads are encoded fairly well (i.e., $\Delta_E(y) \leq \epsilon$), and V declares 1 the winner.

2. All the moves that $V(R)$ reads are encoded fairly well, and V declares 0 the winner.

3. One of the moves y that $V(R)$ reads is badly encoded (i.e., $\Delta_E(y) > \epsilon$).

When player 1 wins D, case (2) cannot happen. Thus we only need to handle cases (1) and (3). In all cases, $1'$ writes down the encoding of each the moves that $0'$ played, i.e., if neither $1'$ nor $0'$ is cheating, these are the encodings of the encodings of the moves that 0 would have played.

Furthermore, $1'$ writes down a PCP$(\log n, 1)$ proof for each bit of each move that $0'$ played. The $(i,j)^{th}$ of these proofs proves that the string encoded by $1'$ has the same value as the j^{th} bit of the i^{th} move played by $0'$. These proofs enable the verifier V' to check with high probability that $1'$ properly encoded $0'$'s move. Note that all these proofs depend on only one encoded (by $1'$) string, and therefore it is possible to construct PCP$(\log n, 1)$ proofs using [1, 3].

Lastly, $1'$ provides a PCP$(\log n, q(n))$ proof that either

1. when the moves played by $1'$ and $0'$ are decoded and used as the debate tape in D, $V(R)$ outputs 1, or

3. at least one move by $0'$ is badly encoded. We argue below that this can be proven in NP.

The inputs to the above statement are $O(q)$ moves by $1'$, corresponding to the $O(q)$ moves of player 1 read by $V(R)$, and also the last move of player $1'$, which contains the encoding of the moves of player $0'$. Observe that all the moves have been encoded by $1'$. Since the above statement is in NP (in fact it is in P), $1'$ can use [1, 3] to write down the PCP$(\log n, q(n))$ proof.

The verifier V' chooses a random string R for V. Then V' probabilistically checks that all the moves that correspond to moves of 0 that $V(R)$ reads have been properly encoded by $1'$. After this verification, V' is convinced with high probability that, if y is the string played by $0'$ and y' is the string that $1'$ claims is the string played by $0'$, then $\Delta_E(y, y') < \epsilon/48$.

Furthermore V' probabilistically checks the proof that $V(R)$ outputs 1 or that some move of $0'$ was badly encoded.

It remains to be shown that the set of badly encoded moves is in NP. More precisely, we show that there exists a polynomial-time computable function F and an $\epsilon > 0$ such that

- for every y, $\Delta_E(y) \le \epsilon/48 \Rightarrow F(y) = 1$, and
- for every y, $\Delta_E(y) > \epsilon \Rightarrow F(y) = 0$.

Thus F determines whether y is an approximately well encoded string or a badly encoded string.

On input $y = (y_1, \ldots, y_{p^l})$, F first decodes each field element (i.e., for each i, finds the z_i that minimizes $\mathrm{Ham}(y_i, E_R(z_i))$). If $\mathrm{Ham}(y, (E_R(z_1), \ldots, E_R(z_{p^l}))) > \epsilon|y|/2$, then F outputs 0. Otherwise, y determines a function $Y : Z_p^l \to Z_p$. For each line (i.e., a pair $\alpha, \beta \in Z_p^l$), F computes whether a low-degree polynomial will interpolate 2/3 of the points on the line, using the method in [5]. If so, F finds this polynomial, say $q'_{\alpha,\beta}$. F now computes an approximation (within a factor of 3) of the probability Q that the low-degree test of [16] described in Section 2 rejects:

$$Q = Prob_{\alpha,\beta,t}[Y(t\alpha + (1-t)\beta) \neq q_{\alpha,\beta}(t)].$$

For every α, β, F computes

$$u_{\alpha,\beta} := \begin{cases} Prob_t[Y(t\alpha + (1-t)\beta) \neq q'_{\alpha,\beta}(t)] & \text{if } q'_{\alpha,\beta} \text{ exists} \\ 1/3 & \text{otherwise.} \end{cases}$$

Then F lets Q' be the average of all the $u_{\alpha,\beta}$. Note that $Q/3 \le Q' \le Q$.

If $\Delta_E(y) > \epsilon$, then $\Delta_{E_P}(y) > \epsilon/2$. From the low-degree test, we then know that $Q > \epsilon/4$. Hence, if $\Delta_E(y) > \epsilon$, then $Q' > \epsilon/12$.

On the other hand, when $\Delta_E(y) \le \epsilon/48$, then $\Delta_{E_P}(y) \le \epsilon/24$ and that implies that $Q \le \epsilon/12$. Thus $Q' \le \epsilon/12$.

The above argument shows that F can reject when $Q' > \epsilon/12$.

Let us show that the debate (D', V') is a PCDS. First assume that x belongs to L. Then there exists a strategy for 1 such that $V(R)$ accepts for all R chosen by V. This gives a strategy for $1'$ in D'. It follows from the definitions that V' accepts with probability 1.

If $x \notin L$, then 0 has a strategy such that V rejects with high probability. Thus $1'$ has to cheat for every R such that $V(R)$ rejects in order for V' to accept. First note that all $0'$-moves are encoded properly; therefore, if $1'$ tries to change one $0'$-move to make $F(y') = 0$, V' will discover this with high probability. Hence the conclusion follows from the properties of PCPS's. ∎

If $r(n) = o(\log n)$, then we cannot conclude that GPCD($r(n)$,$q(n)$) $\subseteq$ PCD($r(n)$,$q(n)$). For example, it is clear that GPCD(0,1) is the entire polynomial-time hierarchy, whereas PCD(0,1) is just P.

Let $\Phi = \exists x_1 \forall x_2 \ldots \exists x_n \phi(x_1, \ldots, x_n)$ be an instance of QBF; without loss of generality, we assume that quantifiers alternate strictly. The assignment tree A for Φ is the complete binary tree of depth n, where one edge from every internal node is labeled "true" and the other "false." Each path P in the tree corresponds to an assignment of the variables; we say P satisfies ϕ if this assignment satisfies ϕ. Call edges at odd-numbered levels (that is, corresponding to existentially quantified variables) 1-edges and edges at even-numbered levels 0-edges. An $\exists$-assignment subtree A_1 is a subtree of A that has two 0-edges from each node at each even level and one 1-edge from each node at each odd level. Similarly, a $\forall$-assignment subtree A_0 is a subtree of A that has two 1-edges from each node at each odd level and one 0-edge from each node at each even level. An $\exists$-assignment subtree A_1 is *optimal* if it maximizes (over all $\exists$-assignment subtrees) the number of paths that satisfy ϕ. Similarly, a $\forall$-assignment subtree A_0 is optimal if it maximizes (over all $\forall$-assignment subtrees) the number of paths that do not satisfy ϕ. Note that if $\Phi \in$ QBF, all paths of an optimal $\exists$-assignment subtree satisfy ϕ, whereas if $\Phi \notin$ QBF, then no path of an optimal $\forall$-assignment subtree satisfies ϕ.

Theorem 3.2 PSPACE = GPCD$(\log n, 1)$.

Proof: The direction GPCD($\log n$,1) $\subseteq$ PSPACE is straightforward. To prove the other direction, we show that QBF $\in$ GPCD($\log n$,1). Let $\Phi = \exists x_1 \forall x_2 \ldots \exists x_n \phi(x_1, \ldots, x_n)$ be an instance of QBF in which quantifiers alternate strictly. Let A_0, A_1 be optimal $\forall$- and $\exists$- assignment subtrees of Φ, respectively. Note that Φ is a true QBF if and only if the unique path P of length n that is in both A_0 and A_1 satisfies ϕ.

We give a debate format and a protocol of players 0 and 1 that enable the players to record parts of the assignment subtrees A_0 and A_1 in such as way that a verifier can efficiently check whether or not path P satisfies ϕ.

In the debate, the players alternately play rounds, starting with player 1. Roughly, in one round, player i does two things: presents a challenge to player $1-i$ and responds to previous challenges written by player $1-i$. A *challenge* by player $1-i$ to player i is simply a path of the assignment subtree A_i that ends in a $(1-i)$-edge. The *response* of player i to this challenge is the edge that extends this path in A_i (if the path is not already of length n).

Note that a challenge *to* player 1 (resp. 0) has to be a path of A_1 (resp. A_0). How can player 0 write down such a path *without knowing* A_1? Essentially, in round t, player 0 must present paths that are consistent with what player 1 wrote in rounds 1 through $t-1$. We will elaborate on this point below.

Play proceeds as follows: In round t, one of the players writes a tree T_t. If player i is honest, then in round $t \equiv i \bmod 2$, player i writes the (unique) smallest subtree T_t of A_i such that T_t contains all challenges by player $1-i$ at rounds $j < t$. The debate format is thus $(f(n), g(n))$, where $g(n)$ is chosen to make the error probability small enough and $f(n)$ is chosen to allow encoding of binary trees of the appropriate form; we will explain in detail how to choose f and g when we prove correctness of the protocol.

Before specifying the algorithm of the verifier V, we give some examples of debates in which player 1 is honest. The values assigned to the variables depend on the input formula and are unimportant for the purposes of this discussion.

Example 1: Suppose that player 0 is honest as well.

In round t, $1 \leq t \leq n$, player $i \equiv t \bmod 2$ assigns a value to x_t by writing down a path of length t that extends the path written in round $t-1$. The path written in round n is P, the intersection of A_1 and A_0. In rounds $n+1$ through g, the path P is repeated in each round. V will declare player 1 the winner (i.e., accept the input) if and only if P satisfies ϕ. (See Figure 1.)

Example 2: Suppose that Φ is a true QBF and that player 0 cheats in an effort to convince V to reject. One way player 0 may do this is to play a move that is not a subtree of A_1 – that is, to lie about player 1's previous moves. (See Figure 2.)

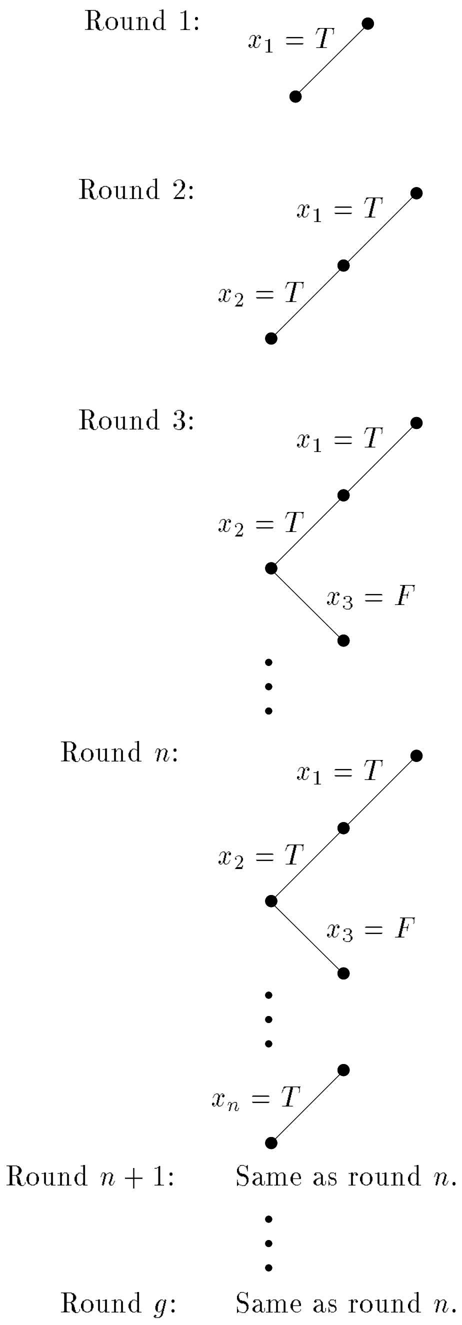

Figure 1: Player 0 is honest

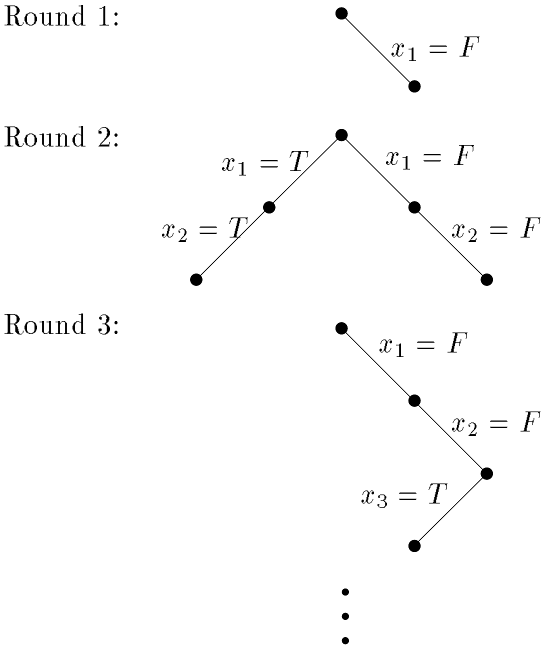

Figure 2: Player 0 lies about some move of player 1.

Note that, in round 3, the honest player 1 need only extend the FF path of the tree played by 0 in round 2. Because the TT path that appears in round 2 is not in A_1, i.e., because it is not consistent with player 1's move in round 1, it does not satisfy the definition of a challenge.

Example 3: Once again, suppose that Φ is true and that player 0 cheats. This time, player 0 does so by lying about player 0's own moves rather than those of player 1. (See Figure 3.)

In this example, both the FF path of round 4 and the FT path of round 2 require a response by player 1 in round 5, because both are legitimate challenges, i.e., neither is inconsistent with player 1's previous moves.

A move T_t by an honest player i must have the following properties: (i) at most one edge from every node is an i-edge (because T_t is a subtree of A_i) and (ii) the edge to every leaf of depth $< n+1$ is an i-edge (because player i responds to any recorded challenge by player $1-i$). A tree T_t satisfying these two properties is *valid*. (In Example 3 above, player 0 should choose which of the paths FTT or FFF of round 5 to extend in round 6, because extending both of them would result in an invalid move T_6.) Also, we define a

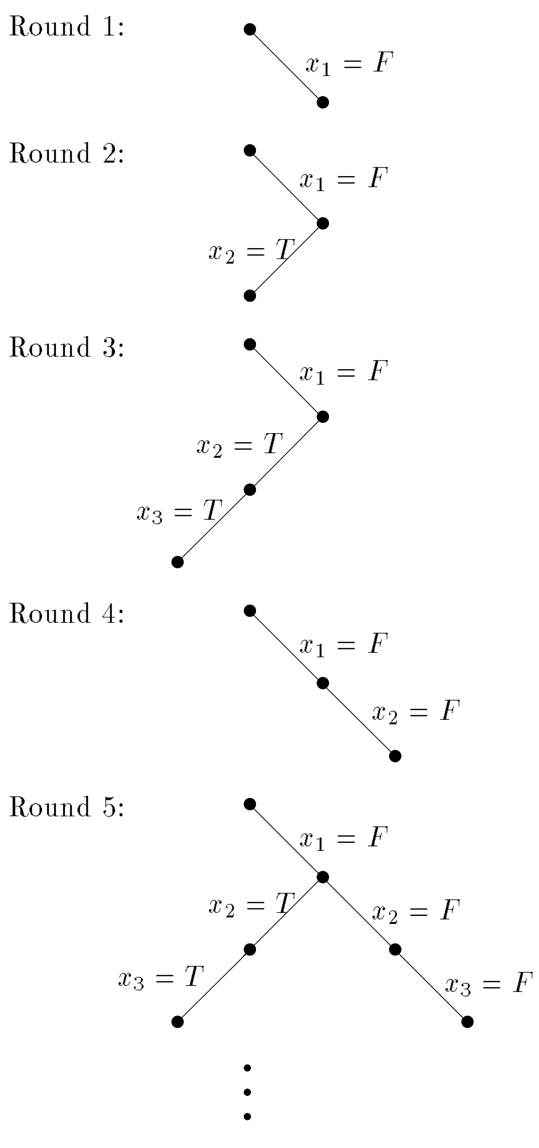

Figure 3: Player 0 lies about its own previous move.

valid challenge by player $1-i$ to player i at round j to be the (unique) longest challenge that lies in T_j, if T_j is valid.

Note that if player i is honest, then T_{t-2} is a subtree of T_t. This is because both T_{t-2} and T_t respond to all valid challenges from rounds $j < t-2$. Also, T_t has at most one more leaf than T_{t-2}, namely the leaf of the path that contains the valid challenge at round $t-1$, if it lies on a different path from previous valid challenges.

We take the number of rounds to be $g = g(n) > 4n$; this will ensure that the error probability is no more than $2(n-1)/(g-3)$, as explained below. We let $f(n)$, the length of a round, be such that any binary tree of depth at most n with at most $g(n)$ leaves can be described using $f(n)$ bits, via some simple encoding of trees to strings. This is sufficient, since the number of leaves of $T_{g(n)}$ is at most $g(n)$.

Below we state V's algorithm formally and prove it correct, but first we explain intuitively how V can catch a cheating player by examining only a constant number of rounds of the debate. Suppose that the input formula is true and hence that player 1 follows the protocol honestly. If they are valid, the trees T_g (player 1's last move) and T_{g-1} (player 0's last move) intersect in a unique path, say $P(g)$. If it is of length n, then $P(g)$ satisfies ϕ, and V will certainly declare player 1 the winner. Thus player 0 must try to "stall" and prevent $P(g)$ from growing to length n. Consider a move T_t, where $t < g-1$ is even. T_t contains a (unique) longest prefix of $P(g)$, say $P(t)$. Informally, we say that player 0 "cooperates" if $|P(t)| > |P(t-1)|$. Player 0 can cooperate in fewer than n rounds, or else $P(g)$ is of length n. Essentially, our formal proof shows that, if player 0 indeed cooperates in fewer than n rounds, then many intermediate moves T_t are either invalid or not subtrees of the final move T_{g-1}. Either way, V can detect that player 0 is cheating by examining only a constant number of randomly chosen moves.

We now give a formal statement of V's algorithm and a proof of its correctness. V first examines the last subtrees, T_{g-1} and T_g, written by players 0 and 1 respectively. If T_{g-1} is not valid, V accepts. If T_g is not valid, V rejects. Otherwise, let $P(g)$ be the (unique) path in both T_{g-1} and T_g. If the length of $P(g)$ is n and $P(g)$ satisfies ϕ, V accepts. If the length of $P(g)$ is n and $P(g)$ does not satisfy ϕ, then V rejects.

Otherwise, the length of $P(g)$ is less than n. In this case, V chooses a random round $t, 1 < t < g$, in which player 1 plays, and examines rounds t and $t-1$. If T_t is not valid or is not a subtree of T_g, V rejects. Similarly, if T_{t-1} is not valid or is not a subtree of T_{g-1}, V accepts. Otherwise, let P' be the longest path in both T_{t-1} and T_t. If the last edge of P' is not a 0-edge, then V rejects. Otherwise, V accepts. This completes the statement of V's algorithm.

Now, suppose that $\Phi \in$ QBF and that player 1 is honest. We show that V accepts with probability 1. This is true if the length of $P(g)$ is n, since in this case $P(g)$ is a path in A_1 and hence satisfies ϕ. If the length of $P(g)$ is less than n, then for all rounds t in which player 1 plays, T_t is valid and a subtree of T_g. Suppose that T_{t-1} is also valid and a subtree of T_{g-1}. It remains to show that the longest path P' in both T_{t-1} and T_t must end in a 0-edge.

First, note that the length of P' must be $< n$; otherwise P' is contained in both T_{g-1} and T_g and therefore $P' = P(g)$, which contradicts our assumption that the length of $P(g)$ is $< n$. Also, since T_{t-1} is valid, all paths of T_{t-1} of length $< n$ that end in a 1-edge are followed by a 0-edge. Furthermore, if P' ends in a 1-edge, the 0-edge following P' in T_{t-1} must also be in T_t, since the path formed by P' and this 0-edge is a prefix of the valid challenge of player 0 to player 1 at round $t-1$, and player 1 is honest. This contradicts the fact that P' is the longest path in both T_{t-1} and T_t. Hence P' must end in a 0-edge.

We next show that if $\Phi \notin$ QBF and player 0 is honest, then V accepts with probability at most $2(n-1)/(g-3)$. If the length of $P(g)$ is n, V rejects, since in this case $P(g)$ is a path in A_0 and hence does not satisfy ϕ. Hence suppose that the length of $P(g)$ is less than n; thus player 1 cannot be honest. We claim that in this case, there can only be $n-1$ values of t such that V accepts when rounds t and $t-1$ are examined. Since V chooses t randomly and uniformly from $(g-3)/2$ choices, the error probability is at most $2(n-1)/(g-3)$. The number of choices for t is $(g-3)/2$, because V never chooses player 1's first move, since it is not preceded by a move of player 0.

If T_t is valid, let $p(t)$ be the length of $P(t)$, where $P(t)$ is the longest prefix of $P(g)$ in T_t. For any t such that player 1 plays in round t, we show that, if V accepts on examining rounds t and $t-1$, then $p(t) > p(j)$ for all $j < t$ such that T_j is valid. This implies the claim, since then V accepts only the first round for which $p(i) = 1, p(i) = 2, \ldots, p(i) = n-1$, and there are at most $n-1$ such rounds.

Suppose, then, that V accepts on examining rounds t and $t-1$. Suppose that $j < t$ is a round of player 1, where T_j is valid. We need to show $p(j) < p(t)$. We first show that $P(j)$ is a prefix of $P(t-1)$, which implies that $p(j) \leq p(t-1)$. This is because $P(j)$, with the last edge removed if it is a 0-edge, is the prefix of a valid challenge of player 1 to player 0 at round j, and since player 0 is honest, player 0 responds to this challenge at round $t-1$. To complete the proof, we show that $p(t-1) < p(t)$. Note that it must be the case that P', the longest path in both T_{t-1} and T_t, ends in a 0-edge, since V accepts. Also, this path must be a prefix of $P(t-1)$ and $P(t)$. In fact, this path must equal $P(t-1)$. (Otherwise, the 1-edge following P' in $P(t-1)$ must be in T_g. Since T_t is a subtree of T_g, this 1-edge must be the 1-edge of T_t following P', contradicting the fact that P' is the longest path in both T_{t-1} and T_t.) Finally, the 1-edge following P' in T_t must be in T_g (since V checks for this) and also in T_{g-1} (since player 0 is honest). Thus $P(t)$ contains $P(t-1)$ as a prefix, and it also contains an additional 1-edge. This implies that $p(t) > p(t-1) \geq p(j)$, as required. ∎

Corollary 3.3 PSPACE = PCD($\log n, 1$).

4 Approximating PSPACE-Hard Functions

In this section, we give many examples of PSPACE-hard functions that are hard to approximate. We consider maximization versions of the problem of deciding whether a quantified Boolean formula is true and show that one version can be approximated within ratio $1/2$, yet there is some constant $\epsilon < 1/2$ such that approximating the function within ratio ϵ is PSPACE-hard. We prove even stronger results for the maximization versions of several other PSPACE-complete problems. For example, there is a constant $\epsilon > 0$ such that approximating Finite Automata Intersection (cf. Kozen [12]) and Generalized Geography (cf. Schaefer [17]) within ratio n^ϵ is PSPACE-hard.

We first consider variants of a well-known PSPACE-complete problem, that of deciding whether a quantified Boolean formula is true. In what follows, we consider quantified (Boolean) formulas in CNF (conjunctive normal form); that is, quantified formulas of the form

$$\Phi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi(x_1, x_2, \ldots, x_n),$$

where each $Q_i \in \{\exists, \forall\}$, each x_i is a Boolean variable, and ϕ is in conjunctive normal form. If each clause of ϕ has exactly 3 literals, we say the quantified formula is in 3CNF. Let QSAT and Q3SAT be the sets of true quantified formulas in CNF and 3CNF, respectively.

Suppose that the variables of the formula are chosen, in order of quantification, by two players 0 and 1, where player 0 chooses the universally quantified variables and player 1 chooses the existentially quantified variables. If player 1 can guarantee that k clauses of ϕ will be satisfied by the resulting assignment, regardless of what player 0 chooses, we say that k clauses of Φ are *simultaneously satisfiable.* We let MAX QSAT (resp. MAX Q3SAT) be the function whose domain is the set of quantified formulas that maps a quantified formula (resp. quantified 3CNF formula) Φ to the maximum number of simultaneously satisfiable clauses.

Theorem 4.1 *There is a constant* $0 < \epsilon < 1$ *such that approximating* MAX Q3SAT *within ratio* ϵ *is* PSPACE-*hard.*

Proof: Let L be a language in PSPACE. We showed in Section 3 that L is in PCD($\log n$, 1). We reduce the problem of deciding whether a string x is in L to the problem of approximating the number of simultaneously satisfiable assignments of a quantified 3CNF formula.

Let (D, V) be a PCDS for L, where V is polynomial-time bounded and uses $r(n) = O(\log n)$ random bits and $O(1)$ queries. Let $D = (f(n), g(n))$. Without loss of generality, we can assume that $f(n)$ and $g(n)$ are polynomials.

Given an instance x of L, say of length n, we construct a quantified formula from (D, V) as follows. There are $f(n)g(n)$ ordered variables, one for each bit of a debate corresponding to the debate format. The first $f(n)$ variables, which correspond to the first round of a debate, are existentially quantified, the next $f(n)$ variables, which correspond to the second round, are universally quantified, and so on.

For each sequence of random bits R of length $r(n)$, there is a subformula with $s = O(1)$ clauses, with variables corresponding to the bits of a debate that are queried on random sequence R. The subformula is satisfied by a truth assignment to the variables if and only if the verifier outputs 1, when the query bits are as in the truth assignment.

If x is accepted by (D, V), then there exists a debate subtree such that, on each debate (or path of the tree), V outputs 1 on all of the random strings. Thus, player 1 can choose the values of the existential variables so that all clauses of the subformulas are simultaneously satisfiable.

If the input x is not accepted by (D, V), then in any debate subtree, there is a debate on which V outputs 1 on at most 1/3 of the random strings. Thus, no matter what truth assignment player 1 chooses for the existential variables, there is a choice for the universal variables such that at most 1/3 of the subformulas are satisfied. Hence, at most 1/3 of the subformulas are simultaneously satisfiable. Since each subformula contains $O(1)$ clauses, it follows that at most a constant fraction < 1 of the clauses are simultaneously satisfiable. ∎

Let MAX-Q-Formula be the function whose domain is the set of quantified formulas in which the "clauses" are general formulas instead of being in CNF. The above result can be extended to prove the following.

Theorem 4.2 *There is a constant $\epsilon > 0$ such that approximating* MAX-Q-FORMULA *within ratio n^ϵ is* PSPACE*-hard.*

We next consider a variant of Q3SAT called *Balanced Q3SAT*. We say a quantified formula is *balanced* if every clause of the formula contains some existentially quantified variable. Balanced Q3SAT consists of those true quantified formulas in 3CNF form that are also balanced. This language is easily seen to be PSPACE-complete, by the following reduction from Q3SAT. An instance

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n \phi(x_1, x_2, \dots, x_n)$$

of MAX QSAT, where ϕ has m clauses, is mapped to the instance

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n \exists w_1 \dots \exists w_m \phi'(x_1, x_2, \dots, x_n, w_1, \dots, w_m)$$

where ϕ' is obtained from ϕ by replacing each clause $C_j = (l_1 \vee l_2 \vee l_3)$ of ϕ by the two clauses $(w_j \vee l_1 \vee l_2)$ and $(\bar{w}_j \vee l_3 \vee l_3)$, $1 \leq j \leq m$.

We define the corresponding function MAX Balanced Q3SAT to be the function MAX Q3SAT, restricted to the domain of balanced quantified formulas. This provides an example of a function that *can* be approximated to within some constant ratio but *cannot* be approximated to within an arbitrary constant ratio, unless PSPACE = P.

Lemma 4.3 *There is a polynomial-time algorithm that approximates* MAX Balanced Q3SAT *within ratio* 1/2.

Proof: Let $\Phi = Q_1 x_1 \ldots Q_n x_n \phi(x_1, \ldots, x_n)$ be a balanced quantified formula in 3CNF. Let ϕ' be the formula obtained from ϕ by eliminating all universally quantified variables. Since ϕ is balanced, note that the number of clauses of ϕ' is equal to the number of clauses of ϕ (however, clauses may now have only one literal).

Johnson [11] showed that a truth assignment to the variables of ϕ' that satisfies at least 1/2 of the clauses can be found in polynomial time. Player 1 can use this assignment to ensure that at least 1/2 of the clauses of ϕ are satisfied, no matter what the values of the universally quantified variables are. ▮

Lemma 4.4 *There is a constant* $0 < \epsilon < 1$ *such that approximating* MAX Balanced Q3SAT *within ratio* ϵ *is* PSPACE*-hard.*

Proof: In Theorem 4.1, we reduced the problem of deciding whether an input x is accepted by a PCDS (D, V) to the problem of approximating a quantified formula Φ. Φ can be converted into a balanced subformula as described in the discussion preceding Lemma 4.3. It is straightforward to show that the resulting balanced quantified formula Φ' also has the property that, if x is accepted by (D, V), then all clauses of Φ' are simultaneously satisfiable, but if x is not accepted, at most a constant fraction < 1 are simultaneously satisfiable. ▮

We next consider two problems from automata theory. Let *FA-INT* be the set of sequences $A_1, A_2, \ldots, A_m$ of deterministic finite-state automata having the same input alphabet Σ such that there exists a string w that is accepted by all the automata. Kozen [12] showed the problem to be PSPACE-complete.

The function MAX FA-INT has as its domain the set of all sequences $A_1, A_2, \ldots, A_m$ of deterministic finite state automata having the same input alphabet Σ and maps such a sequence to the largest number k such that there exists a string w that is accepted by k

of the automata. A related function is MAX Set FA-INT. An instance of the domain of this function is a sequence of *sets* of deterministic finite state automata $\{A_{11}, \ldots, A_{1l}\}$, $\{A_{21}, \ldots, A_{2l}\}$, ..., $\{A_{m1}, \ldots, A_{ml}\}$ in which all the automata in all of the sets have the same input alphabet. This sequence is mapped to the largest number k such that there exists a string w that is accepted by *all* automata in k sets of the sequence. We first prove a non-approximability result for MAX Set FA-INT and then extend our proof to MAX FA-INT.

Theorem 4.5 *There is a constant $\epsilon > 0$ such that approximating* MAX Set FA-INT *within ratio n^ϵ is* PSPACE-*hard.*

Proof: We describe a reduction from a new variant of MAX Q3SAT. The function MAX FIX-QSAT differs from MAX Q3SAT in two ways. First, the domain is the set of quantified formulas, where the "clauses" are now the conjunction of $O(\log n)$ "subclauses", each the disjunction of 3 literals. Second, given an instance of this domain, the function outputs the maximum size k of a set of clauses that player 1 can guarantee will be satisfied, regardless of what assignment player 0 chooses for the universal variables. Thus for MAX FIX QSAT, the set of k satisfied clauses must be fixed in advance, that is, the set must be the same for all assignments of player 0. However, for MAX Q3SAT, the set of k simultaneously satisfied clauses may depend on the assignments of player 0. The proof of Theorem 4.1 can be extended to show that there is some constant $\epsilon > 0$ such that approximating MAX FIX-QSAT to within ratio n^ϵ is PSPACE-hard.

We now describe our reduction from MAX FIX-QSAT to MAX Set FA-INT such that an instance of MAX FIX-QSAT has a set of k clauses that player 1 can guarantee will be satisfied, if and only if the instance of MAX Set FA-INT has k sets of automata such that all automata in the sets accept the same string. Let $\Phi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi(x_1, x_2, \ldots, x_n)$ be an instance of MAX FIX-QSAT, where ϕ has m clauses.

We first describe a set *Valid* of strings w; roughly, each string in this set describes possible choices of player 1 for the existentially quantified variables, against all possible choices of player 0 for the universally quantified variables. We will later construct m sets of automata, one set per clause, such that all automata in each of k sets accept some string w, which happens to lie in the set Valid, if and only if the quantified formula Φ

has k simultaneously satisfiable clauses.

Each string w in the set Valid is of the form $\$w_1\$w_2\$\ldots\$w_N\$$, where each $w_i = w_{i1}w_{i2}\ldots w_{in}$ is a binary string of length n, corresponding to a truth assignment to the variables of ϕ, and $N = 2^u$, where u is the number of universally quantified variables in Φ. Moreover, w must have properties 1, 2 and 3 below.

Roughly, these properties are necessary and sufficient to ensure that w is in Valid if and only if the strings w_i correspond to paths of an $\exists$-assignment subtree that describes the assignments of player 1 against all possible assignments of player 0 (as described in the paragraph preceding Theorem 3.2). Note that such a tree has N leaves. Moreover, these paths are enumerated in order from left to right of the assignment subtree. We now list the three properties.

1. Suppose that x_j is universally quantified. Then, $w_{1j} = 0$ and $w_{Nj} = 1$. Moreover, if w_i is such that $w_{ij} = 1$ for all j such that x_j is universally quantified, then $i = N$.
2. Suppose that x_j is universally quantified and $i > 1$. Then, $w_{ij} = \bar{w}_{i-1,j}$ if for all $j' > j$ such that $x_{j'}$ is universally quantified, $w_{i-1,j'} = 1$, and $w_{ij} = w_{i-1,j}$ otherwise.
3. Suppose that x_j is existentially quantified and $i > 1$. Then, $w_{ij} = w_{i-1,j}$, unless for all $j' > j$ such that $x_{j'}$ is universally quantified, $w_{i-1,j'} = 1$. In the latter case, there is no restriction on w_{ij}.

We say w is *not valid* at index j, if property (2) or (3) fails for j.

There are two types of automata: "syntax checking" automata and "clause checking" automata. We will later group these automata into m sets. For $1 \leq j \leq n$, automaton S_j checks that the jth bit of each w_i has the property (2) or (3) above, depending on whether j is universally or existentially quantified. S_{n+1} checks that the \$'s are separated by strings of length exactly n and that property (1) above holds. Clearly, a string is accepted by all $n+1$ of the automata if and only if it is in the class Valid. Moreover, each of these automata can be constructed to have $poly(n)$ states.

There are m clause checking automata $C_1, \ldots, C_m$, each with $poly(n)$ states, such that a string $w \in$ Valid is accepted by C_j if and only if on all paths of the corresponding assignment subtree, the j^{th} clause is satisfied.

Finally, we let the j^{th} set of automata be $\{S_1, \dots S_{n+1}, C_j\}$, $1 \leq j \leq m$. ∎

We next extend the construction of Theorem 4.5 to the problem MAX FA-INT.

Theorem 4.6 *There is a constant $\epsilon > 0$ such that approximating* MAX FA-INT *within ratio n^ϵ is* PSPACE*-hard.*

Proof: The proof is similar to that of Theorem 4.5. Let Φ be of the form described that proof. Moreover, assume without loss of generality that each variable appears in some clause.

We now construct m automata. The i^{th} automaton A_i does the following checks on its input string. We refer to the automata constructed in Theorem 4.6.

1. Perform the check done by automaton S_{n+1}.
2. If x_j or $\bar{x}_j$ is a literal of C_i, then perform the check of S_j. (In this case, say that A_i *examines* bit x_j.)
3. Perform the check done by automaton C_i.

Check 2 can be done by an automaton with $poly(n)$ states. Roughly, for each i, the automaton stores the the bits $w_{i-1,j}$ and $w_{i,j}$, where bit x_j is examined by C_i. Also, the position of the rightmost universal index with value 0 in w_i is stored. This information is sufficient to perform the check of S_j on the string w_i. Since A_i performs the checks of a three automata of size $poly(n)$, A_i is also of size $poly(n)$.

As in Theorem 4.5, if player 1 can guarantee that a fixed set of k clauses of ϕ are satisfied, for all assignments of player 0, then there is a corresponding string w that is accepted by k automata.

Conversely, suppose that $k > 0$ automata all accept some string w. Note that w may not be a member of Valid. However, w must pass check (1), because $k > 0$. Hence, suppose that w is not valid at I indices. We prove by induction on I that there exists a string w' in Valid such k automata accept w'. From this it follows that there is a set of k clauses of Φ that player 1 can guarantee will be satisfied.

The base case, when $I = 0$, is immediate, for in that case $w = w'$. Suppose $I > 0$, and let j be the smallest invalid index. In what follows, if x_j is existentially (universally) quantified, we say that j is an existential (universal) index.

If j is an existential index, we define w'_i as follows for $1 \leq i \leq N$: Let $w'_{ij} = 0$ and let $w'_{is} = w_{is}$ for $s \neq j$. The resulting string has $I - 1$ invalid indices. Furthermore, it is still accepted by k automata. (This is because none of the k accepting automata can possibly examine bit j.) We can now apply induction to complete the proof.

Next, suppose that j is a universal index. The procedure to construct w' from w is more complicated in this case. We do it in stages. For each set of bits $b_1 \ldots b_{j-1}$, we consider separately the (unique) longest contiguous substring $w_r\$ \ldots \$w_{r'}\$$ of w (if any) such that $b_1 \ldots b_{j-1}$ is a prefix of each w_i, $r \leq i \leq r'$. Call this substring $w[b_1 \ldots b_{j-1}]$. In the next paragraph, we describe a new substring $w'[b_1 \ldots b_{j-1}]$ that is obtained from $w[b_1 \ldots b_{j-1}]$. We then let w' be the string obtained by concatenating the substrings $w'[b_1 \ldots b_{j-1}]$ in the appropriate order. Our construction ensures that w' has $I-1$ invalid indices and is accepted by all k automata that accept w.

Therefore, fix $b_1 \ldots b_{j-1}$, and consider only the substring $w_r\$ \ldots \$w_{r'}\$$. Note that, for all universal indices $j' > j$, $w_{rj'} = 0$ if j' is valid, and $w_{r'j'} = 1$. (If this were not the case, it would contradict the facts that $1 \ldots j-1$ are valid indices and that $w_r\$ \ldots \$w_{r'}\$$ is the longest substring of w such that $b_1 \ldots b_{j-1}$ is a prefix of each w_i.) Let l be the smallest number such that, for all universal indices $j' > j$, $w_{lj'} = 1$.

Let $w'_r\$ \ldots \$w'_l\$$ be such that, for $1 \leq i \leq l$, $w'_{ij} = 0$, and, for $s \neq j$, $w'_{is} = w_{is}$. Similarly, let $w''_r\$ \ldots \$w''_l\$$ be such that, for $1 \leq i \leq l$, $w''_{ij} = 1$ and, for $s \neq j$, $w''_{is} = w_{is}$.

Then, the new string $w'[b_1 \ldots b_{j-1}]$ is $w'_r\$ \ldots \$w'_l\$w''_r\$ \ldots \$w''_l\$$.

This completes the description of w'. The construction guarantees that (i) w' has $I-1$ invalid indices (namely, those invalid indices $j' > j$ of w) and (ii) w' is accepted by the k automata that accept w. Again, induction can be applied to complete the proof. ∎

Generalized Geography is an abstraction of a popular car game in which two players alternately list the names of countries, each beginning with the last letter of the previous country, until one player cannot list a new country. A corresponding game can be played on a directed graph G that has a distinguished start node s. A marker is initially placed on s, and two players, 0 and 1, alternately move it along an edge, with the constraint that player 1 starts and each edge can be used only once. The first player unable to move loses. Schaefer [17] defines GGEOG to be the set of pairs (G, s) such that player 1 has a winning strategy.

Informally, a natural optimization version of GGEOG is to compute how long player 1 can keep the game going, even if player 1 does not eventually win the game. We say (G, s) can be played for k rounds if player 1 has a strategy that causes the marker to move along k edges of the graph before the game ends. We define MAX GGEOG to be the function whose domain is the set of pairs (G, s), where G is a directed graph with node s, that maps a pair (G, s) to the maximum number k of rounds that (G, s) can be played.

Theorem 4.7 *There is a constant $\epsilon > 0$ such that approximating* MAX GGEOG *within ratio n^ϵ is* PSPACE-*hard.*

Proof: We first modify Schaefer's reduction [17] to obtain a reduction from MAX Q3SAT to MAX GGEOG. We later describe a simple modification of our construction, to obtain a reduction from MAX Q-Formula to MAX GGEOG.

Let $\Phi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi(x_1, \ldots, x_n)$ be an instance of MAX 3QSAT, where ϕ contains m clauses. In what follows, we assume that n is even and that $Q_n = \forall$; the reduction can easily be modified to handle the other cases.

We construct from Φ an instance (G, s) of MAX GGEOG, such that, if a maximum of k clauses of Φ are simultaneously satisfiable, then (G, s) can be played for a maximum of $4n + kn^2 + O(1)$ steps. From this property, it follows that, given an approximate value for the length of the generalized geography game, an approximate value for the number of satisfiable clauses can be deduced. The graph G is composed of a "variable-setting" component, a "clause-testing" component and a "line." We describe each of these components in turn and also describe how they are interconnected.

We first describe the variable-setting component. The node set V_1 is

$$\{x_i, \bar{x}_i, u_i, v_i \mid Q_i = \exists, 1 \le i \le n\} \cup \{x_i, \bar{x}_i, u_i, v_i, w_i, \bar{w}_i, z_i \mid Q_i = \forall, 1 \le i \le n\} \cup \{u_{n+1}\}.$$

Node u_1 is the start node s. The nodes $x_i, \bar{x}_i, 1 \leq i \leq n$ are referred to below as "literal" nodes. The edge set is

$$E_1 = \{(u_i, x_i), (u_i, \bar{x}_i), (x_i, v_i), (\bar{x}_i, v_i), (v_i, u_{i+1}) \mid Q_i = \exists, 1 \leq i \leq n\} \cup$$
$$\{(u_i, w_i), (u_i, \bar{w}_i), (w_i, x_i), (\bar{w}_i, \bar{x}_i)(x_i, v_i), (\bar{x}_i, v_i), (v_i, z_i), (z_i, u_{i+1}) \mid Q_i = \forall, 1 \leq i \leq n\}.$$

Thus, the variable-setting component consists of n "diamond" shaped gadgets that are strung together. The construction ensures that, for each i, the choice of whether to follow the path through x_i or $\bar{x}_i$ is made by player 0 if x_i is a universally quantified variable and by player 1 if x_i is an existentially quantified variable. Informally, this choice determines a truth assignment to the variable x_i.

We next describe the clause-testing component. The node set is

$$V_2 = \{y_k, y'_k, y''_k \mid 1 \leq k \leq m\} \cup \{y_{kj} \mid 1 \leq k \leq m, 1 \leq j \leq n^2 - 3\}.$$

The edge set is

$$E_2 = \{(u_{n+1}, y_k) \mid 1 \leq k \leq m\} \cup$$
$$\{(y_k, y'_k), (y_k, y''_k), (y'_k, y_{k1}) \mid 1 \leq k \leq m\} \cup \{(y_{kj}, y_{kj+1}) \mid 1 \leq k \leq m, 1 \leq j \leq n^2 - 4\} \cup$$
$$\{(y_{kn^2-3}, u_{n+1}) \mid 1 \leq k \leq m\}.$$

Note that, because we are assuming that the last quantifier is $\forall$, player 1 chooses, from u_{n+1}, an edge to some y_k. Informally, node y_k corresponds to a clause C_k, which player 1 claims is true. At that point, player 0 can either move to y''_k, in which case player 0 is challenging player 1 that clause C_k is false, or y'_k, in which case player 0 is not challenging. If player 0 does not challenge, a path of length n^2 is followed, back to u_{n+1}, where this is repeated.

Other interconnections between the variable-setting component and the clause-testing component are as follows:

$$E_{12} = \{(y''_k, x_i) \mid x_i \text{ occurs unnegated in clause } k\} \cup$$
$$\{(y''_k, \bar{x}_i) \mid x_i \text{ occurs negated in clause } k\}.$$

Thus, if player 0 challenges clause C_k, player 1 chooses a literal of the clause. If the literal is false, player 0 can follow an edge of the diamond and force player 1 to lose in one more move.

We finally describe the line. The node set is $V_3 = \{l_i \mid 1 \leq i \leq n^4\}$. The edge set is $E_3 = \{(l_j, l_{j+1}) \mid 1 \leq j \leq n^4 - 1\}$. Thus, this is simply a line with $n^4 - 1$ edges.

The following edges connect each literal node of the variable-testing component to the line:

$$E_{13} = \{(x_i, l_1), (\bar{x}_i, l_1) \mid 1 \leq i \leq n\}.$$

The purpose of this line is to ensure that player 0 never challenges player 1 on a clause that is actually true. Otherwise, player 1 can force the game to end up on the line and thus take n^4 steps, and also player 0 loses.

Thus, the whole reduction gives $G = (V, E)$ where $V = V_1 \cup V_2 \cup V_3$ and $E = E_1 \cup E_2 \cup E_{12} \cup E_3 \cup E_{13}$.

This completes the reduction from MAX Q3SAT, which implies that there is a constant ϵ such that approximating MAX GGEOG within ratio ϵ is PSPACE-hard. By modifying the reduction, so that it is from MAX-Q-Formula, we improve the result to ratio n^ϵ. In this new reduction, y''_k is connected to a subgraph that simulates the kth formula, ϕ_k, in the following way. There is a node for each the operators of ϕ_k and possibly some auxiliary nodes.

If $\phi_k = \phi'_k \vee \phi''_k$, we make sure (adding a possible auxiliary node) that player 1 makes the move from the node corresponding to the $\vee$ operator. Thus, player 1 chooses the subformula ϕ'''_k such that $\phi'''_k = 1$. If $\phi_k = \phi'_k \wedge \phi''_k$ we make sure (adding a possible auxiliary node) that player 0 makes the move from the node corresponding to the $\wedge$ operator. If ϕ_k is a literal, we connect it to the diamonds, as before. The paths from the nodes y'_k to u_{n+1} are longer than before and depend on ϵ. ∎

5 Open Questions and Subsequent Related Work

Open questions include:

Question 5.1 Is there a Collapse Theorem for PCDS's? That is, is PCD(r, q) with debate format $(f(n), g(n))$ equal to PCD(r, q) with debate format $(f'(n), g(n)/2)$, for some $f' = poly(f)$? It can be shown that debate format $(poly(n), k)$ is *not* equivalent to debate format $(poly(n), k')$, for certain constants $k \neq k'$, unless the polynomial-time hierarchy collapses. What if $g(n) \geq n$?

Question 5.2 We have given examples of PSPACE-hard functions that do not have polynomial-time approximation schemes, unless some unexpected collapse occurs. It is straightforward to define a PSPACE-hard function that *does* have a PTAS, but the straightforward examples are artificial. Is there a natural PSPACE-hard function that has a PTAS?

Question 5.3 Each hardness-of-approximation result given in Section 4 requires a tailor-made reduction. Is there a PSPACE version of the function class MAX-SNP? If there were, it might allow us to prove in one fell swoop that an entire class of maximization problems are PSPACE-hard to approximate within certain ratios, just as Arora et al. [1] prove that all MAX-SNP problems are NP-hard to approximate within certain ratios.

A polynomial-round Arthur-Merlin game [4] can be thought of as a PCDS in which $r(n)$ and $q(n)$ are both arbitrary polynomials and one of the debators simply makes random moves. In this context, the fact that AM(*poly*) = PSPACE (cf. [13, 18]) means that, if $r(n)$ and $q(n)$ are both arbitrary polynomials, then the universal debator in a PCDS can be replaced by a random debator without loss of generality. In a forthcoming paper, we show that, even if $r(n) = \log n$ and $q(n) = 1$, we can replace the universal debator by a random debator and still retain the power to recognize any language in PSPACE. This fact has implications for the hardness of approximating *stochastic* PSPACE-hard functions, of the type studied by Papadimitriou [14].

6 Acknowledgements

We thank Lance Fortnow and Mario Szegedy for helpful discussions in the formative stages of this work. We thank Jin-yi Cai, Lenore Cowen, Uriel Feige, David Johnson, Madhu Sudan, and Mihalis Yannakakis for their comments on earlier versions. Finally, we thank Nick Reingold for helping us typeset Section 3.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof Verification and Hardness of Approximation Problems*, Proc. 33rd Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, 1992, pp. 14–23.

[2] S. Arora and M. Safra, *Probabilistic Checking of Proofs*, Proc. 33rd Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, 1992, pp. 2–13.

[3] L. Babai, L. Fortnow, L. Levin, and M. Szegedy, *Checking Computations in Polylogarithmic Time*, Proc. 23rd Symposium on Theory of Computing, ACM, New York, 1991, pp. 21-31.

[4] L. Babai and S. Moran, *Arthur-Merlin Games: A Randomized Proof System and a Hierarchy of Complexity Classes*, J. Comput. System Sci., 36 (1988), pp. 254–276.

[5] Berlekamp and Welch, 1991. Appears in [10].

[6] A. Condon, J. Feigenbaum, C. Lund, and P. Shor, *Probabilistically Checkable Debate Systems and Approximation Algorithms for PSPACE-Hard Functions*, AT&T Bell Laboratories Technical Memorandum, November 24, 1992.

[7] A. K. Chandra, D. C. Kozen and L. J. Stockmeyer, *Alternation*, J. ACM, 28 (1981), pp. 114-133.

[8] U. Feige, S. Goldwasser, L. Lovász, M. Safra, and M. Szegedy, *Approximating Clique is Almost NP-Complete*, Proc. 32nd Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, 1991, pp. 2–12.

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

[10] P. Gemmell and M. Sudan. *Highly resilient correctors for polynomials*, Inf. Proc. Letters, 43 (1992), pp. 169–174.

[11] D. S. Johnson, *Approximation algorithms for combinatorial problems*, J. Comput. System Sci., 9 (1974), pp. 256-278.

[12] D. Kozen, *Lower bounds for natural proof systems*, Proc. 18th Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, 1977, pp. 254-266.

[13] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. *Algebraic methods for interactive proof systems*, J. ACM, 39 (1992), pp. 859–868.

[14] C. Papadimitriou. *Games Against Nature*, J. Comput. System Sci., 31 (1985), pp. 288–301.

[15] C. Papadimitriou and M. Yannakakis. *Optimization, Approximation, and Complexity Classes*, J. Comput. System Sci., 43 (1991), pp. 425–440.

[16] R. Rubinfeld and M. Sudan. *Testing polynomial functions efficiently and over rational domains*, Proc. 3rd Symposium on Discrete Algorithms, ACM/SIAM, 1992, pp. 23–32. A more efficient tester is to appear in journal version.

[17] T. J. Schaefer, *On the Complexity of Some Two-Person Perfect-Information Games*, J. Comput. System Sci., 16 (1978), pp. 185–225.

[18] A. Shamir. *IP = PSPACE*, J. ACM, 39 (1992), pp. 869–877.