RANDOM DEBATERS AND THE HARDNESS OF APPROXIMATING STOCHASTIC FUNCTIONS*

ANNE CONDON†, JOAN FEIGENBAUM‡, CARSTEN LUND§, AND PETER SHOR¶

Abstract. A *probabilistically checkable debate system* (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. It is known that there is a PCDS for L in which V flips $O(\log n)$ coins and reads $O(1)$ bits of the debate if and only if L is in PSPACE ([Condon *et al.*, Proc. 25th ACM Symposium on Theory of Computing, 1993, pp. 304–315]). In this paper, we restrict attention to RPCDS's, which are PCDS's in which Player 0 follows a very simple strategy: On each turn, Player 0 chooses uniformly at random from the set of legal moves. We prove the following result.

Theorem: L has an RPCDS in which the verifier flips $O(\log n)$ coins and reads $O(1)$ bits of the debate if and only if L is in PSPACE.

This new characterization of PSPACE is used to show that certain *stochastic* PSPACE-hard functions are as hard to approximate closely as they are to compute exactly. Examples of such functions include optimization versions of Dynamic Graph Reliability, Stochastic Satisfiability, Mah-Jongg, Stochastic Generalized Geography, and other "games against nature" of the type introduced in [Papadimitriou, J. Comput. System Sci., 31 (1985), pp. 288–301].

Key words. Approximation Algorithms, Complexity Theory, Probabilistic Games, Proof Systems, PSPACE

AMS subject classification. 68Q15

1. Introduction.

Recently, there has been great progress in understanding the precision with which one can approximate solutions to NP-hard problems efficiently. Feige *et al.* [13], Arora *et al.* [2, 3] and others proved strong negative results for several fundamental problems such as Clique and Satisfiability. This progress has led to renewed study of approximation algorithms for PSPACE-hard problems.

Not surprisingly, PSPACE-hard problems also display a wide variation in the precision with which they can be approximated efficiently. The following results in the literature show that, with respect to a particular performance guarantee, some PSPACE-hard problems have efficient approximation algorithms, others have such algorithms if and only if NP = P, and still others have such algorithms if and only if PSPACE = P. One interesting class of results concern problems on hierarchically defined structures, such as graphs. Problems on these structures, and on a related class of periodic structures, arise in many VLSI and scheduling applications (see, for example, [18] for references to these and other applications). Because such representations can implicitly describe a structure of exponential size, using just polynomial space,

* These results first appeared in our Technical Memorandum [11]. They were presented in preliminary form at the 9th Annual IEEE Conference on Structure in Complexity Theory, Amsterdam, The Netherlands, June 1994.

† University of Wisconsin, Computer Sciences Department, 1210 West Dayton Street, Madison, WI 57306 USA, condon@cs.wisc.edu. Supported in part by NSF grants CCR-9100886 and CCR-9257241.

‡ AT&T Laboratories, Room 2C473, 600 Mountain Avenue, Murray Hill, NJ 07974-0636 USA, jf@research.att.com.

§ AT&T Laboratories, Room 2C324, 600 Mountain Avenue, Murray Hill, NJ 07974-0636 USA, lund@research.att.com.

¶ AT&T Laboratories, Room 2D149, 600 Mountain Avenue, Murray Hill, NJ 07974-0636 USA, shor@research.att.com.

the associated problems are often PSPACE-hard. As early as 1981, Orlin [20] claimed a negative result on approximating a PSPACE-hard periodic version of the Knapsack problem, namely that a fully polynomial approximation scheme exists for this problem only if NP = P. Orlin did not address whether in fact it might be PSPACE-hard to approximate this function. On the positive side, Marathe *et al.* [18] recently developed constant factor approximation algorithms for PSPACE-hard problems, including the Max Cut and Vertex Cover problems for certain restricted classes of hierarchically represented graphs. In related work, Marathe *et al.* [19] applied results of Arora *et al.*[2] to show that these same problems have polynomial-time approximation schemes if and only if NP = P.

In [10], we considered optimization versions of several other problems in PSPACE, including Quantified Satisfiability, Generalized Geography, and Finite Automata Intersection. Building on the techniques of [2, 3, 13], we showed that it is in fact PSPACE-hard to approximate these problems closely (where "closely" depends on the problem). Using direct reduction arguments, Hunt *et al.* [14] showed that some generalized quantified satisfiability problems (for example, satisfiability of quantified formulas in which "clauses" are not restricted to be disjunctions of literals) are PSPACE-hard to approximate.

An important class of PSPACE-hard problems not previously addressed in this literature is a class of *stochastic* problems that involve decision-making under uncertainty, as in the games against nature of Papadimitriou [21]. In this paper, we develop a new technique for showing that these stochastic PSPACE-hard problems are hard to approximate. Informally, these are problems in which the instances involve probabilities in some essential way. The probabilities may describe failures of arcs in a digraph or moves of one of the players in a game. Examples of functions that we prove are hard to approximate include optimization versions of Stochastic Satisfiability [21], Stochastic Generalized Geography, Dynamic Graph Reliability [21], and Mah-Jongg. We describe two of these problems in more detail below. Precise definitions of all of the functions of interest can be found in Section 3.

Our technique for proving that these problems are hard to approximate is based on a new characterization of PSPACE in terms of debates between one powerful and one random player that are checked by a resource-limited verifier. Just as the "games against nature" model is a powerful tool in obtaining hardness results for stochastic problems such as those mentioned above, our new model of PSPACE proves to be a very useful and natural tool in obtaining nonapproximability results for these problems.

We also use the identity IP = PSPACE [16, 23] to derive nonapproximability results for other stochastic PSPACE-hard functions. Examples of the functions that yield to this technique include optimization versions of Dynamic Markov Process [21] and Stochastic Coloring, as well as a different optimization version of Stochastic Satisfiability.

We now give two examples that illustrate the kind of problems to which our new technique applies. The first is a variant of Graph Reliability, a $\#P$-complete problem studied by Valiant [25]: Given a directed, acyclic graph G, source and sink vertices s and t, and a failure probability $p(v,w)$ for each arc (v,w), what is the probability that there is a path from s to t consisting exclusively of arcs that have not failed? Papadimitriou [21] defines Dynamic Graph Reliability (DGR) as follows: The goal of a strategy is still to traverse the digraph from s to t. Now, however, for each vertex x and arc (v,w), there is a failure probability $p((v,w),x)$; the interpretation is that, if the current vertex is x, the probability that the arc (v, w) will fail before the next move is $p((v, w), x)$. DGR consists of those digraphs for which there exists a strategy for getting from s to t with probability at least 1/2. A natural optimization problem is MAX-PROB DGR: Given a graph, vertices s and t, and a set $\{p((v, w), x)\}$ of failure probabilities, what is the probability of reaching t from s under an optimal strategy? We show in Section 3.3 below that there is a constant $c > 0$ such that it is PSPACE-hard to approximate MAX-PROB DGR within ratio 2^{-n^c}. This implies, for example, that if there is a polynomial-time algorithm that, on input x, outputs a number in the range $[2^{-n^c}\text{MAX-PROB DGR}(x), 2^{n^c}\text{MAX-PROB DGR}(x)]$, then PSPACE=P.

Our second example is a solitaire version of Mah-Jongg that is widely available as a computer game. Mah-Jongg tiles are divided into sets of four matching tiles; we assume that there are an arbitrarily large number of tiles. Initially, the tiles are organized in a preset arrangement of rows, and the rows may be stacked on top of each other. As a result, some tiles are hidden under other tiles. A tile that is not hidden and is at the end of a row is said to be available. In a legal move, any pair of available matching tiles may be removed. The player wins the game if all tiles are removed by a sequence of legal moves. An instance of the Mah-Jongg game describes the arrangement of the rows, and, in addition, the tiles that are not hidden. We let MAH-JONGG(x) be the maximum probability of winning the Mah-Jongg game with initial arrangement x of the tiles, assuming that the hidden tiles are randomly and uniformly permuted. We show that approximating the function MAH-JONGG within some factor n^{-c} is PSPACE-hard, where $c < 1$ is some constant. Thus, if there is a polynomial-time algorithm that, on input x, outputs a number in the range $[n^{-c}\text{MAH-JONGG}(x), n^{c}\text{MAH-JONGG}(x)]$, then PSPACE=P.

Our new characterization of PSPACE, which is used in proving nonapproximability results for these problems, builds on techniques developed in our previous work on *probabilistically checkable debate systems* (PCDS's) [10], which in turn builds on techniques of Arora *et al.* [2], Lund *et al.* [16] and Shamir [23]. In a PCDS for L, there are two computationally powerful players, 1 and 0, and a probabilistic polynomial-time verifier V. Players 1 and 0 play a game in which they alternate writing 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 $O(r(n))$ 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$(r(n), q(n))$. We showed in [10] that PCD$(\log n, 1)$ = PSPACE. That is, any language in PSPACE can be recognized by a PCDS in which the verifier uses $O(\log n)$ coin flips and queries only a constant number of bits of the debate.

We now restrict attention to PCDS's in which Player 0 follows a very simple strategy – that of tossing coins. Specifically, whenever Player 0 writes a string of length $f(n)$ on the debate tape π, the string is chosen uniformly at random from $\{0,1\}^{f(n)}$. We call such a debate system an RPCDS and denote by RPCD$(r(n), q(n))$ the class of languages recognized by RPCDS's in which the verifier flips $O(r(n))$ coins and reads $O(q(n))$ bits of π. We note that an Arthur-Merlin game [4] is an RPCDS in which the verifier is deterministic and $q(n)$ is an arbitrary polynomial; that is, V reads the entire debate between Arthur (Player 0) and Merlin (Player 1) before deciding whether the input is in the language. Thus, the class of languages accepted by Arthur-Merlin games is by definition RPCD$(0, \text{poly}(n))$ and is commonly denoted by IP. It is known that RPCD(0, poly(n)) = PSPACE [16, 23]. In this paper, we prove the following result.

Theorem: RPCD($\log n$, 1) = PSPACE.

This theorem shows that a verifier that tosses $O(\log n)$ coins does not have to read the entire debate between Arthur and Merlin. In fact, only a constant number of bits of the debate are needed. Our result is yet another in a sequence of results on polynomial-time interactive complexity classes, starting with the result that IP = PSPACE, that show that "universal quantification" can be replaced by "random quantification with bounded error" without changing the complexity class.

In the rest of this section, we first define precisely the PCDS and RPCDS models. We then describe previous work on related complexity classes.

1.1. Preliminaries. In this section, we define both the PCDS model of [10] and the new RPDCS model. We conclude with some definitions relating to the approximability of PSPACE-hard functions.

A *probabilistically checkable debate system*, or PCDS, consists of a *verifier* V and a *debate format* D. 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 polynomial-time computable 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 $i \bmod 2$ chooses a string of length $f(n)$.

For each x of length 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 vertex, one edge is labeled 0 and the other is labeled 1. A *debate* is any binary 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 x of length n, a *debate subtree* is a subtree of the debate tree of depth $f(n)g(n)$ such that each vertex 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, or, more succinctly, a "strategy" of Player 1.

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.

1. For all x in L, there is a debate subtree on which, 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).

2. 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 co-RP (see, for example, Johnson [15] for a definition). The main result of [10] also holds 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.

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 x 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 $L \in \mathrm{PCD}(r(n), q(n))$. The classical result of Chandra *et al.* [9] that PSPACE is equal to Alternating Polynomial Time can be restated as $\mathrm{PCD}(0, \mathrm{poly}(n)) = \mathrm{PSPACE}$. In [10], we showed that PSPACE is also equal to $\mathrm{PCD}(\log(n), 1)$.

We now focus on PCDS's in which Player 0 follows a very simple strategy – that of tossing coins. Informally, an RPCDS with debate format $D = (f(n), g(n))$ is a PCDS in which, at each even-numbered round, Player 0 moves by choosing a string in $\{0,1\}^{f(n)}$ uniformly at random and writing it on the debate tape. Formally, for a given a debate subtree (i.e., strategy of Player 1), we define the *overall probability* that V outputs 1 to be the average over all debates π in the tree, of the probability that V outputs 1 on debate π. A language L has an RPCDS with error probability ϵ if

$1'$. For all x in L, there is a debate subtree for which the overall probability that V outputs 1 is 1. Again, we say that x is accepted by (D, V).

$2'$. For all x not in L, on all debate subtrees, the overall probability that V outputs 1 is at most ϵ. In this case, we say that x is rejected by (D, V).

Note that item $1'$ is equivalent to item 1 above. If L has an RPCDS with error probability $1/3$ in which V flips $O(r(n))$ coins and reads $O(q(n))$ bits of π, we say that $L \in \mathrm{RPCD}(r(n), q(n))$.

To conclude this section, we review 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 algorithm A computes the function g, we also say that g approximates f within ratio ϵ.

We say that a function g is *PSPACE-hard* if $\mathrm{PSPACE} \subseteq \mathrm{P}^g$, 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.

1.2. Related Work. The theory of probabilistically checkable debate systems developed here and in [10] plays the role for PSPACE that the theory of probabilistically checkable proof systems (PCPS's) plays for NP. A PCPS is simply a PCDS with just one player, Player 1, in which case the "debate" corresponds to a "proof." (See Arora *et al.* [2, 3] or Sudan [24] for an overview of PCPS's.) If the verifier is deterministic and reads all bits of the proof, the model is equivalent in power to a nondeterministic Turing machine, where the proof corresponds to the nondeterministic moves. Thus, by definition, $\mathrm{NP} = \mathrm{PCP}(0, \mathrm{poly}(n))$. Arora *et al.* [2] showed that $\mathrm{PCP}(\log n, 1) = \mathrm{NP}$. Their result shows that there is a dramatic trade-off between the number of random bits available to the verifier of such a proof and the number of bits of the proof that the verifier has to read. The techniques used to prove this result are used heavily in our results on PCDS's and RPCDS's. In the next two paragraphs, we discuss these results and the relationships between the proofs.

In [10], we developed the PCDS in order to extend the work of Arora *et al.* to PSPACE. The PCDS model is related to the alternating Turing machine model of Chandra *et al.* [9], just as the PCP model is related to the nondeterministic Turing machine model. Chandra *et al.* showed that PSPACE is precisely the set of languages recognized by two-player, perfect information games, in which the referee is a deterministic polynomial-time machine that examines the entire game before deciding who wins. The alternating Turing machine model formalizes their notion of such a game. The PCDS model generalizes this game model of Chandra *et al.* by allowing the referee to flip coins; this generality allows one to study the trade-off between the number of coins used and the number of bits of the game, or debate, that are examined by the referee. In current notation, the result of Chandra *et al.* is that PSPACE $= \mathrm{PCD}(0, \mathrm{poly}(n))$. In [10], we showed that $\mathrm{PCD}(\log n, 1) = \mathrm{PSPACE}$. Thus, we get the same trade-off between random bits and queries as was previously shown for proof systems. The proof that $\mathrm{PSPACE} \subseteq \mathrm{PCD}(\log n, 1)$ is done in two parts. One part shows that PSPACE is contained in the class of languages accepted by PCDS's in which the verifier reads only a constant number of *rounds* of the debate. The second part then shows how the constant number of *rounds* in this result can be replaced by a constant number of *bits*. This second part is proved by extending the work of Arora *et al.* on PCPS's.

The result IP = PSPACE [16, 23] shows that Chandra *et al.*'s characterization of PSPACE is true even if one of the two players uses the unsophisticated strategy of simply selecting a move at random. That is, polynomial-time alternating Turing machines and polynomial-round Arthur-Merlin games accept the same class of languages. The main result of this paper shows that this assumption that one player plays randomly does not destroy the tradeoff between the referee's use of randomness and the number of bits of the game that the referee reads. Namely, we prove that $\mathrm{RPCD}(\log n, 1) = \mathrm{PSPACE}$. Again, the proof that $\mathrm{PSPACE} \subseteq \mathrm{RPCD}(\log n, 1)$ is done in two parts. The first shows that PSPACE is contained in the class of languages accepted by PCDS's in which the verifier reads only a constant number of *rounds* of Player 1 and a constant number of *bits* of Player 0. The second part, just as in [10], shows how the constant number of rounds in this result can be replaced by a constant number of bits.

All of these results on PCPS's, PCDS's and RPCDS's can be used in different ways to prove nonapproximability results for hard problems. The result of Arora *et al.* that $\mathrm{NP} = \mathrm{PCP}(\log n, 1)$ has been applied to prove that several NP-hard problems are hard to approximate closely. These problems include optimization versions of Satisfiability, Independent Set [2, 3], Clique [13], and Colorability [17]. For example, the MAX SAT function maps a boolean formula in 3-conjunctive normal form to the maximum number of clauses of that formula that are simultaneously satisfied by some assignment to the variables. Bellare *et al.* [5] showed that there is no polynomial-time algorithm that can approximate MAX SAT within factor 112/113, unless NP = P. More recently, Bellare and Sudan [6] improved 112/113 to 64/65, but their assumption is weaker than NP = P.

Similarly, the result that $\mathrm{PCD}(\log n, 1) = \mathrm{PSPACE}$ yields nonapproximability results for optimization versions of PSPACE-hard problems, including Quantified Satisfiability, Generalized Geography and Finite Automata Intersection [10]. For example, the optimization version of Quantified Satisfiability is defined as follows. Suppose that the variables of the formula are assigned values, in order of quantification, by two players 0 and 1. Players 0 and 1 assign values to the universally and existentially quantified variables, respectively. If Player 1 can guarantee that k clauses of the formula will be satisfied, regardless of what Player 0 chooses, we say that k clauses of the formula are simultaneously satisfiable. The function MAX QSAT maps a quantified formula to its maximum number of simultaneously satisfiable clauses. In [10], we show that approximating MAX QSAT within some constant factor $c < 1$ is PSPACE-hard.

The tools developed in [10] are useful in proving nonapproximability results for PSPACE-hard problems that can be cast as two-person games between two powerful players. However, these tools do not seem to lead to similar proofs for the stochastic PSPACE-hard problems that are considered in this paper. Our new results on RPCDS's are used to obtain such proofs in Section 3.

There has been other very recent work, both on approximation algorithms and on nonapproximability results for PSPACE-hard problems. Using direct reductions from variations of the Quantified Satisfiability problem, Hunt *et al.* [14] and Marathe *et al.* [18] showed that several PSPACE-hard problems are hard to approximate, unless PSPACE = P. These include algebraic problems and graph problems on hierarchically defined graphs. Marathe *et al.* [19] proved that several graph problems such as vertex cover and independent set, when restricted to planar, hierarchically defined graphs, are PSPACE-hard and yet *do* have polynomial-time approximation schemes. They also developed approximation algorithms for restricted optimization problems on periodically defined graphs. Such problems were proved to be PSPACE-hard by Orlin [20].

The rest of this paper is organized as follows. Our main result that RPCD($\log n, 1$) = PSPACE is proven in Section 2. In Section 3, we prove several nonapproximability results for stochastic functions, using this characterization of PSPACE. Finally, in Section 4, we prove additional nonapproximability results, using the result that IP = PSPACE.

2. Language-Recognition Power. In this section, we prove our main result, that RPCD($\log n, 1$) = PSPACE. To prove the (harder) direction that PSPACE $\subseteq$ RPCD($\log n, 1$), we build on several techniques of Lund *et al.* [16], Shamir [23], Arora *et al.* [3, 2] and Condon *et al.* [10]. We first describe these results, and in Lemma 2.3 we put them together to prove PSPACE $\subseteq$ RPCD($\log n, 1$).

The first result we need, Lemma 2.1, shows that in order to prove that PSPACE $\subseteq$ RPCD($\log n, 1$), it is sufficient to show that any language in PSPACE can be recognized by a RPCDS in which the verifier uses $O(\log n)$ random bits, reads a constant number of *rounds* of Player 1, and a constant number of bits of Player 0.

LEMMA 2.1. *Suppose L is accepted by a RPCDS (D, V) in which the verifier uses $O(\log n)$ random bits, reads a constant number of rounds of Player 1, and a constant number of bits of Player 0. Then L is accepted by a RPCDS (D', V') in which the verifier uses $O(\log n)$ random bits, reads a constant number of bits of Player 1, and a constant number of bits of Player 0.*

Proof. (Sketch) We showed in [10, Theorem 3.2] that any language recognized by a PCDS in which V flips $O(\log n)$ coins and reads $O(1)$ *rounds* of the debate is also recognized by a PCDS in which V flips $O(\log n)$ coins and reads $O(1)$ *bits* of the debate. The lemma follows by a straightforward modification to the proof of [10, Theorem 3.2].

Briefly, if D has N rounds on a given input, then D' has $N + 1$ rounds. Let the players of (D, V) be 0 and 1 and the players of (D', V') be $0'$ and $1'$. Roughly, the idea is that in rounds 1 through N, player $1'$ plays as player 1 does in debate D, except that each move is encoded using a special encoding function, known as the low degree polynomial code [2]. The string written by $1'$ in round $N+1$ contains a proof, π_R, for each random string R of V. The proof π_R shows that V outputs 1 given random string R and the decoded debate of rounds $1, \ldots, N$. Moreover, V' can check this proof while examining only a constant number of bits.

The verifier V' chooses a random seed R and computes the indices $i_1, i_2, \ldots, i_q$ of rounds that V queries using the random seed R. Using a protocol of Arora *et al.* [2, 3], V' need only examine a constant number of bits of each round $i_1, i_2, \ldots, i_q$ and a constant number of bits of round $N+1$ in order to verify that V outputs 1 on random string R and the decoded debate of rounds $1, \ldots, N$.

The correctness of this protocol follows from Arora *et al.* [2, 3]. For details, see [10]. □

The next lemma is implicit in the proof that IP = PSPACE [16, 23].

LEMMA 2.2. *Let L be a language in PSPACE and $x = x_1 x_2 \ldots x_n$ be an input. Then, there is a sequence of multivariate polynomials*

$$g_1(y_{1,1}, \ldots, y_{1,n_1}) \ (\textit{where } n_1 = n),\ g_2(y_{2,1}, \ldots, y_{2,n_2}), \ldots, g_m(y_{m,1}, \ldots, y_{m,n_m})$$

with the following properties.

1. *If $x \in L$, then $g_1(x_1, \ldots, x_n) = 1$, and, if $x \notin L$, then $g_1(x_1, \ldots, x_n) = 0$. (Thus, membership in L can be reduced to a g_1-question.)*

2. *There exist polynomial-time computable functions $h_{1,i}$, $h_{2,i}$, and f_i such that $g_i(y) = f_i(g_{i+1}(h_{1,i}(y)), g_{i+1}(h_{2,i}(y)))$ for all y. (That is, one g_i-question reduces to two g_{i+1}-questions.)*

3. *Finally, g_m is a polynomial-time computable function, and the degree of all the polynomials is bounded by $d = \mathrm{poly}(n)$.*

In [16, 23], each g_i is a polynomial that can be explicitly written as a formula of polynomial length in terms of sums and products but may result in a formula of exponential length when these sums and products are expanded.

Finally, we describe a technique, called the **polynomial verification technique**, that was proposed by Babai as a generalization of a technique first used by Lund *et al.* [16]. Roughly, this method "reduces" a *set* of questions of the form: "Is the value of multivariate polynomial g at point a equal to v?" to *one* such question.

To describe this technique precisely, we need the following definitions. Given a (multivariate) polynomial g over a finite field F, a *g-question* is a pair (a, v), where a is an assignment of values in F to the indeterminates in g, and v is a value in F. Let $\{(a_1, v_1), (a_2, v_2), \ldots, (a_l, v_l)\}$ be a set of g-questions. Let $L(t)$ be the interpolated polynomial curve of degree $l-1$ such that $L(j) = a_j$ for every $j = 1, 2, \ldots, l$. (In "$L(j)$," the symbol "j" is used as an abbreviation for "the j^{th} element of the finite field F.") Let p be the (univariate) polynomial $g(L(t))$. Note that $p(j) = g(a_j)$ for $j = 1, 2, \ldots, l$.

The polynomial verification technique receives as input the set of g-questions and a polynomial p'. The technique outputs one g-question $(L(r), p'(r))$, where r is chosen uniformly at random from F. Suppose that $p'(j) = v_j$ for all j. Then, the output has the following property. If all input g-questions are *good*, that is, $g(a_j) = v_j, 1 \le j \le l$, and $p' = p$, the output g-question is also good. Otherwise, with probability at least $1 - d(l-1)/|F|$, the output g-question is not good. Correctness in the latter case follows from the fact that two distinct polynomials of degree $d(l-1)$ can agree on at most $d(l-1)$ points. Note that since $p'(j) = v_j$ and $p(j) = g(a_j)$ for all j, then $p' \neq p$, because at least one question is not good. Thus, the probability that a random point on p' agrees with p is low (at most $d(l-1)/|F|$). This completes the description of the polynomial verification technique.

To motivate our proof that PSPACE $\subseteq$ RPCD($\log n$,1), it is useful to review the proof that PSPACE $\subseteq$ IP. In that proof, a RPCDS for a language L in PSPACE is obtained by combining the polynomial verification technique and Lemma 2.2. Roughly, in each odd numbered round $2k-1$ of the debate, Player 1 writes a g_k-question, where g_k is defined as in Lemma 2.2. In the first round, the g_1-question should be $((x_1,\ldots,x_n),1)$, which is equivalent to claiming that $x \in L$. In round $2k+1$ where $k \geq 1$, the g_{k+1}-question written by Player 1 should be obtained from the g_k-question written by Player 1 in round $2k-1$ by first reducing the g_k-question to two g_{k+1}-questions, as in part (2) of Lemma 2.2, and then by randomly reducing these two questions to one g_{k+1}-question, using the polynomial verification technique. The random number r needed in this reduction is provided by Player 0 in round $2k$ and the polynomial p' is provided by Player 1 in round $2k-1$ (in addition to the g_k-question). To verify that x is indeed in L, the verifier does three things. First, the verifier checks that the g_1-question written by Player 1 is $((x_1,\ldots,x_n),1)$. Second, the verifier checks that for each $k \geq 1$, the g_{k+1}-question in round $2k+1$ is correctly computed from the g_k-question in round $2k-1$, according to the above description. Finally, the verifier checks that the g_m-question written in the last round of the debate is good; by property (3) of Lemma 2.2, this can be done in polynomial time. If all three checks are passed, the verifier accepts, else the verifier rejects. For details, see [16, 23].

Unfortunately, this protocol requires that the verifier read all rounds of the debate system. In Lemma 2.3, building on the ideas of the above protocol, we describe a new protocol in which the verifier need only read a constant number of rounds of Player 1, and a constant number of bits of Player 0. The key idea of the new protocol is to add much "redundancy" in the rounds of Player 1 in order to enable the verifier to check correctness while looking at only a constant number of rounds. Roughly, this is achieved by requiring that Player 1 write in each odd-numbered round not just a single g_k-question but also all the questions written in previous rounds.

In each odd-numbered round k and for each $i = 1, 2, \ldots, m$, Player 1 actually writes a (possibly empty) set of g_i-questions, all from some finite field F. In the first round, the only question that Player 1 writes should be $((x_1,\ldots,x_n),1)$, which is a g_1-question, equivalent to claiming that $x \in L$. Player 1 claims that all g_i-questions written are good. In order to enable the verifier to verify this, Player 1 furthermore writes a univariate polynomial p_i for each i such that the set of g_i-questions is not empty in round k. Player 1 claims that, for every i, the polynomial p_i is the polynomial $g_i(L_i(t))$, where L_i is the polynomial interpolating the domains of the g_i-questions, as in the polynomial verification technique.

Player 0's random move at round $k+1$ supplies the random point $r \in F$ on the curve L to be used in the polynomial verification technique, for each i. By first using the polynomial verification technique and then by reducing the resulting g_i-question to two g_{i+1}-questions, as in Lemma 2.2, any polynomial number of g_i-questions can be probabilistically reduced to two g_{i+1}-questions in one round. The list of g_{i+1}-questions at round $k+2$ is the list of g_{i+1}-questions at round k, plus these additional two questions. Since at most two new g_i-questions are introduced in each odd round, the total number of g_i-questions in any round is polynomial. The main technical contribution of our proof shows that the resulting "redundancy" is sufficient to enable the verifier to check the debate while examining $O(1)$ rounds.

Also, in round $4m+1$, Player 1 writes a sequence of strings $(r_2, r_4 ..., r_{4m})$. Player 1 claims that these are the random moves of Player 0 in the even-numbered rounds. This enables the verifier to read any of the random strings of Player 0 by examining only one round (of Player 1). The verifier can verify with high probability that Player 1 writes the correct strings, by reading just a constant number of bits of the rounds of Player 0.

LEMMA 2.3. PSPACE $\subseteq$ RPCD$(\log n, 1)$.

Proof. Let L be a language in PSPACE. Lemma 2.1 shows that it is sufficient to construct a RPCDS for L in which the verifier uses $O(\log n)$ random bits, reads a constant number of rounds of Player 1, and a constant number of bits of Player 0.

The RPCDS is constructed as follows. The debate system has $4m+1$ rounds. In each odd-numbered round, for every $i = 1, 2, ..., m$, Player 1 writes a set of g_i-questions

$$\{(a_{i,1}, v_{i,1}), (a_{i,2}, v_{i,2}), \ldots, (a_{i,l_i}, v_{i,l_i})\},$$

where $a_{i,j} \in F^{n_i}$ and $v_{i,j} \in F$, for some finite field F whose size will be determined later. Here, $l_1 = 1$; that is, there is only one g_1-question $(a_{1,1}, v_{1,1})$. Also, for $i > 1$, $l_i = 0$ if $k \leq 2i - 3$ and otherwise, l_i increases by 2 in each subsequent odd-numbered round. That is, if $k > 2i - 3$, the number of g_i-questions in odd-numbered round k is 2 more than the number of g_i-questions in round $k - 2$. Furthermore, for each i such that $l_i > 0$, Player 1 writes a univariate polynomial $p_i(t)$ of degree at most $d(l_i - 1)$. Finally, in round $4m + 1$, Player 1 also writes a sequence of strings $(r_2, r_4 ..., r_{4m})$. This completes the description of the debate.

Before describing the verifier, we need one definition. Let $\{(a'_{i,j}, v'_{i,j}),\ 1 \leq i \leq m,\ 1 \leq j \leq l'_i\}$ be the g_i-questions and $\{p'_i\}$ be the polynomials in odd-numbered round $k-2$, and let $\{(a_{i,j}, v_{i,j}), 1 \leq i \leq m, 1 \leq j \leq l_i\}$ be the g_i-questions and $\{p_i\}$ be the polynomials in round k, where $1 < k \leq 4m+1$. Let L'_i be the polynomial curve interpolating the points $\{a'_{i,j}\}$ in round $k-2$, as in the polynomial verification technique. For odd $k > 1$, we say that round k is *locally consistent with respect to* r if the following holds. First, for all i and j, $1 \leq i \leq m$, $1 \leq j \leq l_i$, $p_i(j) = v_{i,j}$. Also, for all $i, 1 \leq i \leq m$, if $l'_{i-1} = 0$ then $l_i = 0$, and if $l'_{i-1} > 0$ then

$$(a_{i,1}, ..., a_{i,l_i}) = (a'_{i,1}, ..., a'_{i,l'_i}, h_{1,i-1}(L'_{i-1}(r)), h_{2,i-1}(L'_{i-1}(r))),$$

and

$$(v_{i,1}, ..., v_{i,l_i}) = (v'_{i,1}, ..., v'_{i,l'_i}, w_{i,1}, w_{i,2}),$$

where $p'_{i-1}(r) = f_{i-1}(w_{i,1}, w_{i,2})$. Note that $(L'_{i-1}(r), p'_{i-1}(r))$ is the single g_{i-1}-question obtained by applying the polynomial verification technique to the set of g_{i-1}-questions in round $k-1$. Applying property (2) of Lemma 2.2, one can check whether this single g_{i-1}-question is good by checking that $p'_{i-1}(r) = f_{i-1}(w_{i,1}, w_{i,2})$ and that the two g_i-questions

$$(h_{1,i-1}(L'_{i-1}(r)), w_{i,1}) \text{ and } (h_{2,i-1}(L'_{i-1}(r)), w_{i,2})$$

are good. Thus, round k is locally consistent with respect to r if for each $i, 1 \leq i \leq m$, the list of g_i-questions at round k consists of the list of g_i-questions at round $k-2$, plus two additional questions, which can used to verify that the g_{i-1}-questions at round $k-2$ are good.

We now describe the protocol of the verifier V. V first reads the final round $4m+1$. V checks that, for all pairs $(a_{m,j}, v_{m,j})$ in round $4m+1$, $g_m(a_{m,j}) = v_{m,j}$ and also that $(a_{1,1}, v_{1,1}) = ((x_1, ..., x_n), 1)$. If not, V rejects. Otherwise, V reads a random odd-numbered round $k > 1$, and checks that (i) round k is *consistent* with the final round $4m+1$: That is, every pair $(a_{i,j}, v_{i,j})$ written in round k is also written in round $4m+1$; and that (ii) round k is locally consistent with respect to the string r_{k-1} written by Player 1 in round $4m+1$. Finally, V reads a random bit of round $k-1$ of Player 0 and checks that it is equal to the corresponding bit of the string r_{k-1} written by Player 1 in round $4m+1$. If all of these checks are satisfied, V accepts; else V rejects.

It is straightforward to show that if $x \in L$, then Player 1 has a strategy that causes V to accept with probability 1. Hence, suppose that $x \notin L$. For a given run of the RPCDS, if in the final round of the debate, $(a_{1,1}, v_{1,1}) \neq ((x_1, ..., x_n), 1)$, then V rejects; hence suppose that in the final round, $(a_{1,1}, v_{1,1}) = ((x_1, ..., x_n), 1)$. Let S be the set of odd-numbered rounds $k > 1$ that are locally consistent with respect to the string r_{k-1} written by Player 1 in the final round, are consistent with the final round, and have $\Delta(r_{k-1}, s_{k-1}) < 1/3$, where s_{k-1} is the move of Player 0 in round $k-1$ and $\Delta(r, s)$ is the fraction of bits that differ in r and s. We say that such a run is *S-consistent*. We will show that if Player 1 is S-consistent for some S with $|S| \geq m$, then, with very high probability, Player 1's move in round $4m+1$ contains a pair $(a_{m,j}, v_{m,j})$ for which $g_m(a_{m,j}) \neq v_{m,j}$. This follows from the next claim and the fact that the run is S-consistent.

Claim: Let S contain the elements $i_1 < i_2 < \cdots < i_m$. If Player 1 is S-consistent and $g_1(x_1, ..., x_n) = 0$, then, for every $k = 1, 2, \ldots, m$, with probability at least $1 - 4mkd/|F|^{1/18}$ (computed over Player 0's coin tosses), there exists some j such that $g_k(a_{k,j}) \neq v_{k,j}$, where $(a_{k,j}, v_{k,j})$ are played in the i_k^{th} round.

Proof. The claim is proven by induction on k. For $k = 1$, the claim holds: Because player 1 is S-consistent, the pair $((x_1, ..., x_n), 1)$ is written by Player 1, whereas by assumption $g_1(x_1, ..., x_n) = 0$.

Thus assume that, for some pair $(a_{k-1,j}, v_{k-1,j})$ in round i_{k-1}, $g_{k-1}(a_{k-1,j}) \neq v_{k-1,j}$. This happens with probability at least $1 - 4m(k-1)d/|F|^{1/18}$, by the inductive hypothesis. Because the run is S-consistent, this pair is also written in round $i_k - 2$. This implies that the polynomial p'_{k-1} played in round $i_k - 2$ is not the polynomial $g_{k-1}(L'_{k-1}(t))$. Thus if $r = r_{i_k - 1}$ is not a root of the polynomial $p'_{k-1}(t) - g_{k-1}(L'_{k-1}(t))$, then $p'_{k-1}(r) \neq g_{k-1}(L'_{k-1}(r))$ implying that either

$$g_k(h_{1,k-1}(L'_{k-1}(r))) \neq w_{k,1} \text{ or } g_k(h_{2,k-1}(L'_{k-1}(r))) \neq w_{k,2}.$$

Otherwise $p'_{k-1}(r) \neq f_{k-1}(w_{k,1}, w_{k,2})$, which contradicts the fact that round i_k is locally consistent with respect to r.

The claim now follows, because the polynomial $p'_{k-1}(t) - g_{k-1}(L'_{k-1}(t))$ has at most $4md$ roots (since l'_{k-1} is always at most $4m$), and the number of s's such that $\Delta(s, r) < 1/3$ for some root r is less than $|F|^{17/18}$ (using Chernov bounds). □

The above claim shows that if Player 1 plays S-consistently for any fixed S with $|S| \geq m$, then with probability at most $\text{poly}(n)/|F|^{1/18}$, the verifier V accepts. Thus, to bound the probability that V accepts, on a run in which Player 1 plays S-consistent rounds for some S, where $|S| \geq m$, sum over all possible S to obtain a bound of $2^{2m}\,\text{poly}(n)/|F|^{1/18}$. The error is thus less than $1/3$ for sufficiently large F.

If Player 1 does not play S-consistently for any S of size greater than m, the verifier rejects with probability at least $1/6$. This is because with probability at least $1/2$, V checks a round k that is not S-consistent, and then, with probability at least $1/3$, V detects that this round is not S-consistent. □

Our main result on the complexity class $\text{RPCD}(\log n, 1)$ now follows easily.

THEOREM 2.4. $\mathrm{RPCD}(\log n, 1) = \mathrm{PSPACE}$.

Proof. Lemma 2.3 proves one direction, that PSPACE $\subseteq \mathrm{RPCD}(\log n, 1)$. In order, to prove the other direction, first note that $\mathrm{RPCD}(\mathrm{poly}(n), \mathrm{poly}(n)) \subseteq \mathrm{RPCD}(0, \mathrm{poly}(n))$, because, in the last round of the debate, Player 0 can supply the verifier of a RPCDS with random bits; hence the verifier needs no random bits as long as it can read all bits in the debate. Combining this with the result that $\mathrm{RPCD}(0, \mathrm{poly}(n)) =$ IP = PSPACE [16, 23] gives Theorem 2.4. □

3. Nonapproximability Results based on RPCD($\log n, 1$). We prove, in this section, that several PSPACE-hard problems are hard to approximate. The PSPACE-complete language SSAT, or "stochastic satisfiability," introduced by Papadimitriou [21], plays an important role in the proofs in this section. In Section 3.1, we define the language SSAT and, using our result that PSPACE = $\mathrm{RPCD}(\log n, 1)$, we show that the corresponding optimization problem is hard to approximate. In the following three sections, we prove that optimization versions of stochastic Generalized Geography, Dynamic Graph Reliability and Mah-Jongg are all hard to approximate.

3.1. Stochastic Satisfiability (SSAT). An SSAT instance is a boolean formula ϕ over the set of variables $\{x_1, \ldots x_n\}$, in conjunctive normal form with three literals per clause. The instance is in the language if there is a choice of boolean value for x_1 such that, for a random choice (with true and false each chosen with probability 1/2) of x_2, there is a choice for x_3, etc., so that the probability that ϕ is satisfied is greater than 1/2. Think of an instance as a game between an existential player and a random player. For each odd value of i, the existential player chooses an optimal boolean value for x_i, where "optimal" means "maximizes the number of clauses of ϕ that are satisfied." For each even value of i, the random player flips a fair coin to get a boolean value for x_i. The odd-numbered variables are called existential variables, and the even-numbered variables are called random variables. The players choose boolean values in order, by increasing value of i. We define the function MAX-CLAUSE SSAT, whose value on a given instance ϕ is the expected number of clauses that are satisfied if the existential player follows an optimal strategy.

THEOREM 3.1. *There is a constant $0 < c < 1$ such that approximating* MAX-CLAUSE SSAT *within ratio c is* PSPACE-*hard.*

Proof. Let L be a language in PSPACE. From Section 2, there is a RPCDS (D, V) 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 $f(n)$ is even for all n. We reduce the problem of deciding whether a string x is accepted by (D, V) to the problem of approximating the expected number of simultaneously satisfiable assignments of a quantified 3CNF formula within a constant factor.

To do this, it is sufficient to construct a formula from x and (D, V), such that if $x \in L$ then all clauses are simultaneously satisfiable, but if $x \notin L$, then the expected number of simultaneously satisfiable clauses is a constant fraction < 1 of the total number of clauses. Let $|x| = n$. The formula has $f(n)g(n)$ variables, corresponding to the bits of a debate between Players 0 and 1, ordered as they appear in the debate. By adding extra variables (which will not appear in the clauses), we can ensure that the variables corresponding to rounds of Player 1 are existential (odd-numbered) variables and those corresponding to rounds of Player 0 are random (even-numbered) variables.

For each sequence of random bits R of length $r(n)$, there is a subformula with $s = O(1)$ clauses. The subformula is satisfied by a truth assignment to the variables if and only if V outputs 1, when the query bits are as in the truth assignment. Such a subformula can be constructed using those variables corresponding to the bits of a debate that are queried on random sequence R. A constant number of additional existential variables are needed so that the subformula is in 3CNF form; these should be ordered after all variables corresponding to bits of the debate.

If x is accepted by (D, V), then there is a debate subtree for which the overall probability that V outputs 1 is 1. This implies that the expected fraction of simultaneously satisfiable clauses of the formula is 1, if the existential player assigns values to the existential variables according to this debate subtree. If x is not accepted by (D, V), then for any debate subtree, the overall probability that V outputs 1 is at most $1/3$. Thus, no matter how the existential variables are chosen, the expected fraction of subformulas satisfied is at most $1/3$. Since each subformula contains $O(1)$ clauses, it follows that the expected fraction of clauses that can be simultaneously satisfied is a constant fraction < 1. ☐

3.2. Stochastic Generalized Geography (SGGEOG). In this section, we derive a nonapproximability result for a stochastic version of Generalized Geography. Generalized Geography, as defined by Schaefer [22], is a game played on a directed graph G with a distinguished vertex s. A marker is initially placed on s, and two players, 1 and 0, alternately move the marker along arcs of the graph, with the constraints that Player 1 moves first and that each arc can be used at most once. The first player unable to move loses. The language GGEOG is the set of pairs (G, s) on which Player 1 has a winning strategy.

In previous work [10], we defined the function MAX GGEOG that maps a pair (G, s) to the largest integer k such that Player 1 can force the game to be played for k moves. (Note that Player 1's objective is to keep the game going as long as possible, whether or not Player 1 ultimately wins.) Here we consider a variation, Stochastic Generalized Geography, in which Player 0 plays randomly. At each even-numbered move of the game, Player 0 simply chooses an arc uniformly at random among all of the unused arcs out of the vertex at which the marker currently sits. The objective of Player 1 is to maximize the length of the game. The function MAX SGGEOG maps a pair (G, s) to the expected length of the game that is achieved when Player 1 follows an optimal strategy.

THEOREM 3.2. *For any constant $0 < c < 1/2$ it is* PSPACE-*hard to approximate* MAX SGGEOG *within ratio n^{-c}, where n is the number of vertices of the graph.*

Proof. We present an approximability-preserving reduction from MAX-CLAUSE SSAT. A key part of our construction is a gadget (directed graph) with the following properties, where $\epsilon > 0$ is some constant. Let ϕ be an instance of SSAT with n variables and m clauses. The gadget constructed from ϕ has a source vertex and a destination vertex. If $\phi \in$ SSAT, then Player 1 has a strategy that guarantees that the destination is reached from the source, with the last move being by Player 0, if the geography game is played from the source vertex, starting with Player 0. However, if $\phi \notin$ SSAT then on each strategy of Player 1, one of two facts hold: (i) the destination is reached from the source, with probability at most $1-\epsilon$, with the last move being by Player 0, or (ii) the destination is reached from the source, with probability at most $(1-\epsilon)^n$, with the last move being by Player 1.

Let the size of this gadget be bounded by $p(|\phi|)$ for some polynomial p of degree > 1. Given this gadget, the reduction builds an instance $((V, A), s)$ of MAX SGGEOG as follows. The directed graph (V, A) contains n independent copies of the gadget, say $g_1, \ldots, g_n$. The distinguished vertex s is an additional vertex, with a single arc to the source vertex of g_1. In addition, A contains a directed edge from the destination vertex of of gadget g_i to the source vertex of gadget g_{i+1}, for $1 \leq i < n$. Finally,

(V,A) contains a "tail" starting at the destination vertex of g_n. The length of the tail is $(np(|\phi|))^K$, where K is some constant that we will determine later. That is, the tail consists of $(np(|\phi|))^K$ ordered vertices, with an edge between the ith and the $(i+1)$st, and one additional edge between the destination vertex of the nth gadget g_n and the first vertex of the tail.

Then, if $\phi \in$ SSAT, Player 1 has a strategy that guarantees that the tail is reached from s. Namely, Player 1 first moves to the source vertex of g_1, and then uses the strategy that guarantees that the destination vertex of g_1 is reached on a move of Player 0. From there, Player 1 follows the single arc to the source vertex of g_2, and so on, until the destination vertex of g_n is finally reached. At that point, the tail is traversed, and so the length of the game is at least $(np(|\phi|))^K$, that is, the length of the tail. However, if $\phi \notin$ SSAT, then the properties of gadgets above guarantee that on all strategies of Player 1, the probability that the tail is reached is at most $(1-\epsilon)^n$. Hence, the expected length of the game is at most $np(|\phi|)$ (that is, the size of the graph (V,A), excluding the tail) $+(1-\epsilon)^n(np(|\phi|))^K$. Thus for sufficiently large ϕ, a multiplicative factor of approximately $(np(|\phi|))^{K-1}/2$ separates the expected length of the game in the cases $\phi \in$ SSAT and $\phi \notin$ SSAT respectively. The size (i.e., the number of nodes) of the new instance is $N = np(|\phi|) + (np(|\phi|))^K$. Thus an approximation within a factor of $N^{-(K-2)/2K} = N^{-(1/2-1/K)}$ would distinguish the two cases. This immediately yields the theorem.

It remains to explain how the gadget is constructed. Assume without loss of generality that n (the number of variables of ϕ) is odd and that MAX-CLAUSE SSAT$(\phi) \geq 1$. In the gadget (V_g, A_g), the vertex set V_g consists of V_1, the "variable-assignment vertices," V_2, the "clause vertices," V_3, the "staircase vertices," and in addition, a source vertex s_g and a destination vertex d_g.

$$V_1 = \left(\bigcup_{i=1}^{n} \{u_i, q_i, \overline{q_i}, w_i, \overline{w_i}, z_i, z_i'\} \right) \cup \{u_{n+1}\}.$$

V_2 consists of $\{y_1, \ldots, y_m\}$. V_3 contains $2n$ sets of vertices, each of size $3t$ where t will be chosen later; we denote them by

$$\{w_{i,1}, w_{i,1}', w_{i,1}'', \ldots, w_{i,t}, w_{i,t}', w_{i,t}''\} \text{ and } \{\overline{w_{i,1}}, \overline{w_{i,1}'}, \overline{w_{i,1}''}, \ldots, \overline{w_{i,t}}, \overline{w_{i,t}'}, \overline{w_{i,t}''}\},$$

for $1 \leq i \leq n$.

The arc set A_g also consists of four parts, A_1, A_2, A_3, and A_4, that serve the following functions. A_1 connects the vertices of V_1 into a sequence of hexagons: For each odd value of i, $1 \leq i \leq n$, the digraph contains the arcs (u_i, q_i), $(u_i, \overline{q_i})$, (q_i, w_i), $(\overline{q_i}, \overline{w_i})$, (w_i, z_i), $(\overline{w_i}, z_i)$, (z_i, z_i'), (z_i', u_{i+1}); for each even value of i, $1 \leq i \leq n-1$, it contains the arcs (u_i, w_i), $(u_i, \overline{w_i})$, (w_i, q_i), $(\overline{w_i}, \overline{q_i})$, (q_i, z_i), $(\overline{q_i}, z_i)$, (z_i, z_i'), (z_i', u_{i+1}). A_2 connects the vertex u_{n+1} to the clause vertices and the clause vertices back to the hexagons as follows. For $1 \leq j \leq m$, there is an arc (u_{n+1}, y_j). For each literal in the j^{th} clause of ϕ, there is an arc (y_j, v), where v is the "w-vertex" corresponding to this literal. For example, if the 17^{th} clause of ϕ is $x_2 \vee \overline{x_7} \vee x_{12}$, the arcs (y_{17}, w_2), $(y_{17}, \overline{w_7})$, and (y_{17}, w_{12}) are present. A_3 creates $2n$ "staircases" hanging off the vertices w_i and $\overline{w_i}$: For $1 \leq i \leq n$, the arcs $(w_i, w_{i,1})$, $(w_{i,1}, w_{i,1}')$, and $(w_{i,1}, w_{i,1}'')$ are present, as are the arcs $(w_{i,j}'', w_{i,j+1})$, $(w_{i,j+1}, w_{i,j+1}')$, and $(w_{i,j+1}, w_{i,j+1}'')$, for $1 \leq j \leq t-1$; analogous arcs form staircases emanating from the $\overline{w_i}$'s. Finally, A_4 contains the arc (s_g, u_1), that connects the source vertex s_g to the first variable-assignment vertex u_1, and also arcs $(w_{i,t}'', d_g)$ and $(\overline{w_{i,t}''}, d_g)$, $1 \leq i \leq n$, that connect the end of each

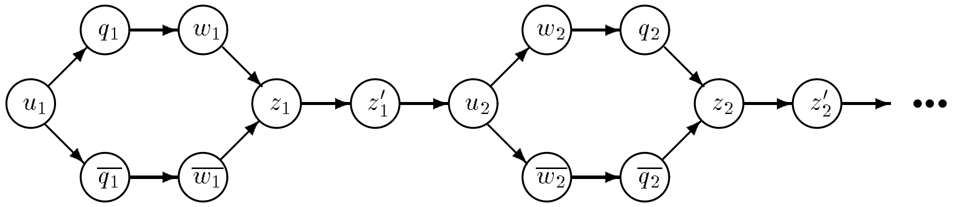

FIG. 1. *The start of the variable assignment (hexagon) section*

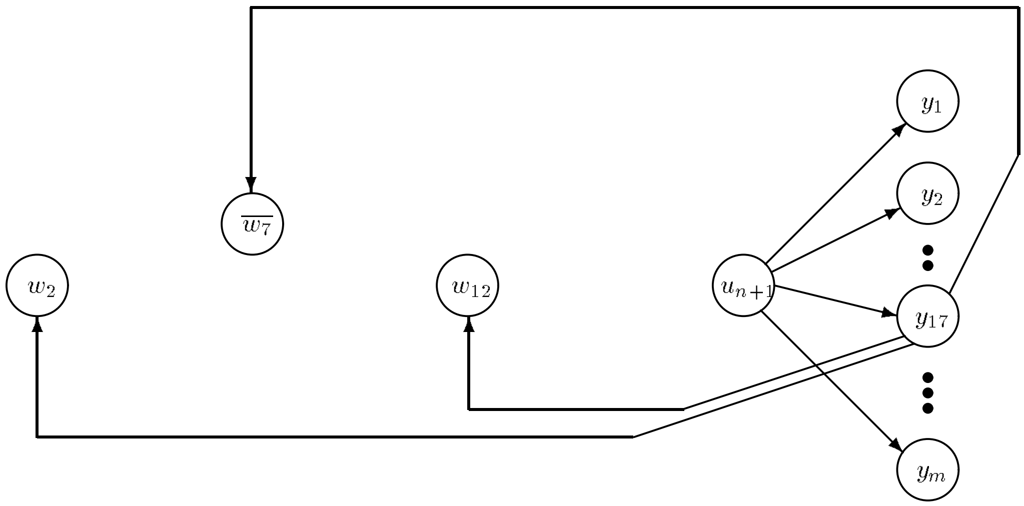

FIG. 2. *The arcs* (u_{n+}, y_j) *and those from* y_{17} *back to the hexagons, where* $c_{17} = x_2 \vee \overline{x_7} \vee x_{12}$

staircase to the destination vertex d_g. Figures 1 through 3 give examples of hexagons, clause-arcs, and staircases.

Within a gadget (V_g, A_g), the Stochastic Generalized Geography game plays out as follows. We assume that Player 0 initially moves from the source vertex, s_g along the single arc to vertex u_1. Player 1 (the existential player) assigns (optimally) either true or false to x_1. If Player 1 assigns true, the next two moves of the game are (u_1, q_1), played by 1, and (q_1, w_1), played by 0. Note that 0 has no choice but to move the marker along (q_1, w_1). Then Player 1 plays (w_1, z_1), 0 plays (z_1, z_1'), and 1 plays (z_1', u_2). We will consider later the case where Player 1 chooses the arc $(w_1, w_{1,1})$. If Player 1 assigns false to x_1, then the analogous moves are made using the vertices $\overline{w_1}$, $\overline{q_1}$, etc. After (z_1', u_2) has been traversed, Player 0 (the random player) assigns (with equal probability) true or false to x_2. If Player 0 assigns true, the next move of the game is (u_2, w_2), played by 0. Play continues along the arcs (w_2, q_2), (q_2, z_2), (z_2, z_2'), (z_2', u_3), played by 1, 0, 1, 0. As before, if Player 0 assigns false to x_2, analogous moves are made using the vertices $\overline{w_2}$, $\overline{q_2}$, etc. From u_3, play continues in this fashion until Player 1 moves along either (z_n', u_{n+1}) or $(\overline{z_n'}, u_{n+1})$, depending on the boolean value chosen for x_n. This ends the "variable assignment" phase of the game.

Player 0 now chooses uniformly at random among the arcs (u_{n+1}, y_1),(u_{n+1}, y_2), ..., (u_{n+1}, y_m). Suppose Player 0 chooses (u_{n+1}, y_j). If the j^{th} clause of ϕ is satisfied by the assignment chosen in the first phase of the game, then Player 1 chooses an arc (y_j, w_i) (resp. $(y_j, \overline{w_i})$) corresponding to a true literal x_i (resp. $\overline{x_i}$) that appears in

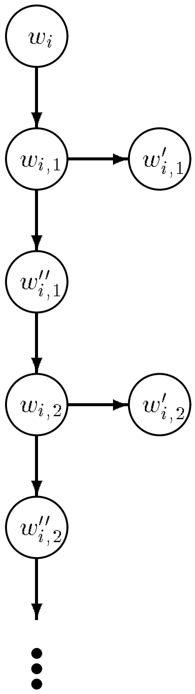

FIG. 3. *The start of the staircase emanating from* w_i

the j^{th} clause. The game then takes $2t$ more steps, because it follows the "spine" of the staircase emanating from w_i (resp. $\overline{w_i}$); Player 0 will have no choice but to take a step down the spine, and Player 1 will always choose to go one step further down the spine instead of off onto a "stair," because the game ends as soon as a stair is chosen. The end of the spine is reached on a move of Player 1, and finally Player 0 follows the single arc to the destination d_g. If the j^{th} clause is not satisfied, then Player 1 must choose an arc (y_j, w_i) (resp. $(y_j, \overline{w_i})$) corresponding to a false literal x_i (resp. $\overline{x_i}$). For concreteness, suppose that Player 1 chooses the arc (y_j, w_i). Both of the arcs from w_i are unused in this case; in the previous case, when x_i was a true literal, only the arc $(w_i, w_{i,1})$ was unused. Player 0 chooses each unused arc with probability 1/2. If Player 0 chooses $(w_i, w_{i,1})$, the game continues for $2t$ moves as before to destination d_g; otherwise the game ends in $O(1)$ moves and d_g is not reached.

The probability of reaching d_g is thus equal to 1, if $\phi \in$ SSAT, since in this case Player 1 can guarantee that all clauses are satisfied by the variable assignment. However, if $\phi \notin$ SSAT, then the probability that a given clause is satisfied by the variable assignment chosen in the first phase of the gadget game is at most k/m, where $k =$ MAX-CLAUSE SSAT(ϕ). If in the second phase of the game, Player 0 chooses a clause that is not satisfied, then with probability 1/2, the destination is not reached. Hence, the probability of reaching d_g is at most $k/m + (1 - k/m)/2 \leq 1 - \epsilon$, for some $\epsilon > 0$, by Theorem 3.1. Thus if $\phi \notin$ SSAT, then on a strategy of Player 1 that never goes off into a staircase, fact (i) of the first paragraph of the proof holds. It remains to consider those strategies of Player 1 that go off into a staircase instead of continuing down the string of hexagons. If Player 1 chooses, say, the arc $(w_i, w_{i,1})$ on the first move from w_i, then Player 0 chooses between the stair arc $(w_{i,1}, w'_{i,1})$ and the spine arc $(w_{i,1}, w''_{i,1})$. With probability 1/2, the stair arc would be chosen, and the game would end after one more step. If the spine arc were chosen, the same choice would confront Player 0 after one more step. Thus, the probability of reaching the end of the spine is 2^{-t}, where t is the number of stairs. We choose t so that $2^{-t} < (1-\epsilon)^n$, to ensure that in this case, fact (ii) holds. □

We note that the result of Theorem 3.2 could also be proved using a reduction from the MAX-PROB SSAT function, defined in the Section 4.

3.3. Dynamic Graph Reliability (DGR). Graph Reliability is a $\#P$-complete problem studied by Valiant [25]: Given a directed, acyclic graph G, source and sink vertices s and t, and a failure probability $p(v,w)$ for each arc (v,w), what is the probability that there is a path from s to t consisting exclusively of arcs that have not failed? Papadimitriou [21] defines Dynamic Graph Reliability (DGR) as follows: The goal of a strategy is still to traverse the digraph from s to t. Now however, for each vertex x and each arc (v,w), there is a failure probability $p((v,w),x)$; the interpretation is that, if the current vertex is x, the probability that the arc (v,w) will fail before the next move is $p((v,w),x)$. The language DGR consists of those digraphs for which there exists a strategy for getting from s to t with probability at least $1/2$. A natural optimization problem is MAX-PROB DGR: Given a graph, vertices s and t, and a set $\{p((v,w),x)\}$ of failure probabilities, what is the probability of reaching t from s under an optimal strategy?

To obtain a nonapproximability result for MAX-PROB DGR, we need the following variant of Theorem 3.1.

THEOREM 3.3. *Consider the restriction of SSAT to instances in which the random variables appear only nonnegated, and there is at most one random variable per clause. There are constants $0 < c_1 < c_2 < 1$ such that, given such a restricted instance with m clauses, it is PSPACE-hard to decide whether on average at least $c_2 m$ clauses are satisfiable or whether at most $c_1 m$ clauses are satisfiable.*

Proof. From the proof of Theorem 3.1, we can conclude that, given a formula ϕ, it is PSPACE-hard to distinguish between the cases that the expected number of simultaneously satisfiable clauses is at most $c_1 m$ or at least $c_2 m$, for some constants c_1, c_2 such that $0 < c_1 < c_2 < 1$. It also follows from this proof, together with the proof of Theorem 2.4, that this is true for instances with only one random variable per clause. To see this, note that in the proof of Theorem 2.4, the verifier queries only one of Player 0's bits; thus each clause of any subformula in the construction of Theorem 3.1 has only one random variable.

We now show how to modify that construction so that no random variable is negated in the resulting formula. To do this, we use the identity

$$\bar{x}yz = \bar{x} + yz + xy\bar{z} + x\bar{y}z + x\bar{y}\bar{z} - 3$$

where concatenation means "or," and "+" and "−" mean addition and subtraction over $\mathbf{Z}$. Suppose ϕ contains n variables and m clauses, of which r contain a negated random variable. We replace any clause $\bar{x}yz$ containing a negated random variable x and literals y and z by the four clauses yz, $xy\bar{z}$, $x\bar{y}z$ and $x\bar{y}\bar{z}$. We now have a formula ϕ' containing n variables and $m + 3r$ clauses. We claim that

$$\text{MAX-CLAUSE SSAT}(\phi') = \text{MAX-CLAUSE SSAT}(\phi) + 5r/2,$$

This formula is derived from the above identity; the term $\bar{x}$ is satisfied exactly half the time, and after subtracting 3, we find that we expect $5/2$ additional clauses to be satisfied in ϕ' for each of the r substitutions that we made. □

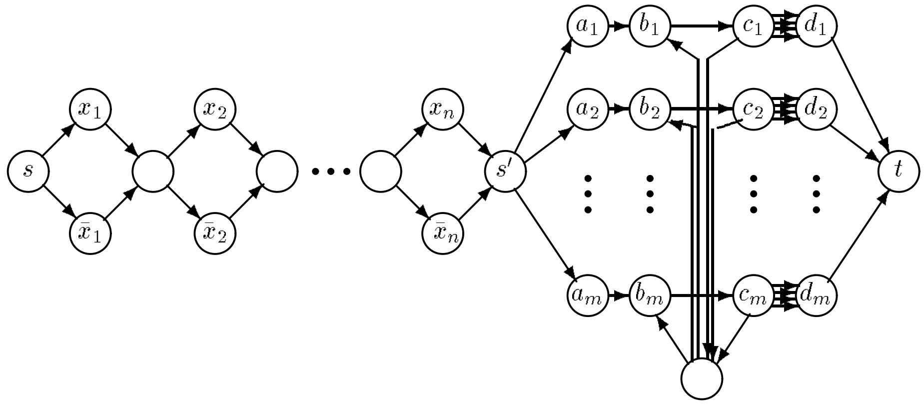

FIG. 4. *The graph constructed in reducing MAX-CLAUSE SSAT to DGR. Note that all edges are directed from left to right except those incident to r, which go from right to left.*

THEOREM 3.4. *There is a constant $c > 0$ such that approximating MAX-PROB DGR within ratio 2^{-n^c} is PSPACE-hard.*

Proof. We show how to reduce the problem of approximating the MAX-CLAUSE SSAT function on an instance of the form given in Theorem 3.3 to the problem of approximating the Dynamic Graph Reliability function. The target DGR instance is composed of two parts, a variable-setting component and a set of clause-testing components. The construction is illustrated in Figure 4. The variable-setting component consists of the n "diamonds" in this figure, and the ith clause-testing component lies between vertices a_i and d_i. There are four edges between each pair c_i and d_i. Three of these correspond to the three literals in the clause, and the fourth is a "bypass edge."

Each path from s through the variable-setting component corresponds to an assignment of the variables in order. If x_i is an existential variable, both of the edges into vertices x_i and $\bar{x}_i$ are available, so the strategy determines which vertex is in the path. If the vertex x_i is chosen, all of the edges in the clause-testing components corresponding to $\bar{x}_i$ fail with probability 1, and similarly if $\bar{x}_i$ is chosen, all of the edges corresponding to x_i in the clause-testing components fail. If x_i is a random variable, upon reaching the vertex immediately before x_i, with probability $1/2$ the edge to x_i fails. If it does, the path must go through $\bar{x}_i$, in which case all of the edges corresponding to x_i in the clause-testing components fail. Otherwise, the path may go through $\bar{x}_i$ or x_i. Since $\bar{x}_i$ never appears in any clauses, it can only help to go through the x_i node, so we may assume that the strategy is consistent with this.

From vertex s', the path leaves the variable-setting component and enters one of m clause-testing components. When s' is reached, each edge $(b_i, c_i), 1 \leq i \leq m$, fails with probability $1 - 1/m$. If all edges (b_i, c_i) fail, clearly t cannot be reached from s'. We next show that if if two or more of these edges survive, t is always reachable, and if exactly one of these edges survives, t is reachable only if the corresponding clause is satisfied. Thus, t is reachable with probability approximately $1 - 2/e + k/(me)$, where k is the number of satisfied clauses and e is 2.71828... . From Theorem 3.3, it now follows that for some constants $c_1 < c_2$, it is PSPACE-hard to distinguish between the cases MAX-PROB DGR$(x) < c_1$ and MAX-PROB DGR$(x) > c_2$.

Recall that three of the four edges between nodes c_i and d_i correspond to the three literals in the clause, and each will be present when s' is reached if the corresponding literal is true. If a_i is visited, then the bypass edge for clause i fails. If exactly one edge (b_i, c_i) survives, clearly t can be reached only if the corresponding clause is satisfied, in which case the path contains $s', a_i, b_i, c_i, d_i, t$. Finally, if two edges (b_i, c_i) and (b_j, c_j) survive, then the path $s', a_i, b_i, c_i, r, b_j, c_j, d_j, t$ reaches t. Note that in this case the bypass edge between c_j and d_j is available because node a_j is never visited.

Finally, we strengthen our result by making several copies of this construction and repeating it in series and in parallel. To show this, we use a result of Ajtai and Ben-Or [1] (see Theorem 3.14 in [8]): For every probabilistic circuit C of size s that accepts a language L with error probability $\epsilon < 1/2$, there is a probabilistic circuit C' of size s poly(N) that accepts L with error probability $< 2^{-N}$. In fact, the circuit C' has the following structure:

$$\bigvee^{2\log N}\left(\bigwedge^{2N^2\log N}\left(\bigvee^{N^3}\left(\bigwedge^{N} C\right)\right)\right).$$

To apply this result to our construction, we replace $\wedge$ by repeating the construction in series (i.e., taking two copies and connecting the sink in the first copy with the source in the second) , and replace $\vee$ by repeating the construction in parallel (i.e., taking two copies and connecting s' in the first copy with the source in the second and the sink in the first copy with the sink in the second). The number of edges of the resulting construction is then $O(sN^{6+\epsilon})$ for any $\epsilon > 0$, where s is the size of the construction above. From this, the theorem follows, where c is any constant less than 1/6. (Assuming that the the instance is encoded such that the length is $O(sN^{6+\epsilon}\log(sN))$, which can easily be done.) □

3.4. A Solitaire Game using Mah-Jongg tiles (MAH-JONGG). Solitaire Mah-Jongg, widely available as a computer game, is played roughly as follows. The game uses Mah-Jongg tiles, which are divided into sets of two or four matching tiles. We generalize the standard game simply by assuming that there are an arbitrarily large number of tiles. Initially, the tiles are organized in a preset arrangement of rows, and the rows may be stacked on top of each other. As a result, some tiles are hidden under other tiles; it is assumed that each possible arrangement of the hidden tiles is equally likely. A tile that is not hidden and is at the end of a row is said to be available. In a legal move, any pair of available matching tiles may be removed, resulting in a new configuration of the tiles in which up to two previously hidden tiles are uncovered. The player wins the game if all tiles are removed by a sequence of legal moves. MAH-JONGG(x) is defined to be the maximum probability of winning the generalized Mah-Jongg game, from initial arrangement x of the tiles, assuming that the hidden tiles are randomly and uniformly permuted.

To define the game precisely, we define a set of Mah-Jongg tiles to be $\mathcal{T} = \cup_i \mathcal{T}_i$, where $\mathcal{T}_1, \ldots, \mathcal{T}_t$ are disjoint sets of tiles, each set being of size 2 or 4. We say tiles T_1 and T_2 *match* if and only if for some i, $T_1, T_2 \in \mathcal{T}_i$. A *configuration* C is a set of positions (i, j, k), where each of i, j, k is a non-negative integer, satisfying the following constraints.

1. If $(i, j, k) \in C$ and $(i, j', k) \in C$ where $j < j'$, then for every j'' in the range $[j, j']$, $(i, j'', k) \in C$.
2. If $(i, j, k) \in C$ where $k > 0$ then $(i, j, k-1) \in C$.

Intuitively, this captures the fact that tiles are arranged in three dimensions. Tiles can be stacked on up of each other; all tiles with common k are at the same height. Tiles at the same height, with common i index, form a row. The first condition ensures that there cannot be "gaps" in a row; the second, that a tile at height $k > 0$ must have a tile underneath it.)

With respect to a given configuration, a position (i,j,k) is *hidden* if $(i,j,k+1)$ is also in the configuration. An *arrangement* consists of a set of Mah-Jongg tiles $\mathcal{T}$, a configuration C of size $|\mathcal{T}|$, and a 1-1 function from the positions of C that are not hidden into $\mathcal{T}$. If this function maps position (i,j,k) to tile T, we say T is *in position* (i,j,k). With respect to a given arrangement, we say a position (i,j,k) is *available* if it is not hidden, and either position $(i,j-1,k)$ or $(i,j+1,k)$ is not in the configuration. An arrangement is *empty* if $\mathcal{T}$ is empty.

Let $x = (\mathcal{T}, C, f)$ be an arrangement. Each pair $\{(i_1,j_1,k_1),(i_2,j_2,k_2)\}$ of available positions at which there are matching tiles T_1, T_2 defines a *match* of x. We say arrangement x' is obtainable from x via match $\{(i_1,j_1,k_1),(i_2,j_2,k_2)\}$ if $x' = (\mathcal{T}', C', f')$, where $\mathcal{T}' = \mathcal{T} - \{T_1,T_2\}$, $C' = C - \{(i_1,j_1,k_1),(i_2,j_2,k_2)\}$ and finally, f' and f agree on the set of positions which are not hidden both in C and C'. (Note that f' is not defined on positions (i_1,j_1,k_1) and (i_2,j_2,k_2), and f' is defined on position (i_1,j_1,k_1-1) if $k_1 > 0$ and on position (i_2,j_2,k_2-1), if $k_2 > 0$, since these are no longer hidden.) Corresponding to each match $\{(i_1,j_1,k_1),(i_2,j_2,k_2)\}$ of x is a *move*, which results in an arrangement x', chosen uniformly and randomly from the set of arrangements obtainable from x via match $\{(i_1,j_1,k_1),(i_2,j_2,k_2)\}$. A sequence of moves from arrangement x leads to a win if it results in the empty arrangement. A *strategy* on x associates a match (if any) with each possible arrangement obtainable from x via any sequence of moves.

An instance of the Mah-Jongg game is an arrangement x. MAH-JONGG(x) is the maximum probability of a win from x, where the maximum is taken over all strategies on x.

In Theorem 3.6, we show that it is PSPACE-hard to approximate the MAH-JONGG function within ratio n^{-c} for some constant $c < 1$. Our result holds for instances x in which tiles are never stacked more than two deep (see Figure 5). We use the following lemma in our reduction.

LEMMA 3.5. *Let $q(n)$ be any polynomial and let $\epsilon > 0$ be any constant. Let ϕ' be an instance of SSAT with no negated random variables. Then there is an instance SSAT ϕ with no negated random variables that can be efficiently constructed from ϕ', with the following properties.*

(a) If the expected fraction of simultaneously satisfiable clauses of ϕ' is at most $c_1 - \epsilon(\geq 0)$, then on any strategy for assigning values to the existential variables of ϕ, with probability at least $1 - 1/q(|\phi'|)$ the fraction of satisfied clauses of ϕ is at most c_1.

(b) If the expected fraction of simultaneously satisfiable clauses of ϕ' is at least $c_2 + \epsilon(\leq 1)$, then there is a strategy for assigning values to the existential variables of ϕ such that with probability at least $1 - 1/q(|\phi'|)$, the fraction of satisfied clauses of ϕ is at least c_2.

Proof. ϕ is simply composed of many independent copies of ϕ', as follows. If

$$\phi' = \exists x_1 \forall x_2 \ldots \exists x_{n'} f(x_1, \ldots x_{n'})$$

then for some $p = p(|\phi|)$ to be chosen later,

$$\phi = \exists x_{11} \forall x_{12} \ldots \exists x_{1n'} \ldots \exists x_{p1} \forall x_{p2} \ldots \exists x_{pn'} f(x_{11}, \ldots x_{1n'}) \wedge \ldots \wedge f(x_{p1}, \ldots x_{pn'}).$$

Fix any strategy for assigning values to the existential variables of ϕ. Let random variable $X_i, 1 \leq i \leq p$ be the fraction of clauses that are satisfied in the subformula

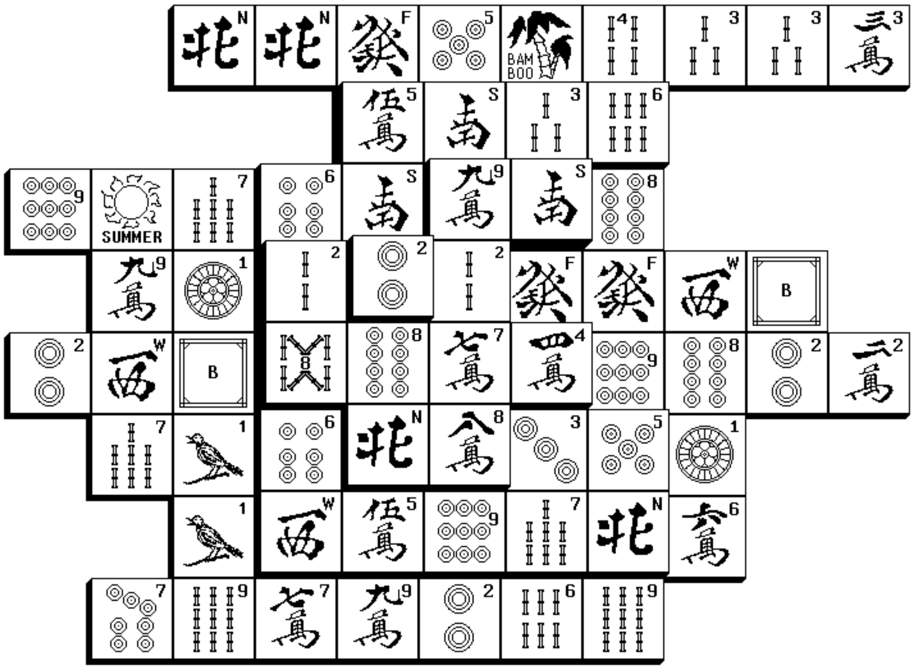

FIG. 5. *The arrangement of tiles part way through a game of Xmahjongg. Possible moves include removing the six of dots in the third row and the six of dots in the sixth row, removing the pair of twos of bamboo in the fourth row, and removing any two of the three available North tiles (in the first, sixth and seventh rows). Xmahjongg is copyright 1989 by Jeff S. Young. Tile designs are copyright 1988 by Mark A. Hohn.*

$f(x_{i1}, \ldots x_{in'})$. Let random variable Y be the fraction of satisfied clauses of ϕ. Then $Y = (1/p)\sum_{i=1}^{p} X_i$. If $\mathrm{Var}(X_i)$ is the variance of X_i, then the variance of Y is

$$\mathrm{Var}(Y) = \sum_{i=1}^{p} \mathrm{Var}(X_i/p) \leq 1/p.$$

To prove part (a), suppose that ϕ' is such that the expected fraction of simultaneously satisfiable clauses is at most $c_1 - \epsilon$. Then, the expected value of X_i, and hence of Y, is at most $c_1 - \epsilon$ for all i. Thus, from Chebyshev's inequality, $\mathrm{Prob}[Y \geq c_1] \leq 1/(\epsilon^2 p)$. If $p(|\phi'|) \geq (1/\epsilon^2)q(|\phi'|)$, then with probability at least $1 - 1/q(|\phi|)$, $Y \leq c_1$. The proof of part (b) is similar. □

THEOREM 3.6. *There is a constant $0 < c < 1$ such that approximating MAH-JONGG within ratio n^{-c} is PSPACE-hard.*

Proof. To prove this result, we show how to reduce the problem of approximating the MAX-CLAUSE SSAT function on an instance ϕ' of the one of the two types given in Theorem 3.3 to the problem of approximating the MAH-JONGG function on some instance. We first convert ϕ' into formula ϕ of Lemma 3.5, and then apply the following reduction to ϕ. We assume without loss of generality that the number of literals in each of the m clauses of ϕ is exactly three and that the number of variables, n, is even. Given such an instance ϕ, the main structures of the corresponding Mah-Jongg game are roughly as follows.

Corresponding to each variable is a set of *variable-setting rows*. Removing a

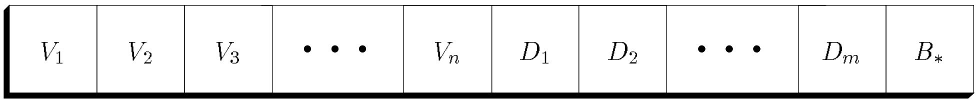

FIG. 6. *The regulating row.*

particular tile from one of these rows corresponds to "setting" the variable. Corresponding to each clause is a set of *clause rows*. Most of the hidden tiles are in the clause rows. The *treasure chest* consists of a single row. On the left end of this row is a special *treasure-key* tile; the game can be won only by removing this tile. The matching treasure-key tile can only be accessed by uncovering a "good" subset of *guard*-tiles, which are initially hidden. Thus, the goal of a good strategy is to maximize the number of hidden tiles uncovered so as to maximize the chances of finding a good subset of guard-tiles, and hence of removing the treasure-key. This strategy corresponds to "setting" the existential variables so as to maximize the number of hidden tiles in the clause rows that are uncovered. (We will see that the random variables are "set" randomly.)

A *regulating row* of tiles is used to ensure that variables are set in the correct order and that no tiles can be removed from a clause row until all variables are set. Furthermore, the regulating row, together with the variable-setting rows, ensures that the only tiles in the clause rows that can legally be removed are those tiles in rows of clauses that have been satisfied during the variable-setting phase. Thus, the probability of winning the game depends on the number of clauses that are satisfied by the variable assignment chosen in the variable-setting phase.

We now describe the tiles and their arrangement in detail. For each existential variable x_i, there are four tiles labeled V_i (one of which is in the treasure chest) and for, each random variable x_i, there are two tiles labeled V_i. For each clause c_j, there are four tiles C_j and two tiles D_j. There are also b pairs of *blocking* tiles; again b will be specified later. The kth pair of blocking tiles is labeled B_k. One of each pair is in the treasure chest.

Figure 6 describes the regulating row. Here and in what follows, we represent a blocking tile by B_*; the subscript is assumed to be different in each occurrence of B_*, and the position of a particular subscript is arbitrary. If the treasure chest is not open, in order to remove all tiles in regulating row, the V_i must first be matched in order, followed by the D_i in order.

We next describe the variable-setting structures. In addition to the tiles defined above, the variable-setting structures contain for each variable x_i, a pair of tiles labeled $M_{i,k}$, where k ranges between 1 and the number of clauses in which either x_i or $\bar{x}_i$ occurs.

If x_i is existential, the ith variable-setting structure is described in Figure 7. If at some point in the game the treasure chest is not open and V_i is the leftmost tile in the regulating row, then this tile must be matched to the V_i in either the first or the second row of this structure. If it is matched to the V_i in the first row, this is equivalent to setting x_i to true. By removing the M-tiles in this row, the player can make available the tiles C_j in this structure, for which clause c_j contains literal x_i. Similarly, setting x_i to false makes available the tiles C_j for which c_j contains $\bar{x}_i$. As the fourth V_i tile is in the treasure chest, the V_i that is not matched cannot be removed until the treasure chest is opened.

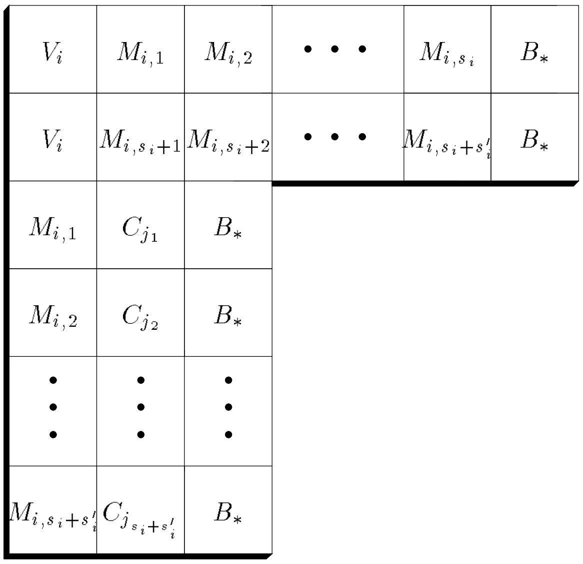

FIG. 7. *The variable-setting structure for existential variable i, where $j_1, j_2, \ldots, j_{s_i}$ are the clauses containing x_i and $j_{s_i+1}, \ldots, j_{s_i+s'_i}$ are the clauses containing $\bar{x}_i$.*

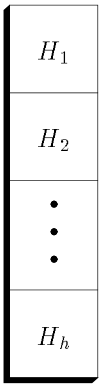

FIG. 8. *The remaining tiles.*

If the variable x_i is a random variable, the variable-setting structure is quite similar, except that *hidden* tiles are used to ensure that the setting of x_i is randomly chosen. Each pair of hidden tiles is labeled H_k or H'_k, for k in the range 1 to h (where h will be specified later). One tile from each pair is hidden. Each remaining tile H'_k is in the treasure chest and each remaining tile H_k forms a singleton row (See Figure 8). Then, if x_i is random, the ith variable-setting structure is described in Figure 9. If the hidden tile under V_i is H_*, then it can be matched to a tile in a singleton row. We will see that this happens with probability approximately 1/2. As before, if x_i is contained in clause c_j, then tile C_j in this structure can be made available. Recall that $\bar{x}_i$ never appears if x_i is a random variable.

We next describe the clause rows (See Figure 10). Again, we need additional tiles. For each clause j, there is a pair of tiles labeled $M'_{j,k}$, where k ranges between 1

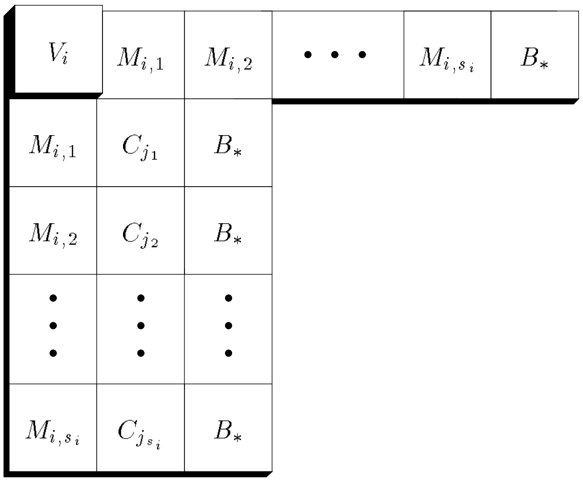

FIG. 9. *The variable-setting structure for random variable i, where $j_1, j_2, \ldots, j_{s_i}$ are the clauses in which x_i appears.*

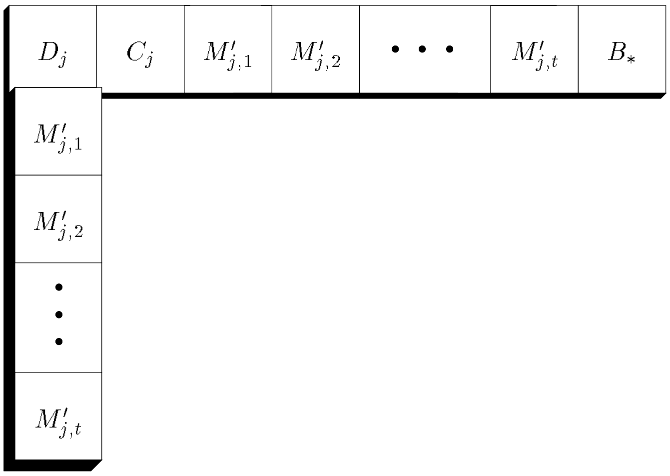

FIG. 10. *The clause row for the ith clause.*

and t (where t will be chosen later). There is a tile hidden under each M'-tile. This arrangement ensures that if the jth clause is true, then all t of the hidden tiles under all $M'_{j,k}$ can be removed.

Before describing the treasure chest and the arrangement of guard-tiles, we state some properties of the game structures constructed so far. In what follows, by "with high probability", we mean with probability at least $1 - 1/p(|\phi'|)$, for some large polynomial p. First, the number $t = t(|\phi'|)$ of tiles hidden in each clause component, is sufficiently large, we can ensure that with high probability, only tiles of the type H_k or H'_k are uncovered in the variable-setting phase. Second, assuming that the random variables are true and false with probability $1/2$, it follows from Lemma 3.5 that if $q(|\phi'|)$ is sufficiently large, then with high probability, the fraction of hidden tiles uncovered in the clause components is either $\geq c_1$ or $\leq c_2$, depending on the

K	L	B_1	$\cdots$	B_l	V_1	$\cdots$	V_n	H'_1	$\cdots$	H'_h	L

FIG. 11. *The treasure chest, where* $l = 4m + (3n/2) + 1 + (3v - 1)/2$.

type of ϕ'. The difficulty here is that after several hidden tiles have been revealed, the probability of a random variable's being true or false is no longer $\frac{1}{2}$ but depends on the relative numbers of H_i and H'_i already revealed. However, if t is sufficiently large, say $t \geq r^k$, the chance of any particular variable being true will change by a factor of at most $1 \pm k!/r^{k-1}$. Thus, the probability of any particular assignment of trues and falses to the random variables will be changed by a factor of at most $(1 \pm k!/r^{k-1})^r \approx 1 \pm k!/r^{k-2}$, which won't affect the probabilities enough to change the result. (This is the reason for the tiles D_i: They ensure that the hidden tiles associated with true clauses cannot be revealed, thus changing the probability of a variable being true, until after the variables have been set.)

The treasure chest is described in Figure 11. On the left end is the treasure-key K. Rows called *guard rows* control access to the single key K which matches the treasure-key in the following way. There are v guard rows (where v is a power of 3 and will be chosen later). Associated with the kth guard row are w pairs of guard-tiles $G_{kj}, 1 \leq j \leq w$, one of which is initially hidden. The other guard-tile from each pair is in the guard row. By finding all guard-tiles for one guard row, the treasure-key can be accessed and the treasure chest opened. We describe the structure that enforces this later; we first show how such a structure, with appropriate choice of v and w, can guarantee that the reduction is "approximation preserving".

First, suppose that the expected fraction of simultaneously satisfiable clauses of ϕ' is at most $c_1 - \epsilon$, so that by Lemma 3.5, on any strategy for assigning values to the existential variables of ϕ, with probability at least $1 - 1/q(|\phi'|)$ the fraction of satisfied clauses of ϕ is at most c_1. Then with high probability (at least $1 - 1/p(|\phi'|)$), on any play of the Mah-Jongg game, at most a fraction c_1 of the hidden tiles in the clause components are uncovered. To remove all guard-tiles from a given guard row, all w guard-tiles must be uncovered. If at most a fraction c_1 of the hidden tiles in the clause components are uncovered, the probability of finding these tiles in the clause components is at most c_1^w. Hence, the probability that the treasure-chest is not opened is at least the probability that no guard-tiles are found during the variable-setting phase, times the probability that at most a fraction c_1 of the hidden tiles in the clause components are uncovered, times the probability that for all guard rows, it is not the case that all guard-keys in that guard row are uncovered. This is at least $(1 - 1/p(|\phi'|)^2(1 - c_1^w)^v)$.

In the case that the expected fraction of simultaneously satisfiable clauses of ϕ' is at least $c_2 + \epsilon$, a slightly different argument shows that the probability that the treasure-chest is opened is at least $(1 - 1/p(|\phi'|)^2(1 - (1 - c_2^w)^v)$. We need to choose w and v so that $(1 - c_1^w)^v$ is large and $(1 - c_2^w)^v$ is small. We let $w = \log_{1/c_1} |\phi'|$ and let $v = |\phi'|$. Then, $(1 - c_1^w)^v = (1 - 1/|\phi'|)^{|\phi'|} \approx 1/e$, and $(1 - c_2^w)^v \leq n^{-c}$, for some $c > 0$. This completes the proof that the reduction is approximation-preserving.

It remains to describe the structure that ensures that the treasure-key can only be accessed by finding all guard-tiles for one guard row. One way to ensure this in our construction would be to put a key K matching the treasure-key in each guard row, which could be accessed once all the guard-tiles of the guard-row are removed.

G_{11}	$\cdots$	G_{1w}	K_{01}	B_*
G_{21}	$\cdots$	G_{2w}	K_{01}	B_*
G_{31}	$\cdots$	G_{3w}	K_{01}	B_*
G_{41}	$\cdots$	G_{4w}	K_{02}	B_*
G_{51}	$\cdots$	G_{5w}	K_{02}	B_*
G_{61}	$\cdots$	G_{6w}	K_{02}	B_*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$G_{v-2,1}$	$\cdots$	$G_{v-2,w}$	K_{0i}	B_*
$G_{v-1,1}$	$\cdots$	$G_{v-1,w}$	K_{0i}	B_*
G_{v1}	$\cdots$	G_{vw}	K_{0i}	B_*

FIG. 12. *The guard rows.*

However, this requires an unbounded number of treasure-keys. The following scheme achieves the same goal while ensuring that the number of tiles of any one type is at most 4 (see Figures 12 and 13).

The guard rows are arranged in groups of three; each row in the ith group has a tile K_{0i} to the right of all of its guard-tiles; this is followed by a blocking tile B_*. For each group i, a fourth tile K_{0i} appears as the leftmost tile in a row with two other tiles. Again, these "first-level" rows are grouped in three's. Each row in the i'th group has as its middle tile a tile $K_{1i'}$, and a blocking B_*-tile as its rightmost tile. Again for each i', the fourth $K_{1i'}$ tile appears as the leftmost tile in a "second-level" row with two other tiles and these rows are grouped in three's. Each row in the i''th group has as its middle tile a tile $K_{2i''}$, and a blocking B_*-tile as its rightmost tile. Further levels of rows are constructed in this way; the number of rows at each level of this structure decreases by a factor of three, until there is only one row left. This row again has three tiles, but the middle one matches the treasure-key K. By opening any one of the guard rows, a row at each level can successively be opened until the treasure-key is finally accessed.

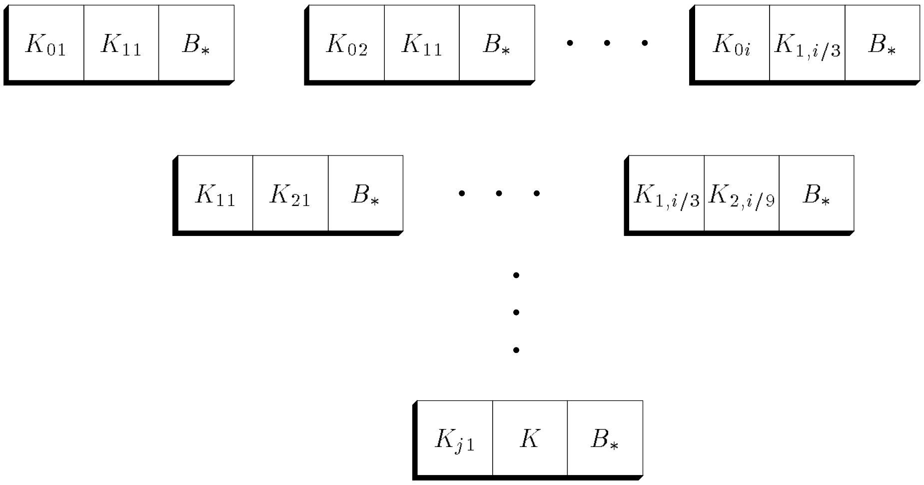

FIG. 13. *Levels of rows which guard access to the treasure chest. If all guard-tiles from one guard row are removed, the key K_{0j} in that row can be used to remove keys at successive levels of this structure until the treasure-key K is removed at the last level.*

To account for the number of pairs of blocking tiles needed, note that $(3v-1)/2$ are needed for the guard rows, as well as $4m + 3n/2$ for the clause and variable-setting rows, and 1 for the regulating row. Therefore, the total number of blocking tiles needed is $4m + 3n/2 + 1 + (3v-1)/2$. Also, once t, v and w are selected to ensure correctness of the reduction, h, the number of pairs of initially hidden tiles of the form H_k or H'_k for some k, can be determined as follows. The total number of hidden tiles is $2h + vw$, that is, one tile from each pair of the form H_k or H'_k, and one of each of the vw pairs of guard-tiles. Also, the number of locations in which tiles are hidden initially is $r + mt$, that is, one in each variable-setting structure for the r random variables, and t in each of the m clause structures. Hence h is chosen so that $2h + vw = r + mt$. □

4. Nonapproximability Results using IP. The proofs of all of the nonapproximability results in the previous section were based on the characterization RPCD$(\log n, 1)$ = PSPACE. In this section, we prove nonapproximability results for different PSPACE-hard functions, based on the characterization IP = PSPACE obtained in [16, 22].

The first problem we consider here is also based on the language SSAT. Once again, an instance of SSAT can be thought of as a game between an existential player and a random player, but this time the objective of the existential player is to maximize the probability that ϕ is satisfied. The value of MAX-PROB SSAT on a given instance is the probability that ϕ is satisfied if the existential player follows an optimal strategy. The language SSAT, as defined in [21], consists of all instances ϕ for which MAX-PROB SSAT$(\phi) > 1/2$.

The reductions in this section are based on the following fact, which is a direct consequence of the proof that IP = PSPACE.

FACT 4.1. *For any language L in PSPACE and any $\epsilon < 1$, there is a polynomial-time reduction f from L to SSAT such that*

$$x \in L \Rightarrow \textit{MAX-PROB SSAT}(f(x)) = 1, \ \textit{and}$$

$$x \notin L \Rightarrow \textit{MAX-PROB SSAT}(f(x)) < 1/2^{n^\epsilon},$$

where n is the number of variables in $f(x)$.

The next theorem is a direct consequence of Fact 4.1.

THEOREM 4.2. *There is a constant $c > 0$ such that approximating MAX-PROB SSAT within ratio 2^{-n^c} is PSPACE-hard.*

Papadimitriou [21] defines the language Dynamic Markov Process (DMP). An instance is a set S of states and an $n \times n$ stochastic matrix P, where $n = |S|$. Associated with each state s_i is a set D_i of decisions, and each $d \in D_i$ is assigned a cost $c(d)$ and a matrix R_d. Each row of R_d must sum to 0, and each entry of $P + R_d$ must be nonnegative. The result of making decision d when the process is in state s_i is that a cost of $c(d)$ is incurred, and the probability of moving to state s_j is the $(i,j)^{th}$ entry of $P + R_d$. A strategy determines which decisions are made over time; an optimal strategy is one that minimizes the expected cost of getting from state s_1 to state s_n. The language DMP consists of tuples $(S, P, \{D_i\}, c, \{R_d\}, B)$ for which there is a strategy with expected cost at most B. A natural optimization problem is MIN DMP, the function that maps $(S, P, \{D_i\}, c, \{R_d\})$ to the expected cost of an optimal strategy. Papadimitriou [21] proves that DMP is PSPACE-complete by providing a reduction from SSAT. In fact, the reduction has the property that if SSAT instance ϕ is mapped to DMP instance $(S, P, \{D_i\}, c, \{R_d\}, B)$, and MAX-PROB SSAT($\phi$) $= p$, then MIN DMP$(S, P, \{D_i\}, c, \{R_d\}) = p/4$. The next result thus follows from Theorem 4.2.

THEOREM 4.3. *There is a constant $c > 0$ such that approximating MIN DMP within ratio 2^{-n^c} is PSPACE-hard.*

The complexity of coloring games was studied by Bodlaender [7], motivated by scheduling problems. An instance of a coloring game consists of a graph $G = (V, E)$, an ownership function o that specifies which of two players, 0 and 1, owns each vertex, a linear ordering f on the vertices, and a finite set C of colors. This instance specifies a game in which the players color the vertices in the order specified by the linear ordering. When vertex i is colored, its owner chooses a color from the set of *legal* colors, i.e., those in set C that are *not* colors of the colored neighbors of i. The game ends either when all vertices are colored, or when a player cannot color the next vertex in the linear ordering f because there are no legal colors. Player 1 wins if and only if all vertices are colored at the end of the game. The length of the game is the number of colored vertices at the end of the game.

We consider a stochastic coloring game (SCG) in which one player, say 0, randomly chooses a color from the set of legal colors at each stage. Two corresponding optimization problems are to maximize the following functions: MAX-PROB SCG(G, o, f, C), which is the maximum probability that Player 1 wins the game (G, o, f, C), and MAX-LENGTH SCG(G, o, f, C), which is the maximum expected length of the game (both maxima are taken over all strategies of Player 1).

We show that MAX-PROB SCG is PSPACE-hard to approximate within a factor of 2^{-n^c}, for some constant $c > 0$, and that MAX-LENGTH SCG is PSPACE-hard to approximate within a factor of $n^{-c'}$, for some constant $c' > 0$.

THEOREM 4.4. *There is a constant $c > 0$ such that approximating MAX-PROB SCG within ratio 2^{-n^c} is PSPACE-hard.*

Proof. We describe a reduction from MAX-PROB SSAT to MAX-PROB SCG that adapts the original construction [7]. By Fact 4.1, we can restrict our attention to instances ϕ such that either MAX-PROB SSAT(ϕ) $= 1$ or MAX-PROB SSAT(ϕ)

$< 1/2^{n^\epsilon}$, for some $\epsilon > 0$. We construct an instance $G = (V, E)$ of MAX-PROB SCG as follows.

$$V = \{true, false, X\} \cup \{x_i, \bar{x}_i \mid 1 \le i \le n\}$$

$$\cup \{c_{j,k} \mid 1 \le j \le m, 1 \le k \le n\} \cup \{d\}.$$

The linear ordering f of the vertices is as follows:

$$true, false, X, x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n \bar{x}_n, c_{1,1}, c_{1,2}, \ldots, c_{m,n}, d.$$

The ownership function is specified as follows. Player 1 owns the vertices $true, false$, and X and also the vertices $x_i, \bar{x}_i$ where x_i is existentially quantified in the formula ϕ. Player 0 owns the remaining vertices. The set of colors C is $\{true, false, X\}$.

It remains to specify the set of edges.

$$E = \{\{true, false\}, \{true, X\}, \{false, X\}\} \tag{1}$$

$$\cup \{\{x_i, \bar{x}_i\}, \{X, x_i\}, \{X, \bar{x}_i\} \mid 1 \le i \le n\} \tag{2}$$

$$\cup \{\{l_i, c_{j,k}\}, \{false, c_{j,k}\} \mid l_i \text{ is a literal in } c_j, 1 \le i \le n, 1 \le j \le m, 1 \le k \le n\} \tag{3}$$

$$\cup \{\{c_{j,k}, d\} \mid 1 \le j \le m, 1 \le k \le n\} \cup \{\{d, false\}, \{d, X\}\}. \tag{4}$$

The set of edges (1) ensures that vertices $true, false, X$ are colored with three distinct colors. Without loss of generality, suppose that they are colored $true, false$ and X, respectively.

The set of edges (2) ensures that, for each pair $x_i, \bar{x}_i$, one is colored $true$ and the other $false$. Thus, each coloring of these vertices corresponds to a truth assignment of the variables $x_1, \ldots, x_n$.

The set of edges (3) ensures that, if clause c_j is true with respect to the truth assignment corresponding to the coloring of vertices $x_1, \ldots, x_n$, then each vertex $c_{j,k}$ is colored X. If c_j is false, then each vertex $c_{j,k}$ is independently colored X or true, each with probability 1/2.

The set of edges (4) ensures that d can be colored only if all the $c_{j,k}$ are colored X.

We claim that deciding whether MAX-PROB SAT(ϕ) is equal to 1 or is at most $1/2^{n^\epsilon}$ can be reduced in polynomial time to approximating MAX-PROB SCG(G, o, f, C) within ratio 2^{-n^c}, where c is a positive constant that depends on ϵ. To see this, note that if MAX-PROB SSAT(ϕ) = 1, then there is a way to choose a truth assignment to the existentially quantified variables that ensures that all clauses of ϕ are true. Hence, by definition of the ownership function, Player 1 has a strategy for coloring vertices such that, for every choice of colors of Player 0, the corresponding truth assignment to $x_1, \ldots, x_n$ satisfies all clauses. Hence, all vertices $c_{j,k}$ are colored X, and so vertex d can be colored. Thus, Player 1 has a strategy that wins with probability 1.

On the other hand, if MAX-PROB SSAT(ϕ) $< 1/2^{n^\epsilon}$, then on any strategy of Player 1, d can be colored with only exponentially small probability. This is because, on all strategies of Player 1, with probability at least $1 - 1/2^{n^\epsilon}$, the truth assignment corresponding to the variable coloring fails to satisfy $f(x)$. Suppose that clause c_j is false. Then with probability at least $1 - 1/2^n$, one of the vertices $c_{j,k}, 1 \leq k \leq n$, is colored *true*. As a result, d can be colored with probability at most $1-(1-1/2^n)(1-1/2^{n^\epsilon})$. □

The proof of Theorem 4.4 can easily be modified to show the following.

THEOREM 4.5. *There is a constant $c' > 0$ such that approximating* MAX-LENGTH SCG *within ratio $n^{-c'}$ is* PSPACE*-hard.*

Proof. Simply modify the construction of Theorem 4.4 so that the vertex set is V', consisting of the vertices of V plus $|V|n^{2c'}$ additional vertices $d_1, \ldots, d_{|V|n^{2c'}}$. The ownership of these vertices can be specified arbitrarily, and they are ordered after the vertices in V. No additional edges are needed.

The expected length of the game is now either $|V|-1+|V|n^{2c'}$ or $|V|-1+(1-(1-1/2^n)(1-1/2^{n^\epsilon}))(|V|n^{2c'})$, depending on the two possibilities for the probability that ϕ is satisfied. □

5. Open Problems. There are many other two-player games that can be made into stochastic functions by letting Player 0 play randomly. Examples include the following.

1. In the Node Kayles game of Schaefer [22], the input is a graph. A move consists of putting a marker on an unoccupied vertex that is not adjacent to any occupied vertex. The first player unable to move loses. We can define Stochastic Node Kayles (SNK) in the same way, except that Player 0, rather than choosing optimally among all unoccupied vertices that are not adjacent to occupied vertices, instead chooses uniformly at random from the same set. Player 1's objective is to keep the game going as long as possible. MAX SNK is the expected length of the game under an optimal strategy of Player 1.

2. In the Generalized Hex game of Even and Tarjan [12], the input is a graph with two distinguished nodes n_1 and n_2. A move for Player 1 (0) consists of putting a white (black) marker on a vertex; the player is free to choose any unoccupied vertex except n_1 or n_2. After all vertices have been chosen, Player 1 wins if and only if there is a path from n_1 to n_2 along only white-occupied vertices. Stochastic Generalized Hex (SGH) is, as usual, the same game in which Player 0 places a black marker on a random unoccupied vertex rather than an optimal unoccupied vertex. A natural stochastic function is MAX-PROB SGH, which maps a graph to the probability, under an optimal strategy of Player 1, that there will be a white path from n_1 to n_2.

Superficially, these games differ from games like Generalized Geography and Stochastic Coloring in that there is no "locality" requirement on the moves of the random player. In Stochastic Generalized Geography, the random player must choose from among the arcs out of the current vertex, and in SCG the random player must color the vertex specified by the ownership function. The reductions that we use to prove MAX SGGEOG, MAX-PROB SCG, and MAX-LENGTH SCG hard to approximate make essential use of this locality. In Node Kayles, say, the random player may choose to mark a vertex that is very far away from the vertex just marked by the existential player.

We have not proven nonapproximability results for any functions without locality. Are they easy to approximate within some constant ratio? Are they in fact easy to compute exactly? It would be interesting to settle the difficulty of approximating these functions and, more generally, to characterize precisely those PSPACE-complete languages that give rise to stochastic functions that are hard to approximate.

6. Acknowledgements. We thank the anonymous referees for their very helpful comments, in particular, for pointing out that our bounds on the nonapproximability of SSGEOG and MAH-JONGG could be improved.

REFERENCES

[1] M. AJTAI AND M. BEN-OR, *A Theorem on Probabilistic Constant Depth Circuits*, Proc. 16th Annual ACM Symposium on Theory of Computing, ACM, New York, 1984, pp. 471-474.

[2] 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.

[3] 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.

[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] M. BELLARE, S. GOLDWASSER, C. LUND AND A. RUSSELL, *Efficient Probabilistic Checkable Proofs and Applications to Approximation*, Proc. 25th Symposium on Theory of Computing, ACM, New York, 1993, pp. 286-293.

[6] M. BELLARE AND M. SUDAN, *Improved Non-approximability Results*, Proc. 26th Symposium on Theory of Computing, ACM, New York, 1994, pp. 184-193.

[7] H. L. BODLAENDER, *On the Complexity of Some Coloring Games*, Intl. J. Foundations Comput. Sci., 2 (1991), pp. 133-147.

[8] R. B. BOPPANA AND M. SIPSER, *The Complexity of Finite Functions*, in Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity, J. van Leeuwen (ed.), MIT Press/Elsevier, 1990, pp. 757–800.

[9] A. K. CHANDRA, D. C. KOZEN, AND L. J. STOCKMEYER, *Alternation*, J. ACM, 28 (1981), pp. 114-133.

[10] A. CONDON, J. FEIGENBAUM, C. LUND, AND P. SHOR, *Probabilistically Checkable Debate Systems and Nonapproximability of PSPACE-Hard Functions*, Chicago J. Theoretical Comp. Sci., vol. 1995, no. 4. Extended abstract, entitled *Probabilistically Checkable Debate Systems and Approximation Algorithms for PSPACE-Hard Functions*, in Proc. 25th Symposium on Theory of Computing, ACM, New York, 1993, pp. 305-314.

[11] A. CONDON, J. FEIGENBAUM, C. LUND, AND P. SHOR, *Random Debaters and the Hardness of Approximating Stochastic Functions (Extended Abstract)*, AT&T Bell Laboratories Technical Memorandum, Murray Hill NJ, May 1993.

[12] S. EVEN AND R. TARJAN, *A Combinatorial Problem which is Complete in Polynomial Space*, J. ACM, 23 (1976), pp. 710–719.

[13] 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.

[14] H. HUNT III, M. MARATHE AND R. STEARNS, *Generalized CNF Satisfiability Problems and Non-Efficient Approximability*, Proc. 9th Conference on Structure in Complexity Theory, IEEE Computer Society Press, Los Alamitos, 1994, pp. 356-366.

[15] D. S. JOHNSON, *A Catalog of Complexity Classes*, in Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity, J. van Leeuwen (ed.), The MIT Press/Elsevier, 1990, pp. 67–162.

[16] C. LUND, L. FORTNOW, H. KARLOFF, AND N. NISAN, *Algebraic methods for interactive proof systems*, J. ACM, 39 (1992), pp. 859–868.

[17] C. LUND AND M. YANNAKAKIS, *On the Hardness of Approximating Minimization Problems*, J. ACM, 41 (1994), pp. 960–981.

[18] M. MARATHE, H. HUNT III, AND S. RAVI, *The Complexity of Approximating PSPACE-Complete Problems for Hierarchical Specifications*, Proc. 20th International Colloquium On Automata, Languages and Programming, A. Lingas, R. Karlsson, and S. Carlsson (eds.), Lecture Notes in Comput. Sci., vol. 700, Spinger, Berlin, 1993, pp. 76-87.

[19] M. MARATHE, H. HUNT III, R. STEARNS, AND V. RADHAKRISHNAN, *Hierarchical Specifications and Polynomial-Time Approximation Schemes for PSPACE-Complete Problems*, Proc. 26th Symposium on Theory of Computing, ACM, New York, 1994, pp. 468–477.

[20] J. ORLIN, *The complexity of Dynamic Languages and Dynamic Optimization Problems*, Proc. 13th Symposium on Theory of Computing, ACM, New York, 1981, pp. 218–227.

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

[22] T. J. SCHAEFER, *On the Complexity of Some Two-Person Perfect-Information Games*, J. Com-

put. System Sci., 16 (1978), pp. 185–225.
[23] A. SHAMIR, *IP = PSPACE*, J. ACM, 39 (1992), pp. 869–877.
[24] M. SUDAN, *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*, PhD Thesis, University of California, Computer Science Division, Berkeley CA, 1992.
[25] L. VALIANT, *The Complexity of Enumeration and Reliability Problems*, SIAM J. Comput., 8 (1979), pp. 410–421.