A Preliminary version of this paper appeared in *Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1990).

Randomness in Interactive Proofs

Mihir Bellare* Oded Goldreich† Shafi Goldwasser‡

(August 24, 1991)

Abstract

This paper initiates a study of the quantitative aspects of randomness in interactive proofs. Our main result, which applies to the equivalent form of IP known as Arthur-Merlin (AM) games, is a randomness-efficient technique for decreasing the error probability. Given an AM proof system for L which achieves error probability 1/3 at the cost of Arthur sending $l(n)$ random bits per round, and given a polynomial $k = k(n)$, we show how to construct an AM proof system for L which, in the same number of rounds as the original proof system, achieves error $2^{-k(n)}$ at the cost of Arthur sending only $O(l+k)$ random bits per round.

Underlying the transformation is a novel sampling method for approximating the average value of an arbitrary function $f : \{0,1\}^l \rightarrow [0,1]$. The method evaluates the function on $O(\epsilon^{-2} \log \delta^{-1})$ sample points generated using only $O(l + \log \delta^{-1})$ coin tosses to get an estimate which with probability $\geq 1-\delta$ is within ϵ of the average value of the function.

Keywords: Interactive proof systems, Arthur-Merlin games, randomness, sampling methods, error reduction, expander graphs, pairwise independent random variables.

* IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598. e-mail: mihir@watson.ibm.com. Work done while author was at MIT supported in part by NSF grant No. CCR-8719689 and DARPA grant No. N00014-89-J-1988.

† Computer Science Department, Technion, Haifa, Israel. e-mail: oded@cs.technion.ac.il. Partially supported by grants No. 86-00301 and 89-00312 from the United States – Israel Binational Science Foundation (BSF).

‡ MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139. e-mail: shafi@theory.lcs.mit.edu. Supported in part by NSF grant No. CCR-8657527, DARPA grant No. DAAL03-86-K-0171 and grants No. 86-00301 and 89-00312 from the United States – Israel Binational Science Foundation (BSF).

1 Introduction

The notion of an "efficiently verifiable proof" plays a fundamental role in the study of efficient computation. Traditionally this notion is associated with the complexity class NP [22], the set of languages with short (polynomial length) and polynomial-time verifiable proofs of membership. In other words, NP as a proof system consists of a prover who provides a short proof and a verifier who checks it. How the prover found the proof is immaterial; as long as the verifier can check it via some mechanical procedure then the proof is valid.

More recently, Goldwasser, Micali and Rackoff suggested that the notion of "efficiently verifiable proofs" be extended to include *interactive proof systems* [31]. Interactive proof systems augment NP proof systems with two ingredients: *interaction* and *randomness.* The verifier is now no longer a silent partner; he is allowed to ask the prover questions. Thus the two parties exchange a short (polynomial in the theorem length) sequence of messages and it is only after the completion of this exchange that the verifier decides whether or not he believes the theorem valid. Furthermore, both parties may flip coins, and the proof is probabilistic: there is some small chance that the verifier accepts the proof of a false theorem.

Since their inception interactive proofs have proven to be very useful. The original motivation for their introduction was the introduction of *zero-knowledge* interactive proofs [31], which in turn were suggested as a principal tool for cryptographic protocol design [31]. The wide applicability of this tool was demonstrated by Goldreich, Micali and Wigderson [30], yielding a dramatic effect on the theory and practice of cryptography. However, in the above mentioned development, the potential power of interactive proofs to serve as proof systems for languages outside NP was not taken advantage of. First indications to such potential were presented by Babai [5] and Goldreich, Micali and Wigderson [30]. But it is only in the last two years that a full understanding of the power of interactive proofs has emerged. And here the truth was startling: interactive proofs exist for any language which can be recognized in polynomial space (cf. Lund, Fortnow, Karloff and Nisan [40] and Shamir [45]). Hence, IP (the class of languages possessing interactive proofs of membership) is probably much larger than NP.

We conclude that interaction and randomness make a potent brew. We are well motivated then, to understand more the precise role played by these ingredients in the granting of so much power. The principal issue here is to determine how the power of interactive proofs varies with the amount of interaction (measured by the number of rounds of message exchange) or the number of coin tosses (of the verifier).

We can begin by observing that the proof of IP=PSPACE yields proof systems using polynomially many rounds and polynomially many coin tosses. On the other hand it is easy to see that in the absence of either ingredient the power of IP vanishes: if there were no interaction then the verifier is left to decide on his own and IP would equal BPP, and if there were no randomness then the prover could anticipate all the verifier's moves and IP would collapse to NP. So the real question lies in understanding what happens in between these extremes.

In this regard the role of interaction has received a lot of study [2, 5, 6, 7, 15]. The role of randomness remained, in contrast, unaddressed. This paper initiates a study of the role of randomness in interactive proofs.

Below we begin by reviewing some definitions and background, and briefly review what is known on the role of interaction in interactive proof systems. Next we turn to randomness, and describe our results and techniques. We conclude by discussing related work and possible avenues for further investigation.

1.1 Interactive Proofs and Arthur-Merlin Games

An interactive proof system [31] consists of two communicating parties, the prover and the verifier. The verifier is restricted to probabilistic polynomial time while the prover is not computationally restricted. On a common input w they exchange a sequence of messages, with the number of messages polynomial in the length n of w. At the end of this exchange the verifier either accepts or rejects the common input. Associated to a prover-verifier pair (P, V) and an input w is a probability that V accepts w which we denote by $\mathbf{P}[(P, V) \text{ accepts } w]$.

We say that a language L has an interactive proof if there is a strategy for the verifier under which he can be convinced to accept inputs in L but rejects inputs not in L no matter what strategy the prover follows. More precisely, L has an interactive proof if there is a verifier V such that two conditions hold. First, there is a prover P such $\mathbf{P}[(P, V) \text{ accepts } w] \geq 2/3$ for every $w \in L$ (the completeness condition). Second, $\mathbf{P}[(\widehat{P}, V) \text{ accepts } w] \leq 1/3$ for every prover $\widehat{P}$ and $w \notin L$ (the soundness condition). This error probability can be decreased to $2^{-k(n)}$ for any polynomial $k(n)$ by standard techniques. Let IP denote the class of languages that possess interactive proofs.

Arthur-Merlin (AM) games, introduced by Babai [5] (see also Babai and Moran [6]), can be viewed as special kinds of interactive proof systems; they are sometimes called interactive proof systems with public coins. Merlin is the prover and Arthur the verifier. Merlin plays just like a prover in an interactive proof system; the specialty is in the role of Arthur. During the interaction, Arthur is restricted to tossing coins and sending their outcome to Merlin. At the end of the interaction he computes a deterministic polynomial time predicate of the common input and the conversation and decides whether or not to accept based on the value of this predicate. AM proof systems for a language L are defined just like interactive ones.

Goldwasser and Sipser [32] showed that any language having an interactive proof system also has an AM proof system; the two systems are thus of equal power as far as language recognition is concerned. Moreover, the equivalence established by [32] preserves the number of rounds. The advantage of the AM formulation lies in its simpler combinatorial structure, and it is via AM games that most structural results (including ours and [5, 2, 6, 15, 29, 10]) are proved. In the rest of this paper we consider AM proof systems.

Let $\mathrm{AM}[g(n)]$ denote the class of languages which possess AM proofs of membership of $g(n)$ rounds of interaction. Much attention has been devoted to understanding the hierarchy $\mathrm{AM} = \mathrm{IP} = \bigcup_{c \geq 0} \mathrm{AM}[n^c]$. We now briefly review what is known.

Babai [5] showed that any constant number of rounds is equivalent to two: $\mathrm{AM}[k] = \mathrm{AM}[2]$ for any integer $k \geq 2$. Probably the most interesting positive result to date however is that of Babai and Moran [6] who showed that the number of rounds in an Arthur-Merlin game can always be cut in half. That is, AM[$2g(n)$] = AM[$g(n)$] for any polynomial $g(n)$. It is important to note that in constructing a $g(n)$ round Arthur-Merlin proof system for L from a $2g(n)$ round one the proof of [6] blows up the message lengths of both parties by polynomial factors, and thus we cannot iterate the collapse more than a constant number of times. Whether or not AM = AM[2] remains an open question.

Boppana, Håstad and Zachos [15] provide some indication that co-NP does not have constant round interactive proofs by showing that if it did then the polynomial time hierarchy would collapse to the second level.

Aiello, Goldwasser and Håstad [2] show that if $\alpha(n) \to \infty$ then there are oracles separating AM[$g(n)$] from AM[$g(n)/\alpha(n)$]. However, we note that the results of [40, 45] indicate that relativized separations are not evidence of real separations, for although IP=PSPACE, a random oracle separates them [33, 20]. So a better than constant factor collapse in IP need not be ruled out by the relativized separation results of [2].

1.2 Our Results

The results in this paper represent the first attempt to understand the role of randomness in interactive proofs in a quantitative sense. The specific problem we consider is that of reducing the error probability of an Arthur-Merlin game in a randomness-efficient manner. Let briefly describe the problem and then our contribution.

The Problem. Recall that an AM game is an AM proof system for the language L if the error probability on any input w (the probability that Arthur accepts if $w \notin L$ or rejects if $w \in L$) is $\leq 1/3$. It is well known that this error probability can be decreased to 2^{-k} for any polynomially bounded $k = k(n)$ by running the game $O(k)$ times in parallel and taking a majority vote on the outcomes [5, 6]. Note that this maintains the number of rounds. Supposing that Arthur sent $l = l(n)$ random bits per round in the original game, this results in a game in each round of which Arthur sends $O(lk)$ random bits.

Error-reduction is the most basic and most often used transformation of AM games and a natural place to begin to investigate the possibility of saving coins. The problem we consider is to accomplish round-preserving error-reduction using fewer coin tosses than the standard method.

Summary of Results. Given a $g = g(n)$ round AM proof system for L in which Arthur sends $l = l(n)$ random bits per round, and given a polynomially bounded function $k = k(n)$, our main result is that we can construct a g round AM proof system for L which achieves error probability $\leq 2^{-k}$ at the cost of Arthur sending only $O(l + k)$ random bits per round.

We also show that if we only need to decrease the error probability to $n^{-O(1)}$ then a constant factor more coins per round suffices. More precisely, given a $g = g(n)$ round AM proof system for L in which Arthur sends $l = l(n)$ random bits per round, and given a function $\epsilon = \epsilon(n)$ with polynomially bounded inverse, we show how to construct a g round Arthur Merlin proof system for L which achieves error probability $\leq \epsilon$ at the cost of Arthur sending only $O(l)$ random bits per round.

Both results extend to the case where the error probability of the original game was $\frac{1}{2} - n^{-O(1)}$

rather than 1/3.

The value 1/3 in the bound on the error probability in the definition of L having an AM proof system is not crucial: equivalent definitions are derived by letting the bound on the error probability be either $\frac{1}{2} - n^{-O(1)}$ or $n^{-\omega(1)}$. However, until now it was not known whether this equivalence preserves the amount of randomness used (even up to a constant multiplicative factor).

1.3 Comparison with the Case of RP and BPP

The problem of reducing the error probability in a randomness-efficient manner has received much attention in the context of the randomized complexity classes RP and BPP.

That such randomness-efficient error-reduction was possible was pointed out, non-constructively, by Sipser [47] and Santha [43]. The first constructions were obtained by Karp, Pippinger and Sipser [38] and Chor and Goldreich [19] who showed that the error-probability of a BPP algorithm which used $r = r(n)$ coin tosses could be reduced to n^{-c} for any given constant c while using only $O(r)$ coin tosses. Cohen and Wigderson [21] and Impagliazzo and Zuckermann [37], using techniques of Ajtai, Komlos and Szemeredi [3], showed how the error could be decreased to 2^{-k} at the cost of $O(r + k)$ coins for any polynomially bounded $k = k(n)$.

Furthermore, pseudo-random sequences [14, 48] can be used to decrease the error of any BPP algorithm to n^{-c} while using only n^{ϵ} coins, for any constants ϵ and c. Coupling this with the results of [21, 37] and [36, 34] we get that the existence of one-way functions implies that the error of any BPP algorithm can be reduced to 2^{-k} while using only $n^{\epsilon} + O(k)$ coins, for any constant ϵ.

Error-reduction for AM games, however, seems more difficult to handle. The above mentioned techniques [47, 43, 38, 19, 21, 37, 3] are not directly applicable here as we are not dealing with witness-sets which are fixed beforehand, but rather with an adversary (cheating Merlin) that dynamically guides, by his responses to the verifiers coins, the search of the verifier for rejecting computations. Furthermore, techniques based on assumptions of computational difficulty are of no use against a prover who has the power to invert one-way functions. Thus the verifier cannot use pseudo-random sequences in place of random ones. We note however that our construction will exploit techniques from [19] and [3] but in a different manner.

In general, not every result for RP and BPP translates easily (or at all) to a result on the class IP. Notable examples are results such as BPP equals almost-P [11] and BPP is contained in non-uniform P [1]. The IP counterpart of the first was open for several years and finally proved in [42], while the IP analogue of the second (i.e. IP is contained in non-uniform NP) is not believed to be true.

1.4 Techniques

Our result involves a novel techniques for randomness-efficient sampling. More specifically, we consider the task of estimating the average value $\mathbf{E}[f] \stackrel{\text{def}}{=} 2^{-l} \sum_{x \in \{0,1\}^l} f(x)$ of a given function $f\colon \{0,1\}^l \to [0,1]$. Our interest is in primitives for this task which we call (l, ϵ, δ)-approximators.

An (l, ϵ, δ)-approximator is a two stage process. In a first, randomized stage, we pick a collection of sample points $x_1, \ldots, x_t \in \{0,1\}^l$. In a second, deterministic stage, we compute, as a function of $f(x_1), \ldots, f(x_t)$, an estimate. We require that with probability $\geq 1-\delta$ this estimate is within ϵ of $\mathbf{E}[f]$.

The straightforward construction of an (l, ϵ, δ)-approximator is to select $t = O(\epsilon^{-2} \log \delta^{-1})$ independent and uniformly distributed sample points and use as the estimate the average value of the function on these sample points. This requires $O(tl)$ coin tosses and m function evaluations.

The construction of randomness-efficient (l, ϵ, δ)-approximators has been a subject of much research, but existing constructions [19, 10, 21, 37, 41] that save coins over the standard method suffer from various restrictions (see §6 for details). We present a new construction which requires the same (up to a constant multiplicative factor) number of sample points (and function evaluations) as the standard construction, but these sample points will be generated using only $O(l + \log \delta^{-1})$ coin tosses. Our technique is optimal in the number of sample points, and, amongst the techniques that use this number of sample points, optimal in the number of coin tosses (both up to constant factors). It works for all functions $f\colon \{0,1\}^l \to [0,1]$.

The tools we rely on are low independence distributions and random walks on explicitly constructed expander graphs.

It is interesting to note that $(l, 1/6, \delta)$-approximators for Boolean functions would suffice for error-reduction in BPP whereas our error-reduction for IP relies on (l, ϵ, δ)-approximators of arbitrary functions ranging in $[0,1]$ with ϵ^{-1} being a polynomial.

1.5 Related Work and Avenues for Further Investigation

Bellare and Rompel [10] have investigated the randomness complexity of the operation of reducing the number of rounds of an AM game by one-half. Given a $2g(n)$ round AM proof system for L in which Arthur sends $l(n)$ random bits per round an Merlin responds with a $q(n)$ bit string, they show how to construct a $g(n)$ round AM proof system for L in which Arthur sends only $O(l + q \log l)$ random bits per round. This improves on the construction of [6] in which Arthur sent $O(lqg^3 \log g)$ random bits per round in the $g(n)$ round game.

We looked at how the power of IP varies with the amount of randomness. A different direction was taken by Schrift [44] who investigated how the power of an interactive proof varies with the *quality* of the random bits used. She shows that interactive proofs retain their power even when the parties do not have access to truly random bits, but rather to certain kinds of sources of weak randomness.

We list in §7 some open questions relating directly to our results. Let us indicate here some more global future directions.

Our results are about AM games. Although AM games and interactive proofs are equivalent in language recognition power this does not mean that our results generalize directly to interactive proofs. This is because there is a cost in randomness in transforming an interactive proof to an AM game. So one direction of research is to generalize our results to general interactive proofs. For example, is there a randomness-efficient way of reducing the error probability of an interactive proof? Or is there a randomness-efficient way to transform any interactive proof into an AM game?

A second and major direction of research is to investigate the role of randomness in zero-knowledge interactive proofs. The current situation for zero-knowledge exactly parallels that which existed for IP before our work. Namely, although there are many results known on reducing interaction [31, 8, 9, 26, 27, 23, 16, 13, 12], nothing is known about reducing randomness. A specific question is to find a randomness-efficient technique of reducing the error probability of a zero-knowledge proof.

2 Preliminaries

We review definitions and notation for the Arthur-Merlin games of Babai [5] and Babai and Moran [6], introduce the accepting probability function, and conclude by showing we can without loss of generality restrict our attention to a special case.

2.1 Arthur-Merlin Games

An Arthur-Merlin game has two players called Merlin and Arthur. They have a common input; we will denote it by w and its length by n. A designated start player makes the first move and after that the players alternate moves. The total number of moves is a polynomially bounded, polynomial time computable function of the input length which we usually denote by G. A player's move consists of sending the other player a message. Merlin's message in any of his moves is a $q(n)$ bit string, computed as an arbitrary function of the common input and Arthur's previous messages. Arthur's messages are more special: each consists of the outcomes of $l(n)$ independent, unbiased coins. Here l and q are fixed polynomially bounded, polynomial time computable functions. When Merlin's last move is completed, Arthur applies a polynomial time computable binary predicate ρ to the common input w and the transcript c of the conversation, and is said to accept if and only if the value of $\rho(w, c)$ is 1. We call ρ Arthur's *decision predicate*. Merlin is said to win the game if Arthur accepts w. Note that there is no restriction on Merlin's computational power.

We will use a bit $s \in \{0, 1\}$ to denote the start player, with 0 standing for Merlin and 1 for Arthur. We call (ρ, G, l, q, s) a *Arthur strategy.* A *Merlin strategy* for the game defined by the Arthur strategy $A = (\rho, G, l, q, s)$ is a function which, given the common input and the l bit messages received so far from Arthur, returns a q bit string which is Merlin's next message.

We write G, l, q for $G(n), l(n), q(n)$ respectively whenever the input length n is understood. Arthur's i-th message will be typically denoted r_i while Merlin's will be denoted y_i. G_A will denote the number of Arthur moves and $G_M = G - G_A$ the number of Merlin moves in the game.

A *round* consists of an Arthur move followed by a Merlin one, or vice-versa. To simplify notation we usually assume that the game consists of $G/2$ rounds with Arthur playing first and Merlin second in each round. We call such a game *symmetric.* We stress that the consideration of symmetric games is only for notational ease: all our theorems extend to the general setting. When $A = (\rho, G, l, q, s)$ is the Arthur strategy of a symmetric game we let $g = G/2$ denote the number of rounds. For the rest of this section we assume the game is symmetric.

Let $A = (\rho, G, l, q, s)$ be a Arthur strategy and let $t \leq g$. An *A Arthur history* is a sequence of strings $r_1 y_1 \cdots r_t y_t$ where $r_j \in \{0, 1\}^l$ and $y_j \in \{0, 1\}^q$ for $j = 1, \ldots, t$. A *A Merlin history* is a sequence of strings of the form $r_1 y_1 \cdots r_{t-1} y_{t-1} r_t$ where $r_j \in \{0,1\}^l$ and $y_j \in \{0,1\}^q$ for $j = 1, \ldots, t$. An A history is either an A Arthur history or an A Merlin history. If $t = g$ we call an A Arthur history an A *conversation* and otherwise we call it a *proper* A Arthur history. When A is understood we omit mention of it and speak of Arthur and Merlin histories, and conversations.

Let M be a Merlin strategy for the game defined by A and let $t \leq g$. An Arthur history $r_1 y_1 \cdots r_t y_t$ in which $y_j = M(w, r_1 \ldots r_j)$ for $j = 1, \ldots, t$ is called an (A, M) Arthur history. A Merlin history $r_1 y_1 \cdots r_{t-1} y_{t-1} r_t$ in which $y_j = M(w, r_1 \ldots r_j)$ for $j = 1, \ldots, t-1$ is called an (A, M) Merlin history. If $t = g$ we call a (A, M) Arthur history a (A, M) conversation. In the game between A and M each (A, M) conversations occurs with probability 2^{-lg}.

The assumption that the players send the same number of bits in each of their moves (l for Arthur and q for Merlin) is made only for notational simplicity. All our results generalize to games in which the number of bits sent in move i is a function of i.

2.2 Arthur-Merlin Proof Systems

With respect to any fixed Merlin strategy, there is for each w a probability (defined by Arthur's random moves) that Arthur will accept. One can now define Arthur-Merlin proof systems for a language L just like one defines interactive proof systems. Namely, we say that an Arthur strategy $A = (\rho, G, l, q, s)$ defines an Arthur-Merlin proof system for L if whenever $w \in L$ there exists a strategy for Merlin under which Arthur accepts with probability $\geq 2/3$ (the completeness condition) and whenever $w \notin L$ the probability that Arthur accepts is $\leq 1/3$ regardless of Merlin's strategy (the soundness condition). Note that since Merlin is computationally unbounded we may, for both conditions, simply assume he plays an optimal winning strategy. Note also that this strategy is deterministic, which justifies our assuming Merlin deterministic in the first place.

As we said above, it actually suffices to consider an "optimal Merlin" that chooses all its messages in a way maximizing Arthur's accepting probability. This leads to the much more convenient formulation of Arthur-Merlin games in terms of max-average combinatorial games as developed by [5, 6]. We capture this formulation below with the definition of the accepting probability function of the game.

Note that the optimal Merlin strategy depends on the Arthur strategy A. For any Arthur strategy A we fix a particular optimal Merlin strategy which we denote by $M_A^{\rm opt}$.

2.3 The Accepting Probability Function

The game tree and its accepting probability function which we now describe are a convenient way of analyzing an Arthur Merlin game (cf. [5, 6]).

Fix an input length n. We visualize a tree of depth equal to the number of moves $2g(n)$. Nodes in this tree are named according to their level: those at even levels are called Arthur nodes while those at odd levels are called Merlin nodes. An Arthur node has $2^{l(n)}$ children, and the set of branches that lead to these children are labeled by the strings from $\{0,1\}^{l(n)}$. Similarly, a Merlin node has $2^{q(n)}$ children, and the set of branches that lead to these children are labeled by the strings from $\{0,1\}^{q(n)}$.

The execution of the game defines a path beginning at the root. The first message r_1 that Arthur sends corresponds to picking the branch out of the root labeled r_1, and the game moves to the corresponding child of the root. It is now Merlin's turn, and his response y_1 similarly selects one of the children of this node. This goes on, with the players alternating, until a leaf is reached.

We label each node of the game tree with an accepting probability chosen so that it bounds the probability that Arthur will accept in the remaining part of the game (the probability is over Arthur's messages in the rest — that is in the subtree rooted at this node — of the game). More formally, we have the following definition of the *accepting probability function* for an Arthur strategy $A = (\rho, g, l, q, 1)$. This is a variation of what [6] call the payoff function.

- The value at a conversation (leaf of the tree) is the value of A's deciding predicate: $acc_A(w, r_1 y_1 \ldots r_g y_g) = \rho(w, r_1 y_1 \ldots r_g y_g)$
- The value at a proper Arthur history (even level internal node of the tree) is the *maximum* value of all possible extensions by one move of Merlin: $acc_A(w, r_1 y_1 \ldots r_{t-1} y_{t-1} r_t) = \max_y acc_A(w, r_1 y_1 \ldots r_{t-1} y_{t-1} r_t . y)$ for $t = g(n), \ldots, 1$.
- Finally the value at a Merlin history (odd level internal node of the tree) is the *average* value of all its extensions by one move of Arthur: $acc_A(w, r_1 y_1 \ldots r_{t-1} y_{t-1}) = \mathbf{E}_r\, acc_A(w, r_1 y_1 \ldots r_{t-1} y_{t-1} . r)$ for $t = g(n), \ldots, 1$.

The following fact is a direct consequence of the definition.

Proposition 2.1 *Let A be an Arthur strategy. Let $t \leq g$ and let $r_1 y_1 \ldots r_t y_t$ be a A Arthur history. Then*

$$acc_A(w, r_1 y_1 \ldots r_{t-1} y_{t-1} . r_t y_t) \;\leq\; acc_A(w, r_1 y_1 \ldots r_{t-1} y_{t-1} . r_t)$$

with equality holding when $y_j = M_A^{\mathrm{opt}}(r_1 \ldots r_{j-1})$ for all $j = 1, \ldots, t$.

A's accepting probability on input w is defined as $acc_A(w) \stackrel{\mathrm{def}}{=} acc_A(w, \lambda)$, where λ is the empty string. The *error probability* of A on input w with respect to a language L is defined as

$$err_A^L(w) = \begin{cases} 1 - acc_A(w) & \text{if } w \in L \\ acc_A(w) & \text{otherwise.} \end{cases}$$

The error probability of A with respect to L is the map from $\mathsf{N} \to [0,1]$ whose value at n is $\sup_{|w|=n} err_A^L(w)$. Thus an Arthur strategy A defines a proof system for L if its error probability with respect to L is $\leq 1/3$.

2.4 Restriction to a Special Case

For technical reasons it will be convenient to assume that Arthur messages are of length $\geq c \log n$ (for a specific constant c that will arise in our construction). This assumption does not reduce the generality of our results, since for every $c > 0$, any Arthur-Merlin game can transformed — without increasing the number of rounds or, upto constant factors, the total number of coin tosses — into one in which Arthur's messages are of length $\geq c \log n$. More precisely, we have the following

Proposition 2.2 *There is a constant α such that the following is true. Let c be > 0 and suppose $A = (\rho, G, l, q, s)$ defines an Arthur-Merlin proof system for L in which $l(n) \leq c \lg n$. Then we can construct $A^* = (\rho^*, G^*, l^*, q^*, 1)$ which defines an Arthur-Merlin proof system for L in which $c \lg n \leq l^*(n) \leq \alpha c \lg n$, the number of moves G^* is $\leq G$, and the total number of coins flipped by Arthur increases by a factor of at most α.*

Proof: The idea is to group consecutive rounds of the given game into blocks in such a way that Arthur is sending just over $c \lg n$ bits per block, and then "collapse" each block into a single round. Merlin begins this round by sending his responses to all possible messages of A for the block, and Arthur then selects one sequence of responses at random. A more precise specification follows.

For simplicity we assume the given game is symmetric and let $g = G/2$ be the number of rounds. Without loss of generality we assume the error probability of the given game is $\leq 1/6$; this can be achieved by standard error-reduction as per Theorem 3.1 at a constant cost in message lengths and no cost in rounds. Let $b(n)$ be the least integer such that $b(n)l(n) \geq c \lg n$ and let $l^* = bl$. Let $g^*(n)$ be the least integer such that $b(n)(g^*(n) + 1) \geq g(n)$. We view the given game as being divided into g^* blocks of b rounds each, followed by a block of $\leq b$ rounds. The new game consists of g^* rounds followed by a final Merlin move. Merlin plays first and Arthur second in each of the g^* rounds.

The execution of the g^* rounds will define a A Arthur history $h_1 \ldots h_{g^*}$ which is built up round by round, with round i adding the sequence of strings h_i which corresponds to moves of the original players in block i. Initially this history is the empty string. Assuming the first t rounds of the game have been played and $h_1 \ldots h_{t-1}$ is defined, here is how it is extended. Merlin's message in round t of the new game is (the encoding of) a l-ary tree of depth b each node of which is labeled with a q bit string; this represents his responses to each possible sequence of A moves in block t of the old game with history $h_1 \ldots h_{t-1}$. Arthur's round t response is a $l^* = bl$ bit random string which selects a branch of the tree and thus defines a particular sequence h_t of b moves of the original game in this block. This sequence is appended to the history, and the parties then move to the next round.

For the last block, Merlin's message is (the encoding of) a l-ary tree of depth $\leq b$ each node of which is labeled with a q bit string; this represents his responses to each possible sequence of A moves in block $g^* + 1$ of the old game with history $h_1 \ldots h_{g^*}$. At the conclusion of this move, Arthur has a polynomial number of A conversations. He evaluates ρ on each of these, and accepts if and only if a majority are accepting, and, additionally, if Merlin's message for each round was indeed (the encoding of) a l-ary tree of the appropriate depth labeled with q bit strings.

The fact that the original game had error probability $\leq 1/6$ implies that if $w \in L$ and Merlin provides responses corresponding to those of M_A^{opt} then the majority of conversations extending a history $h = h_1 \ldots h_{g^*}$ are accepting for at least $2/3$ of these histories. Similarly if $w \notin L$ then regardless of how Merlin responds, the majority of conversations extending a history $h = h_1 \ldots h_{g^*}$ are accepting for at most $1/3$ of these histories. So our final game is an Arthur-Merlin proof system for L. ∎

$$
\left.\begin{array}{lcccc}
 & \underline{\text{subgame } 1} & \underline{\text{subgame } 2} & \dots & \underline{\text{subgame } m} \\
\text{Arthur's message:} & r_1^1 & r_1^2 & \dots & r_1^m \\
\text{Merlin's response:} & y_1^1 & y_1^2 & & y_1^m \\
\vdots & \vdots & \vdots & & \vdots \\
\text{Arthur's message:} & r_g^1 & r_g^2 & & r_g^m \\
\text{Merlin's response:} & y_g^1 & y_g^2 & \dots & y_g^m
\end{array}\right\} g \text{ rounds}
$$

Figure 1: *Framework of the Standard Error-Reduction Protocol*

3 Error-Reduction: The Basic Template

We call *error-reduction* the process of reducing the error probability of an Arthur-Merlin proof system from $\leq 1/3$ to $\leq 2^{-k}$ for a given polynomially bounded $k = k(n)$. As an introduction to our error-reduction technique it is helpful to review the standard one [5, 6].

3.1 The Standard Error-Reduction Technique

Given $A = (\rho, g, l, q)$ defining an Arthur-Merlin proof system for L with error $\leq 1/3$ we are required to design A^* defining an error $\leq 2^{-k}$ Arthur-Merlin proof system for L. The solution is to play in parallel $m = O(k)$ *independent* copies of the old game (the one defined by strategy A). The independence of Arthur's moves in the various "subgames" is used to prove that the error probability decreases exponentially with the number of subgames.

More concretely, A^* will, in round t, send ml random bits to Merlin. These bits are regarded as a sequence $r_t^1 \dots r_t^m$ of m different round t messages of A. Merlin then responds with strings $y_t^1 \dots y_t^m$, and y_t^i is regarded as the response of Merlin to r_t^i in the i-th subgame ($i = 1, \dots, m$). This continues for g rounds (see Figure 1). Finally, A^* will accept in the new game iff a majority of the subgames were accepting for the original A. The result is

Theorem 3.1 [5, 6] *There is a constant α such that the following is true. Let $A = (\rho, g, l, q, s)$ be an Arthur strategy which has error $\leq 1/3$ with respect to L. Let $k\colon \mathsf{N} \to \mathsf{N}$ be polynomially bounded and polynomial time computable. Let $m = \alpha k$. Then the Arthur strategy $A^* = (\rho^*, g, ml, mq, s)$ has error probability $\leq 2^{-k}$ with respect to L, where*

$$
\rho^*(w, \vec{r}_1\vec{y}_1 \cdots \vec{r}_g\vec{y}_g) = \begin{cases} 1 & \textit{if } \ |\{\, i \in [m] \,:\, \rho(w, r_1^i y_1^i \dots r_g^i y_g^i) = 1 \,\}| \geq \frac{m}{2} \\ 0 & \textit{otherwise}\,. \end{cases}
$$

Here Merlin's round t message $\vec{y}_t \in \{0,1\}^{mq}$ is parsed as $\vec{y}_t = y_t^1 \dots y_t^m$ with $y_t^i \in \{0,1\}^q$.

The bound on the error probability of the new game follows from the fact that the coin tosses used by Arthur in the different subgames are independent. However, the cost of this argument is in the large number of coin tosses used by A^*; namely $O(lk)$ coin tosses per round (to be contrasted with the l coin tosses used in each round of the original game).

$$
\begin{array}{rccccc}
 & \underline{\text{subgame } 1} & \underline{\text{subgame } 2} & \cdots & \underline{\text{subgame } m} & \\
\text{Arthur's message } s_1 \text{ specifies:} & r_1^1 & r_1^2 & \cdots & r_1^m & \multirow{5}{*}{$\left.\begin{array}{c}\\ \\ \\ \\ \\ \end{array}\right\} g \text{ rounds}$} \\
\text{Merlin's response:} & y_1^1 & y_1^2 & & y_1^m & \\
\vdots & \vdots & \vdots & & \vdots & \\
\text{Arthur's message } s_g \text{ specifies:} & r_g^1 & r_g^2 & & r_g^m & \\
\text{Merlin's response:} & y_g^1 & y_g^2 & \cdots & y_g^m &
\end{array}
$$

Figure 2: *Framework of Our Error-Reduction Protocol*

3.2 Framework of Our Solution

We will stay within the template of playing several copies of the original game in parallel, evaluating each of these subgames individually at the conclusion of the game, and, based on the outcomes of the subgames, deciding whether or not to accept. Our subgames, however, will be *dependent* copies of the original game, and we will prove that although these copies depend on one another the error probability decreases exponentially with our "investment" in the randomness of each round.

More precisely, Arthur's message in round t will consist of a randomly chosen "seed" s_t. This seed, via an appropriate deterministic process, which we will later describe, specifies a sequence of (statistically dependent) strings $r_t^1 \dots r_t^m$, where $m = m(n)$ is a function of our deterministic process. These strings will play the role of the original Arthur's round t messages for the different subgames. The Merlin of the new game computes the sequence of messages specified by the seed and replies with a sequence of m strings $y_t^1 \dots y_t^m$ that will be interpreted as his answers in the corresponding m subgames (see Figure 2). At the end, A^* evaluates $\rho(w, r_1^i y_1^i \dots r_g^i y_g^i)$ for each $i = 1, \dots, m$ and decides whether or not to accepts based on some function of these values.

We stress the Arthur's moves in the different rounds are still statistically independent (a new random seed is selected at each round), and that A^* actually sends in round t the (uniformly selected) seed s_t and both parties compute the sequence $r_t^1, \dots, r_t^m$ specified by the seed s_t.

Within this template we rely on a combination of two different ideas. The first idea is to specify $r_t^1, \dots, r_t^m$ so that this sequence is pairwise independently distributed. We show that we can approximate the average accepting rate of independent subgames and thereby reduce the error to $\epsilon = n^{-O(1)}$; the restriction on ϵ comes from the fact that the message lengths in the game we construct here are polynomial in ϵ^{-1}. The second idea is to play many copies of this new game in each of which the error is non-negligible (i.e. $n^{-O(1)}$) and take the median value, relying on the fact that the probability that a majority of the games are not representative is exponentially small. In the implementation of this step we rely on random walks on explicitly constructed expander graphs.

We now elaborate on of these ideas in turn.

4 Reducing Error to any Non-Negligible Fraction

Here we show how to reduce the error probability to $n^{-O(1)}$ at the cost of multiplying by a constant factor the number of coin tosses used by Arthur in each round. More precisely, we will prove the following

Theorem 4.1 *There is a constant α such that the following is true. Let c be > 0 and let $A = (\rho, G, l, q, s)$ be an Arthur strategy with error probability $\leq \frac{1}{2} - n^{-c}$ with respect to the language L. Let ϵ: $\mathsf{N} \to [0,1]$ have polynomially bounded inverse. Then we can construct another Arthur strategy $A^* = (\rho^*, G, \alpha l, G_A^3 n^{2c} \epsilon^{-1} q, s)$ with error probability $\leq \epsilon$ with respect to L.*

We will see that if we restrict our attention to games in which Arthur's message length l is $\geq \lg(G_A^3 n^{2c} \epsilon^{-1})$ then the value of the constant α can be taken to be 2. In particular this is true if $l = \omega(\log n)$. Otherwise we have to apply Proposition 2.2 to make $l \geq \lg(G_A^3 n^{2c} \epsilon^{-1})$ and this incurs extra constant factors in coins.

We now proceed to the proof. For notational simplicity we will assume the given game is symmetric and let $g = G/2$ be the number of rounds. We begin with a review of the constructions of pairwise independent sequences. We then give an overview of the protocol and the intuition behind the analysis. We conclude with a more formal proof of correctness.

4.1 Pairwise Independent Generators

For integer m we let $[m] \stackrel{\text{def}}{=} \{1, \ldots, m\}$.

Definition 4.2 Let $X_1, \ldots, X_m$ be random variables defined over a common probability space and assuming values in a common range R. We say that $X_1, \ldots, X_m$ are pairwise independent if $\mathbf{P}[X_{i_1} = x_1, X_{i_2} = x_2] = \mathbf{P}[X_{i_1} = x_1] \cdot \mathbf{P}[X_{i_2} = x_2]$ for all $x_1, x_2 \in R$ and all distinct $i_1, i_2 \in [m]$.

Often we will be interested in random variables which assume all values in their range equiprobably. In the conventional terminology we have

Definition 4.3 A random variable is said to be uniform over a finite set R if it assumes each value in R with probability $1/|R|$.

We will need (deterministic) procedures which take a $O(l)$ bit string s and, in time polynomial in l and m, specify a sequence of $m \leq 2^l$ strings of l bits each with the property that if s is chosen at random then the resulting sequence is pairwise independent and uniform over $\{0,1\}^l$ (cf. [18]). Accordingly, we make the following

Definition 4.4 Let $P(\cdot, \cdot, \cdot)$ be a polynomial time algorithm which outputs strings of length equal to the length of its first input. We say that P is a *pairwise independent generator* if there is a constant $c_P > 0$ such that $P_{l,1}, \ldots, P_{l,2^l}$ are pairwise independent and uniform over $\{0,1\}^l$ for each l, where $P_{l,i}$: $\{0,1\}^{c_P l} \to \{0,1\}^l$ is defined by $P_{l,i}(s) = P(1^l, i, s)$ and is regarded as a random variable over the uniform distribution on $\{0,1\}^{c_P l}$.

It is well known that pairwise independent generators exist. For completeness, let us sketch two implementations.

The first implementation [19] uses finite fields. We identify the sets $\{0,1\}^l$ and $[2^l]$ with $\mathrm{GF}(2^l)$. Using Shoup's (deterministic) algorithm [46] we can find a degree l irreducible polynomial over $\mathrm{GF}(2)$ in time polynomial in l, and this yields the ability to do polynomial time arithmetic in the field. We now regard $s \in \{0,1\}^{2l}$ as the concatenation of two field elements a and b and then let $P(1^l, i, s) = ai + b$, the arithmetic being in the finite field. The pairwise independence follows from the fact that for any pair of points $(i_1, x_1), (i_2, x_2)$ with $i_1 \neq i_2$ there is a unique polynomial of degree ≤ 1 which passes through these points.

It is easy to see that if P is a pairwise independent generator then its associated constant c_P is ≥ 2. So the above implementation is optimal in the number of random bits used.

Goldreich and Levin [28] propose an alternative implementation which, although it uses slightly more random bits, has the advantage of being more efficient in practice. In order to describe it we first recall that a l-by-l matrix $[a_{ij}]$ is *Toeplitz* if $a_{i,j} = a_{i-1,j-1}$ for each $i, j = 2, \ldots, l$, and thus such a matrix is specified by its first row and column. The implementation of [28] consists of using a $3l - 1$ bit seed s to specify an l bit vector b and an l by l $(0,1)$-Toeplitz matrix M and setting $P(1^l, i, s) = Mi + b$, where i is being regarded as a l bit vector and the arithmetic is mod 2.

4.2 The Protocol: Overview

We play $m = g^3 n^{2c} \epsilon^{-1}$ copies of the original game in parallel, with the sequence of strings to play the role of A's messages being pairwise independently distributed. More precisely, in each round t Arthur sends a (random) $c_P l$ bit long seed s_t and this is used, via P, to specify the sequence of m strings $r_t^1 \ldots r_t^m \in \{0,1\}^l$ which will play the role of A's round t messages. A^* accepts iff a majority of the subgames accept.

The idea of the analysis is to guarantee that at each round the sequence of messages specified by the seed approximates, with very high probability, the average accepting probability of a sequence of independently chosen messages. That is, for each $t = 1, \ldots, g$, assuming $s_1, \ldots, s_{t-1}$ have been chosen, we guarantee that with high probability

$$\frac{1}{m}\sum_{i=1}^{m} acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i . r_t^i) \;\approx\; \frac{1}{m}\sum_{i=1}^{m} \mathbf{E}_r\, acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i . r)$$

for the random choice of s_t, where $r_j^1 \ldots r_j^m$ is the sequence specified by s_j. Indeed if for all rounds we can guarantee that all seeds selected provide good approximations in this sense then the fraction of accepting subgames in the new game will approximate the accepting probability in the original game.

More precisely we call a seed s_t *bad* for a history $s_1\vec{y}_1 \cdots s_{t-1}\vec{y}_{t-1}$ if the values $\frac{1}{m}\sum_{i=1}^{m} acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i . r_t^i)$ and $\frac{1}{m}\sum_{i=1}^{m} acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i)$ differ by more than $n^{-c} g^{-1}$, and then show that the fraction of seeds bad for any history is inversely proportional to m. The appropriate choice of m yields a game in which with high probability conversations have all seeds good for their corresponding histories, and these conversations are "representative".

The restriction that ϵ have polynomially bounded inverse comes from the fact that Merlin's message length in the game we construct here is polynomial in ϵ^{-1}.

Let us now describe all this in more detail.

4.3 The Protocol: Specification and Analysis

We fix a pairwise independent generator P and denote by c_P its associated constant.

Definition 4.5 Let $m \le 2^l$ be a polynomial. Then we let A_m denote the Arthur strategy $(\rho_m, g, c_P l, mq, s)$, where

$$\rho_m(w, s_1\vec{y}_1 \cdots s_g\vec{y}_g) = \begin{cases} 1 & \text{if } |\{\, i \in [m] : \rho(w, r_1^i y_1^i \ldots r_g^i y_g^i) = 1 \,\}| \ge \frac{m}{2} \\ 0 & \text{otherwise}. \end{cases}$$

Here $r_t^i = P(1^l, i, s_t)$ and Merlin's round t message $\vec{y}_t \in \{0,1\}^{mq}$ is parsed as $\vec{y}_t = y_t^1 \ldots y_t^m$ with $y_t^i \in \{0,1\}^q$, for $i = 1, \ldots, m$ and $t = 1, \ldots, g$.

We call Arthur's round t message in this game a *seed*, and continue to denote Merlin's round t message by $\vec{y}_t = y_t^1 \ldots y_t^m$ with $y_t^i \in \{0,1\}^q$. We will show that by choosing m to be a suitable polynomial in g and ϵ^{-1} the error probability of A_m with respect to L can be made $\le \epsilon$. We begin with the following

Definition 4.6 Let $h = s_1\vec{y}_1 \cdots s_{t-1}\vec{y}_{t-1}$ be an A_m Arthur history, and $s_t \in \{0,1\}^{c_P l}$ a seed. We say that s_t is *bad* for h if

$$\left| \frac{1}{m} \sum_{i=1}^{m} \left[acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i . r_t^i) - acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i) \right] \right| \ge \frac{1}{n^c g}$$

where $r_j^i = P(1^l, i, s_j)$ for $i = 1, \ldots, m$ and $j = 1, \ldots, t$. We say s_t is *good* for h if it is not bad for h. We say that a A_m conversation $s_1\vec{y}_1 \cdots s_g\vec{y}_g$ is *representative* if for each $t = 1, \ldots, g$ it is the case that s_t is good for $s_1\vec{y}_1 \cdots s_{t-1}\vec{y}_{t-1}$.

The virtue of representative conversations is that they always yield the correct outcome: A_m accepts a representative conversation iff the input is in the language. More precisely, we have the following

Lemma 4.7 *Let $s_1\vec{y}_1 \cdots s_g\vec{y}_g$ be a representative A_m conversation. Then*

(1) *Suppose $w \in L$ and $\vec{y}_t = y_t^1 \ldots y_t^m$ is the particular sequence of messages defined for $t = 1, \ldots, g$ and $i = 1, \ldots, m$ by $y_t^i = M_A^{\mathrm{opt}}(w, r_1^i \ldots r_t^i)$, where $r_t^i = P(1^l, i, s_t)$. Then $\rho_m(w, s_1\vec{y}_1 \cdots s_g\vec{y}_g) = 1$.*

(2) *Suppose $w \notin L$. Then $\rho_m(w, s_1\vec{y}_1 \cdots s_g\vec{y}_g) = 0$.*

Proof: Suppose first that $w \in L$ and $y_t^i = M_A^{\mathrm{opt}}(w, r_1^i \ldots r_t^i)$ where $r_t^i = P(1^l, i, s_t)$. Then

$$acc_A(w) - \frac{1}{m} \sum_{i=1}^{m} acc_A(w, r_1^i y_1^i \ldots r_g^i y_g^i)$$

$$
\begin{aligned}
&= \sum_{t=1}^{g} \frac{1}{m} \sum_{i=1}^{m} \left[acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i) - acc_A(w, r_1^i y_1^i \ldots r_t^i y_t^i)\right] \\
&= \sum_{t=1}^{g} \frac{1}{m} \sum_{i=1}^{m} \left[acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i) - acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i . r_t^i)\right] \\
&< \sum_{t=1}^{g} \frac{1}{m} \sum_{i=1}^{m} \frac{1}{n^c g} \;=\; \frac{1}{n^c} .
\end{aligned}
$$

Here the second equality is by Proposition 2.1 and the inequality is by the fact that the conversation $s_1\vec{y}_1 \cdots s_g\vec{y}_g$ is representative. Noting that $w \in L$ implies $acc_A(w) \geq \frac{1}{2} + n^{-c}$ we get

$$
\frac{1}{m}\sum_{i=1}^{m} acc_A(w, r_1^i y_1^i \ldots r_g^i y_g^i) \;>\; acc_A(w) - \frac{1}{n^c} \;\geq\; \frac{1}{2} ,
$$

and hence $\rho_m(w, s_1\vec{y}_1 \cdots s_g\vec{y}_g) = 1$. The argument for $w \notin L$ is much the same, but for completeness let us give the details. Regardless of the values of $\vec{y}_1, \ldots, \vec{y}_g$ we have

$$
\begin{aligned}
&\frac{1}{m}\sum_{i=1}^{m} acc_A(w, r_1^i y_1^i \ldots r_g^i y_g^i) - acc_A(w) \\
&\quad= \sum_{t=1}^{g} \frac{1}{m} \sum_{i=1}^{m} \left[acc_A(w, r_1^i y_1^i \ldots r_t^i y_t^i) - acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i)\right] \\
&\quad\leq \sum_{t=1}^{g} \frac{1}{m} \sum_{i=1}^{m} \left[acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i . r_t^i) - acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i)\right] \\
&\quad< \sum_{t=1}^{g} \frac{1}{m} \sum_{i=1}^{m} \frac{1}{n^c g} \;=\; \frac{1}{n^c} ,
\end{aligned}
$$

again by Proposition 2.1 and the fact that the conversation $s_1\vec{y}_1 \cdots s_g\vec{y}_g$ is representative. But this time $acc_A(w) \leq \frac{1}{2} - n^{-c}$ and thus we get

$$
\frac{1}{m}\sum_{i=1}^{m} acc_A(w, r_1^i y_1^i \ldots r_g^i y_g^i) \;<\; acc_A(w) + \frac{1}{n^c} \;\leq\; \frac{1}{2} ,
$$

and hence $\rho_m(w, s_1\vec{y}_1 \cdots s_g\vec{y}_g) = 0$. ■

Next we show that the fraction of seeds bad for any particular history is inversely proportional to m.

Lemma 4.8 *Let $m \leq 2^l$ and $h = s_1\vec{y}_1 \cdots s_{t-1}\vec{y}_{t-1}$ a A_m Arthur history. Then at most a $g^2 n^{2c}/m$ fraction of the seeds $s_t \in \{0,1\}^{c_P l}$ are bad for h.*

Proof: Let $r_j^i = P(1^l, i, s_j)$ for $i = 1, \ldots, m$ and $j = 1, \ldots, t-1$. For each $i = 1, \ldots, m$ define X_i: $\{0,1\}^{c_P l} \to [0,1]$ by $X_i(s) = acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i . P(1^l, i, s))$. Note that $X_1, \ldots, X_m$ are pairwise independent when regarded as random variables over the uniform distribution on $\{0,1\}^{c_P l}$. It follows that $\mathrm{Var}[\sum_{i=1}^m X_i] = \sum_{i=1}^m \mathrm{Var}[X_i]$, and since $\mathrm{Var}[X_i] \leq 1$ we have $\mathrm{Var}[\sum_{i=1}^m X_i] \leq m$. So by Chebyshev's inequality we get

$$
\mathbf{P}\left[\, \left|\tfrac{1}{m}\textstyle\sum_{i=1}^m (X_i - \mathbf{E}[X_i])\right| \geq \tfrac{1}{n^c g} \,\right] \;=\; \mathbf{P}\left[\, \left|\textstyle\sum_{i=1}^m (X_i - \mathbf{E}[X_i])\right| \geq \tfrac{m}{n^c g} \,\right]
$$

$$\leq \frac{\mathrm{Var}[\sum_{i=1}^{m} X_i]}{(m/(n^c g))^2}$$
$$\leq \frac{g^2 n^{2c}}{m} .$$

To conclude the proof we need only note that $\mathbf{E}[X_i] = acc_A(w, r_1^i y_1^i \ldots r_{t-1}^i y_{t-1}^i)$ by definition of the accepting probability function. ■

We can use the above lemma to show that representative conversations occur with high probability when m is appropriately chosen, and thereby derive the desired conclusion.

Lemma 4.9 *Let $m = g^3 n^{2c} \epsilon^{-1}$ and assume $m \leq 2^l$. Then A_m has error probability $\leq \epsilon$ with respect to L.*

Proof: Fix any Merlin strategy M for the game defined by A_m. By Lemma 4.8 the probability that a (A_m, M) conversation is representative is at least

$$1 - g \cdot \frac{g^2 n^{2c}}{m} = 1 - \epsilon .$$

The conclusion now follows from Lemma 4.7. ■

If $l \geq \lg(g^3 n^{2c} \epsilon^{-1})$ and we use the finite field implementation of pairwise independent generators then this says we can reduce the error probability to ϵ at the cost of doubling the number of coin tosses used per round. To complete the proof of Theorem 4.1 we need now only note that by Proposition 2.2 we may assume $l \geq \lg(g^3 n^{2c} \epsilon^{-1})$ at the cost of a constant factor in coins.

5 Error Reduction at Logarithmic Cost

We now show how to reduce the error probability to 2^{-k} using $O(k)$ additional random bits per round, where $k = k(n)$ is $\omega(\log n)$ and polynomially bounded. More precisely, we prove the following

Theorem 5.1 *There is a constant α such that the following is true. Let c be > 0 and let $A = (\rho, G, l, q, s)$ be an Arthur strategy with error probability $\leq \frac{1}{2} - n^{-c}$ with respect to the language L. Let $k\colon \mathsf{N} \to \mathsf{N}$ be polynomially bounded and $\omega(\log n)$. Then we can construct another Arthur strategy $A^* = (\rho^*, G, \alpha(l+k), 2000 \cdot G_A^7 k^5 n^{2c} q, s)$ with error probability $\leq 2^{-k}$ with respect to L.*

For the proof again assume for simplicity that the given game is symmetric and let $g = G/2$. The construction uses explicit constructions of expander graphs.

5.1 Expander Graphs and the Expander Path Lemma

Definition 5.2 We call $\mathcal{G} = \{G_l\}_{l \geq 1}$ a *family of graphs* if G_l is for each integer $l \geq 1$ a (undirected) graph on the vertex set $\{0,1\}^{2l}$. We say that $\mathcal{G}$ is *explicitly constructible* if there is a polynomial time algorithm which on input $x \in \{0,1\}^{2l}$ outputs the list of neighbors of x. We say that $\mathcal{G}$ is d-regular if each G_l is d-regular, and bipartite if each G_l is bipartite.

Definition 5.3 Let $\mathcal{G} = \{G_l\}_{l \geq 1}$ be a family of d-regular graphs. We denote by $A(G_l)$ the matrix obtained by dividing every entry of the adjacency matrix of G_l by d and we denote by $\lambda_2(G_l)$ the second eigenvalue of $A(G_l)$. We call $\lambda_2(\mathcal{G}) = \sup_{l \geq 1} \lambda_2(G_l)$ the *second eigenvalue of* $\mathcal{G}$.

Definition 5.4 Let $\mathcal{G}$ be a family of d-regular graphs. We call $\mathcal{G}$ a (family of) *expanders* if $\lambda_2(\mathcal{G}) < 1$.

Gabber and Galil [24] demonstrate the existence of families of explicitly constructible, d-regular, bipartite expanders. It will be convenient for us to assume the degree d is a power of 2. By adding edges we can easily modify the construction of [24] to achieve this; via Alon's result [4] we can be sure that the resulting graph is still an expander. To summarize:

Theorem 5.5 [24] *There exists an explicitly constructible family of d-regular expanders with the degree d being a power of* 2.

Let G be a d-regular undirected graph and A the matrix obtained by dividing every entry of the adjacency matrix of G by d. A random walk on G is the sequence of vertices visited by a "token" which starts at a random vertex and then moves according to the following transition rule: if at time t the token is at vertex x then at time $t+1$ it moves to a random neighbor of x. In other words, if X_t is the random variable which describes the position of the token at time t then $\{X_t\}_{t \geq 0}$ is the Markov chain whose transition probability matrix is A and whose initial position X_0 is uniform over the vertex set of G.

In a *modified* random walk, the token at any point in time first flips a coin, and if the coin value is 1 it performs the random walk transition. Otherwise it stays where it is. In other words its position is described by the Markov chain whose transition probability matrix is $\bar{A} = \frac{1}{2}(I + A)$ (where I is the N by N identity matrix) with the initial position again being uniform over the vertex set.

Note that a modified random walk of length k is specified by $2l + k(1 + \lg d) = 2l + O(k)$ random bits. Let us now be a little more precise.

Definition 5.6 Let d be a power of 2 and G a (undirected) d-regular graph on the vertex set $\{0,1\}^{2l}$. Assume a canonical order on the edges out of any given vertex. Let $x \in \{0,1\}^{2l}$ and for $i = 1, \ldots, k$ let $e_i \in \{0,1\}^{\lg d}$ and $b_i \in \{0,1\}$. The *modified walk* specified by the string $xb_1e_1b_2e_2 \ldots b_ke_k$ is the sequence of vertices $x_0, x_1, \ldots, x_k$ where $x_0 = x$ and

$$x_i = \begin{cases} \text{the } e_i\text{-th neighbor of } x_{i-1} & \text{if } b_i = 1 \\ x_{i-1} & \text{otherwise} \end{cases}$$

for each $i = 1, \ldots, k$. We call k the *length* of the walk and x_i the vertex visited at time i. Picking $xb_1e_1 \ldots b_ke_k$ at random yields a *modified random walk.*

A property of modified random walks on expanders is the main tool of this section.

Lemma 5.7 (Expander Path Lemma) *For any family $\mathcal{G} = \{G_l\}$ of expander graphs there is a constant $\eta \geq 1$ such that the following is true. Let ϵ be $< 1/2$ and let $L = \eta \lg \epsilon^{-1}$. Let $v \in \mathsf{N}$ and let $B_1, \ldots, B_v$ be subsets of the vertex set $\{0,1\}^{2l}$ which have density $\leq \epsilon$. Let b be an integer $\leq v$ and let $1 \leq j_1 < \ldots < j_b \leq v$ be a sequence of indices between 1 and v. Consider a modified random walk of length Lv on the expander and denote by Y_j the vertex visited at time Lj for $j = 1, \ldots, v$. Then $\mathbf{P}[Y_{j_1} \in B_{j_1}, \ldots, Y_{j_b} \in B_{j_b}] \leq (2\epsilon)^{b/2}$. We call η the* expansion constant *of $\mathcal{G}$; it depends only on the second eigenvalue of $\mathcal{G}$.*

A proof of Lemma 5.7, following the ideas of [3], appears in Appendix A. A variant of this lemma appears in [25].

5.2 The Protocol: Overview

We will play in parallel $v = O(k/\log(kg))$ copies of the game presented in §4, setting m such that the error in each of these games is at most an appropriately chosen ϵ. Since each of our subgames consists itself of m subgames, we will be playing a total of vm subgames which are arranged in v blocks each consisting of m subgames. The sequence of v seeds which specify the original Arthur's messages in each round is itself specified by a random walk on the expander.

More precisely, Arthur's message in the t-th round consists of a "super-seed" s_t of length $c_P l + O(k)$ (the constant in the O depends on the second eigenvalue and the degree of the expander). This super-seed is used to specify a random walk of length $O(k)$ on the expander. We denote the vertices visited at intervals of length $L = O(\log \epsilon^{-1})$ by $s_t^1, \ldots, s_t^v \in \{0,1\}^{2l}$. Each s_t^j specifies via P a sequence of m strings of l bit each (as in §4). We denote this sequence by $r_t^{j,1} \ldots r_t^{j,m}$ where $m = g^3 n^{2c} \epsilon^{-1}$. The string $r_t^{j,i}$ is regarded as A's round t move in the i-th subgame of the j-th block. Merlin's (answer) message in round t has the form $y_t^{1,1} \ldots y_t^{1,m} \cdots y_t^{v,1} \ldots y_t^{v,m}$ where $y_t^{j,i}$ is called Merlin's answer in the i-th subgame of the j-th block. After all g rounds are completed Arthur evaluates ρ in each of the subgames and accepts iff in a majority of blocks there is a majority of accepting conversations.

To analyze the game, we remind the reader that in each block of m subgames, for each round, at most a $g^2 n^{2c}/m = \epsilon$ fraction of the seeds are bad for the current history (Lemma 4.8). By the Expander Path Lemma, the probability that a particular sequence of b seeds is bad (in a sequence of v seeds generated by the random walk) is bounded above by $(2\epsilon)^{b/2}$. Thus, the probability that in the g rounds of the game at least $b = v/2$ of the vg seeds are bad is bounded above by $\binom{vg}{b} \cdot (2\epsilon)^{b/2}$ and by appropriate choices of the parameters this can be made $\leq 2^{-k}$. It follows that with probability $\geq 1 - 2^{-k}$, the conversations in $\geq 1/2$ of the blocks are representative. We can conclude by applying the analysis of §4.

Let us now specify all this in more detail.

5.3 The Protocol: Specification and Analysis

We fix an explicitly constructible family of d-regular expander graphs $\{G_l\}$ with d being a power of 2, and let η denote its expansion constant. We also fix a pairwise independent generator P

whose associated constant c_P we assume (without loss of generality) to be even. To specify the new Arthur strategy we first define the parameters

$$\begin{aligned}
\epsilon &= \frac{1}{200\,k^4 g^4} \\
L &= \eta \lg \epsilon^{-1} && = O(\log(kg)) \\
v &= \frac{10\eta k}{L} && = \frac{10k}{\lg \epsilon^{-1}} && = O\left(\frac{k}{\log(kg)}\right) \\
m &= g^3 n^{2c} \epsilon^{-1} \\
\beta &= 10\eta(1+\lg d) && = O(1) && .
\end{aligned}$$

Definition 5.8 For $t = 1, \ldots, g$ let $s_t \in \{0,1\}^{c_P l + \beta k}$ denote Arthur's round t message; we call s_t a *super-seed*. For $j = 1, \ldots, v$ let $s_t^j \in \{0,1\}^{c_P l}$ be the vertex visited at time Lj by the modified random walk on $G_{c_P l/2}$ that is specified by s_t. For $i = 1, \ldots, m$ let $r_t^{j,i} = P(1^l, i, s_t^j)$. We let A^* denote the Arthur strategy $(\rho^*, G, c_P l + \beta k, mvq, s)$, where

$$\rho^*(w, s_1 \vec{y}_1^1 \ldots \vec{y}_1^v \cdots s_g \vec{y}_g^1 \ldots \vec{y}_g^v) = \begin{cases} 1 & \text{if } |\{\, j \in [v] : \rho_m(w, s_1^j \vec{y}_1^j \cdots s_g^j \vec{y}_g^j) = 1 \,\}| \geq \frac{v}{2} \\ 0 & \text{otherwise .} \end{cases}$$

Here ρ_m is the decision predicate of the game of Definition 4.5 and the mvq bit string that is Merlin's round t message is being parsed as $\vec{y}_t^1 \ldots \vec{y}_t^v$ where $\vec{y}_t^j = y_t^{j,1} \ldots y_t^{j,m}$ with $y_t^{j,i} \in \{0,1\}^q$.

Definition 5.9 If $C = s_1 \vec{y}_1^1 \ldots \vec{y}_1^v \cdots s_g \vec{y}_g^1 \ldots \vec{y}_g^v$ is a A^* conversation and $j \in [v]$, we call $s_1^j \vec{y}_1^j \cdots s_g^j \vec{y}_g^j$ a *sub-conversation* of C.

Note that a sub-conversation of a A^* conversation is an A_m conversation. Our goal will be to show that if M is any Merlin strategy for the game defined by A^* then at least half the sub-conversations of each (A^*, M) conversation are representative with probability $\geq 1 - 2^{-k}$. We begin with the following

Definition 5.10 Let M be a Merlin strategy for the game defined by A^*. Let $s_1, \ldots, s_{t-1}$ be a sequence of super-seeds, and let $\vec{y}_u^1 \ldots \vec{y}_u^v = M(w, s_1 \ldots s_u)$ for $u = 1, \ldots, t-1$. For each block $j = 1, \ldots, v$ and each round $t = 1, \ldots, g$ let

$$B_t^j(M; s_1, \ldots, s_{t-1}) \;=\; \{\, s_t^j \in \{0,1\}^{c_P l} : s_t^j \text{ is bad for the } A_m \text{ history } s_1^j \vec{y}_1^j \cdots s_{t-1}^j \vec{y}_{t-1}^j \,\}$$

denote the set of seeds which are bad for the current history in this block.

We note that if $C = s_1 \vec{y}_1^1 \ldots \vec{y}_1^v \cdots s_g \vec{y}_g^1 \ldots \vec{y}_g^v$ is a (A^*, M) conversation and for each $t = 1, \ldots, g$ it is the case that $s_t^j \notin B_t^j(M; s_1, \ldots, s_{t-1})$ then the sub-conversation $s_1^j \vec{y}_1^j \cdots s_g^j \vec{y}_g^j$ is representative.

Lemma 5.11 *Let M be a Merlin strategy for the game defined by A^*. Then*

$$\mathbf{P}\left[\, |\{\, (t,j) \in [g] \times [v] : s_t^j \in B_t^j(M; s_1, \ldots, s_{t-1}) \,\}| \geq \tfrac{v}{2} \,\right] \;\leq\; 2^{-k} ,$$

where the probability is over Arthur's random choice of $s_1, \ldots, s_g$.

Proof: By Lemma 4.8 and our choice of ϵ we know that the density of $B_t^j(M; s_1, \ldots, s_{t-1})$ is at most ϵ for each block j and round t. Now fix t and assume $s_1, \ldots, s_{t-1}$ have been chosen. Let $b \leq v$ and suppose $1 \leq j_1 < \ldots < j_b \leq v$ is a sequence of b indices between 1 and v. By Lemma 5.7 we know that

$$\mathbf{P}\left[\, s_t^{j_b} \in B_t^{j_1}(M; s_1, \ldots, s_{t-1}), \ldots, s_t^{j_b} \in B_t^{j_b}(M; s_1, \ldots, s_{t-1}) \right] \;\leq\; (2\epsilon)^{b/2} \,,$$

the probability being over Arthur's random choice of s_t. It follows that

$$\mathbf{P}\left[\, |\{\, (t, j) \in [g] \times [v] \,:\, s_t^j \in B_t^j(M; s_1, \ldots, s_{t-1}) \,\}| \geq \tfrac{v}{2} \right] \;\leq\; \binom{vg}{v/2} \cdot (2\epsilon)^{v/4} \,,$$

the probability being over Arthur's random choice of $s_1, \ldots, s_g$. We bound this last expression by $(vg)^{v/2}(2\epsilon)^{v/4} = (vg\sqrt{2\epsilon}\,)^{v/2}$. Substituting $\sqrt{2\epsilon} = (10k^2g^2)^{-1}$ and $v = 10k/\lg \epsilon^{-1}$ this is

$$\leq\; \left(\frac{10kg}{10k^2g^2 \lg \epsilon^{-1}}\right)^{\frac{10k}{2\lg \epsilon^{-1}}} \;=\; \left(\frac{1}{kg \lg \epsilon^{-1}}\right)^{\frac{5k}{\lg \epsilon^{-1}}} \;=\; \left(2^{-k}\right)^{\frac{5 \lg(kg \lg \epsilon^{-1})}{\lg \epsilon^{-1}}} \,.$$

But $1 \leq \lg \epsilon^{-1} \leq O(1) + 4\lg(kg)$ so $5\lg(kg \lg \epsilon^{-1}) \geq \lg \epsilon^{-1}$ for large enough n and hence the above is $\leq 2^{-k}$ as desired. ■

Now fix an arbitrary Merlin strategy for the game defined by A^*. Lemma 5.11 implies that

$$\mathbf{P}\left[\, |\{\, j \in [v] \,:\, s_t^j \notin B_t^j(M; s_1, \ldots, s_{t-1}) \text{ for all } t = 1, \ldots, g \,\}| > \tfrac{v}{2} \right] \;\geq\; 1 - 2^{-k} \,,$$

or, in other words, at least half the sub-conversations of any (A^*, M) conversation are representative with probability $\geq 1 - 2^{-k}$. So by Lemma 4.7 and the definition of ρ^* it follows that the error probability of A^* with respect to L is $\leq 2^{-k}$. This concludes the proof of Theorem 5.1.

6 Randomness-Efficient Approximation

Implicit in the previous sections is a new sampling method. An application of particular interest is to the problem of approximating the average value of an arbitrary real valued function.

6.1 Definitions

For $f\colon \{0,1\}^n \to [0,1]$ we let $\mathbf{E}[f] \stackrel{\text{def}}{=} 2^{-n} \sum_{x \in \{0,1\}^n} f(x)$ denote the average value of f.

An (l, ϵ, δ)-approximator is a two stage process. In a first, randomized, *sampling* stage, we pick a collection of sample points $x_1, \ldots, x_t \in \{0,1\}^l$. In a second, deterministic, *estimation* stage, we compute, as a function of $f(x_1), \ldots, f(x_t)$, an estimate. We require that with probability $\geq 1 - \delta$ this estimate is within ϵ of $\mathbf{E}[f]$.

Let us now specify this more precisely. We begin with the sampling stage.

Definition 6.1 Let $l, t, \sigma\colon \mathsf{N} \to \mathsf{N}$. An (l, t, σ)-*sampler* is a (deterministic) algorithm $S(\cdot, \cdot)$ which runs in time polynomial in its first input, and, on input 1^n and a *seed* $s \in \{0,1\}^{\sigma(n)}$, outputs a sequence of strings $r_1, \ldots, r_{t(n)} \in \{0,1\}^{l(n)}$ which we call *sample points.* The *number of sample points* is t. For $i = 1, \ldots, t(n)$ we will denote by $S_i(1^n, s)$ the i-th sample point output by S on input 1^n and $s \in \{0,1\}^{\sigma(n)}$.

As we will see when we complete the specification of (l, ϵ, δ)-approximation, the seed for the sampler will be chosen at random. So the seed length σ will represent the number of coin tosses that the sampler uses.

We now turn to the estimation stage.

Definition 6.2 Let $l, t \colon \mathsf{N} \to \mathsf{N}$. A (l,t)-*estimator* is a (deterministic) polynomial time algorithm $E(\cdot,\cdot)$ which on input 1^n and $(x_1, \ldots, x_{t(n)}) \in \mathsf{R}^{l(n)}$ outputs a real number between 0 and 1.

If F_n is for each integer n a set of functions mapping $\{0,1\}^n$ to $[0,1]$ then we will refer to $\mathcal{F} = \{F_n\}$ as a *class* of functions. We can now say what is an (l, ϵ, δ)-approximator for a class of functions $\mathcal{F}$.

Definition 6.3 Let $l \colon \mathsf{N} \to \mathsf{N}$ and $\epsilon, \delta \colon \mathsf{N} \to [0,1]$. Let S be a (l, t, σ)-sampler and E a (l,t)-estimator, for some $t, \sigma : \mathsf{N} \to \mathsf{N}$. For each n and $f \colon \{0,1\}^{l(n)} \to [0,1]$ we define $A_{n,f} \colon \{0,1\}^{\sigma(n)} \to [0,1]$ by

$$A_{n,f}(s) \;=\; E(1^n, (f(S_1(1^n,s)), \ldots, f(S_{t(n)}(1^n,s))))$$

and regard it as a random variable over the uniform distribution on $\{0,1\}^{\sigma(n)}$. We say that $A = (S,E)$ is an (l, ϵ, δ)-*approximator for the class of functions* $\mathcal{F} = \{F_n\}$ if

$$\mathbf{P}\left[\, |A_{n,f} - \mathbf{E}[f]| \le \epsilon(n) \,\right] \;\ge\; 1 - \delta(n)$$

for each n and each $f \in F_n$. The *number of sample points* used is t and the *number of coin tosses* used is σ.

Note that both S and E are independent of the function f we are trying to approximate.

We assume that the real numbers $f(r_1), \ldots, f(r_t)$ that are provided to the estimator are truncated to $O(\log \epsilon^{-1})$ bits; this level of approximation is suitable for our purposes and henceforth we will assume total accuracy.

The parameters to consider in designing an (l, ϵ, δ)-approximator are the number of sample points t (which by the above definition is also the number of function evaluations), the number of coin tosses σ used (by S) to generate these sample points, and the class $\mathcal{F}$ of functions that can be approximated.

For the purpose of the discussion that follows it is helpful to have the following

Definition 6.4 [10] An (l, ϵ, δ)-approximator (S,E) is called *oblivious* if

$$E(1^n, (x_1, \ldots, x_{t(n)})) \;=\; \frac{1}{t(n)} \sum_{i=1}^{t(n)} x_i$$

for all n and $(x_1, \ldots, x_{t(n)}) \in \mathsf{R}^{l(n)}$, where t is the number of sample points used by (S,E).

Obliviousness is useful in some applications (cf. [10]).

6.2 Approximation Methods

The standard approximation method is to select $t = O(\epsilon^{-2} \log \delta^{-1})$ independent and uniformly distributed sample points and use as estimate the average value of the function on these sample points. An application of the Chernoff bound shows that this yields an (l, ϵ, δ)-approximator which works for all functions. However, it uses $O(tl)$ coin tosses.

A large savings in the number of coin tosses is possible by the use of pairwise independence. With $O(l)$ coin tosses we can specify $O(\epsilon^{-2}\delta^{-1})$ pairwise independent sample points and again use as estimate the average value of the function on these sample points; an application of Chebyshev's inequality shows that this yields a (l, ϵ, δ)-approximator for all functions (cf. Chor and Goldreich [19]). The cost is in the number of sample points which now grows in proportion to δ^{-1} rather than the $\lg \delta^{-1}$ of the standard method; if we want the number of sample points to be polynomial in n then we cannot attain exponentially small (in n) error.

Using higher independence and an iterated sampling technique, Bellare and Rompel [10] are able to remove this restriction. They construct an (l, ϵ, δ)-approximator for all functions which, with $O(l + \log \delta^{-1} \cdot \log l)$ coin tosses, specifies $O(\epsilon^{-6} \log^6 l + \log^3 \delta^{-1})$ sample points and then uses as estimate the average value of the function on these sample points. Thus they do save coins compared to the standard method and use a number of sample points which is polynomial in (l, ϵ^{-1} and) $\lg \delta^{-1}$, but the number of sample points is much more than that of the standard method.

Using the techniques of Impagliazzo and Zuckermann [37] one can construct an (l, ϵ, δ)-approximator for the class of boolean functions which uses $O(\epsilon^{-2} \log \delta^{-1})$ sample points and $l + O(\epsilon^{-4} \log^2 \delta^{-1})$ coin tosses as long as $l = \omega(\epsilon^{-2} \log \delta^{-1})$.

The (l, ϵ, δ)-approximator that we will present in §6.3 improves on all of these: with only $O(l + \log \delta^{-1})$ coin tosses it specifies the same (up to constant factors) $O(\epsilon^{-2} \log \delta^{-1})$ number of sample points as the standard method, and works for all functions. It is optimal in the number of sample points and coin tosses. We note, however, that our approximator will not be oblivious, in contrast to all the ones mentioned above.

A somewhat coarser kind of approximation to the average of a boolean functions $f\colon \{0,1\}^l \to \{0,1\}$ can be obtained via a random walk on a 2^l node explicitly constructed expander graph (cf. [3, 21, 37]). For example, if $\mathbf{E}[f] \geq 99/100$ then we can use $O(l + \log \delta^{-1})$ random bits to generate $O(\log \delta^{-1})$ sample points such that the average of f on these sample points is $\geq 2/3$ with probability $\geq 1 - \delta$. This is useful for many applications such as randomness-efficient error-reduction for BPP algorithms.

Other randomness-efficient sampling techniques include that of Nisan [41] who shows how $O(l \log t)$ random bits suffice to produce a sequence of l bit strings $r_1, \ldots, r_t$ such that $|\mathbf{P}[f(r_1) = \cdots = f(r_t) = 1] - \mathbf{E}[f]^t| < 2^{-l}$ for all boolean functions $f\colon \{0,1\}^l \to \{0,1\}$. Recently Impagliazzo [35] presented a general technique to extend these kinds of results to real valued functions.

6.3 Our (l, ϵ, δ)-approximator

We construct an (l, ϵ, δ)-approximator which uses $O(l + \log \delta^{-1})$ coin tosses and $O(\epsilon^{-2} \log \delta^{-1})$ sample points, and works for all functions.

Up to constant factors, the number of sample points used by our method is optimal. Namely, any (l, ϵ, δ)-approximator (even for just the class of boolean functions) must use $\Omega(\epsilon^{-2} \log \delta^{-1})$ sample points (cf. [17]). Furthermore, among methods using the optimal number of sample points, our method is nearly optimal in the number of coin tosses used (for a wide range of natural choices for the parameters). Namely, any (l, ϵ, δ)-approximator (even for boolean functions) which uses $O(\epsilon^{-2} \log \delta^{-1})$ sample points must use at least $l + \lg \delta^{-1} - \lg \epsilon^{-2} - O(1)$ coin tosses (cf. [17] and consider $\delta < \epsilon^{\omega(1)}$).

Let us now proceed to our theorem.

Theorem 6.5 *Let $l\colon \mathsf{N} \to \mathsf{N}$ and $\epsilon, \delta\colon \mathsf{N} \to [0,1]$ with l, ϵ^{-1} and $\lg \delta^{-1}$ polynomially bounded and polynomial time computable. Then there exists an (l, ϵ, δ)-approximator for all functions which uses $O(\epsilon^{-2} \log \delta^{-1})$ sample points and $O(l + \log \delta^{-1})$ coin tosses.*

Proof: We fix an explicitly constructible family of d-regular expander graphs $\{G_l\}$ with d being a power of 2, and let η denote its expansion constant. We also fix a pairwise independent generator P whose associated constant c_P we assume (without loss of generality) to be even. We define the parameters

$$\begin{array}{rcl}
L &=& 9\eta \quad = O(1) \\
v &=& \lg \delta^{-1} \\
m &=& 2^9 \epsilon^{-2} \quad = O(\epsilon^{-2}) \\
t &=& vm \quad = O(\epsilon^{-2} \lg \delta^{-1}) \\
\beta &=& 9\alpha \lg d \quad = O(1)
\end{array}$$

If $m > 2^l$ then we can trivially get a perfect estimate without expending any coin tosses because we are allowed enough sample points that exact computation becomes feasible. Formally, the sampler would output a list of all the $2^l \le t$ points in $\{0,1\}^l$, and the estimator on input 1^n and $(x_1, \ldots, x_t)$ would output the average of the t values $x_1, \ldots, x_t$. So we may assume $m \le 2^l$.

The sampler S chooses $s \in \{0,1\}^{c_P l + \beta \lg \delta^{-1}}$ at random. Let s^j denote the vertex visited at time Lj on the (modified) random walk on $G_{c_P l/2}$ that is specified by s, and let $r^{j,i} = P(1^n, i, s^j)$ for $j = 1, \ldots, v$ and $i = 1, \ldots, m$. The output of the sampler is $r^{1,1}, \ldots, r^{1,m}, \ldots, r^{v,1}, \ldots, r^{v,m}$. Thus the approximator uses $vm = O(\epsilon^{-2} \lg \delta^{-1})$ sample points and $c_P l + \beta \lg \delta^{-1} = O(l + \lg \delta^{-1})$ coin tosses.

The estimator E, on input 1^n and $(x^{1,1}, \ldots, x^{1,m}, \ldots, x^{v,1}, \ldots, x^{v,m})$, evaluates $x^j = \frac{1}{m} \sum_{i=1}^m x^{j,i}$ for $j = 1, \ldots, v$ and then outputs the median value of $x^1, \ldots, x^v$.

For the analysis, fix $f\colon \{0,1\}^l \to [0,1]$. Call a seed $s \in \{0,1\}^{c_P l}$ *bad* if

$$\left| \frac{1}{t} \sum_{i=1}^{t} f(r^i) - \mathbf{E}[f] \right| \ge \epsilon ,$$

where $r^i = P(1^n, i, s)$ for $i = 1, \ldots, m$. We know that $m \le 2^l$, so if s is selected at random then $f(r^1), \ldots, f(r^m)$ are pairwise independent random variables. By Chebyshev's inequality it follows that

$$\mathbf{P}\left[\, |\tfrac{1}{t} \textstyle\sum_{i=1}^t f(r^i) - \mathbf{E}[f]| \ge \epsilon \,\right] \;\le\; \frac{m}{m^2 \epsilon^2} = 2^{-9} ,$$

the probability being over a random choice of the seed. In other words, the density of $B^j \stackrel{\text{def}}{=} \{ s : s \text{ is bad} \}$ is $\leq 2^{-9}$ for each $j = 1, \ldots, v$. By Lemma 5.7, the probability that a majority of the seeds are bad is bounded above by

$$\binom{v}{v/2} \cdot (2 \cdot 2^{-9})^{v/4} \;\leq\; (2 \cdot (2^{-8})^{1/4})^v \;=\; 2^{-\log \delta^{-1}} \;=\; \delta\ .$$

Thus when $x^{j,i} = f(r^{j,i})$, with probability $\geq 1 - \delta$ a majority of the values x^j are within ϵ of $\mathbf{E}[f]$. So with probability $\geq 1 - \delta$ the median of these values is within ϵ of $\mathbf{E}[f]$. ■

7 Conclusion and Open Problems

We recall our main result. Suppose we are given an Arthur-Merlin game of $g = g(n)$ rounds in which Arthur sends $l = l(n)$ (random) bits per round, and suppose this game is a proof system for L with error probability $\leq \frac{1}{3}$. Then given a polynomially bounded function $k = k(n)$ we can construct a new Arthur Merlin games in which the error probability is reduced to 2^{-k} while maintaining the number of rounds and using $O(l + k)$ coin tosses per round. Here are some areas for further investigation.

- For unbounded g, is there a construction which achieves error 2^{-k} with Arthur flipping a *total* of only $O(gl+k)$ coins? Note that the total number of coins flipped by Arthur in the given game is gl, and the total number of coins flipped in the new game that our result yields is $O(g(l+k))$.

  We know that the analogue of the question we pose is true for BPP algorithms. Namely, a BPP algorithm which has error $\leq \frac{1}{3}$ with respect to L and uses $r = r(n)$ coins can be transformed into one which has error $\leq 2^{-k}$ and uses $O(r + k)$ coins [3, 21, 37].

- Can our result be generalized to (general) interactive proof systems? We don't know of any direct non-trivial error-reduction technique for general interactive proof systems. By "direct" we mean without first transforming the interactive proof into an Arthur-Merlin game (a transformation which requires exponentially small error probability), and by "non-trivial" we mean more efficient in terms of coin tosses than playing independent copies of the same interactive proof.

- Is it possible to achieve error $\leq 2^{-k}$ using only $l + O(k)$ coins per round? Note that the best our technique yields is $2l + O(k)$.

  Again, the analogue is true for BPP. A BPP algorithm which has error $\leq \frac{1}{3}$ with respect to L and uses $r = r(n)$ coins can be transformed, for any constant $c > 0$, into one which has error $\leq n^{-c}$ and still uses only r coins (a result of [38] described in [21]) and one can then apply [21, 37] to reduce the error to 2^{-k} at the cost of $O(k)$ additional coins.

Acknowledgments

We thank John Rompel for many valuable ideas and discussions in the early stages of this research, and Silvio Micali for helpful discussions.

References

[1] L. Adleman. Two Theorems on Random Polynomial Time. *Proceedings of the 19th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1978).

[2] W. Aiello, S. Goldwasser and J. Håstad. On the Power of Interaction. *Combinatorica* **10**(1), 3–25 (1990).

[3] M. Ajtai, J. Komlos and E. Szemeredi. Deterministic Simulation in Logspace. *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing,* ACM (1987).

[4] N. Alon. Eigenvalues and Expanders. *Combinatorica* **6**(2), 83–96 (1986).

[5] L. Babai. Trading Group Theory for Randomness. *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing,* ACM (1985).

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

[7] L. Babai and S. Moran. Proving Properties of Interactive Proofs by a Generalized Counting Technique. *Information and Computation* **82**, 185–197 (1989).

[8] M. Bellare, S. Micali and R. Ostrovsky. Perfect Zero-Knowledge in Constant Rounds. *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing,* ACM (1990).

[9] M. Bellare, S. Micali and R. Ostrovsky. The True Complexity of Statistical Zero-Knowledge. *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing,* ACM (1990).

[10] M. Bellare and J. Rompel. Randomness-Efficient Sampling of Arbitrary Functions. *MIT LCS Tech. Memo.* TM–433.

[11] C. Bennet and J. Gill. Relative to a random Oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \mathrm{coNP}^A$, with probability 1. *SIAM J. on Computing* **10**, 96–113 (1981).

[12] M. Blum, A. De Santis, S. Micali and G. Persiano. Non-Interactive Zero-Knowledge. *MIT LCS Tech. Memo.* TM–430.b.

[13] M. Blum, P. Feldman and S. Micali. Non-Interactive Zero-Knowledge and its Applications. *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing,* ACM (1988).

[14] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing* **13** (4), 850-864 (1984).

[15] R. Boppana, J. Hastad and S. Zachos. Does coNP have Short Interactive Proofs? *Info. Processing Letters* **25**, 127–132 (1987).

[16] G. Brassard, C. Crépeau and M. Yung. Everything in NP can be Argued in Perfect Zero-Knowledge in a Constant Number of Rounds. *Proceedings of ICALP 89*.

[17] R. Canetti, G. Even and O. Goldreich. Lower bounds for sampling algorithms for estimating the average. TR-686, Computer Science Dept., Technion, Haifa, Israel, August 1991.

[18] L. Carter and M. Wegman. Universal Classes of Hash Functions. *J. Computer and System Sciences* **18**, 143–154 (1979).

[19] B. Chor and O. Goldreich. On the Power of Two–Point Based Sampling. *J. of Complexity* **5**, 96–106 (1989).

[20] B. Chor, O. Goldreich and J. Hastad. The Random Oracle Hypothesis is False. TR-631, Computer Science Dept., Technion, Haifa, Israel, March 1990.

[21] A. Cohen and A. Wigderson. Dispersers, Deterministic Amplification, and Weak Random Sources. *Proceedings of the 30th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1989).

[22] S. Cook. The Complexity of Theorem Proving Procedures. *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing,* ACM (1971).

[23] U. Feige and A. Shamir. Witness Indistinguishable and Witness Hiding Protocols. *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing,* ACM (1990).

[24] O. Gabber and Z. Galil. Explicit Construction of Linear Sized Superconcentrators. *J. Computer and System Sciences* **22**, 407–420 (1981).

[25] O. Goldreich, R. Impagliazzo, L. Levin, R. Venkatesan and D. Zuckermann. Security Preserving Amplification of Hardness. *Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1990).

[26] O. Goldreich and A. Kahan. Using Claw-Free Permutations to Construct Constant Round Zero-Knowledge Proofs for NP.

[27] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *Proceedings of ICALP 90,* Lecture Notes in Computer Science Vol. 443, Springer Verlag (1990).

[28] O. Goldreich and L. Levin. A Hard-Core Predicate for any One-Way Function. *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing,* ACM (1989).

[29] O. Goldreich, Y. Mansour, and M. Sipser. Interactive Proof Systems: Provers that never Fail and Random Selection. *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1987).

[30] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity. Technical Report #498, Technion, 1988. Preliminary version in *Proceedings of the 27th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1986).

[31] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proofs. *SIAM J. Computing* **18**(1), 186–208 (1989).

[32] S. Goldwasser and M. Sipser. Private Coins versus Public Coins in Interactive Proof Systems. *Proceedings of the 18th Annual ACM Symposium on the Theory of Computing,* ACM (1986).

[33] J. Hartmanis, R. Chang, D. Ranjan and P. Rohatgi. Structural Complexity Theory: Recent Surprizes. *Technical Report 90-1117, Dept. of Computer Science, Cornell University* (April 1990).

[34] J. Håstad. Pseudo-random generators under Uniform Assumptions. *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing,* ACM (1990).

[35] R. Impagliazzo. Private communication (December 1990).

[36] R. Impagliazzo, L. Levin and M. Luby. Pseudo-Random Generation from One-Way Functions. *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing,* ACM (1989).

[37] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. *Proceedings of the 30th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1989).

[38] R. Karp, N. Pippinger and M. Sipser. A Time-Randomness Tradeoff. In *AMS Conference on Probabilistic Computational Complexity*, Durham, New Hampshire (1985).

[39] A. Lubotzky, R. Phillips and P. Sarnak. Explicit Expanders and the Ramanujan Conjectures. *Proceedings of the 18th Annual ACM Symposium on the Theory of Computing,* ACM (1986).

[40] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1990).

[41] N. Nisan. Pseudorandom Generators for Space Bounded Computation. *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing,* ACM (1990).

[42] N. Nisan and A. Wigderson. Hardness vs. Randomness. *Proceedings of the 29th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1988).

[43] M. Santha. On Using Deterministic Functions to Reduce Randomness in Probabilistic Algorithms. *Information and Computation* **74**(3), 241–249 (1987).

[44] A. Schrift. Chapter 4 of Ph.D Thesis, Dept. of Applied Mathematics and Computer Science, Weizmann Institute.

[45] A. Shamir. IP=PSPACE. *Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1990).

[46] V. Shoup. New Algorithms for Finding Irreducible Polynomials over Finite Fields. *Proceedings of the 29th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1988).

[47] M. Sipser. Expanders, Randomness, or Time Versus Space. *Structure in Complexity Theory* (1986).

[48] A. Yao. Theory and Applications of Trapdoor Functions. *Proceedings of the 23rd Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1982).

A Appendix: Proof of the Expander Path Lemma

Fix l and let $N = 2^{2l}$ denote the number of nodes of G_l. Let $A = A(G_l)$ and $\delta = 1 - \lambda_2(\mathcal{G})$. We denote by $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$ the eigenvalues of A. The fact that all line sums of A equal 1 implies that the first eigenvalue λ_1 equals 1 and the spectral radius of A (maximum of the absolute values of all the eigenvalues of A) is 1. Moreover, the expander property says that $\lambda_2 \leq 1 - \delta < 1$.

We let $\bar{A} = \frac{1}{2}(I + A)$ denote the transition probability matrix of the modified random walk. Its eigenvalues are $\bar{\lambda}_1 \geq \bar{\lambda}_2 \geq \ldots \geq \bar{\lambda}_N$ where $\bar{\lambda}_i = \frac{1}{2}(1 + \lambda_i)$. It follows that the first eigenvalue of $\bar{A}$ is 1, all its eigenvalues are ≥ 0, and $\bar{\lambda}_2 \leq 1 - \delta/2 < 1$.

Since $\bar{A}$ is a real symmetric matrix there is an orthonormal basis of R^N which consists of eigenvectors of $\bar{A}$. We fix such a basis $u_1, \ldots, u_N$ with u_i being an eigenvector of $\bar{A}$ with eigenvalue $\bar{\lambda}_i$. Note that $\bar{\lambda}_1 = 1$ implies that $u_1 = (N^{-1/2}, \ldots, N^{-1/2})$. We let V_1 be the space spanned by u_1 and V_2 the space orthogonal to V_1 which is spanned by $u_2, \ldots, u_N$.

Let $\|x\|$ denote the Euclidean norm of $x \in \mathsf{R}^N$.

Lemma A.1 $\|\bar{A}x\| \leq (1 - \delta/2)\|x\|$ *for any* $x \in V_2$*.*

Proof: Since $u_2, \ldots, u_N$ is a basis for V_2 there are real numbers $c_2, \ldots, c_N$ such that $x = \sum_{s=2}^N c_s u_s$. But $\bar{A}u_s = \bar{\lambda}_s u_s$ and the vectors $u_2, \ldots, u_N$ are orthonormal, so

$$\|\bar{A}x\|^2 \;=\; \|\textstyle\sum_{s=2}^N c_s \bar{A} u_s\|^2 \;=\; \|\textstyle\sum_{s=2}^N c_s \bar{\lambda}_s u_s\|^2 \;=\; \textstyle\sum_{s=2}^N c_s^2 \bar{\lambda}_s^2 \;.$$

Since $\bar{\lambda}_2 \geq \bar{\lambda}_3 \geq \ldots \geq \bar{\lambda}_N \geq 0$ this implies

$$\|\bar{A}x\|^2 \;\leq\; \bar{\lambda}_2^2 \textstyle\sum_{s=2}^N c_s^2 \;=\; \bar{\lambda}_2^2 \|x\|^2 \;\leq\; (1 - \delta/2)^2 \|x\|^2$$

which proves the lemma. ■

Let e_i be the N-vector with 1 in position i and zeroes elsewhere. Define the projection matrix P_j as having its i-th column equal to e_i if $i \in B_j$ and the 0 vector otherwise. Let $\eta \stackrel{\text{def}}{=} \frac{1}{2}[\lg \frac{2}{2-\delta}]^{-1}$ and note that this is a constant which depends only on the second eigenvalue of $\mathcal{G}$. Let $L \stackrel{\text{def}}{=} \eta \lg \epsilon^{-1}$.

Lemma A.2 $\|P_j \bar{A}^i x\| \leq \sqrt{2\epsilon}\,\|x\|$ *for any* $x \in \mathsf{R}^N$ *and any* $j = 1, \ldots, v$ *and* $i \geq L$*.*

Proof: Let $x = x_1 + x_2$ where $x_1 = c_1 u_1 \in V_1$ and $x_2 \in V_2$. Then

$$\begin{aligned}
\|P_j \bar{A}^i x\| &\leq \|P_j \bar{A}^i x_1\| + \|P_j \bar{A}^i x_2\| \\
&\leq \|P_j x_1\| + \|\bar{A}^i x_2\| \\
&\leq [2\,(\|P_j x_1\|^2 + \|\bar{A}^i x_2\|^2)]^{1/2} \;.
\end{aligned}$$

Here the first inequality is by the triangle inequality. The second uses the fact that $\bar{A}x_1 = x_1$ and $\|P_j y\| \leq \|y\|$ for any $y \in \mathsf{R}^n$. The third is just an application of the inequality $a + b \leq [2(a^2 + b^2]^{1/2}$. Now the fact that B_j has density $\leq \epsilon$ and $x_1 = c_1 u_1$ implies that $\|P_j x_1\|^2 \leq \epsilon \|x_1\|^2$. On the other hand, since A maps V_2 into itself we can apply Lemma A.1 repeatedly to conclude that $\|\bar{A}^i x_2\| \leq (1 - \delta/2)^i \|x_2\|$. Our choice of L implies that $(1 - \delta/2)^{2L} \leq \epsilon$ and hence it follows that

$\|\bar{A}^i x_2\|^2 \le \epsilon \|x_2\|^2$. Putting all this together we get

$$\|P_j \bar{A}^i x\| \;\le\; [2\,(\epsilon\|x_1\|^2 + \epsilon\|x_2\|^2)]^{1/2} = \sqrt{2\epsilon}\,\|x\|$$

as desired. ■

Let $\|x\|_1$ denote the L_1 norm (that is, the sum of the absolute values of the components) of $x \in \mathsf{R}^N$. Now let $x = (1/N, \ldots, 1/N) = N^{-1/2} u_1$ be the N vector corresponding to the uniform distribution and set

$$y \;=\; P_{j_b} \bar{A}^{(j_b - j_{b-1}) \cdot L} \cdots P_{j_2} \bar{A}^{(j_2 - j_1) \cdot L} P_{j_1} \bar{A}^{j_1 \cdot L} x \;.$$

Lemma A.2 implies that $\|y\| \le (2\epsilon)^{b/2}\|x\| = (2\epsilon)^{b/2} N^{-1/2}$. Thus, the probability that a random walk, starting at the uniform distribution x, and terminating after Lv steps at distribution y, visits a vertex in the set B_{j_i} at time Lj_i for $i = 1, 2, ..., b$ is

$$\|y\|_1 \;\le\; \sqrt{N}\,\|Y\| \;\le\; (2\epsilon)^{b/2} \;.$$