2-LOCAL RANDOM REDUCTIONS TO 3-VALUED FUNCTIONS

A. Pavan and N. V. Vinodchandran

Abstract. Yao (in a lecture at DIMACS Workshop on structural complexity and cryptography, 1990) showed that if a language L is 2-locally random reducible to a *Boolean function*, then $L \in \text{PSPACE/}\texttt{poly}$. Fortnow & Szegedy quantitatively improved Yao's result to show that such languages are in fact in $\text{NP/}\texttt{poly}$ (*Information Processing Letters*, 1992). In this paper we extend Yao's result to show that if a language L is 2-locally random reducible to a target function which takes values in $\{0, 1, 2\}$, then $L \in \text{PSPACE/}\texttt{poly}$.

Keywords.
Random reductions, lower bounds, upper bounds, computational complexity, complexity classes.

Subject classification. 68Q15, 68Q17

1. Introduction

Informally a language L is *locally random reducible* (in short *lrr*) to a target function f if, for any input x, membership of x in L can be efficiently reduced to k random instances of f. If the reduction makes k queries then it is called a k-local random reduction. Additionally if the target function is the language L itself then it is called a *random self-reduction*. Such random reducibility notions have been very useful in many areas of theoretical computer science including, complexity theory, cryptography, and private information retrieval. For example, it is known that if a language is random-self reducible, then its worst-case complexity is same as the average-case complexity. These notions also plays an important role in the design of program checkers. Refer to the paper by Feigenbaum (1993) for a survey on locally random reductions and their applications.

For a language L, is there a function f so that L locally random reduces to f? Beaver & Feigenbaum (1990) showed that this is true: any language L is $(n+1)$-locally random reduces to a specific function f. Beaver *et al.* (1997)

improved this result to show that in fact, for any language L, there is a function f so that L $(n/\log n)$-locally random reduces to f. In these constructions the range of the target function is very large (in fact $\Omega(2^n)$).

A natural question that arises is whether the result of Beaver *et al.* (1997) is optimal when we consider the number of queries? For every language L, is there a function f such that L locally random reduces to f using less than $n/\log n$ queries? Are there languages that are not k-lrr to any function, for a constant k? For $k = 1$, Abadi *et al.* (1989) showed that a language that is 1-lrr to some function is in NP/`poly`. Thus there exist languages that are not 1-lrr to any function.

What happens when we allow two queries? Is there a language that is not 2-lrr to any function? Surprisingly, this question turns out be difficult. Yao considered the following weaker question: Is there a language that is not 2-lrr to any *Boolean function*—a function whose range is $\{0, 1\}$. He showed that any language that is 2-locally random reducible to a *Boolean function* is in PSPACE/`poly`. Fortnow & Szegedy extended Yao's result to show that such languages are in fact in NP/`poly` $\cap$ co-NP/`poly` (Fortnow & Szegedy 1992). Thus there exist languages that are not 2-lrr to any Boolean function. For the general case when the target function is not restricted to be Boolean the question of whether there are languages that are not 2-lrr is still open.

It is to be mentioned that in the case of random self reductions more results are known. Feigenbaum *et al.* (1990) showed that any function (need not be a language/Boolean function) that is uniform 2-random self-reducible is in fNP/`poly` (fNP is the functional version of NP). Feigenbaum & Fortnow (1993) showed that if an NP-complete set such as SAT is `poly`(n)-random self-reducible with nonadaptive queries, then the Polynomial Hierarchy collapses.

In this paper we consider the following question: can we extend the results of Yao, and Fortnow and Szegedy, for target functions other than Boolean? Building on the work of Yao, and Fortnow and Szegedy, we show that there exist languages that are not 2-locally random reducible to any function whose range is $\{0, 1, 2\}$ (3-valued function). We obtain this result by showing that every language that is 2-lrr to a 3-valued function is in PSPACE/`poly`.

THEOREM 1.1. *If a language L is 2-locally random reducible to functions g and h that take values in $\{0, 1, 2\}$, then $L \in$* PSPACE/`poly`.

Locally random reductions are closely related to the notion of *locally decodable codes*. An error-correcting code $C : \Sigma^n \to \Sigma^m$ is a *k-locally decodable code*, if every single bit of the original data x can be recovered by probing k locations of a word y that is close to $C(x)$. Locally decodable codes have found

many applications in complexity theory. We refer the reader to the excellent survey paper by Trevisan (2004) for more on locally decodable codes and their applications.

Ideally we would like to design locally decodable codes for which m is comparable to n. This raises the following question: If $C : \Sigma^n \to \Gamma^m$ is a locally decodable code, then what is the minimum value of m? This question was first raised by Katz & Trevisan (2000). They showed that if C is a k-locally decodable code, then m must be super linear in n. Goldreich *et al.* (2002) obtained an exponential lower bound for the special case of linear 2-locally decodable codes. Deshpande *et al.* (2002) extended the results of Katz and Trevisan to the case of adaptive codes. Kerenidis & de Wolf (2003) and Wehner & de Wolf (2005) generalized the result of Goldreich *et al.* (2002) to the case of all 2-locally decodable codes. More specifically, they showed that $m = 2^{\Omega(n)}$. Their proof interestingly uses quantum information theory.

Here we consider *perfectly smooth codes*, a special case of locally decodable codes. Results of Kerenedis & de Wolf and Wehner & de Wolf when applied to the special case of 2-perfectly smooth codes yield a lower bound of $2^{n/4}$. We observe that Yao-Fortnow-Szegedy proof can be adapted to show that if $C : \{0,1\}^n \to \{0,1\}^m$ is a 2-perfectly smooth code, then $m \geq 2^{n-1}$.

THEOREM 1.2. *If $C : \{0,1\}^n \to \{0,1\}^m$ is a 2-perfectly smooth code, then $m \geq 2^{n-1}$.*

It appears that extending our result and the results of Yao, and Fortnow & Szegedy for 3-lrr is difficult as this might lead to an exponential lower bound for the length of 3-locally decodable codes which is an outstanding open problem.

2. Lower bound for locally random reductions

DEFINITION 2.1 (locally random reduction). *A language L is k-locally random reducible (k-*lrr*) to functions $g_1, \ldots, g_k$ if there are polynomial-time functions σ and f and a polynomial q so that*

- $\forall x \in \{0,1\}^*, \forall r \in \{0,1\}^{q(|x|)}$, $L(x) = f(g_1(\sigma(1,x,r)), \ldots, g_k(\sigma(k,x,r)), x, r)$
- *for all i, $\sigma(i,x,r)$ and $\sigma(i,y,r)$ are identically distributed when $|x| = |y|$ and $r \in \{0,1\}^{q(|x|)}$ is chosen uniformly at random.*

Now we will state and prove our main result.

THEOREM 1.1. *If a language L is 2-locally random reducible to functions g and h that take values in $\{0,1,2\}$, then $L \in$ PSPACE/`poly`.*

First we give an informal overview of the proof.

2.1. Overview of the proof. For this informal description, we assume that the target functions g and h are the same. We first briefly explain the idea behind Yao-Fortnow-Szegedy's proof as our proof builds on this. The basic idea in both proofs is to compute answers of all the possible queries asked on inputs at a fixed length. A polynomial amount of information is given as advice for this computation.

Let L be a language that is 2-lrr to a Boolean function g. Given a string x, let p and q be two queries produced by the reduction using a random string r. Suppose the membership of x in L depends only on p, and does not depend on q. Then by knowing $g(p)$ we can decide x. In this case, we say p *sets* x. Suppose p does not set x. This means the membership of x in L depends on both p and q. The crucial observation is the following: When p does not set x, given $g(p)$, whether x belongs to L or not precisely depends on whether $g(q)$ is 0 or 1. Thus, in this case, by knowing $g(p)$, and $L(x)$ we can deduce the value of $g(q)$. We call this scenario p *forces* q *via* x. Now, the proof of Yao-Fortnow-Szegedy goes as follows. Let $P = \{p_1, \cdots p_m\}$ be a set of queries whose answers we know, i.e, we know the values of $g(p_i), 1 \leq i \leq m$. Let x be the input string. If there exists a $p \in P$ that sets x then we can decide the membership of x in L easily. On the other hand if no p in P sets x, then for every p there exists a query q such that p forces q via x. Thus by giving $L(x)$ as advice we can compute the value of $g(q)$ for all these q's. Thus the set of queries whose answers we know has doubled if we are given the value of $L(x)$. Applying this argument iteratively we can decide answers to all the queries by specifying $L(x)$ for a polynomially small set of x's. This idea can be implemented to show that L is in NP/`poly`.

Now let us consider when g is a 3-valued function. The above proof does not go through directly. In particular, if membership of x in L depends on both p and q, as g is 3-valued, we cannot deduce the value of $g(q)$ from the values of $g(p)$ and $L(x)$. We get around this problem by using a *majority argument*.

Let $P = \{p_1, \cdots p_m\}$ be a set of queries whose answers we know. Let x be the input string. If there exists a $p \in P$ that sets x then we can decide the membership of x in L. So consider the case where no $p \in P$ set x. For a $p \in P$, let q be the second query. By considering all three possible values (0, 1, and 2) for $g(q)$, we compute possible values for $L(x)$. Since $L(x)$ is Boolean it has only two possible values and hence there exist two possible values of $g(q)$ that predicts $L(x) = a$ and one possibility for $g(q)$ that predicts $L(x) = \overline{a}$, for an $a \in \{0, 1\}$. A crucial point is that in the latter case we can deduce the

value of $g(q)$ from the values of $g(p)$ and $L(x)$. Let a_p denote the predicted value of $L(x)$ for a majority over 3 possible settings for the value of $g(q)$. Let $a_x = \text{Majority}_{p \in P}\{a_p\}$. Our strategy is the following: if $a_x \neq L(x)$ then we will give the value of $L(x)$ as advice. In this case, for a majority of $p \in P$, there is only one value of $g(q)$ that result in $L(x)$ and for all the qs corresponding to these ps, we can deduce the value of $g(q)$. Thus by giving $L(x)$ as advice, we can deduce values of additional $\frac{m}{2}$ queries. We can apply this argument a polynomial number of times to compute the values of all possible queries. We show how this strategy can be implemented to get a PSPACE/`poly` algorithm for L. A detailed proof follows.

2.2. Detailed proof.

PROOF. Since L is 2-locally random reducible to g and h, there exist polynomial-time computable functions f, σ such that for every x,

$$\forall r, f(g(\sigma(1,x,r)), h(\sigma(2,x,r)), x, r) = L(x).$$

Moreover, for every x, y, $|x| = |y|$, the random variables $\sigma(1,x,r)$ and $\sigma(1,y,r)$ are identically distributed, and the random variables $\sigma(2,x,r)$ and $\sigma(2,y,r)$ are also identically distributed.

We will first introduce some notation that we use for the rest of the proof. $\sigma(1,x,r)$ and $\sigma(2,x,r)$ will be denoted by $\sigma_1(x,r)$ and $\sigma_2(x,r)$ respectively. We will use p's to denote the first queries and q's to denote the second queries.

Given x, let $P(x)$ denote the multi-set of all possible first queries, and $Q(x)$ denote the multi-set of all possible second queries. That is $P(x) = \{\sigma_1(x,r) | r \in \{0,1\}^{q(|x|)}\}$ and $Q(x) = \{\sigma_2(x,r) | r \in \{0,1\}^{q(|x|)}\}$. We stress that both $P(x)$ and $Q(x)$ are multi-sets.

Without loss of generality, we assume that we can easily distinguish first queries from second queries. Let m be a bound on the length of queries in $P(x)$ and $Q(x)$. Since, σ_1, and σ_2 are polynomial time-computable, $m = \texttt{poly}(n)$. Since L is 2-locally random reducible to g and h, if x and y are of same length, then $P(x) = P(y)$ and $Q(x) = Q(y)$. We will denote $P(0^n)$ with P_n and $Q(0^n)$ with Q_n.

Given a multi set A consisting of zeros and ones, let $\texttt{MAJ}(A) = 1$ if at least half the members of A are 1, else $\texttt{MAJ}(A) = 0$.

DEFINITION 2.2. *For an x and r, let $p = \sigma_1(x,r)$.*

- *We say* p sets x *if the membership of* $L(x)$ *can be decided by knowing only* $g(p)$*, i.e.,*

$$f(g(p), 0, x, r) = f(g(p), 1, x, r) = f(g(p), 2, x, r) = L(x).$$

- *We say* p and x force q *if* $h(q)$ *can be computed by knowing* $g(p)$ *and* $L(x)$*. I.e., there exists unique* $b \in \{0, 1, 2\}$ *such that*

$$f(g(q), b, x, r) = L(x).$$

- *We say that* p weakly sets x *if* $\mathtt{MAJ}(\{f(g(p), 0, x, r), f(g(p), 1, x, r), f(g(p), 2, x, r)\}) = L(x)$.

Similarly we can define setting, forcing, and weak setting for second query $q = \sigma_2(x, r)$.

Observation: Note that if p does not set or weakly set x then p and x force q.

For each n we will construct a "query tree" T_n. Each node of the tree is labeled with a query from the multi sets P_n or Q_n and each level of the tree will be associated with a string $x \in \{0, 1\}^n$. The tree will be such that by knowing $L(x)$ for each of these x's, we will be able to get answers to any query in the query tree using a PSPACE procedure.

We now give the construction of T_n. Consider strings in $\{0, 1\}^n$ in the lexicographic order. For a string x, $x + 1$ denotes string which follows x in this ordering. Let $\perp$ be a special string and assume $\perp < x$ for every x.

Construction Level 1
1. Fix a query p_0 from P_n and label the root of the tree with p_0

Let $x_0 = \perp$. Assume that we have built k levels of the tree and let x_1, x_2, $\cdots, x_{k-1}$ be the strings associated with each level of the tree. We now define the tree at level $k+1$ and associate a string x_k with level k. Define the following multi-sets.

$$Q'_k = \{q \mid q \in Q_n, \text{ and } q \text{ is at level } k\},$$
$$P'_k = \{p \mid p \in P_n, \text{ and } p \text{ is at level } k\}.$$

If $x_{k-1} = 1^n$ (we have exhausted all the input strings), or $Q'_k = Q_n$ and $P'_k = P_n$ (we have exhausted all the queries), then T_n is completely defined. Else, we construct level $k + 1$ by the following procedure.

CONSTRUCTION Level $k+1$

1. $x \leftarrow x_{k-1} + 1$
2. If either (a). There exists a query q at level k that sets x, or
   (b). There exists r such that both $\sigma_1(x,r)$ and $\sigma_2(x,r)$ are in level k
   then $x \leftarrow x+1$ and Goto Step 2
3. For each $p \in P'_k$, let r_p be the lexicographically least r such that $\sigma_1(x,r_p) = p$
   For each $q \in Q'_k$, let r_q be the lexicographically least r such that $\sigma_2(x,r_q) = q$
   Let $a_p = \texttt{MAJ}\{f(g(p),0,x,r_p), f(g(p),1,x,r_p), f(g(p),2,x,r_p)\}$
   $a_q = \texttt{MAJ}\{f(0,h(q),x,r_q), f(1,h(q),x,r_q), f(2,h(q),x,r_q)\}$
4. Consider the multi set $A = \{a_q \mid q \in Q'_k\} \cup \{a_p \mid p \in P'_k\}$
   If $\texttt{MAJ}(A) = L(x)$, then $x \leftarrow x+1$ and Goto Step 2
5. Else /* `define nodes at level` $k+1$`, and associate a string with level` k */
   Associate the string x with level k
   For every $v \in Q'_k \cup P'_k$ such that $a_v = L(x)$, v has only one child with label v
   For each $p \in P'_k$ such that $a_p \neq L(x)$, p has two children with labels p and $\sigma_2(x,r_p)$
   For each $q \in Q'_k$ such that $a_q \neq L(x)$, q has two children with labels $\sigma_1(x,r_q)$ and q

We now show some properties of the tree T_n. Let x_k be the string associated with level k. We first start with the following observation.

CLAIM 2.3. *Let $p \in P'_k$, let $q = \sigma_2(x_k, r_p)$. If $a_p \neq L(x_k)$, then x_k and p force q. Similarly, let $q \in Q'_k$, let $p = \sigma_1(x_k, r_q)$. If $a_q \neq L(x_k)$, then x_k and q force p.*

PROOF. Let

$$M = \{f(g(p),0,x_k,r_p), f(q(p),1,x_k,r_p), f(g(p),2,x_k,r_p)\}.$$

Since x_k is the string associated with level k, p does not set x_k. Thus, all the elements in the multi-set M can not be the same. Since $a_p = \texttt{MAJ}(M) \neq L(x_k)$, there exists a unique $b \in \{0,1,2\}$ such that $L(x_k) = f(g(p),b,x_k,r_p)$. Thus knowing the values of $L(x_k)$ and $g(p)$ gives the value of $h(q)$. Thus x_k and p force q. Proof of the the second part of the claim is identical. □

Let S_k be the multi-set consisting of all nodes at level k. Next we show that size of S_{k+1} is considerably larger than the size of S_k.

CLAIM 2.4. $\forall k, |S_{k+1}| \geq \frac{3}{2}|S_k|$.

PROOF. Let x_k be the string associated with level k. Step 5 of the above construction defines the nodes in level $k+1$. Observe that every node at level k has either one or two children. Every $q \in Q'_k$, for which $a_q = L(x_k)$, has exactly one child, and every $q \in Q'_k$, for which $a_q \neq L(x_k)$, q has exactly two children. Similarly, every $p \in P'_k$, for which $a_p = L(x_k)$, has exactly one child, and every $p \in P'_k$, for which $a_p \neq L(x_k)$, p has exactly two children.

By definition of x_k, we know $L(x_{k+1}) \neq \mathtt{MAJ}(A)$, where $A = \{a_q \mid q \in Q'_k\} \cup \{a_p \mid p \in P'_k\}$. Thus for more than half nodes v in level k, $a_v \neq L(x_k)$. All such nodes have exactly two children. Thus $|S_{k+1}| \geq \frac{3}{2}|S_k|$. Note that here we use the fact that the sets S_k, S_{k+1}, P'_k, and Q'_k are multi sets. □

COROLLARY 2.5. *Depth of T_n is* $\mathtt{poly}(n)$.

CLAIM 2.6. *Let u be a first query, and let k be a level at which u appears. Given $g(u)$ and $L(x_k)$, there is a* PSPACE *algorithm that computes the children of u. Moreover, for each child v of u, $g(v)$ ($h(v)$ if v is a second query) can be computed in* PSPACE. *A similar claim holds if u is a second query.*

PROOF. Let k be a level at which u appears. Compute the least r such that $\sigma_1(x_k, r) = u$. Let $v = \sigma_2(x_k, r)$. Note that r can be computed in PSPACE and given r, v can be computed in polynomial-time. Compute

$$a_u = \mathtt{MAJ}(\{f(g(u), 0, x_k, r), f(g(u), 1, x_k, r), f(g(u), 2, x_k, r\}).$$

By the construction of T_n, if $a_u = L(x_k)$, u has only one child with label u. In this case we know the value of $g(u)$. Else $a_u \neq L(x_k)$. In this case u has two children with labels u and v. Since $a_u \neq L(x_k)$, by Claim Claim 2.3, u and x_k force v. Since $g(u)$ and $L(x_k)$ are known, $h(v)$ can be computed. Thus $h(v)$ can be computed in PSPACE. □

Next we claim that given $g(p_0)$, $\langle x_1, L(x_1)\rangle$, $\cdots$, $\langle x_d, L(x_d)\rangle$ where d is the height of the tree T_n, the tree T_n can be traversed in PSPACE. Given a node u, let $value(u) = g(u)$ if u is a first query, otherwise $value(u) = h(u)$.

CLAIM 2.7. *Let u a node. Given $g(p_0)$, $\langle x_1, L(x_1)\rangle$, $\cdots$, $\langle x_d, L(x_d)\rangle$ where d is the height of the tree T_n, and a level k, there is a* PSPACE *algorithm checks if u appears at level of T_n. If u appears at level k, then this algorithm computes $value(u)$.*

PROOF. Let REACH$(w, value(w), u, s)$ be a subroutine that returns true if u can be reached from w in exactly s steps. Consider the following recursive algorithm.

REACH(*w, value(w), u, s*)
1 `Current` $\leftarrow w$;
2 For each child v of `Current`
3 Compute $value(v)$;
4 If REACH$(v, value(v), u, s-1) = true$, then return *true*.

By Claim Claim 2.6, Step 3 can be done in PSPACE. The recursion terminates when $s = 1$. Again by Claim Claim 2.6, given $L(x_k)$, REACH$(node, value(node), u, 1)$ can be computed in PSPACE. By Claim Claim 2.6, it follows that $value(u)$ can also be computed in PSPACE. Since the maximum out degree of T_n is 2, the above procedure can implemented in PSPACE. □

Now we are ready to give a PSPACE/`poly` algorithm for L. The algorithm is given the advice $\langle x_1, L(x_1)\rangle, \cdots, \langle x_d, L(x_d)\rangle$, p_0 and $g(p_0)$. Let x be the given input. If x is one of $x_1, \cdots, x_d$, then $L(x)$ can be computed trivially. Let $x_{k-1} < x < x_k$. Since x is not the string associated with level $k+1$, one of the following must hold:

1. There exists a node v at level k that sets x.
2. There exists a r such that $\sigma_1(x, r)$ and $\sigma_2(x, r)$ appear at level k.
3. Majority of nodes at level k weakly set x.

We can check if Statement 1 holds or not as follows: For each r compute $v = \sigma_1(x, r)$. Check if v appears in level k of T_n and if it appears compute $g(v)$. By, Claim Claim 2.7 this can be done in PSPACE. Now, we can check if v sets x. If none of the first queries set x, check if any of the second queries sets x. If we succeed in finding a v that sets x, then we know $L(x)$.

If Statement 1 does not hold, we can check if Statement 2 holds as follows: For each r compute $u = \sigma_1(x, r)$ and $v = \sigma_2(x, r)$. By Claim Claim 2.7, in PSPACE, we can find if u and v occur at level k. If so, we can also compute $g(u)$ and $h(v)$ and thus compute $L(x)$. Thus if Statement 2 holds, then also $L(x)$ can be computed in PSPACE.

If Statement 1 and Statement 2 both do not hold, then Statement 3 must hold. For each node v at level k compute a_v, also compute majority of a_v's. This can be done in PSPACE as we can systematically generate each node at

level k. Since Statement 3 holds, majority of nodes at level k weakly set x. Thus majority of a_v's is the value of $L(x)$.

Thus L is in PSPACE/poly. □

2.3. Discussion on difficulty in extending to the 4-valued case. Can we extend our techniques to show upper bounds on 2-local random reduction to k-valued functions for $k > 3$? In our proof it was crucial that we could extend the notion of *forcing* from Boolean case to the 3-valued case. It appears to be difficult to extend this notion even to the 4-valued case. For this discussion, we use the notation used in Section 2.1. On an input x and random string r, let p and q be the two queries asked by the reduction. As before, let us assume that we know the value of $g(p)$ and $L(x)$ and we are trying to deduce the value of $g(q)$. Possible values for $g(q)$ are $\{0, 1, 2, 3\}$. If it were the case that out of 4 settings of the values of $g(q)$ there is only one value that will correctly predict $L(x)$, then p and x will force q. But the scenario where there are two setting for $g(q)$ which gives rise to $L(x)$ and two settings for $g(q)$ which gives rise to $\overline{L}(x)$ we cannot uniquely determine the value of $g(q)$ by knowing $g(p)$ and $L(x)$. At present we do not know how to overcome this situation.

3. Relations to perfectly smooth codes

Notions of locally random reductions, private information retrieval, and locally decodable codes are closely related. For example, it is known that lower bounds on locally decodable codes yield lower bounds on private information retrieval (Goldreich *et al.* 2002). Beigel *et al.* (2006) showed that techniques used to show upper bounds on locally random reductions can be used to obtain lower bounds on private information retrieval. In this section we observe that the techniques used to show upper bounds on locally random reductions yields lower bounds on certain locally decodable codes also.

Definition 3.1. *A code $C : \{0,1\}^n \to \Gamma^m$, is (k, δ, ϵ)-locally decodable if there exists a probabilistic oracle algorithm A such that for every message $X \in \{0,1\}^n$, index $i \in \{0, \cdots, n\}$, and a string y such that $d(y, C(x)) \leq \delta m$, A on input i makes k random queries to y (given as oracle to A) and outputs x_i with probability at least $1/2 + \epsilon$.*

Perfectly smooth codes are a special case of locally decodable codes.

Definition 3.2. *A code $C : \{0,1\}^n \to \Gamma^m$ is a* k-perfectly smooth code *if there is a probabilistic oracle algorithm A such that for every $X \in \{0,1\}^n$ and*

index i, $1 \leq i \leq n$, A on input i makes k random queries to $C(X)$ (given as oracle to A) and outputs x_i with probability 1. Moreover, for all i and j, the j^{th} query is uniformly distributed in $\{1, \ldots, m\}$.

We show that if $C : \{0,1\}^n \to \{0,1\}^m$ is a 2-perfectly smooth code, then $m \geq 2^{n-1}$. This proof is based on Yao and Fortnow & Szegedy's proof. This bound is very close to optimal as Hadamard code is a 2-perfectly smooth code with code length $m = 2^n - 1$.

THEOREM 1.2. *If $C : \{0,1\}^n \to \{0,1\}^m$ is a 2-perfectly smooth code, then $m \geq 2^{n-1}$.*

PROOF. Let $C : \{0,1\}^n \to \{0,1\}^m$ is a 2-*perfectly smooth code*. Consider a Kolmogorov random string $X \in \{0,1\}^n$ (string with Kolmogorov complexity $K(X) = n$). We will argue that if $m < 2^{n-1}$ then $K(X) < n$.

Assume that A is a probabilistic oracle decoding algorithm that we have for the code C. On an index i and a random string r, A makes two queries which we call the first query and the second query. Let $X = x_1 \cdots x_n$ and let $C(X) = y_1 \cdots y_m$ be the code word of X. First we assume that for two random strings r_1 and r_2 producing two distinct first queries p_1 and p_2, the corresponding second queries are also distinct (this assumption is not without loss of generality, but we can argue that since both the first and the second queries are distributed uniformly, there should exists random strings r_1 and r_2 which produce first queries p_1 and p_2 so that if $p_1 \neq p_2$ then the corresponding second queries are also distinct and we can focus on these set of random strings for the rest of the argument).

We will construct a binary tree with nodes labeled by queries (a query is an index between 1 and m). We will associate each level of the tree (except the last level) by a bit of X. The tree is constructed in stages. The root of the tree is labeled with query 1. Assume that the k^{th} level has been constructed and let P_k be the set of first queries and Q_k the set of second queries that label the nodes in level k. Also let $x_{i_1}, \cdots, x_{i_{k-1}}$ be the bits associated with levels 1 through $k-1$. Now we will explain how to associate a bit of X with level k and how to construct and label the nodes of level $(k+1)$. Assume that we have considered the first j bits $x_1 \ldots x_j$ of X. We will associate the k^{th} level with the next bit from the remaining $x_{j+1}, \cdots, x_n$ bits (considered in that order) that is *independent* of the tree constructed so far. Let us explain what we mean by independent. We call the j^{th} bit x_j of X *independent* if *none* of the following three conditions are satisfied: (a) there is a random string r so that both the queries by A on input j and random string r are from $P_k \cup Q_k$ or (b) there is a

random string r such that the first query by A on input j and random string r is in P_k and it *sets* x_j or (c) there is a random string r such that the second query by A on input j and random string r is in Q_k and it sets x_j. Here we can define the notion of setting as in the earlier section: a query p on random string r sets x_j if the output of $A(j)$ with random string r and queries p and q does not depend on the answer to the query q. Associate the k^{th} level with the next x_l that is *independent*. That is x_{i_k} = the first independent x_l in X after $x_{i_{k-1}}$. Now we will explain how to construct the nodes in level $k+1$. For each $p \in P_k$ let r be a random string for which p is the first query made by A on input l and random string r. The left child of p is p itself and the right child of p is q where q is the corresponding second query. Similarly we can define the children for a query $q \in Q_k$. Since x_l is independent $|P_{k+1} \cup Q_{k+1}| = 2 \times |P_k \cup Q_k|$ (each query produces a new query). So the number of distinct queries at level $k+1$ is 2^k. We end the construction when there are no more independent bits of X remaining. Let there be t internal levels for this tree (do not count the last level where the nodes are leaves).

The idea is that if the bit x_j is not independent then it can be effectively constructed from the previous independent bits of x and the first bit y_1 of the codeword $C(X)$. This is because using y_1 and the independent bits of X before x_j, we can deduce the answers to the queries at level k. Therefore the string X can be reconstructed from the string $y_1 x_{i_1} \cdots x_{i_t}$. Since X is Kolmogorov random, we should have $n \leq t+1$. So $m \geq |P_{t+1} \cup Q_{t+1}| = 2^t \geq 2^{n-1}$.

□

We remark that Kerenidis & de Wolf (2003) and Wehner & de Wolf (2005) obtained exponential lower bounds for 2-locally decodable codes using quantum information theory. Their results when applied to the special case of 2-perfectly smooth codes with $|\Gamma| = 2$ yields a lower bound of $2^{n/4}$. The above theorem gives a better lower bound and does not use tools from quantum information theory.

An obvious question is whether the proof of Theorem 1.1 can be used to obtain lower bounds for 2-perfectly smooth codes with $|\Gamma| = 3$. The proof of Theorem 1.1 indeed gives a lower bound for this case also, however this bound is weak. Let C be a 2-perfectly smooth code from $\{0,1\}^n$ to Γ^n, where $|\Gamma| = 3$. Let X be a Kolmogorov random string from $\{0,1\}^n$. Using ideas from Theorem 1.1, we can show that X can be described by a string of length $\log n \log_{3/2} m$. Thus we obtain that $m \geq 2^{n/\log n}$. However, Kerenidis & de Wolf (2003) and Wehner & de Wolf (2005) give $2^{\Omega(n)}$ bound for this case.

4. Concluding Remarks

Some lower bound techniques known for locally decodable codes can be used to prove similar results for locally random reductions. In particular, techniques of Katz & Trevisan (2000) and Deshpande *et al.* (2002) can be used to show that any language that is k-lrr to a function is in PSPACE/$2^{\epsilon n}$ for $\epsilon = 1 - \frac{1}{k}$.

The locally random reduction that we consider in this paper does not allow errors. This is true for lower bound results in Yao (1990), Fortnow & Szegedy (1992), and Feigenbaum *et al.* (1990). Extending these techniques to the general nonzero error case is interesting since it may lead to a non-quantum proof of exponential lower bounds for 2-locally decodable codes: currently the known exponential lower bound proofs for 2-locally decodable codes use quantum information theory (Kerenidis & de Wolf 2003; Wehner & de Wolf 2005).

Acknowledgements

The authors thank the anonymous reviewers for their comments. Research of the first author is supported in part by NSF grants CCR-0344817 and CCF-0430807. Research of the second author is supported in part by NSF grant CCF-0430991, Big 12 Fellowship, and University of Nebraska Layman Award.

References

M. Abadi, J. Feigenbaum & J. Kilian (1989). On hiding information from an oracle. *Journal of Computer and System Sciences* **39**, 21–50.

D. Beaver & J. Feigenbaum (1990). Hiding Instances in Multioracle Queries. In *7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, 37–48. Springer.

D. Beaver, J. Feigenbaum, J. Kilian & P. Rogaway (1997). Locally Random Reductions: Improvements and Applications. *Journal of Cryptology* **10**(1), 17–36.

R. Beigel, L. Fortnow & W. Gasarch (2006). A tight lower bound for restricted PIR protocols. *Computational Complexity* **15**(1), 82–91.

A. Deshpande, R. Jain, T. Kavitha, J. Radhakrishnan & S. Lokam (2002). Better Lower Bounds for Locally Decodable Codes. In *IEEE Conference on Computational Complexity*, 184–193.

J. Feigenbaum (1993). Locally random reductions in interactive complexity theory. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **13**, 73–98.

J. Feigenbaum & L. Fortnow (1993). On the Random-Self-Reducibility of complete sets. *SIAM J. Comput.* **22**, 994–1005.

J. Feigenbaum, S. Kannan & N. Nisan (1990). Lower Bounds on Random-Self-Reducibility. In *Proceedings of the Fifth Annual Structure in Complexity Theory Conference*, 100–109. IEEE Computer Society Press.

L. Fortnow & M. Szegedy (1992). On the power of two-local random reductions. *Information Processing Letters* **44**(6), 303–306.

O. Goldreich, H. Karloff, L. Schulman & L. Trevisan (2002). Lower bounds for linear locally decodable codes and private information retrieval. In *17th Computational Complexity Conference*, 175–183.

J. Katz & L. Trevisan (2000). On the efficiency of local decoding procedures for error-correcting codes. In *In Proc. of 32nd STOC*, 80–86.

I. Kerenidis & R. de Wolf (2003). Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *In 35th Annual ACM Symposium on Theory of Computing*, 106–115.

L. Trevisan (2004). Some Applications of Coding Theory in Computational Complexity. *Quaderni di Matematica* **13**, 347–424.

S. Wehner & R. de Wolf (2005). Improved Lower Bounds for Locally Decodable Codes and Private Information Retrieval. In *Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming*, volume LNCS 3580, 1424–1436.

A. Yao (1990). An Application of Communication Complexity to Cryptography. In *A Lecture at DIMACS Workshop on Structural Complexity and Cryptography.*

Manuscript received

A. Pavan
Department of Computer Science
Iowa State University
Ames, IA 50011.
email: pavan@cs.iastate.edu

N. V. Vinodchandran
Dept. of Computer Science and Engg.
University of Nebraska-Lincoln
Lincoln, NE 68588.
email: vinod@cse.unl.edu.