On the Existence of Hard Sparse Sets under Weak Reductions

JIN-YI CAI [1]   ASHISH V. NAIK [2]   D. SIVAKUMAR [3]

[1]Department of Computer Science, State University of New York at Buffalo, Buffalo, NY 14260. Research supported in part by NSF grants CCR-9057486 and CCR-9319093, and an Alfred P. Sloan Fellowship. Email: cai@cs.buffalo.edu

[2]Computer Science Department, University of Chicago, Chicago, IL 60637. Research supported in part by NSF grant CCR 92-53582. Email: naik@cs.uchicago.edu

[3]Department of Computer Science, State University of New York at Buffalo, Buffalo, NY 14260. Research supported in part by NSF grant CCR-9409104. Email: sivak-d@cs.buffalo.edu

Abstract

Recently a 1978 conjecture by Hartmanis [Har78] was resolved [CS95], following progress made by [Ogi95]. It was shown that there is no sparse set that is hard for P under logspace many-one reductions, unless P = LOGSPACE. We extend the results to the case of sparse sets that are hard under more general reducibilities. Our main results are as follows.

(1) If there exists a sparse set that is hard for P under bounded truth-table reductions, then $\mathrm{P} = \mathrm{NC}^2$.

(2) If there exists a sparse set that is hard for P under randomized logspace reductions with one-sided error, then P = Randomized LOGSPACE.

(3) If there exists an NP-hard sparse set under randomized polynomial-time reductions with one-sided error, then NP = RP.

(4) If there exists a $2^{(\log n)^{O(1)}}$-sparse hard set for P under truth-table reductions, then $\mathrm{P} \subseteq \mathrm{DSPACE}[(\log n)^{O(1)}]$.

As a by-product of (4), we obtain a uniform $O(\log^2 n \log\log n)$ time parallel algorithm for computing the rank of a $2^{\log^2 n} \times n$ matrix over an arbitrary field, generalizing a result of Mulmuley [Mul87]. This algorithm may be of independent interest in solving non-square linear equation systems.

1 Introduction

The study of reductions of complexity classes to sparse sets has a long and rich history in complexity theory. The starting point of this research was provided by Berman and Hartmanis [BH77], when, as a consequence of the famous isomorphism conjecture, they conjectured that NP-complete sets cannot be sparse. As a culmination of considerable previous work [Ber77, For79], Mahaney [Mah82] proved that if there exists a many-one hard sparse set for NP, then P = NP. A decade later, Ogiwara and Watanabe [OW91] succeeded in extending Mahaney's theorem to sparse sets complete under bounded truth-table reductions. Since then, there has been a large body of research to extend the Ogiwara-Watanabe theorem for various weaker reducibilities and for other complexity classes [RR92, OL91, AKM92, HL91] (see [HOW92, You92a, You92b] for a survey).

In contrast, there is another equally prominent conjecture by Hartmanis about sparse sets on which, until recently, not much progress had been made. In 1978, while studying the isomorphism problem for P-complete and NL-complete problems, Hartmanis conjectured that there do not exist P-complete sparse sets under logspace many-one reductions. Hemachandra, Ogiwara and Toda [HOT94] reported the first progress on this question by showing that if there exists a polylog-sparse hard set for P, then P = SC (Steve's Class). The next significant breakthrough was obtained by Ogihara [Ogi95], who showed that if there exists a sparse set that is hard for P under logspace many-one reductions, then $P \subseteq DSPACE[\log^2 n]$. Finally Cai and Sivakumar [CS95] settled Hartmanis' conjecture by showing that there are no sparse hard sets for P under logspace many-one reductions, unless P = LOGSPACE.

Although the isomorphism conjecture was really about many-one completeness, there is an equally compelling reason to study weaker reducibilities. It is well-known that the class of all languages with (non-uniform) polynomial size circuits is precisely the class of languages that are polynomial-time Turing reducible to sparse sets. Using Buss' result that the boolean formula value problem is in uniform NC^1 [Bus87, BCGR92], it can be shown that the class of all languages with (non-uniform) poly-size, log-depth circuits is precisely the class of languages that are uniform-NC^1 reducible to sparse sets. The ultimate objective, going beyond this paper, is to understand the relationship between P and non-uniform NC^1.

In this paper we address the question of existence of hard sparse sets under weaker reducibilities. We show that unless every problem in P has highly parallelizable algorithms, there do not exist sparse sets that are hard for P under several weaker reducibilities. We now describe our results in detail.

First, we consider *truth-table* (tt) reducibility. A language A is said to be reducible to B by an $s(n)$-tt reduction f if for every x, f outputs a list of $s(n)$ queries $w_1, \ldots, w_{s(n)}$, and a boolean predicate e on $s(n)$ variables, such that $x \in A$ iff $e(B(w_1), \ldots, B(w_{s(n)}))$. This is the parallel version of Turing reducibility. Indeed they are the most general kind of reductions computable by NC circuits as defined in [Coo85] (in contrast to the adaptive Turing reductions computable by Turing Machines).

Under the assumption that P has a sparse hard set under logspace-computable bounded truth-table reductions, we first show that $P \subseteq RNC^2$. The proof of this result relies on solving a system of equations where a fraction of the equations could be erroneous. We then derandomize this algorithm, taking advantage of certain *error-correcting* capabilities of the "small-bias sample space" construction used in [CS95]. The generalization is not straightforward, however. It is an indication of the effectiveness of algebraic and derandomization techniques that the proof in [CS95] can be generalized to account for the case of bounded truth-table reductions. We note that in the NP case it took the research community 10 years to generalize Mahaney's result for many-one reducibility to Ogihara-Watanabe's theorem for bounded truth-table reducibility [OW91]. The proof technique for the case of k-tt reductions also gives a weaker collapse result for $O(\log\log n)$-tt reductions: if P has a sparse hard set under $O(\log\log n)$-tt reductions computable in polylogarithmic space, then $\mathrm{P} \subseteq \mathrm{DSPACE}[(\log n)^{O(1)}]$. Further results are mentioned in the paper.

Next, we consider randomized many-one reductions. We show that if P has a sparse hard set under logspace-computable randomized reductions with one-sided error, then P = RLOGSPACE (with one-sided error). Combining this result with the Valiant–Vazirani reduction [VV85] from satisfiability to unique satisfiability, we show that if NP has a sparse hard set under randomized reductions computable in polynomial time, then NP = RP. This answers an open question raised in [RR92]. Results about randomized reductions are described in Section 4.

To obtain sharp space bounds for the truth-table reductions, and for the case of truth-table reductions to *quasipolynomially* sparse sets (that is, sets with density $2^{(\log n)^{O(1)}}$), we need to compute in parallel the rank of an $m \times n$ matrix over $GF(2)$, where $n = o(m)$. For definiteness, suppose $m = n^{\log n}$. Mulmuley's algorithm [Mul87] can be adapted to compute the rank of the above matrix, but we get a circuit of depth $O(\log^4 n)$. An earlier algorithm due to Chistov [Chi85] gives a *nonuniform* circuit of depth $O(\log^2 n \log\log n)$. We build upon these two algorithms and obtain a uniform circuit of depth $O(\log^2 n \log\log n)$ to compute the rank of the above matrix. We also give the construction of a uniform randomized circuit of depth $O(\log^2 n)$ for this problem. Our rank algorithm is described in Section 5.

Finally, we apply our techniques to deterministic and nondeterministic space bounded classes and to circuit classes. These results are described in Section 6.

2 Preliminaries

We will use the standard notation for various complexity classes. DSPACE$[s(n)]$ denotes deterministic space $O(s(n))$, NSPACE$[s(n)]$ denotes nondeterministic space $O(s(n))$, and RSPACE$[s(n)]$ denotes randomized space $O(s(n))$ with one-sided error. We will use LOGSPACE for DSPACE$[\log n]$, RLOGSPACE, or RL, for RSPACE$[\log n]$, and NL for NSPACE$[\log n]$. By a *uniform* circuit family, we usually mean that there is an $O(\log s(n))$-space-bounded transducer that, for each n, outputs an encoding of the circuit for inputs of length n, where $s(n)$ is the size of the circuit produced.

We use standard definitions of boolean circuits. The *circuit value problem* (CVP) is defined as follows: Given a boolean circuit C with m inputs and a binary string x of length m, compute the output of C with input x. Ladner [Lad75] showed that CVP is P-complete under logspace many-one reductions; Cook [Coo85] noted that if CVP is in NC^1, then $\mathrm{P} = \mathrm{NC}^1$.

3 Truth-table reductions

We start by describing our results about sparse sets that are hard under truth-table reductions. In Section 3.2 we illustrate our proof technique by giving the proof for 1-tt reductions.

3.1 Main Results

We first show that if there exists a hard sparse set under $s(n)$-tt reductions, then CVP has a fast *randomized* parallel algorithm.

Theorem 1 *If there exists a sparse set S such that CVP reduces to S under a logspace-computable truth-table reduction that queries at most s strings to the oracle, then CVP can be solved in randomized $O(\max\{2^s \log n, \log^2 n\})$ parallel time using at most $n^{O(2^s)}$ many processors.*

Corollary 2 *If there exists a sparse set that is hard for* P *under bounded truth-table reductions, then* $\mathrm{P} \subseteq \mathrm{RNC}^2$.

To prove Theorem 1 we need to prove an extended version of a probabilistic lemma of Cai and Sivakumar in [CS95].

Given a circuit C with n gates, and an input x, let $g_1, \ldots, g_n$ be the values output by the gates $1, 2, \ldots, n$. Let B^n denote the n-dimensional binary hypercube. Any subset of the gates in C can be represented as a point in the vector space B^n. Let A be defined as follows,

$$A = \{\langle C, x, I, b\rangle \mid I \in B^n, \bigoplus_{i \in I} g_i = b\}.$$

It is easy to see that $A \in \mathrm{P}$.

The idea in the proof for the many-one case is the following. Fix a circuit C and an input x, by trying many independently chosen subsets I, we may hope that the reduction f will produce many "collisions", namely $f(\langle C, x, I, b\rangle) = f(\langle C, x, J, b'\rangle)$. Note that if S is sparse, many collisions do occur, since for every I, there is a unique b such that f maps $\langle C, x, I, b\rangle$ maps to some string in S. When the tuples $\langle C, x, I, b\rangle$ and $\langle C, x, J, b'\rangle$ collide, we obtain the equation

$$\bigoplus_{i \in I \Delta J} g_i(x) = b \oplus b'.$$

In the case of truth-table reductions, we cannot hope for a string in S to be "popular," i.e., to be mapped on by many I, and thus "carry the day." With truth-table reductions, it is not necessary that all (or any of) the "YES" instances map inside the set S, which is what makes the proof in the case of many-one reductions work. Also, it is no longer straightforward to construct equations as in the proofs of the many-one case.

We give a brief sketch of the idea for the truth-table cases. On input $\langle C, x, I, b\rangle$, suppose the reduction from A to S queries the strings $w_1, \ldots, w_s$. We will call w_i the i-color of I. Thus, every point in B^n has at most s colors. We argue that, by randomly selecting polynomially-many points in B^n, with high probability, we can extract a collection $\mathcal{G}_1, \ldots, \mathcal{G}_r$

of large sets of points from the sample such that for every set $\mathcal{G}$ in the collection $\mathcal{G}_1, \ldots, \mathcal{G}_r$ and for every i, $1 \leq i \leq s$, either the i-color is the same for all points in $\mathcal{G}$ or there are many different i-colors in $\mathcal{G}$. If, for some i, there is a large number of different i-colors in $\mathcal{G}$, we *assume* that all these i-colors are outside S. Then, we give a procedure to generate a system of equations for the points in each $\mathcal{G}$. However, our assumption has introduced some erroneous systems of equations. Now, we use the fact that since S is a sparse set, the number of incorrect assumptions would be small, and, in particular, we show that at least one system of equations is completely error-free. We solve all the systems of equations in parallel. Using the details of the circuit C, we can then locally check the solutions to discard the incorrect solutions. Since at least one of the systems is correct whp, the correct solution will be obtained. The complete proof of Theorem 1 is described in the appendix.

We now turn to derandomization. Theorem 1 can be derandomized to show that:

Theorem 3 *If there exists a sparse set S such that CVP reduces to S via a s-tt logspace reduction, then CVP can be solved in* $O(\max\{2^s \log n, \log^2 n\})$ *parallel time using at most* $n^{O(2^s)}$ *many processors.*

Corollary 4 *If there exists a sparse set that is hard for* P *under bounded truth-table reductions, then* $\mathrm{P} = \mathrm{NC}^2$.

Corollary 5 *If there exists a sparse set that is hard for* P *under* $O(\log\log n)$*-truth-table reductions, then* $\mathrm{P} \subseteq \mathrm{DSPACE}[(\log n)^{O(1)}]$.

Corollaries 4 and 5 solve an open question raised by Ogihara [Ogi95].

The proof of Theorem 3 is more involved, even for the case of 1-tt reductions. Without loss of generality, we assume that for each tuple $\langle C, x, I\rangle$, where C is a circuit, x is its input, and $I \subseteq [n]$ is a set of gates of C, the reduction function gives a unique string w, such that for exactly one value $b \in \mathbf{Z_2}$,

$$\bigoplus_{i \in I} g_i(x) = b \iff w \in S.$$

That is, we assume that the 1-tt reduction assigns the same string to $\langle C, x, I, 0\rangle$ and $\langle C, x, I, 1\rangle$ with a "flipped" truth table.

We will use a polynomial-sized set D of vectors $I \in B^n$. The set D will be the same as the one used in [CS95] for derandomizing the algorithm for the many-one case. This construction is formally identical to an *error-correcting code* [vL91] that is obtained by concatenating a certain Reed-Solomon code with a Hadamard code. We will use the error-correcting capabilities of this construction in a crucial way.

The basic idea is the following. For each $I \in D$, we will produce strings $f(\langle C, x, I, b\rangle)$. We then argue that either we will already have enough equations to solve the system, or we must be in a situation where most of the points sampled are outside the sparse set S. We will then *assume* that all such points are outside of S and thus form a set of equations with sufficiently high rank. But due to our *assumption* that all sampled points are outside S, we have introduced some erroneous values for the right hand side of some of the equations, while the coefficient matrix of the equations have no errors. Then we will use the error correcting capability to correct those few errors, and then solve the system.

A proof for the 1-tt case is given in the next subsection.

The passage from 1-tt to the general case of bounded truth-table reductions is even more involved. The extra difficulty can be described as follows: While nested logically in the various cases for the truth-tables, we need to reconstruct linear functions defined on various linear subspaces L, each of logarithmic dimension. However, unlike in the case of 1-truth-table reductions, we no longer have a reliable set of samples from the whole space L, but rather only a set of samples which may be all from a smaller linear subspace of L.

To overcome this difficulty, we need to suitably "shrink" the linear subspace L to a linear subspace $L' \subseteq L$, which is smaller than L, and yet not too much smaller. On L' we can uniquely reconstruct the linear function from sample data that are "essentially correct." We then apply similar procedures in the 1-tt case to finish the proof. The proof requires several new technical lemmas especially concerning the feasibility of reconstructing linear functions from sample data that are partial *and* faulty.

The proof for the general bounded truth-table case is combinatorially complicated and is omitted from this extended abstract. Before we proceed to give the proof for the 1-tt case, we apply Theorem 3 to higher complexity classes to yield the following corollary.

Corollary 6 *Let $\mathcal{C}$ belong to $\{\mathrm{NP}, \mathrm{PP}, \mathrm{C_{=}P}, \mathrm{Mod_2P}, \mathrm{Mod_3P}\ldots\}$. If there exists a sparse set S such that every language in $\mathcal{C}$ reduces to S via logspace-computable bounded truth-table reductions, then $\mathcal{C} = \mathrm{NC}^2$.*

Note that even though it was known [OW91, OL91] that these classes collapse to P if there exists a btt-hard sparse set under polynomial-time reductions, the polynomial-time simulations provided by these proofs are highly serial in nature and even if the reduction is logspace-computable, do not imply Corollary 6.

Can we extend the above results to $O(\log n)$-tt reductions? Theorem 3 does not help us at all with $O(\log n)$ reductions. However, as a first step towards solving this question, we prove a result for a special kind of $O(\log n)$-truth-table reducibility—the *positive* reducibility.

Definition 1 *A language A* positively *reduces to B if A reduces to B via a truth-table reduction f with the property that the boolean predicate e of the reduction is a monotonic boolean function.*

Theorem 7 *If there is a sparse set S that is hard for* P *under $O(\log n)$-tt positive reductions, then* $\mathrm{P} \subseteq \mathrm{RSPACE}[\log^2 n]$.

Corollary 8 *If there exists a $2^{(\log n)^{O(1)}}$-sparse set that is hard for* P *under $(\log n)^{O(1)}$-space computable $(\log n)^{O(1)}$-positive truth-table reductions, then* $\mathrm{P} \subseteq \mathrm{RSPACE}[(\log n)^{O(1)}]$.

3.2 Derandomization for the 1-tt Reductions

Fix a circuit C and an input z. Recall that, we assume the 1-tt reduction assigns, for every I, the same string to $\langle C, z, I, 0\rangle$ and $\langle C, z, I, 1\rangle$ with a "flipped" truth table. If the tuple $\langle C, z, I\rangle$ is assigned the string w then we say that I *receives the color* w. We will identify a subset $I \subseteq [n]$ with its characteristic vector in $\mathbf{Z_2}^n$. Let $p(n)$ be a polynomial such that there are at most $p(n)$ many distinct strings w assigned by the reduction that belong to S.

Suppose two tuples $\langle C, z, I\rangle$ and $\langle C, z, J\rangle$ receive the same color w. Then according to the actual truth table by the evaluator for the reduction on $\langle C, z, I\rangle$ and $\langle C, z, J\rangle$, we get an equation of the form

$$\bigoplus_{i \in I \Delta J} g_i(z) = b,$$

for some appropriate boolean value b, independent of whether the color $w \in S$. On the other hand, if we know the membership for "$w \in S$?", then we can get two equations,

$$\bigoplus_{i \in I} g_i(z) = b_I, \qquad \bigoplus_{i \in J} g_i(z) = b_J,$$

where b_I and b_J depend on the actual truth tables for I and J.

Let $m = 10 + 3\log n + 2\log p(n) = O(\log n)$. Let $\mathbf{F} = GF(2^m)$ be a Galois field of 2^m elements. Let

$$D = \{(\langle 1, v\rangle, \langle u, v\rangle, \ldots, \langle u^{n-1}, v\rangle) \mid u, v \in \mathbf{F}\}.$$

Note that $|D| = 2^{2m} = n^{O(1)}$. Let

$$D_S = \{y \in D \mid \text{ color}(y) \in S\}.$$

Let D_S and D be decomposed into color classes

$$D_S = C_1 \cup C_2 \cup \ldots \cup C_p,$$

and

$$D = C_1 \cup C_2 \cup \ldots \cup C_p \cup \ldots \cup C_q,$$

where the number of color classes p is at most $p(n)$, and $q \le |D|$.

Let the affine span of C_i be denoted as $L_i + d_i$, where L_i is a linear subspace, and d_i is a displacement vector. Let $L = L_1 + L_2 + \cdots + L_q$ be the sum of the linear subspaces. We call L the span of the color classes. The difference of each pair of vectors in any color class C_i gives us an equation mod 2 of the values of the gates of the circuit C with the given input. If we collect a generating set of vectors for each L_i, together they span L. Thus, if $\dim L \ge n - O(\log n)$, we would succeed since by sampling exhaustively in D, we would have obtained a system of linear equations of rank $\ge n - O(\log n)$, which can be solved in NC^2 (cf. [CS95]).

Let $k = 5 + \log n + \log p(n) = O(\log n)$. We now consider two cases.

Case 1. $\dim L \ge n - k$. Then we can solve the problem, as illustrated in [CS95].

Case 2. $\dim L < n - k$. Then, we note that $D_S \subseteq \bigcup_{i=1}^{p}(L_i + d_i) \subseteq \bigcup_{i=1}^{p}(L + d_i)$. Replacing L by a higher dimensional subspace if necessary, we can assume that L is defined by a set of k linearly independent equations,

$$\sum_{j=0}^{n-1} a_{ij} x_j = 0,$$

where $a_{ij} \in \mathbf{Z_2}$, and $i = 1, \ldots, k$. Each affine space $L + d$ is then defined by the linear system

$$\sum_{j=0}^{n-1} a_{ij} x_j = b_i \tag{1}$$

for some $b(d) = (b_1, \ldots, b_k)^T \in \mathbf{Z_2}^k$, a right hand side (RHS) vector depending on d.

Now consider any such system of k independent linear equations with an arbitrary RHS vector b. Denote this affine space by Π. Denote the point in D specified by u, v as $D(u, v)$. It can be shown that each equation in the linear system (1) cuts down the sample set D successively into two roughly equal subsets, so that the fraction of the points of D that fall in Π is roughly $1/2^k$. More precisely,

$$\left| \Pr_{u,v \in \mathbf{F}} [D(u, v) \in \Pi] - 1/2^k \right| \leq \frac{(2^k - 1)(n - 1)}{2^m} \left(1 - \frac{1}{2^k}\right).$$

As D_S is covered by at most $p(n)$ many affine translations of L, we have

$$\frac{|D_S|}{|D|} \leq p(n) \cdot \left(\frac{1}{2^k} + \frac{n}{2^{m-k}}\right),$$

which by our choice of values for k and m, is $< 1/(8n)$.

Now for each I we will *assume* the target $w \notin S$, which gives us an equation with a certain RHS. For precisely those $I \in D_S$ we get an incorrect RHS. For $I \in D - D_S$, the assumption that $w \notin S$ is correct and thus the RHS for that equation is correct.

For each $u \in \mathbf{F}$, we will first try to obtain a correct equation of the form

$$\sum_{j=0}^{n-1} u^j x_j = \gamma,$$

for some $\gamma \in \mathbf{F}$. We do it by taking a majority vote for each bit.

Let v' range over $\{0, 1\}^{m-1}$. Let $v = 0v'$ and $1v'$ denote the corresponding elements in $\mathbf{F}$ with the specified bit pattern. Then we will set the 1st bit of γ to 0 if the majority of the RHS of $\sum_{j=0}^{n-1} \langle u^j, 0v' \rangle x_j$ and $\sum_{j=0}^{n-1} \langle u^j, 1v' \rangle x_j$ agree, and we set it to 1 otherwise. Similarly we set the other $m - 1$ bits of γ.

Note that, for the given $u \in \mathbf{F}$, if less than 1/4 of the RHS bit b_{uv} in the equation

$$\sum_{j=0}^{n-1} \langle u^j, v \rangle x_j = b_{uv} \in \mathbf{Z_2},$$

are incorrect, as v ranges over all $\mathbf{F}$, then our correction scheme succeeds in finding a correct RHS γ in

$$\sum_{j=0}^{n-1} u^j x_j = \gamma.$$

Now among the 2^m equations $\sum_{j=0}^{n-1} u^j x_j = \gamma$ as u ranges over $\mathbf{F}$, less than a fraction of $1/(2n)$ of them could have an incorrect RHS γ. For, otherwise, the fraction of incorrect equations in the original system is $\geq 1/(8n)$, contradicting the fact that

$$\frac{|D_S|}{|D|} < \frac{1}{8n}.$$

Finally we will correct those remaining values γ via polynomial self-reducibility.

Note that if we set the polynomial $f(U) = \sum_{j=0}^{n-1} \alpha_i U^i$, where α_i is the actual value of the ith gate in the circuit C with input z, then the equations are valuations of f at various points $u \in \mathbf{F}$.

Fix any $u_0 \in \mathbf{F}$. We will break all 2^m equations of the form

$$\sum_{j=0}^{n-1} u^j x_j = \gamma,$$

as u varies over $\mathbf{F}$, into $2^m/n$ blocks of n equations each. Then we use polynomial interpolation formula to evaluate f at u_0 within each block of n equations, and then take a majority vote. Since the fraction of incorrect RHS is $< 1/(2n)$, the majority vote is guaranteed to be correct for $f(u_0)$. Note that u_0 is arbitrary. We can then solve a Vandermonde system for the actual values of the gates, after we have evaluated at n points $u_1, u_2, \ldots, u_n$ in parallel.

This completes our treatment of Case 2.

To summarize, our NC^2 deterministic algorithm works as follows. It constructs D and forms two sets of linear equation systems in parallel. One set of equations is formed similar to the many-one reducibility case by considering all pairs of (I, J) which are mapped to the same string w. The second set of equations is formed by *assuming* all queried strings $w \notin S$. We try to solve both systems and then check out the results, using local information of the circuit such as $z_i = 0$, or $z_i = 1$, or $g_j(z) = g_k(z) \wedge g_\ell(z)$, etc. If the first set of equations has sufficiently high rank $n - O(\log n)$, then it will succeed. Otherwise the second set of equations will succeed.

4 Randomized many-one reductions

We apply the techniques of Cai and Sivakumar [CS95] to the case of many-one reductions with one-sided error. We say that a language A is *rp-reducible* to B if there is a function $f(\cdot,\cdot)$ and a constant $\epsilon > 0$ such that for all x,

$$x \in A \Rightarrow \Pr_z[f(x,z) \in B] \geq \epsilon, \qquad \text{and} \qquad x \notin A \Rightarrow \Pr_z[f(x,z) \in B] = 0.$$

We will show that even if P rp-reduces in logspace to a sparse set S with success probability $\epsilon = 1/n^{O(1)}$, then P = Randomized LOGSPACE.

Theorem 9 *If there is a sparse set S that is hard for* P *under rp-reductions with success probability $\epsilon = 1/n^{O(1)}$, then CVP can be solved by a logspace-uniform family of randomized circuits of size $n^{O(1)}$ and depth $O(\log n)$, with $n^{O(1)}$ many parallel calls to the reduction.*

A sketch of the proof for Theorem 9 is given in the appendix. Theorem 9 has an application for NP: it is known by a result of Ranjan and Rohatgi [RR92] (see also [AKM92]) that if NP is reducible to a sparse set via a *co-rp* reduction, then NP = RP. The question of whether the same consequence can be proved under the hypothesis that NP is rp-reducible to a sparse set was left open. We combine the above theorem with the Valiant–Vazirani randomized reduction from satisfiability to unique satisfiability to settle this question.

Theorem 10 *If there is a sparse set S that is hard for* NP *under rp-reductions computable in polynomial time with success probability $1/n^{O(1)}$, then* NP = RP.

Proof Sketch. Under the hypothesis that NP is rp-reducible to a sparse set, we show that $SAT \in$ RP. Given a boolean formula $\varphi(x_1, x_2, \ldots, x_n)$, we first apply the reduction of Valiant and Vazirani. This produces a list of boolean formulae $\varphi_1, \varphi_2, \ldots, \varphi_k$, where $k = n^{O(1)}$, and where each φ_i is a formula in $n^{O(1)}$ variables. If φ is satisfiable, then with probability $1 - e^{-n}$, at least one of the φ_i's has a unique satisfying assignment. If φ is not satisfiable, then no φ_i is satisfiable. Assume that φ^* is a boolean formula in t variables that has a unique satisfying assignment; it suffices to show how to construct this assignment using the reduction of NP to a sparse set S. We define the language

$$L = \left\{ \langle \psi, 1^m, u, v \rangle \mid m = 2 \cdot 3^\ell, u, v \in GF(2^m), (\exists \vec{a} = (a_1, \ldots, a_t))[\psi(\vec{a}) \ \wedge \ \sum_{i=0}^{n-1} a_i u^i = v] \right\}.$$

Clearly $L \in$ NP, therefore L reduces to S. Now, using the same ideas as in the case of rp-reductions from P to a sparse set, we can construct the satisfying assignment $\vec{a}^*$ of φ^* in randomized polynomial time. $\square$

Using the fact that the Valiant–Vazirani reduction can be computed in NC^1, we obtain:

Corollary 11 *If there is a sparse set S that is hard for* NP *under rp-reductions computable in* NC^1*, then* NP = RNC^1.

5 Computing the Rank of Non-Square Matrices

To obtain sharp bounds in our results concerning sparse sets that are hard under unbounded truth-table reductions, and concerning sparse sets of superpolynomial density, we need to compute the rank of $2^{(\log n)^{O(1)}} \times n$ matrices over $GF(2)$. In this section, we give deterministic circuits of depth $O(\log^2 n \log\log n)$ and randomized circuits of depth $O(\log^2 n)$ for this problem. Our results in this section work for arbitrary fields, and may be of independent interest.

Let A be an $m \times n$ matrix over some field $\mathbf{F}$, where $n = o(m)$, and suppose we wish to compute the rank of A. For definiteness, we will focus on the case $m = n^{\log n}$ as an example. Two efficient parallel algorithms are known for this problem:

(1) Mulmuley gives a uniform parallel algorithm that requires $N^{4.5}$ processors and that takes $O(\log^2 N)$ time, where $N = m + n$. This blows up to $O(\log^4 n)$-parallel time if $m = n^{\log n}$.

(2) Chistov gives a non-uniform parallel algorithm that requires $M^{O(1)}$ processors and that takes $O\left(\log M + \log^2 q \log\left(\frac{\log M}{\log|\mathbf{F}|}\right)\right)$ time, where $M = mn$ and $q = \min\{m, n\}$. This works out to be an $O(\log^2 n \log\log n)$ time algorithm if $m = n^{\log n}$.

In both cases, operations in the field $\mathbf{F}$ are assumed to take unit time. If they are not, the complexities are appropriately scaled.

We build upon ideas from these two papers, and give a uniform deterministic algorithm that takes $O(\log^2 n \log\log(nm))$ time. We also present an elementary probabilistic argument that can replace Chistov's construction, which he establishes using arguments from algebraic geometry. This gives us a uniform randomized algorithm that takes $O(\log^2 n)$ time. Both algorithms use $\mathrm{poly}(n, m)$ processors. We describe the algorithms in the Appendix.

6 Further Results and Open Questions

We state the following theorems without proof. The proof techniques are similar to the ones used in Sections 3 and 4. More corollaries for randomized and truth-table reductions also follow.

Theorem 12 *If there is a sparse set S that is hard for* LOGSPACE *under* NC^1 *many-one reductions, then* LOGSPACE $= \mathrm{NC}^1$.

Theorem 13 *If there is a sparse set S that is hard for* NL *under* NC^1 *many-one reductions, then* $\mathrm{NL} = \mathrm{RNC}^1$.

Theorem 14 *If there is a sparse set S that is hard for* NL *under logspace many-one reductions, then* $\mathrm{NL} = \mathrm{RL}$.

Theorem 15 *If there is a sparse set S that is hard for* NC^k *under* NC^ℓ *many-one reductions, where $0 < \ell < k$, then* $\mathrm{NC}^k = \mathrm{NC}^\ell$.

Two interesting problems remain open. The first is whether we can obtain similar consequences under the hypothesis that there is a sparse hard set for P under randomized reductions with two-sided error. The other question is whether the results of Section 3 can be improved to the case of $O(\log n)$-tt reductions, and whether we can obtain a collapse to LOGSPACE. It appears that these problems will require more sophisticated techniques, and promise to be interesting questions to study.

References

[AGHP90] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple constructions of almost k-wise independent random variables. In *Proc. 31st FOCS*, pages 544–553, 1990.

[AKM92] V. Arvind, J. Köbler, and M. Mundhenk. On bounded truth-table, conjunctive, and randomized reductions to sparse sets. In *Proc. 12th FST & TCS*, Lecture Notes in Computer Science, Springer-Verlag, 1992.

[Ber77] L. Berman. *Polynomial Reducibilities and Complete Sets.* PhD thesis, Cornell University, 1977.

[BH77] L. Berman and H. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–322, 1977.

[BvzGH82] A. Borodin, J. von zur Gathen, and J. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52:241–256, 1982.

[Bus87] S. Buss. The Boolean formula value problem is in ALOGTIME. In *Proc. 19th STOC*, pages 123–131, 1987.

[BCGR92] S. Buss, S. Cook, A. Gupta, and V. Ramachandran. An optimal parallel algorithm for formula evaluation. *SIAM Journal on Computing*, 21:755–780, 1992.

[CS95] J. Cai and D. Sivakumar. The resolution of a Hartmanis conjecture. *Submitted*, April 1995.

[Chi85] A.L. Chistov. Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In *Proc. 5th FCT*, Lecture Notes in Computer Science, pages 63–69. Springer-Verlag, 1985.

[Coo85] S. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[For79] S. Fortune. A note on sparse complete sets. *SIAM Journal on Computing*, 8:431–433, 1979.

[Har78] J. Hartmanis. On log-tape isomorphisms of complete sets. *Theoretical Computer Science*, 7(3):273–286, 1978.

[HOT94] L. Hemachandra, M. Ogiwara, and S. Toda. Space-efficient recognition of sparse self-reducible languages. *Computational Complexity*, 4:262–296, 1994.

[HOW92] L. Hemachandra, M. Ogiwara, and O. Watanabe. How hard are sparse sets. In *Proc. 7th Structures*, pages 222–238, 1992.

[HL91] S. Homer and L. Longpré On reductions of NP sets to sparse sets. In *Proc. 6th Structures*, pages 79–88, 1991.

[Lad75] R. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):18–20, 1975.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1:103–123, 1975.

[Mah82] S. Mahaney. Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25:130–143, 1982.

[Mul87] K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987.

[NN90] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proc. 22nd STOC*, pages 213–223, 1990.

[Ogi95] M. Ogihara. Sparse hard sets for P yield space-efficient algorithms. Technical Report 569, Dept. of CS, University of Rochester, January 1995.

[OL91] M. Ogiwara and A. Lozano. On one-query self-reducible sets. In *Proc. 6th Structures*, pages 139–151, 1991.

[OW91] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal on Computing*, 20(3):471–483, 1991.

[RR92] D. Ranjan and P. Rohatgi. On randomized reductions to sparse sets. In *Proc. 7th Structures*, pages 239–242, 1992.

[VV85] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. In *Proceedings of 17th ACM Symposium Theory of Computing*, pages 458–463, 1985.

[vL91] J.H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1991.

[You92a] P. Young. How reductions to sparse sets collapse the polynomial-time hierarchy: A primer (Part I). *SIGACT News*, 23(3):107–117, 1992.

[You92b] P. Young. How reductions to sparse sets collapse the polynomial-time hierarchy: A primer (Part II). *SIGACT News*, 23(4):83–94, 1992.

A Proof of Theorem 1

In this section, we prove Theorem 1. We will first give the proof for 2-tt reductions, which we then generalize to bounded truth-table reductions.

Theorem 1 *If there exists a sparse set S such that CVP reduces to S under a logspace-computable truth-table reduction that queries at most s strings to the oracle, then CVP can be solved in randomized $O(\max\{2^s \log n, \log^2 n\})$ parallel time using at most $n^{O(2^s)}$ many processors.*

Proof. We will give the proof only for the 2-tt case. Let C be a boolean circuit with gates $1, 2, \ldots, n$ and input x. Let the output value of the gates in C on input x be denoted by $g_1, \ldots, g_n$. We want to compute the value g_n.

Let B^n denote the n-dimensional binary hypercube; B^n has a vector space structure over the finite field $\mathbf{Z_2}$. Any subset of the gates in C can be represented as a point in the vector space B^n. Let A be defined as follows.

$$A = \{\langle C, x, I, b\rangle \mid \bigoplus_{i \in I} g_i = b\}$$

Let S be a sparse set that is hard for P under 2-tt reductions. Since $A \in$ P, A reduces to S via a logspace-computable 2-tt reduction f. Assume, without loss of generality, that for all I, f on inputs $\langle C, x, I, 1\rangle$ and $\langle C, x, I, 1\rangle$ queries the same set of strings. For all $I \in B^n$, if $f(\langle C, x, I, 0\rangle)$ queries two strings w_1 and w_2 to S such $w_1 \leq w_2$ lexicographically, then w_1 is called the *1-color* of I and w_2 is called the *2-color* of I. A set P of points is said to be *1-monochromatic* if all points in P have the same 1-color and *2-monochromatic* if all points in P have the same 2-color. For any set X, the set of 1-colors in X is the set $\{w \mid w \text{ is the 1-color of some } I \in X\}$. $X_1, \ldots, X_s$ is an *equal partition of X by 1-color* if $X_1, X_2, \ldots, X_s$ is a partition of X such that the 1-colors of all X_i's are pairwise disjoint and of equal size. Each partition is often referred to as a *block*.

Let $q(m)$ be a polynomial that bounds the length of strings that can be queried by f on inputs of size m and let $p(m)$ be a polynomial that strictly bounds the number of strings in S of length at most $q(m)$. For technical reasons, we will assume that p is at least linear. Let k denote $p(|f(\langle C, x, I, b\rangle)|)$.

Now consider the following probabilistic procedure SAMPLE. It starts by choosing a set of k^{16} random points in B^n and outputs a subset of this set. The output is organized in the form of one or many *blocks* such that each block contains at least k^3 points. For every block that is output, we construct a system of linear equations. We will show that one of the system of linear equations is correct and has a high rank of $n - O(\log n)$.

begin SAMPLE
Choose a set $\mathcal{R}$ of k^{16} many points in B^n uniformly and independently at random
  Case I: If $\exists$ 1-monochromatic $\mathcal{G} \subseteq \mathcal{R}, \|\mathcal{G}\| \geq k^8$, **then**
  begin
    Case a: if $\exists$ 2-monochromatic $\mathcal{G}' \subseteq \mathcal{G}, \|\mathcal{G}'\| \geq k^4$, **then**
      Output $\mathcal{G}'$
    Case b: else /* $\mathcal{G}$ contains $> k^4$ 2-colors */
      Obtain k equal partitions of $\mathcal{G}$ by 2-colors
      Output *all* the blocks of $\mathcal{G}$
  end
  Case II: else /* $\mathcal{R}$ contains $> k^8$ 1-colors */
  begin
    Obtain k equal partitions $\mathcal{G}_1, \ldots, \mathcal{G}_k$ of $\mathcal{R}$ by 1-colors
    for all $\mathcal{G}_i$ **do**
    begin
      Case a: if $\exists$ 2-monochromatic $\mathcal{G}_i' \subseteq \mathcal{G}_i, \|\mathcal{G}_i'\| \geq k^3$ **then**
        select $\mathcal{G}_i'$
      Case b: else /* $\mathcal{G}_i$ contains $> k^4$ 2-colors*/
      begin
        Obtain k equal partitions of $\mathcal{G}$ by 2-colors
        select all the blocks of $\mathcal{G}_i$
      end
    end
    Output all the selected blocks
  end

It is easy to see that for all samples $\mathcal{R}$, SAMPLE outputs a set of blocks such that each block contains at least k^3 points. We claim that for each block output by SAMPLE, the dimension of the affine span of the points in the block is at least $n - 18\log k$ with high probability. To prove this claim, it suffices to show that with high probability, every subset $\mathcal{G}$ of $\mathcal{R}$ of size $\geq k^3$ has dimension greater than $n - 18\log k$. This claim follows by the next lemma.

Lemma 2 *Suppose $\mathcal{R}$ is a set of k^{16} independent, identically distributed points in B^n. For all $\mathcal{G} \subseteq \mathcal{R}$, such that $\|\mathcal{G}\| \geq k^3$,*

$$\Pr[(\exists\mathcal{G})\mathit{dim(affine\ span\ of\ }\mathcal{G}) < n - 18\log k] < e^{-k^2}$$

Proof. Let $\mathcal{R}$ be a random sample and let $\mathcal{H} = \{I_1, \ldots, I_r\} \subseteq \mathcal{R}$ such that $r \geq k^3$. Mark the points in $\mathcal{H}$ by 0 or 1 as follows. Let I_1 be marked 0. For all $j \leq r$, I_j is marked 1 if and only if the dimension of the affine span of $I_1, \ldots, I_j$ is greater than the dimension of the affine span of $I_1, \ldots, I_{j-1}$. This defines a 0-1 sequence σ of length r. We want to estimate the probability that the number of 1's is small in σ.

Suppose that we have marked the sequence $I_1, \ldots, I_{j-1}$, and $I_1, \ldots, I_{j-1}$ has less than $n - 18\log k$ many 1's. Then, the conditional probability that σ_i will be marked 0 is less than $\frac{1}{k^{18}}$. Therefore

$$\begin{aligned}\Pr[\dim(\text{affine span of } \mathcal{H}) < n - 18\log k] &< \sum_{j<n-18\log k} \binom{k^3}{j} \left(\frac{1}{k^{18}}\right)^{k^3-(n-18\log k)} \\ &\leq k^{-18k^3+O(n)}\end{aligned}$$

The total number of subsets $\mathcal{H}$ of $\mathcal{R}$ of size k^3 is $\binom{k^{16}}{k^3}$. Hence, the probability that there exists a set $\mathcal{H} \subseteq \mathcal{R}$ of size k^3 and dimension $< n - 18\log k$ is less than the following expression.

$$\begin{aligned} &\leq \binom{k^{16}}{k^3} \cdot k^{-18k^3+O(n)} \\ &\leq k^{-2k^3+O(n)} \\ &\leq e^{-k^2}\end{aligned}$$

$\square$

We now use the SAMPLE procedure to describe an RNC^2 algorithm for recognizing CVP. On input C and x, choose a set of k^{16} points randomly in B^n. Now, in parallel, simulate all the four cases (Ia, Ib, IIa, IIb) in SAMPLE on the set of points chosen. Assuming that each case is true for the sample, a collection of blocks will be output. Next, we generate one or more systems of equations corresponding to each block that is output. One of the output set of points will correspond to the correct assumption and hence will have more than k^3 points in each block. With high probability, the dimension of the affine span of the points in each block of the output will be greater than $n - 18\log k$. We solve all the systems of equations whose rank is greater than $n - 18\log k$ by cycling through (in parallel) all polynomially many possible values of the $O(\log n)$ free variables. Each solution is then locally checked. At least one of the system of equations with an appropriate assignment to the free variables will yield a correct solution to the circuit C.

It remains to describe how equations are constructed and to prove that at least one of the system of equations is correct. The equations constructed depend on which of the four assumptions is the correct assumption for the sample $\mathcal{R}$.

(Ia) In this case, the output is both 1-monochromatic and 2-monochromatic (with 1-color w_1 and 2-color w_2). Simultaneously, assume all possible memberships of w_1 and w_2 in S, and generate an equation for each I under this assumption about w_1 and w_2 (e.g., if the assumption about w_1 and w_2 implies that $\bigoplus_{i \in I} g_i = 0$, then output the above equation.) The correct assumption about w_1 and w_2 will yield a correct system of equations with high rank.

(Ib) In this case, all the points have the same 1-color and there are k blocks of 2-colors, each block has greater than k^3 2-colors. Assume that all the 2-colors are outside S and generate two systems of equations by assuming both possibilities of membership of the first color in S. In one of the two assumptions, we will make the correct assumption about the 1-color. However, there may be some error in both systems of equations due to the 2-colors, since some of the 2-colors may be in S. We now use the fact that the number of erroneous colors is less than k, since the number of strings in S that can be queried by f is bounded by k. Since there are k output blocks, there must be at least one block all of whose 2-colors are correctly assumed to be outside S. Thus the system of equations obtained from at least one of the blocks will be correct, and as Lemma 1 shows, have a high rank. Once again, this system of equations can be solved in NC^2.

(IIa) This case is symmetric to the previous case. Output the systems of equations in a manner analogous to case (Ib).

(IIb) In this case, generate one system of equations for each block assuming that all the colors are outside S. By a similar argument as before, one of the blocks will be right about the 1-color, and in that block, one of the subblocks will be correct about the 2-color. Thus, one of the blocks must output a correct system of equations with a high rank.

This completes the description of the algorithm. Observe that our algorithm implicitly described a probabilistic circuit of size $O(k^{16})$ and depth $O(\log^2 n)$. □

We can generalize this construction to work under the assumption of bounded truth-table hard sparse sets.

Theorem 3 *If there exists a sparse set that is hard for* P *under logspace-computable bounded truth-table reductions, then* $\mathrm{P} \subseteq \mathrm{RNC}^2$.

Sketch of Proof This is a straightforward generalization of the proof mentioned before. Suppose the set A reduces to a sparse set S via a logspace computable function f that asks at most a queries, where a is some fixed constant. Let C be a circuit that we wish to evaluate on input x. For all $I \in B^n$, I now has a colors (instead of 2).

We generalize the SAMPLE procedure to the case of a colors as follows. SAMPLE now performs a rounds of analysis on a random sample, such that it considers the i-th color in the i-th round. In round i, it refines a subset of the sample, and either outputs a large i-monochromatic subset or k equal partition of the subset by i-color. Finally, it outputs a large set (of size $> k^3$) from which some correct set of equations can be obtained. It is not too hard to show that it is sufficient to sample $k^{2^{a+2}}$ random points in B^n to start with, and since a is a constant, this is still bound by a polynomial. The rest of the details are straightforward and omitted. □

B Proof of Theorem 9

Theorem 9 *If there is a sparse set S that is hard for* P *under rp-reductions with success probability $\epsilon = 1/n^{O(1)}$, then CVP can be solved by a logspace-uniform family of randomized circuits of size $n^{O(1)}$ and depth $O(\log n)$, with $n^{O(1)}$ many parallel calls to the reduction.*

Proof Sketch. It is known that the polynomial $X^{2\cdot 3^\ell} + X^{3^\ell} + 1 \in \mathbf{Z_2}[X]$ is an irreducible polynomial over $\mathbf{Z_2}$ for all $\ell \geq 0$ [vL91]. In the following, by a finite field $GF(2^m)$, where $m = 2 \cdot 3^\ell$, we refer explicitly to the field $\mathbf{Z_2}[X]/(X^{2\cdot 3^\ell} + X^{3^\ell} + 1)$.

Let S be a sparse set hard for P under logspace-computable many-one reductions. As before, we will consider a refinement of the circuit-value problem. Define

$$L = \left\{ \langle C, x, 1^m, u, v\rangle \mid m = 2\cdot 3^\ell, u, v \in GF(2^m), \sum_{j=0}^{n-1} u^j g_j = v \right\},$$

where C is a boolean circuit and x is an input to C, and where $g_0, \ldots, g_{n-1}$ are 0-1 variables that denote the values of the gates of C on input x. Here exponentiation and summation are carried out in the finite field $GF(2^m)$. It is easy to see that $L \in$ P, since all the required field arithmetic involved in checking $\sum u^j g_j = v$ can be performed in polynomial time.

Clearly $|\langle C, x, 1^m, u, v\rangle|$ is bounded polynomially in n and m. If f is the logspace-computable function that reduces L to S, the bound on the length of queries made by f on inputs of length $|\langle C, x, 1^m, u, v\rangle|$ is some polynomial $q(n, m)$. Let $p(n, m)$ be a polynomial that bounds the number of strings in S of length at most $q(n, m)$. We will choose the smallest m of the form $2\cdot 3^\ell$ such that $2^m/p(n, m) \geq n$. It is clear that $m = O(\log n)$. Let $\mathbf{F}$ denote the finite extension $GF(2^m)$ of $GF(2)$. When C, x and m are fixed and understood, we will abbreviate $\langle C, x, 1^m, u, v\rangle$ by $\langle u, v\rangle$.

Let $r = r(n) = n^{O(1)}$ denote the number of random bits used to compute the reduction f from L to S. For each $u \in \mathbf{F}$, there is a unique $v_u \in \mathbf{F}$ such that the sum of $u^j g_j$ equals v_u. Moreover, for every $u \in \mathbf{F}$, there exists some $w_u \in S$ such that the number of random strings $z \in \{0,1\}^r$ for which $f(\langle u, v_u\rangle, z) = w_u$ is at least $\epsilon 2^r/p(n)$. Therefore there is some $w \in S$ such that $w = w_u$ for at least $2^m/p(n) \geq n$ many u's. Let $u_1, \ldots, u_n$ denote n such u's—call them *useful*, and let $v_1, \ldots, v_n$ denote the corresponding v_u's. If we can obtain n correct equations $g_j u_i^j = v_i$, then we will have an inhomogeneous system of equations with a Vandermonde coefficient matrix over the field $\mathbf{F}$.

We know that for each i, $1 \leq i \leq n$,

$$\Pr_{z\in\{0,1\}^r}[f(\langle u_i, v_i\rangle, z) = w] \geq \epsilon/p(n).$$

Let $s = s(n) = p(n)/\epsilon$, and note that so long as $\epsilon = 1/n^{O(1)}$, $s = n^{O(1)}$. For all $u, v \in \mathbf{F}$, we will compute $f(\langle u, v\rangle, z)$ with sn^2 many randomly chosen z's. For the useful $u_1, \ldots, u_n$, the probability that w does not appear as the value of $f(\langle u_i, v_i\rangle, \cdot)$ in sn^2 independent trials is at most $(1 - 1/s)^{sn^2}$, which is at most e^{-n^2}. Therefore the probability that w does not appear as the value of $f(\langle u_i, v_i\rangle, \cdot)$ for at least one u_i is bounded by $ne^{-n^2} < e^{-n^2+\ln n} < e^{-n}$.

In other words, with probability at least $(1 - e^{-n})$, we will have a valid system of equations with a Vandermonde coefficient matrix. It is shown in [CS95] how to set up and solve this system in logspace-uniform NC^1. $\square$

C Description of the Rank Algorithm

Recall that the problem we wish to solve is to compute the rank of an $m \times n$ matrix A, where $n = o(m)$, over an arbitrary field $\mathbf{F}$. Arithmetic operations in $\mathbf{F}$ are assumed to take unit time.

A deterministic algorithm

Let $\mathbf{G}$ denote the extension of $\mathbf{F}$ obtained by adjoining to $\mathbf{F}$ an element x that is algebraically independent over $\mathbf{F}$; that is $\mathbf{G} = \mathbf{F}(x)$, the field of rational functions in the indeterminate x, with coefficients from $\mathbf{F}$. The rank of A over $\mathbf{G}$ is the same as the rank of A over $\mathbf{F}$. Let X denote the diagonal matrix $diag(1, x, x^2, \ldots, x^{m-1})$ over the field $\mathbf{G}$. Our first lemma is essentially due to Mulmuley [Mul87], who proved it for symmetric matrices A.

Lemma 10 *$rk(A^T XA) = rk(A)$.*

Proof. That $rk(A^T XA) \le rk(A)$ is obvious, since whenever $Au(x) = 0$ for some $u(x) \in \mathbf{G}^n$, $A^T XAu(x) = 0$. For the other inequality, suppose to the contrary that for some $u(x) \in \mathbf{G}^n$, $Au(x) \neq 0$ but $A^T XAu(x) = 0$. Without loss of generality, we may assume that $u(x) \in (\mathbf{F}[x])^n$, the ring of polynomials in x (otherwise, we can multiply $u(x)$ by the lcm of the denominators in $u(x)$). Let $v(x) = Au(x) \in (\mathbf{F}[x])^m$, $v(x) \neq 0$. Let y be a new indeterminate. Now

$$\begin{aligned}
0 &= u(y)^T A^T XAu(x) \\
&= (v(y))^T Xv(x) \\
&= (v_1(y), v_2(y), \ldots, v_m(y)) \begin{pmatrix} 1 & & & \\ & x & & \\ & & \ddots & \\ & & & x^{m-1} \end{pmatrix} \begin{pmatrix} v_1(y) \\ v_2(y) \\ \vdots \\ v_m(y) \end{pmatrix} \\
&= \sum_{i=1}^{m} v_i(y) x^{i-1} v_i(x).
\end{aligned}$$

Let d be the highest degree over all the $v_i(x)$'s, and let t be the highest index among the v_i's that achieves this. Consider the term corresponding to the monomial $y^d x^{t-1} x^d = y^d x^{d+t-1}$. This term must have non-zero coefficient and cannot be canceled; hence the sum is non-zero, a contradiction. $\square$

Let $B = A^T XA$. B is an $n \times n$ matrix with entries that are polynomials in x of degree at most $m - 1$. Each entry, therefore, is an element of the field $\mathbf{F}(x)$. If m is superpolynomial in n, computing the rank of this matrix using Mulmuley's algorithm will require time $O(\log^2 n)$ times the complexity of arithmetic operations in the field $\mathbf{F}(x)$. When $m = n^{\log n}$, this complexity is at least $O(\log^2 n)$. This results in an $O(\log^4 n)$ time algorithm, which is no improvement!

To avoid this complexity, we introduce the following twist, which is present in a different form in Chistov's paper. The rank of B equals the largest r such that there is a $r \times r$

submatrix of B that has a non-zero determinant[1]. On the other hand, the determinant of every submatrix of B is a polynomial in x of degree at most $n(m-1)$. Let $\mathbf{K}$ be an extension of $\mathbf{F}$ such that $\mathbf{K}$ has at least $\ell = nm - n + 1$ elements. Consider the determinant of the largest non-singular submatrix of B. This determinant must be non-zero when x is substituted by a for at least one element $a \in \mathbf{K}$. We will take $\mathbf{K}$ to be an algebraic extension of $\mathbf{F}$ such that $\ell \leq |\mathbf{K}| \leq \ell^{O(1)}$. For every $a \in \mathbf{K}$, let B_a denote the $n \times n$ matrix obtained by substituting a for x in the matrix B. In parallel, we can compute the rank of B_a for every $a \in \mathbf{K}$. This requires $\ell^{O(1)} = (nm)^{O(1)}$ processors, and takes time $O(\log^2 n \log\log(nm))$. The extra $\log\log(nm)$ factor stems from the fact that arithmetic operations in $\mathbf{K}$ cost $O(\log\log(nm))$ time.

In summary, the algorithm is: Given as input an $m \times n$ matrix A (where, wlog. assume $m > n$), compute $B = A^T X A$, where $X = \mathit{diag}(1, x, x^2, \ldots, x^{m-1})$. Let $\ell = nm - n + 1 = O(nm)$. Take an extension field $\mathbf{K}$ of $\mathbf{F}$ such that $\ell \leq |\mathbf{K}| \leq \ell^{O(1)}$ (the field $\mathbf{K}$ can be pre-constructed offline). Using Mulmuley's algorithm, compute in parallel $\mathit{rk}(B_a)$ for all $a \in \mathbf{K}$, and output the largest rank.

A randomized algorithm

Given an $m \times n$ matrix A, where $n = o(m)$, we give a simple probabilistic trick to convert this to an $r \times n$ matrix D such that $\mathit{rk}(D) = \mathit{rk}(A) = r$. Let Y denote the nullspace of A^T, that is, Y is the set of all rows y such that $yA = 0$. Let B denote a basis for Y over the field $\mathbf{F}$. If a matrix C of row vectors in $\mathbf{F}^m$ extends B to full rank, then since

$$\begin{pmatrix} B \\ C \end{pmatrix} A = \begin{pmatrix} 0 \\ CA \end{pmatrix},$$

it follows that $\mathit{rk}(A) = \mathit{rk}(CA) = \mathit{rk}(C)$. Therefore, if C itself has full row-rank, then C has exactly r rows. Using arguments from algebraic geometry, Chistov [Chi85] gives a collection of $\ell = \mathrm{poly}(m, n)$ matrices $C_{r,i}, 1 \leq r \leq n, 1 \leq i \leq \ell$, such that for any matrix A with rank r, one of the $C_{r,i}$'s satisfies the above properties.

We argue that if r is the rank of A, then a *random* matrix of size $r \times m$ is a good candidate for the choice of C. Of course, we don't know r, so we will try all values of r between 1 and n in parallel, and take the maximum value of $\mathit{rk}(CA)$ that we find. The main computational advantage is that the matrix CA is an $r \times n$ matrix over the same field $\mathbf{F}$, whose rank we can easily determine in $O(\log^2 n)$ parallel time using Mulmuley's algorithm. Notice also that we don't need to know B explicitly. The discussion below is restricted to finite fields; the success probability approaches 1 as the size of the field goes to infinity.

Lemma 11 *With the notations used above, the probability that a random $r \times m$ matrix C satisfies $\mathit{rk}(CA) = \mathit{rk}(A)$ is at least $p^* \doteq e^{-3/(2k-2)}$, where $k = |\mathbf{F}|$. For the smallest non-trivial field, namely $GF(2)$, $p^* > 0.223$.*

Proof. It suffices to show that a random $r \times m$ matrix C extends B to full rank with probability at least p^*. Let C be a list of row vectors $C_1, C_2, \ldots, C_r$. By definition, the

[1]If r is an integer between 1 and n, an $r \times r$ submatrix of B is any $r \times r$ matrix obtained by deleting $(n-r)$ rows and $(n-r)$ columns of B.

dimension of the span of the row vectors in B is exactly $m - r$, and hence there are at least $k^m - k^{m-r}$ points in $\mathbf{F}^m$ that are outside $span(B)$. Therefore, the probability that $B \cup \{C_1\}$ has rank $m - r + 1$, which is the same as the probability that $C_1 \notin span(B)$, is exactly $1 - 1/k^r$. Conditioned on this being the case, the probability that $B \cup \{C_1, C_2\}$ has rank $m - r + 2$, which is the same as the probability that $C_2 \notin span(B \cup \{C_1\})$, is exactly $1 - 1/k^{r-1}$. Continuing this argument, we have

$$\begin{aligned}
& \Pr\left[\begin{pmatrix} B \\ C \end{pmatrix} \text{ has rank } m\right] \\
= \; & \prod_{i=1}^{r}\left(1 - \frac{1}{k^i}\right) \\
= \; & \exp\left(\sum_{i=1}^{r} \ln\left(1 - \frac{1}{k^i}\right)\right) \\
\geq \; & \exp\left(\sum_{i=1}^{r}(-3/(2k^i))\right) && \text{since } \ln(1-x) \geq -\frac{3}{2}x \text{ for } x \leq 1/2 \\
\geq \; & \exp(-3/(2k-2)) && \text{by elementary calculations.}
\end{aligned}$$

$\square$