On the Hardness of Permanent

Jin-Yi Cai *[1], A. Pavan[1], and D. Sivakumar **[2]

[1] Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY 14260. {cai, aduri}@cse.buffalo.edu
[2] Department of Computer Science, University of Houston, Houston, TX 77204. siva@cs.uh.edu

Abstract. We prove that if there is a polynomial time algorithm which computes the permanent of a matrix of order n for any inverse polynomial fraction of all inputs, then there is a BPP algorithm computing the permanent for *every* matrix. It follows that this hypothesis implies $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$. Our algorithm works over any sufficiently large finite field (polynomially larger than the inverse of the assumed success ratio), or any interval of integers of similar range. The assumed algorithm can also be a probabilistic polynomial time algorithm. Our result is essentially the best possible based on any black box assumption of permanent solvers, and is a simultaneous improvement of the results of Gemmell and Sudan [GS92], Feige and Lund [FL92] as well as Cai and Hemachandra [CH91], and Toda (see [ABG90]).

1 Introduction

The *permanent* of an $n \times n$ matrix A is defined as

$$\mathrm{per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)} \ ,$$

where S_n is the symmetric group on n letters, i.e., the set of all permutations of $\{1, \ldots, n\}$.

The permanent function has a rich history in combinatorics and in computational complexity theory. All known general algorithms computing the permanent function over the integers take exponential time. In fact, this exponential time complexity remains true for any general algorithm over any field of characteristic other than two. The best known general computational procedure for the permanent is a formula due to Ryser, which runs in time $O(n^2 2^n)$ [Rys63].

In terms of computational complexity theory, in 1979, Valiant [Val79] proved the seminal result that computing the permanent of integer matrices is complete for the counting complexity class #P, and is therefore NP-hard. A decade later, Toda [Tod89] demonstrated the surprising power of #P; Toda's theorem,

* Supported in part by NSF grant CCR-9634665, and by a Guggenheim Fellowship.
** Supported in part by an NSF CAREER award CCR-9734164.

together with Valiant's result, implies that the permanent is hard for the entire polynomial-time hierarchy.

The permanent has two other fascinating properties that endow it with rich computational structure. Lipton [Lip91] observed that the permanent of a matrix with entries that are low degree polynomials in an indeterminate x is itself a low-degree polynomial. In particular, taking linear polynomials in x as entries, the permanent of an $n \times n$ matrix is a polynomial of degree at most n. It follows that the computation of the permanent of a matrix can be reduced, via polynomial interpolation, to computing the permanent of uniformly distributed random matrices. This property of *random self-reducibility* has the important consequence that the permanent is computationally hard in the worst case if and only if it is hard on the average. The permanent also has the *downward self-reducibility* property, which means that computing the permanent of an $n \times n$ matrix can be reduced (in polynomial time) to the computation of the permanent of $(n-1) \times (n-1)$ matrices, via Laplacian expansion. Lund *et al.* [LFKN90] applied this property together with the random self-reducibility, to obtain the breakthrough result that #P has interactive proof protocols. Very recently, Impagliazzo and Wigderson [IW98] have used these two properties to obtain exciting new connections between computational hardness vs. randomness.

The line of research initiated by Lipton [Lip91] connecting the worst-case and the average-case complexities of the permanent was subsequently pursued by Gemmell et al. [GLRSW91], Gemmell and Sudan [GS92], and by Feige and Lund [FL92]. It is shown in [GS92] that if there is a polynomial time algorithm that can compute the permanent on at least a $(1/2) + (1/\text{poly})$ fraction of all $n \times n$ matrices over the finite field $\mathbf{Z}_p$, then there is a probabilistic polynomial time algorithm that can compute the permanent of every $n \times n$ matrix over $\mathbf{Z}_p$, provided p is sufficiently large with respect to n. Feige and Lund [FL92] improved this and showed a similar result under a weaker hypothesis that assumes an algorithm that can compute the permanent on at least a $(1/2) - 1/n$ fraction of all $n \times n$ matrices over the finite field $\mathbf{Z}_p$. A polynomial reconstruction algorithm (aka. Reed-Solomon decoder) of Berlekamp and Welch [BW] is the crucial technical procedure in both of these papers. The interactive protocol of [LFKN90] is another crucial ingredient in [FL92].

Another result regarding the computational complexity of the permanent is due to Cai and Hemachandra [CH91] and independently due to Toda (see [ABG90]). According to this result, if there is a polynomial time algorithm such that for every matrix of order n, it can enumerate a list of polynomially many values, one of which is the correct value of the permanent, then one can compute the permanent of any matrix in polynomial time.

In this note, we show that if there is a polynomial time algorithm that can compute the permanent on at least an inverse polynomial fraction $(1/n^c)$ of all $n \times n$ matrices over the finite field $\mathbf{Z}_p$, then there is a probabilistic polynomial time algorithm that can compute the permanent of every $n \times n$ matrix over $\mathbf{Z}_p$, provided p is somewhat larger than the inverse of the success ratio of the assumed algorithm. The proof can be extended to the case where the assumed algorithm is also a probabilistic polynomial time algorithm. We also prove the same result for matrices over any finite field, (again assuming that the field size is sufficiently large compared to the inverse of the success ratio), and over integers (with an appropriate restriction on the length of the entry size and a corresponding definition of uniform distribution over such an interval of integers.) These results are a simultaneous improvement of the results in [GS92,FL92] as well as in [CH91] (except replacing probabilistic for deterministic polynomial time algorithm in the latter).

We achieve this improvement by applying an improved Reed-Solomon decoder due to Madhu Sudan [Sud96], building on earlier work by Ar *et al.* [ALRS92]. Our proof uses a family of intermediate matrices whose entries are univariate polynomials in an indeterminate x; this definition unifies two ideas used in previous papers, and makes the proof simple and completely self-contained.

2 Hardness over $\mathbf{Z}_p$

We begin with a theorem concerning the complexity of permanent over the finite field $\mathbf{Z}_p$, where p is somewhat larger than the inverse of the success ratio of the assumed algorithm.

Theorem 1. *For any constant $k > 0$, if there exists a deterministic polynomial time algorithm $\mathcal{A}$ such that for all n and $p \geq 9n^{2k+2}$, $\mathcal{A}$ computes the permanent of order n over the field $\mathbf{Z}_p$ correctly on greater than $1/n^k$ fraction of inputs, then* $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$.

Proof. Assume that the success probability of $\mathcal{A}$ is q. Then $q \geq 1/n^k$. Without loss of generality we assume $k \geq 1$. Let M be an $n \times n$ matrix over $\mathbf{Z}_p$ whose permanent we wish to compute. Let $M_{11}, M_{21}, \ldots, M_{n1}$ be the n minors of M. Consider the following matrix polynomial, defined by choosing the matrices B and C (of dimension $n-1$) uniformly and independently at random,

$$D(x) = \sum_{i=1}^{n} \delta_i(x) M_{i1} + \alpha(x)(B + xC),$$

where each $\delta_i(x)$ is a polynomial of degree $n-1$ such that

$$\delta_i(x) = \begin{cases} 1 \text{ if } x = i \\ 0 \text{ if } x \neq i \text{ and } 1 \leq x \leq n, \end{cases}$$

and $\alpha(x)$ is a degree-n polynomial that vanishes for $x = 1, 2, \ldots, n$, given by $\alpha(x) = (x-1)(x-2)\cdots(x-n)$. Note that $D(i) = M_{i1}$, for $1 \leq i \leq n$.

Let $D = \{D(x) \mid x = n+1, n+2, \ldots, p\}$. The permanent of $D(x)$ is a polynomial of degree less than n^2. If we know the polynomial $\mathrm{per}(D(x))$ then we can compute the permanent of M, by Laplacian expansion. If the matrices B and C are chosen uniformly at random, then all the matrices in D are uniformly distributed. Moreover, if the matrices B and C are chosen uniformly and independently, then it can be shown easily that the matrices in D are pairwise independent. So the algorithm $\mathcal{A}$ has success probability close to q on D also. More precisely, the following lemma can be proved by the Chebyshev inequality.

Lemma 1. *The probability (over random choices of B and C) that the algorithm $\mathcal{A}$ correctly computes the permanent on more than $q/2$ fraction of inputs from D is at least $1 - \frac{1}{(p-n)q^2}$.*

Proof. For $i = n+1, n+2, \ldots, p$, define Z_i to be 1 if $\mathcal{A}$ succeeds on $D(i)$, else 0. We know that the expectation $E(Z_i) = q$ for $i = n+1, n+2, \ldots, p$. Define the random variable $Z = \frac{\sum_{i=n+1}^{p} Z_i}{p-n}$ as the average of the Z_i's. Expectation of Z is also q.

$$\begin{aligned}
\Pr[Z \leq (q/2)] &= \Pr[Z - E(Z) \leq (-q/2)] \\
&\leq \Pr[|Z - E(Z)| \geq (q/2)] \\
&\leq \frac{\mathbf{Var}(Z)}{(q/2)^2} \leq \frac{1}{(p-n)q^2}.
\end{aligned}$$

The last inequality is obtained using the facts that Z_i's are pairwise independent and the variance of a 0-1 random variable is at most 1/4. So with high probability $\mathcal{A}$ works correctly on more than $q/2$ fraction of inputs from D. ∎

We call a set D (which is defined by choosing B and C) *good* if, $\mathcal{A}$ works correctly on more than $q/2$ fraction of inputs from D. In the following discussion assume that D is good.

For a polynomial f, the $Graph(f)$ is the set $\{(i, f(i)) \mid i = 1, 2, \ldots, p\}$. Define a set S as follows, $S = \{(i, \mathcal{A}(D(i))) \mid i = n+1, \ldots, p\}$. Consider the set of all polynomials f of degree less than n^2 such that S intersects the set $Graph(f)$ with at least $(p-n)q/2$ points. The polynomial $\mathrm{per}(D(x))$ is one among them, since $\mathcal{A}$ computes the permanent correctly on $(p-n)q/2$ matrices from D. We can prove that there exist at most polynomially (in n) many such polynomials f.

Lemma 2. *If $p \geq 9n^{2k+2}$, then there are at most $3/q$ many polynomials f of degree less than n^2 that satisfy $|Graph(f) \cap S| \geq (p-n)q/2$.*

Proof. If there exist more than $3/q$ such polynomials, consider a set $\mathcal{F}$ of $N = \lceil 3/q \rceil$ polynomials among them. For a polynomial f define the set, $S_f = \{i \mid (i, f(i)) \in Graph(f) \cap S\}$. By the inclusion-exclusion principle

$$\begin{aligned}
p - n \geq \left| \bigcup_{f \in \mathcal{F}} S_f \right| &\geq \sum_{f \in \mathcal{F}} |S_f| - \sum\nolimits_{f, f' \in \mathcal{F}, f \neq f'} |S_f \cap S_{f'}| \\
&\geq N \frac{(p-n)q}{2} - \frac{N(N-1)}{2}(n^2 - 1)
\end{aligned}$$

The last inequality uses the fact that any two distinct polynomials of degree less than n^2 can agree on no more than $n^2 - 1$ points. If $N = \lceil 3/q \rceil$ then $N - 1 < 3/q$

and $N \geq 3/q$. Since $p \geq 9n^{2k+2}$ and $q \geq 1/n^k$, we have $p - n > 9(n^2-1)/q^2$. Thus we obtain the following contradiction

$$\begin{aligned} p - n &\geq \frac{N}{2}\left((p-n)q - (N-1)(n^2-1)\right) \\ &> \frac{N}{2}\left((p-n)q - \frac{3(n^2-1)}{q}\right) \\ &\geq \frac{3}{2q}\left((p-n)q - \frac{3(n^2-1)}{q}\right) \\ &= p - n + \frac{1}{2}\left(p - n - \frac{9(n^2-1)}{q^2}\right) \;\; > \;\; p - n. \end{aligned}$$

∎

Next we show how all these polynomials can be obtained explicitly by a randomized procedure with high probability. To handle a minor technical complication, we will divide this procedure into two cases. We remind the reader that we are working under the assumption that the set D is good, that is, the fraction of inputs from D for which $\mathcal{A}$ works correctly is more than $q/2 = 1/(2n^k)$.

(Case $9n^{2k+2} \leq p < 161n^{3k+2}$):

In this case, we apply the algorithm $\mathcal{A}$ to the matrix $D(x)$ for every $x = n+1, \ldots, p$. Clearly, this can be accomplished in polynomial time. Since D is assumed to be *good*, $\mathcal{A}$ computes the permanent correctly for at least $(p-n)/(2n^k)$ of these $p-n$ matrices. Letting $L = p - n$ and $d = n^2$, we have a list of L pairs $\{(x_j, y_j) \mid j = 1, \ldots, L\}$, such that the x_j's are all distinct, and moreover, there is a polynomial f of degree at most d (namely $\mathrm{per}(D(x))$) whose graph intersects the list on at least $(p-n)/(2n^k)$ places. The condition $p \geq 9n^{2k+2}$ implies that $(p-n)/(2n^k) > \sqrt{2(p-n)n^2} = \sqrt{2Ld}$, for all n.

(Case $p \geq 161n^{3k+2}$):

Pick $L = 40n^{2k+2}$ many values x uniformly and independently from the set $\{n+1, \ldots, p\}$. For the set of (distinct) x's produced by this process, produce the matrix $D(x)$ and apply the algorithm $\mathcal{A}$ to it. Our goal is to argue that with overwhelming probability, for at least $9n^{k+2}$ *distinct* x's (out of the L choices made), $\mathcal{A}$ will give us the correct value of the permanent. (The reason for the choice of 161 and 9 with respect to 40 will be clear shortly.)

Call an x *lucky* if $\mathcal{A}$ computes $\mathrm{per}(D(x))$ correctly. We define the events E_j, $j = 1, \ldots, L$, as follows: the event E_j occurs if the j-th random choice of x is lucky, *and* the j-th random choice of x is distinct from all the previously chosen lucky x's. The worst case for E_j to occur is when $j = L$ and all the previously chosen x's were distinct and lucky, thus making it least likely that the jth choice from $\{n+1, \ldots, p\}$ is a lucky x distinct from all previous points. Even in this case, there are more than $((p-n)/(2n^k)) - L$ distinct lucky x's that have not been picked, and which, if picked next, would cause E_j to occur. Therefore, for every j, and under any condition on the events $E_1, E_2, \ldots, E_{j-1}$, the probability that E_j occurs is at least

$$\frac{\frac{p-n}{2n^k} - L}{p-n} = \frac{1}{2n^k} - \frac{40n^{2k+2}}{p-n} \geq \frac{1}{2n^k} - \frac{1}{4n^k} = \frac{1}{4n^k},$$

since $p \geq 161n^{3k+2}$. With this estimate we will prove that with high probability at least $9n^{k+2}$ out of the L events occur.

The event "at least $9n^{k+2}$ out of L E_j's occur" is stochastically dominated by the event that we obtain at least $9n^{k+2}$ successes in L Bernoulli trials, each having an independent success probability of $1/(4n^k)$. This can be seen by performing the L Bernoulli trials in the following fashion: first, perform the experiment which defines E_j, where an occurrence of E_j is counted as a success in the jth Bernoulli trial; second, for each j, if E_j did not occur, let the jth Bernoulli trial be a success with probability $1/(4n^k) - e_j$, where e_j is the conditional probability of E_j occurring given the previous occurrences (and non-occurrences) of $E_1, E_2, \ldots, E_{j-1}$.

Now the probability of the event that "at least $9n^{k+2}$ successes in L Bernoulli trials with success probability of $1/(4n^k)$" can be shown to be very close to one, using Chernoff bounds. Therefore, with very high probability we have $9n^{k+2}$ distinct x's for which the value of the polynomial $\mathrm{per}(D(x))$ of degree less than n^2 is available. Of course, these x's and the values of $\mathrm{per}(D(x))$ are part of a list of at most L pairs (x_j, y_j). Once again, recalling that $L = 40n^{2k+2}$ and letting $d = n^2$, we notice that $9n^{k+2} > \sqrt{2Ld} = \sqrt{80n^{2k+4}}$.

To summarize, in either case, with high probability, we have a list $\{(x_j, y_j)\}$ of at most L pairs (for some L which is polynomially bounded in n) such that the graph of the polynomial $\mathrm{per}(D(x))$ of degree at most $d = n^2$, intersects the list on more than $\sqrt{2Ld}$ places. Given such a list, the following lemma of Sudan [Sud96], building on earlier work by [ALRS92], shows how one can construct all the polynomials f whose graphs intersect the list on more than $\sqrt{2Ld}$ places. We note that Sudan's procedure is based on bivariate polynomial factoring. Therefore, it can be implemented in randomized polynomial time in L and d and $\log p$, or in deterministic time polynomial in L, d, and p (see [Kal92]).

Lemma 3 ([Sud96]). *Given a sequence of L distinct pairs $\{(x_i, y_i) \mid 1 \leq i \leq L\}$, where x_i and y_i are elements of a field $\mathbf{F}$. Let t and d be integers such that $t > \sqrt{2Ld}$. Then there is a probabilistic polynomial time algorithm that finds all polynomials f of degree at most d such that the number of i's such that $y_i = f(x_i)$ is at least t.*

Sudan's polynomial reconstruction algorithm is very elegant and simple. For the sake of completeness, we will sketch his algorithm here.

Consider all bivariate polynomials of the form $F(x, y) = \sum_{i,j} a_{i,j} x^i y^j$, where $i + jd \leq \sqrt{2Ld}$. Thus there are exactly

$$\sum_{j=0}^{\lfloor\sqrt{2L/d}\rfloor} \left(\lfloor\sqrt{2Ld}\rfloor - jd + 1\right) = \left(\lfloor\sqrt{2L/d}\rfloor + 1\right)\left(\lfloor\sqrt{2Ld}\rfloor + 1\right) - d\binom{\lfloor\sqrt{\frac{2L}{d}}\rfloor + 1}{2},$$

many coefficients $a_{i,j}$. This quantity can be shown to be strictly greater than L as follows:

Let $I = \lfloor\sqrt{2L/d}\rfloor$, then $\sqrt{2L/d} = I + x$, for some $0 \le x < 1$. Since $\lfloor\sqrt{2Ld}\rfloor + 1 > \sqrt{2Ld}$, the total sum is strictly greater than

$$\left(\sqrt{2L/d} + 1 - x\right)\sqrt{2Ld} - \frac{d}{2}\left(\sqrt{2L/d} + 1 - x\right)\left(\sqrt{2L/d} - x\right),$$

which can be simplified to $L + \sqrt{\frac{Ld}{2}} + \frac{x(1-x)d}{2} > L$.

Now if we set $F(x_i, y_i) = 0$ for all $1 \le i \le L$, we have a homogeneous linear equation system in the coefficients $a_{s,t}$, with more unknowns than L, the number of equations, and hence we can find at least one non-trivial bivariate polynomial F. If we substitute $y = f(x)$ where f is a polynomial of degree at most d and passes through at least t points (x_i, y_i), then we have a univariate polynomial $F(x, f(x))$ of degree at most $\sqrt{2Ld}$, but vanishes on $t > \sqrt{2Ld}$ many points. Hence $F(x, f(x))$ is identically 0 in $\mathbf{F}[x]$. Thus as polynomials in $\mathbf{F}(x)[y]$, $y - f(x) \mid F(x, y)$. But since $y - f(x)$ is monic and in fact belongs to $\mathbf{F}[x][y]$, $y - f(x) \mid F(x, y)$ also in $\mathbf{F}[x, y]$.

Clearly $y - f(x)$ is irreducible in $\mathbf{F}[x, y]$, which is a UFD. In probabilistic polynomial time one can factor a polynomial in $\mathbf{F}[x, y]$ [Kal92], then $y - f(x)$ must be one of the the irreducible factors.

We return to the proof of Theorem 1. Thus we have a randomized procedure that, with high probability, computes a list of at most $3/q = O(n^k)$ polynomials, such that one of them is the permanent of the $(n-1) \times (n-1)$ matrix $D(x)$. The remaining task is to identify the correct polynomial from this list. Two ideas come into play here: First, we can find in deterministic polynomial time a point $v \in \mathbf{Z}_p$, such that all $3/q$ polynomials disagree on v. This is because each pair of polynomials can agree on at most n^2 points, and there are strictly less than $9n^{2k}$ such pairs, and $p \ge 9n^{2k+2}$. Secondly, if we can *somehow* obtain the *correct* value of $\mathrm{per}(D(v))$, then we can eliminate all but at most one polynomial on our list, by cross-checking the values of each polynomial in our list at v against the correct value of $\mathrm{per}(D(v))$. Assuming D is *good*, then with very high probability, the correct polynomial $\mathrm{per}(D(x))$ is on the list of $O(n^k)$ polynomials, thus exactly one polynomial must remain, and the remaining polynomial must be the correct $\mathrm{per}(D(x))$. Furthermore, D is *good* with high probability. Once we have the correct polynomial, by evaluating the correct polynomial $\mathrm{per}(D(x))$ at $x = 1, \ldots, n$, we may compute the permanents of the n minors of the matrix M that we started with, and thus also compute $\mathrm{per}(M)$.

What we have achieved is a reduction, using $\mathcal{A}$ as an auxiliary procedure, of the computation of the permanent of $n \times n$ matrices to the computation of the permanent of $(n-1) \times (n-1)$ matrices. The reduction is probabilistic, and has a high probability of success. In fact, the error probability is bounded by the probability that D is not *good*, which is at most $1/((p-n)q^2) = O(1/n^2)$, plus the probability that given a good D still we did not get a sufficient number of distinct points on which $\mathcal{A}$ evaluates correctly, which is exponentially small, plus the failure probability of the factoring algorithm, which also can be made exponentially small. Thus the overall error probability is $O(1/n^2)$, say c/n^2 for some constant $c > 0$. Therefore, if we carry this process through $n, n-1, \ldots, K$ for some large constant K, the total error probability is bounded by $\sum_{j=K}^{n}(c/j^2)$, which can be made arbitrarily small, say $\epsilon < 1/2$, by choosing a large enough K. This implies that we may use the *same* procedure recursively to compute the correct value of $\mathrm{per}(D(v))$. The recursion terminates when the order of the matrix becomes less than K, and we can compute the permanent directly.

Finally, to show that $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$, we need a probabilistic polynomial time algorithm for the permanent with negligible (less than any inverse polynomial) error probability. We can achieve that by repeating the above algorithm for the permanent a sufficiently large polynomial number of times, and taking the majority vote. By Chernoff bound, this will succeed with exponentially small error probability. ∎

3 Some Extensions

The above proof can be seen to work over any finite field as long as the cardinality of the field is at least as large as $9n^{2k+2}$, and also there is a polynomial length representation of field elements.

Theorem 2. *Fix any constant $k > 0$, if there exists a deterministic polynomial time algorithm $\mathcal{A}$ such that for all n, $\mathcal{A}$ computes the permanent of order n over the finite field $\mathbf{F}$ of characteristic other than two correctly on greater than $1/n^k$ fraction of inputs, where $|\mathbf{F}| > 9n^{2k+2}$, and each field element has a representation of bit length at most $n^{O(1)}$, then $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$.*

Similarly we can extend the proof to the case of integers. Admittedly this is the most interesting case classically. But our proof will reduce this case to the case with a finite field $\mathbf{Z}_p$, for an appropriate prime p.

We must first define properly what is meant to be the uniform distribution of integer $n \times n$ matrices. For our purposes, we will define simply as follows: Consider any bit length ℓ between $\Omega(\log n)$ and $n^{O(1)}$. Then we consider uniform distribution of all integer $n \times n$ matrices where each entry of the matrix is an integer with absolute value bounded by 2^ℓ.

Theorem 3. *For any constant $k > 0$, if there exists a deterministic polynomial time algorithm $\mathcal{A}$ such that for all n, $\mathcal{A}$ computes the permanent of order n over the integers in the interval $[-2^\ell, 2^\ell]$ on greater than $1/n^k$ fraction of inputs, where $(2k+3)\log_2 n \le \ell \le n^{O(1)}$, then $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$.*

We prove this by choosing a prime close to 2^ℓ, and reason in the finite field $\mathbf{Z}_p$. We omit the details.

The proof of our theorems can also be carried out assuming only the existence of a probabilistic polynomial time algorithm $\mathcal{B}$ with the expected success ratio at least inverse polynomial on an inverse polynomial fraction of the inputs. We omit the proof here.

Theorem 4. *The same result holds in the above theorems if we assumed the existence of a probabilistic polynomial time algorithm only.*

Our theorems are an improvement of the results in [GS92,FL92]. They also simultaneously generalize the results in [CH91]. In this latter result, it is assumed that there exists a polynomial time algorithm such that for every matrix of order n, it can enumerate a list of polynomially many values, one of which is the correct value of the permanent. By taking one entry from such a list at random, we obtain a probabilistic polynomial time algorithm with an inverse polynomial success ratio.

In a related development (after this paper was submitted to STACS), Goldreich, Ron, and Sudan published a technical report in ECCC [GRS98] showing that if there is a polynomial time algorithm $\mathcal{B}$ that is able to guess the permanent of a random $n \times n$ matrix on $2n$-bit integers modulo a random n-bit prime with inverse polynomial success probability, then $\mathrm{P}^{\#\mathrm{P}} = \mathrm{BPP}$. To prove this result, they develop algorithmic tools to decode an error correcting code based on the Chinese Remainder Theorem. While their decoding algorithms may be of independent interest, the above result on permanent, which is the main motivation for their algorithm, may be proved directly from our results in this paper.

Specifically, given such an algorithm $\mathcal{B}$, we can show how to produce an algorithm $\mathcal{A}$ that meets the hypothesis of Theorem 3. The idea is to randomly choose $p(n)$ many n-bit primes for a large polynomial $p(n)$, (by choosing polynomially many integers of this size and applying a probabilistic primality test). With high probability, we will be able to find sufficiently many primes p such that the algorithm $\mathcal{B}$ succeeds with a fixed inverse polynomial probability in computing the permanent modulo p on random $n \times n$ matrices. Here "sufficiently many" means that the number of such primes, which we will call *good* primes, is sufficient, via Chinese remaindering, to describe any integer that might be the value of the permanent of $n \times n$ matrices of $2n$-bit integers. Furthermore, by the distribution of primes according to the Prime Number Theorem, with high probability, we will have a sufficiently many good primes in hand, among all the primes generated, which are large enough in value to apply the procedures in the proof of Theorem 1. Of course, we will also have many *bad* primes where the algorithm $\mathcal{B}$ fails to achieve such success rate. The main idea is that, through the use of the random self-reducibility and downward self-reducibility of the permanent (as in the proof of Theorem 1), for any prime, with high probability we will know whether the procedures in the proof of Theorem 1 had succeeded or not. The essential idea is contained in the LFKN protocol. We can amplify the correctness probability of this identification of good primes exponentially close to one, namely $> 1 - e^{-n^c}$ for any constant c. Once we have sufficiently many good primes identified, we may apply the proof ideas of Theorem 1 and for a given matrix M, compute its permanent modulo the good primes, and finally apply an error-free Chinese remaindering to compute the permanent of the matrix M.

The permanent function has played a pivotal role in complexity theory, e.g., see [Lip91,LFKN90,IW98]. Frequently, it is a standard technique to use Yao's XOR Lemma to amplify the unpredictability of some hard function. The theorems presented here has the interesting feature that, for the permanent function, no such amplification is needed if one requires only less than inverse polynomial unpredictability. Perhaps these theorems can be used as an alternative to the standard amplification technique. The advantage is that we do not need any replication of the inputs. This has been an important concern in some recent results due to Impagliazzo and Wigderson [IW98].

References

[ABG90] A. Amir, R. Beigel, and W. Gasarch. Some connections between query classes and non-uniform complexity. In *Proceedings of the 5th Structure in Complexity Theory*, pages 232–243. IEEE Computer Society, 1990.

[ALRS92] S. Ar, R. Lipton, R. Rubinfeld, and M. Sudan. Reconstructing algebraic functions from mixed data. In *Proc. 33rd FOCS*, pages 503–512, 1992.

[BW] E. Berlekamp and L. Welch. Error correction of algebraic codes. US Patent Number 4,633,470.

[CH91] J. Cai and L. Hemachandra. A note on enumerative counting. *Information Processing Letters*, 38(4):215–219, 1991.

[FL92] U. Feige and C. Lund. On the hardness of computing permanent of random matrices. In *Proceedings of 24th STOC*, pages 643–654, 1992.

[GLRSW91] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of 23rd STOC*, pages 32–42, 1991.

[GS92] P. Gemmell and M. Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43:169–174, 1992.

[GRS98] O. Goldreich and D. Ron and M. Sudan. Chinese remaindering with errors. ECCC Technical Report TR 98-062, October 29, 1998. Available at `www.eccc.uni-trier.de`.

[IW98] R. Impagliazzo and A. Wigderson. Randomness vs Time, De-randomization under a uniform assumption. Manuscript, 1998. *To appear in FOCS '98*.

[Kal92] E. Kaltofen. Polynomial factorization 1987–1991. *LATIN '92*, I. Simon (Ed.), LNCS, vol.583, pp294–313, Springer, 1992.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings of 31st FOCS*, pages 2–10, 1990.

[Lip91] R. Lipton. *New directions in testing*, In Distributed Computing and Cryptography, volume 2 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 191–202. AMS, 1991

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[Rys63] H. J. Ryser. *Combinatorial Mathematics.* Carus Mathematical Monograph No 14, Math. Assoc. of America, 1963.

[Sud96] M. Sudan. Maximum likelihood decoding of Reed-Solomon codes. In *Proceedings of the 37th FOCS*, pages 164–172, 1996.

[Tod89] S. Toda. On the computational power of PP and ⊕P. In *Proceedings of the 30th FOCS*, pages 514–519, 1989.

[Val79] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 47(1):85–93, 1979.