Electronic Colloquium on Computational Complexity, Report No. 88 (2000)

A note on the hardness of the characteristic polynomial

Meena Mahajan.
Institute of Mathematical Sciences,
Chennai 600 113, India.
meena@imsc.ernet.in.

V Vinay.
Department of Computer Science and Automation,
Indian Institute of Science,
Bangalore 560 012, India.
vinay@csa.iisc.ernet.in.

November 28, 2000

1 Introduction

In this note, we consider the problem of computing the coefficients of the characteristic polynomial of a given matrix, and the related problem of verifying the coefficents.

Santha and Tan [ST98] show that verifying the determinant (the constant term in the characteristic polynomial) is complete for the class C=L, and ask whether verification becomes easier if all coefficients are given. Hoang and Thierauf [HT00] answer this negatively; they show that verifying all coefficients is also complete for C=L. One of our main contributions here is a considerably simplified proof of this latter result. Our simplified proof is combinatorial, as opposed to the algebraic proof of [HT00].

It is well known [Dam91, Tod91, Tod92, Vin91a, Vin91b, Val92] that computing the determinant i.e. the constant term of the characteristic polynomial is hard for GapL. On the other hand, the coefficients of high degree terms are trivial or easy to compute: the leading coefficient is always one, the second leading coefficient is the trace of the matrix and is hence computable in TC^0. Thus one may ask the question as to how many coefficients are easy to compute. We partially answer this question by showing that, for an $n \times n$ matrix:

1. For constant k, computing the coefficient of x^{n-k} is in TC^0.
2. For constant c and for $k \in O((\log n)^c)$, the coefficient of x^{n-k} can be computed by polynomial size threshold circuits of $O(\log \log n)$ depth.
3. For any fixed $0 < \epsilon \leq 1$, computing the coefficient of x^{n-n^ϵ} is hard for GapL. Verifying it is hard for C=L.

ISSN 1433-8092

2 Verifying the characteristic polynomial

Consider the following functional and verification problems.
V-CHARPOLYNOMIAL: Input: An $n \times n$ matrix A with n-bit integer entries, a sequence $\langle c_n, c_{n-1}, \ldots, c_0 \rangle$. Question: Is $\chi_A(x) = \sum_{i=0}^{n} c_i x^i$?

Input: An $n \times n$ matrix A with n-bit integer entries, and i, j, m, $(1 \leq i, j, m \leq n)$.
POWERELEMENT: Find $A^m[i, j]$.
V-POWERELEMENT: With additional input an integer a, decide if $A^m[i, j] = a$.

Input: An $n \times n$ matrix A with n-bit integer entries.
DETERMINANT: Find $\det(A)$.
V-DETERMINANT: With additional input an integer a, decide if $\det(A) = a$.

(In all the above problems, without loss of generality, one could consider rationals instead of integers.)

Theorem 1 (Theorem 3.2 in [HT00]) *The problem V-CHARPOLYNOMIAL is complete for* C=L *under uniform projections.*

Proof: Since computing the characteristic polynomial is in GapL, verifying it is trivially in C=L. To show hardness, the proof of [HT00] begins with the problem V-POWERELEMENT that is known to be hard for C=L, and initially follows the construction that reduces POWERELEMENT to DETERMINANT.
Stage 1: Treat A as the adjacency matrix of a directed bipartite graph G_0. with n vertices in each partition. Make m copies of G_0 and cascade them to get G_1. That is, vertices of G_1 are (u, k) for $u \in \{1, \ldots, n\}$ and $k \in \{0, \ldots, m\}$. For each edge $\langle u, v \rangle$ of G_0 with weight $A[u, v]$, there are edges $\langle (u, k), (v, k-1) \rangle$ in G_1 with the same weight, where $k \in \{0, \ldots, m-1\}$. G_1 is a directed acyclic layered graph, with m layers of edges.

The weight of a path is the product of the weights of the path edges.

Let s denote the vertex $(i, 0)$, and t denote the vertex (j, m). $A^m[i, j]$ is the sum of weights of paths in G_1 from s to t.
Stage 2: Subdivide each edge, to get graph G_2 with $2m$ layers of edges.
Stage 3: Add an edge from t to s of weight $+1$ to get the graph G_3.
Stage 4: Add a self-loop only at t, with weight -1, to get graph G_4. This graph has $N = n(m+1) + me$ vertices, where e is the number of edges in G_0. Let D be the adjacency matrix of G_4.

In [HT00] it is shown that

$$\chi_D(x) = x^N + x^{N-1} + A^m[i, j] x^{N-(2m+1)}.$$

This is proved by analyzing the effect of a series of row operations on the matrix $D + I$, and then relating χ_{D+I} to χ_D. We now give a significantly simpler and shorter proof of this fact.

We use the simple combinatorial fact that the coefficient of x^{N-i} in χ_D is the sum of the signed weights of all partial cyle covers in the associated graoh G_D that touch exactly i vertices. (See for instance [Str83, SW86, MV99]. A partial cycle cover with l vertices and k

cycles and weight w contributes $(-1)^k w$ to c_l.) Now observe that G_4 is essentially layered, except for the edges $t \longrightarrow s$ and $t \longrightarrow t$. So any cycle in G_4 must use one of these edges. Partial cycle covers need disjoint cycles, and thus can use at most one of these. So the only cycles in G_4 are of these two types: (1) covering one vertex: the cycle $t \longrightarrow t$ of weight -1. This is the sole contribution to c^{N-1}; hence $c_{N-1} = 1$. (2) covering $2m+1$ vertices, one from each layer: a cycle that uses the the $t \longrightarrow s$ edge and then traces out an $s \longrightarrow t$ path. But the total weight of these paths is precisely $A^m[i,j]$. So the total contribution of these cycles to $c^{N-(2m+1)}$ is $A^m[i,j]$. This contribution comes with a negative sign, because each such partial cycle cover has exactly one cycle. ■

Note that in the above construction, the edge added in stage 4 plays no crucial role; it only contributes to $c_{N-1} = 1$. So in fact, if E is the adjacency matrix of G_3, then we have proved that

$$\chi_E(x) = x^N + A^m[i,j]x^{N-(2m+1)}.$$

This suffices to show the required hardness result.

3 Computing individual coefficients of the characteristic polynomial

Computing the determinant i.e. the constant term of the characteristic polynomial is hard for GapL. On the other hand, the coefficients of high degree terms are trivial or easy to compute: the leading coefficient is always one, the second leading coefficient is the trace of the matrix and is hence computable in TC^0. Thus one may ask the question as to how many coefficients are easy to compute. We show upper bounds for the high degree terms and lower bounds for the low degree terms.

More precisely, for an $N \times N$ matrix D, computing the coefficient of $x^{N-f(N)}$ in $\chi(D)$, for $0 \leq f(N) \leq N$, is trivial if $f(N) = 0$ and GapL-hard if $f(N) = N$. As $f(N)$ goes from 0 to N, at what value does this start becoming hard? Starting from the easy end, this remains easy (read in TC^0) if $f(N) = k$ for some constant k. Starting from the hard end, this remains hard (read hard for GapL) if $f(N) = N^c$ for $0 < c \leq 1$.

Theorem 2 *Given an $n \times n$ matrix A and $0 \leq k \leq n$, computing the coefficient of x^{n-k} in the characteristic polynomial of A*

(1.) is reducible, via uniform projections, to computing the product of k matrices of size $2n^2 \times 2n^2$.

(2.) can be done by polynomial size semi-unbounded arithmetic circuits over $(+, \times)$ of depth $O(\log k)$.

(3.) can be done by polynomial size threshold circuits of $O(\log k)$ depth.

Proof: (1.) This follows from the Mahajan-Vinay algorithm [MV97, MV99] for computing the characteristic polynomial. The algorithm constructs partial clow sequences in stages, and all partial clows equences which are not partial cycle covers cancel out, leaving just the

contribution to the characteristic polynomial coefficients. The coefficient of x^{n-k} requires partial cycle covers touching k vertices, and this corresponds to a uniform branching program of length k. Going from the branching program to iterated matric multiplication is standard. (Note that the Samuelson-Berkowitz algorithm [Ber84, Sam42] also reduces characteristic polynomial computation to iterated matrix multiplication. But there, irrespective of which coefficient is required, one has to compute the product of n matrices.)

(2.) This follows from seeing how the hardness of iterated matrix multiplication varies with the number of multiplicands. Note that multiplying two matrices can be performed by a polynomial size depth 2 arithmetic circuit with unbounded fanin + gates and bounded fanin $\times$ gates in an obvious manner. Since matrix multiplication is associative, the result follows.

(3.) Both the arithmetic gates used above (integer multiplication, iterated integer addition) can be realised by Boolean TC^0 circuits, giving the upper bound. ■

In terms of Boolean circuits, for small values of k, the last upper bound above translates as follows.

Corollary 2.1 *1. For constant k, computing the coefficient of x^{n-k} is in* TC^0.

2. For constant $c > 0$ and $k \in O((\log n)^c)$, the coefficient of x^{n-k} is computable by polynomial size threshold circuits of $O(\log\log n)$ depth.

Note that the upper bound for Boolean circuits ((3.) in Theorem 2) is not optimal even within known bounds for large k. For instance, for $k = n$, i.e. for computing the determinant, it gives an upper bound of TC^1 rather than GapL. Even the bound (2.) in Theorem 2 is not tight; as an arithmetic circuit, the resources place it in GapLogCFL as opposed to GapL. The upper bound should really be stated in terms of iterated matrix multiplication; all matrices are of size polynomial in n, and the number of matrices depends on k. Thus the branching program model is what correctly captures the complexity of computing the characteristic polynomial.

On the flip side, we show that for a fairly large range of exponents, the corresponding coefficients of the characteristic polynomial are as hard to compute as the determinant.

Theorem 3 *Given an $N \times N$ matrix D and a fixed $0 < \epsilon \leq 1$, computing the coefficient of x^{N-N^ϵ} in $\chi(D)$ is hard for* GapL. *Verifying it is hard for* C=L.

Proof: We demonstrate a simple reduction from determinant to this problem. As in the previous section, we use the fact that the coefficient of x^{N-i} in χ_D is the sum of the signed weights of all partial cyle covers that touch exactly i vertices. Consider an $n \times n$ matrix A whose determinant we wish to compute. Construct an $N \times N$ matrix B which has A in the top-left corner and all remaining rows and columns have zeroes. Let G and H be the graphs corresponding to A and B. Since B has $N - n$ isolated vertices, any cycle cover of G is a partial cycle cover covering n vertices of B, and these are the only n-partial cycle covers of H. Thus in $\chi(B)$, the coefficient of x^{N-n} is $\det(A)$. (sign?) Choosing $N = n^{1/\epsilon}$ gives the desired result. ■

References

[Ber84] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18:147–150, 1984.

[Dam91] C. Damm. DET=$L^{(\#L)}$. Technical Report Informatik–Preprint 8, Fachbereich Informatik der Humboldt–Universität zu Berlin, 1991.

[HT00] Hoang and Thierauf. The complexity of verifying the characteristic polynomial and testing similarity. In *SCT: Annual Conference on Computational Complexity*, 2000.

[MV97] Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chicago Journal of Theoretical Computer Science*, 1997, December 1997.

[MV99] Meena Mahajan and V. Vinay. Determinant: Old algorithms, new insights. *SIAM Journal on Discrete Mathematics*, 12(4):474–490, November 1999.

[Sam42] P. A. Samuelson. A method of determining explicitly the coefficients of the characteristic polynomial. *Ann. Math. Stat.*, 13:424–429, 1942.

[ST98] Santha and Tan. Verifying the determinant in parallel. *CMPCMPL: Computational Complexity*, 7, 1998.

[Str83] H. Straubing. A combinatorial proof of the Cayley-Hamilton theorem. *Discrete Maths.*, 43:273–279, 1983.

[SW86] D. Stanton and D. White. *Constructive Combinatorics*. Springer-Verlag, 1986.

[Tod91] S. Toda. Counting problems computationally equivalent to the determinant. manuscript, 1991.

[Tod92] S. Toda. Classes of arithmetic circuits capturing the complexity of computing the determinant. *IEICE Trans. Inf. and Syst.*, E75-D:116–124, 1992.

[Val92] L. G. Valiant. Why is boolean complexity theory difficult? In M. S. Paterson, editor, *Boolean Function Complexity*. Cambridge University Press, 1992. London Mathematical Society Lecture Notes Series 169.

[Vin91a] V Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proc. 6th Structure in Complexity Theory Conference*, pages 270–284, 1991.

[Vin91b] V Vinay. *Semi-unboundedness and complexity classes*. PhD thesis, Indian Institute of Science, Bangalore, July 1991.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'