Efficient Computation of the Characteristic Polynomial of a Threshold Graph

Martin Fürer(✉)

Department of Computer Science and Engineering, Pennsylvania State University, University Park, State College, PA 16802, USA
furer@cse.psu.edu
http://www.cse.psu.edu/∼furer

Abstract. An efficient algorithm is presented to compute the characteristic polynomial of a threshold graph. Threshold graphs were introduced by Chvátal and Hammer, as well as by Henderson and Zalcstein in 1977. A threshold graph is obtained from a one vertex graph by repeatedly adding either an isolated vertex or a dominating vertex, which is a vertex adjacent to all the other vertices. Threshold graphs are special kinds of cographs, which themselves are special kinds of graphs of clique-width 2. We obtain a running time of $O(n \log^2 n)$ for computing the characteristic polynomial, while the previously fastest algorithm ran in quadratic time.

Keywords: Efficient algorithms · Threshold graphs · Characteristic polynomial

1 Introduction

The characteristic polynomial of a graph $G = (V, E)$ is defined as the characteristic polynomial of its adjacency matrix A, i.e. $\chi(G, \lambda) = \det(\lambda I - A)$. The characteristic polynomial is a graph invariant, i.e., it does not depend on the enumeration of the vertices of G. The complexity of computing the characteristic polynomial of a matrix is the same as that of matrix multiplication [10,13] (see [2, Chap.16]), currently $O(n^{2.376})$ [4]. For special classes of graphs, we expect to find faster algorithms for the characteristic polynomial. Indeed, for trees, a chain of improvements [12,16] resulted in an $O(n \log^2 n)$ time algorithm [7]. The determinant and rank of the adjacency matrix of a tree can even be computed in linear time [5]. For threshold graphs (defined below), Jacobs et al. [9] have designed an $O(n^2)$ time algorithm to compute the characteristic polynomial. Here, we improve the running time to $O(n \log^2 n)$. As usual, we use the algebraic complexity measure, where every arithmetic operation counts as one step. Throughout this paper, $n = |V|$ is the number of vertices of G.

Threshold graphs [3,8] are defined as follows. Given n and a sequence $b = (b_1, \ldots, b_{n-1}) \in \{0,1\}^{n-1}$, the threshold graph $G_b = (V, E)$ is defined by

M. Fürer—Research supported in part by NSF Grant CCF-1320814.

J. Wang and C. Yap (Eds.): FAW 2015, LNCS 9130, pp. 45–51, 2015.
DOI: 10.1007/978-3-319-19647-3_5

$V = [n] = \{1, \ldots, n\}$, and for all $i < j$, $\{i, j\} \in E$ iff $b_i = 1$. Thus G_b is constructed by an iterative process starting with the initially isolated vertex n. In step $j > 1$, vertex $n - j + 1$ is added. At this time, vertex j is isolated if b_j is 0, and vertex j is adjacent to all other (already constructed) vertices $\{j+1, \ldots, n\}$ if $b_j = 1$. It follows immediately that G_b is isomorphic to $G_{b'}$ iff $b = b'$. G_b is connected if $b_1 = 1$, otherwise vertex 1 is isolated. Usually, the order of the vertices being added is $1, 2, \ldots, n$ instead of $n, n-1, \ldots, 1$. We choose this unconventional order to simplify our main algorithm.

Threshold graphs have been widely studied and have several applications from combinatorics to computer science and psychology [11].

In the next section, we study determinants of weighted threshold graph matrices, a class of matrices containing adjacency matrices of threshold graphs. In Sect. 3, we design our efficient algorithm to compute the characteristic polynomial of threshold graphs. We also look at its bit complexity in Sect. 4, and finish with open problems.

2 The Determinant of a Weighted Threshold Graph Matrix

We are concerned with adjacency matrices of threshold graphs, but we consider a slightly more general class of matrices. We call them weighted threshold graph matrices. Let $M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}}$ be the matrix with the following entries.

$$\left(M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}}\right)_{ij} = \begin{cases} b_i & \text{if } i < j \\ b_j & \text{if } j < i \\ d_i & \text{if } i = j \end{cases}$$

Thus, the weighted threshold matrix for $(b_1 b_2 \ldots b_{n-1}; d_1 d_2 \ldots d_n)$ looks like this.

$$M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}} = \begin{pmatrix} d_1 & b_1 & b_1 & \ldots & b_1 & b_1 \\ b_1 & d_2 & b_2 & \ldots & b_2 & b_2 \\ b_1 & b_2 & d_3 & \ldots & b_3 & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ b_1 & b_2 & b_3 & \ldots & d_{n-1} & b_{n-1} \\ b_1 & b_2 & b_3 & \ldots & b_{n-1} & d_n \end{pmatrix}$$

In order to compute the determinant of $M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}}$, we subtract the penultimate row from the last row and the penultimate column from the last column. In other words, we do a similarity transformation with the following regular matrix

$$P = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & -1 \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{pmatrix},$$

i.e.,

$$P_{ij} = \begin{cases} 1 & \text{if } i = j \\ -1 & \text{if } i = n \text{ and } j = n-1 \\ 0 & \text{otherwise.} \end{cases}$$

The row and column operations applied to $M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}}$ produce the similar matrix

$$P^T M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}} P = \begin{pmatrix} d_1 & b_1 & b_1 & \ldots & b_1 & 0 \\ b_1 & d_2 & b_2 & \ldots & b_2 & 0 \\ b_1 & b_2 & d_3 & \ldots & b_3 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ b_1 & b_2 & b_3 & \ldots & d_{n-1} & b_{n-1} - d_{n-1} \\ 0 & 0 & 0 & \ldots & b_{n-1} - d_{n-1} & d_n + d_{n-1} - 2b_{n-1} \end{pmatrix}$$

Naturally, the determinant of P is 1, implying

$$\det\left(P^T M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}} P\right) = \det\left(M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}}\right).$$

Furthermore, we observe that $P^T M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}} P$ has a very nice pattern.

$$P^T M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}} P = \begin{pmatrix} \boxed{M^{d_1 d_2 \ldots d_{n-1}}_{b_1 b_2 \ldots b_{n-2}}} & \begin{matrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ b_{n-1} - d_{n-1} \end{matrix} \\ 0\ 0\ 0 \ldots 0\ b_{n-1} - d_{n-1} & d_n + d_{n-1} - 2b_{n-1} \end{pmatrix}$$

To further compute the determinant of $P^T M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}} P$, we use Laplacian expansion by minors applied to the last row.

$$\begin{aligned} &\det\left(M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}}\right) = \det\left(P^T M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}} P\right) \\ &= (d_n + d_{n-1} - 2b_{n-1}) \det\left(M^{d_1 d_2 \ldots d_{n-1}}_{b_1 b_2 \ldots b_{n-2}}\right) - (b_{n-1} - d_{n-1})^2 \det\left(M^{d_1 d_2 \ldots d_{n-2}}_{b_1 b_2 \ldots b_{n-3}}\right) \end{aligned}$$

By defining the determinant of the 0×0 matrix $M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}}$ with $n = 0$ to be 1, and checking the determinants for $n = 1$ and $n = 2$ directly, we obtain the following result.

Theorem 1. *$D_n = \det\left(M^{d_1 d_2 \ldots d_n}_{b_1 b_2 \ldots b_{n-1}}\right)$ is determined by the recurrence equation*

$$D_n = \begin{cases} 1 & \text{if } n = 0 \\ d_1 & \text{if } n = 1 \\ (d_n + d_{n-1} - 2b_{n-1}) D_{n-1} - (b_{n-1} - d_{n-1})^2 D_{n-2} & \text{if } n \geq 2 \end{cases}$$

□

This has an immediate implication, as we assume every arithmetic operation takes only 1 step.

Corollary 1. *The determinant of an $n \times n$ weighted threshold graph matrix can be computed in time $O(n)$.*

Proof. Every step of the recurrence takes a constant number of arithmetic operations. □

For arbitrary matrices, the tasks of computing matrix products, matrix inverses, and determinants are all equivalent [2, Chap.16], currently $O(n^{2.376})$ [4]. For weighted threshold graph matrices, they all seem to be different. We have just seen that the determinant can be computed in linear time, which is optimal, as this time is already needed to read the input. The same lower bound holds for computing the characteristic polynomial, and we will show an $O(n \log^2 n)$ algorithm. It is not hard to see that the multiplication of weighted threshold graph matrices can be done in quadratic time. This is again optimal, because the product is no longer a threshold graph matrix, and its output requires quadratic time.

3 Computation of the Characteristic Polynomial of a Threshold Graph

The adjacency matrix A of the n-vertex threshold graph G defined by the sequence $(b_1, \ldots, b_{n-1})$ is the matrix $M^{0\,0\ldots0}_{b_1 b_2 \ldots b_{n-1}}$, and the characteristic polynomial of this threshold graph is

$$\chi(G, \lambda) = \det(\lambda I - A) = \det\left(M^{\lambda\,\lambda\ldots\lambda}_{-b_1 - b_2 \cdots - b_{n-1}}\right).$$

This immediately implies that any value of the characteristic polynomial can be computed in linear time.

The characteristic polynomial itself can be computed by the recurrence equation of Theorem 1. Here all $d_i = \lambda$, and D_n, as the characteristic polynomial of an n-vertex graph, obviously is a polynomial of degree n in λ. Now, the computation of D_n from D_{n-1} and D_{n-2} according to the recurrence equation is a multiplication of polynomials. It takes time $O(n)$, as one factor is always of constant degree. The resulting total time is quadratic. The same quadratic time is achieved, when we compute the characteristic polynomial $\chi(G, \lambda)$ for n different values of λ and interpolate to obtain the polynomial $\chi(G, \lambda)$.

We want to do better. Therefore, we write the recurrence equation of Theorem 1 in matrix form.

$$\begin{pmatrix} D_n \\ D_{n-1} \end{pmatrix} = \begin{pmatrix} d_n + d_{n-1} - 2b_{n-1} & -(b_{n-1} - d_{n-1})^2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} D_{n-1} \\ D_{n-2} \end{pmatrix}$$

Noticing that $D_0 = 1$ and $D_1 = \lambda$, and all $d_i = \lambda$, we obtain the following matrix recurrence immediately.

Theorem 2. *For*

$$B_i = \begin{pmatrix} 2(\lambda - b_i) & -(b_i - \lambda)^2 \\ 1 & 0 \end{pmatrix} \text{ for } i = 1, \ldots, n-1,$$

we have

$$\begin{pmatrix} D_n \\ D_{n-1} \end{pmatrix} = B_{n-1} B_{n-2} \cdots B_1 \begin{pmatrix} \lambda \\ 1 \end{pmatrix}$$

□

This results in a much faster way to compute the characteristic polynomial $\chi(G, \lambda)$.

Corollary 2. *The characteristic polynomial $\chi(G, \lambda)$ of a threshold graph G with n vertices can be computed in time $O(n \log^2 n)$.*

Proof. For every i, all the entries in the 2×2 matrix B_i are polynomials in λ of degree at most 2. Therefore, products of any k such factors have entries which are polynomials of degree at most $2k$. To be more precise, actually the degree bound is k, because by induction on k, one can easily see that the degree of the i, j-entry of any such k-fold product matrix is at most

$$\begin{aligned} k &\text{ for } i = 1 \text{ and } j = 1, \\ k+1 &\text{ for } i = 1 \text{ and } j = 2, \\ k-1 &\text{ for } i = 2 \text{ and } j = 1, \\ k &\text{ for } i = 2 \text{ and } j = 2, \end{aligned}$$

But the bound of $2k$ would actually be sufficient for our purposes. W.l.o.g., we may assume that $n - 1$ (the number of factors) is a power of 2. Otherwise, we could fill up with unit matrices. Now the product $B_{n-1} B_{n-2} \cdots B_1$ is computed in $\log(n-1)$ rounds of pairwise multiplication to reduce the number of factors by half each round. We use the FFT (Fast Fourier Transform) to compute the product of two polynomials of degree n in time $O(n \log n)$ [1].

In the rth round ($r = 1, \ldots, \log(n-1)$), we have $(n-1)2^{-r}$ pairs of matrices with entries of degree at most $2^{r-1} + 1$, requiring $O(n2^{-r})$ multiplications of polynomials of degree at most $2^{r-1} + 1$. With FFT this can be done in time $O(n2^{-r})(2^{r-1} + 1) \log(2^{r-1} + 1) = O(nr)$. Summing over all rounds $r = 1, \ldots, \log(n-1)$, results in a running time of $O(n \log^2 n)$. □

Omitting the simplification of $d_i = \lambda$ in Theorem 2, we see immediately, that also the characteristic polynomial of a weighted threshold graph matrix can be computed in the same asymptotic time of $O(n \log^2 n)$.

4 Complexity in the Bit Model

By definition, the characteristic polynomial of an n-vertex graph can be viewed as a sum of $n!$ monomials with coefficients from $\{-1, 0, 1\}$. Thus all coefficients of

the characteristic polynomial have absolute value at most $n!$, and can therefore be represented by binary numbers of length $O(n \log n)$.

The coefficients of the characteristic polynomials of some graphs can be of this order of magnitude. For an example, one can start with an $n \times n$ Hadamard matrix with only 1's in the first row. Its determinant has an absolute value of $n^{n/2}$. Adding the first row to all other rows and dividing all other rows by 2 results in a 0-1-matrix whose determinant has an absolute value of $2^{-n+1}n^{n/2}$. The bipartite graph G with this bipartite adjacency matrix has a determinant with absolute value $\left(2^{-n/2+1}(n/2)^{n/4}\right)^2 = 4(n/8)^{n/2}$. Thus the constant coefficient of the characteristic polynomial of G has length $\Omega(n \log n)$.

With such long coefficients, the usual assumption of arithmetic operations in constant time is actually unrealistic for large n. Therefore, the bit model might be more useful. We can use the Turing machine time, because our algorithm is sufficiently uniform. No Boolean circuit is known to compute such things with asymptotically fewer operations than the number of steps of a Turing machine.

We use the fast $m \log m 2^{O(\log^* m)}$ integer multiplication algorithm [6] (where m is the length of the factors) to compute the FFT for the polynomials. A direct implementation, just using fast integer multiplication everywhere during a fast polynomial multiplication, results in time

$$(nr)(2^r r^2 2^{O(\log^* r)}) = n 2^r r^3 2^{O(\log^* r)}$$

for the rth round, where $O(n2^{-r})$ pairs of polynomials of degree $O(2^r)$ are multiplied. The coefficients of these polynomials have length $O(2^r r)$. As the coefficients and the degrees of the polynomials increase at least geometrically, only the last round with $r = \log n$ counts asymptotically. The resulting time bound is $n^2 \log^3 n\, 2^{O(\log^* n)}$. Using Schönhage's [14] idea of encoding numerical polynomials into integers in order to do polynomial multiplication, a speed-up is possible. Again only the last round matters. Here a constant number of polynomials of degree $O(n)$ with coefficients of length $O(n \log n)$ are multiplied. For this purpose, each polynomial is encoded into a number of length $O(n^2 \log n)$, resulting in a computation time of

$$n^2 \log^2 n\, 2^{O(\log^* n)}.$$

Actually, because the lengths of coefficients are not smaller than the degree of the polynomials, no encoding of polynomials into numbers is required for this speed-up. In this case, one can do the polynomial multiplication in a polynomial ring over Fermat numbers as in Schönhage and Strassen [15]. Then, during the Fourier transforms all multiplications are just shifts. Fast integer multiplication is only used for the multiplication of values. This is not of theoretical importance, as it results in the same asymptotic $n^2 \log^2 n\, 2^{O(\log^* n)}$ computation time, just with a better constant factor.

5 Open Problems

We have improved the time to compute the characteristic polynomial of a threshold graph from quadratic to almost linear (in the algebraic model). The question

remains whether another factor of $\log n$ can be removed. More interesting is the question whether similarly efficient algorithms are possible for richer classes of graphs. Of particular interest are larger classes of graphs containing the threshold graphs, like cographs, graphs of clique-width 2, graphs of bounded clique-width, or even perfect graphs.

References

1. Aho, A., Hopcroft, J., Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, MA (1974)
2. Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic Complexity Theory, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 315. Springer, Berlin (1997)
3. Chvátal, V., Hammer, P.L.: Aggregation of inequalities in integer programming. In: Studies in Integer Programming (Proceedings Workshop Bonn, 1975). Annals of Discrete Mathematics, vol. 1, pp. 145–162. North-Holland, Amsterdam (1977)
4. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. J. Symb. Comput. **9**(3), 251–280 (1990)
5. Fricke, G.H., Hedetniemi, S., Jacobs, D.P., Trevisan, V.: Reducing the adjacency matrix of a tree. Electron. J. Linear Algebr. **1**, 34–43 (1996)
6. Fürer, M.: Faster integer multiplication. SIAM J. Comput. **39**(3), 979–1005 (2009)
7. Fürer, M.: Efficient computation of the characteristic polynomial of a tree and related tasks. Algorithmica **68**(3), 626–642 (2014). http://dx.doi.org/10.1007/s00453-012-9688-5
8. Henderson, P.B., Zalcstein, Y.: A graph-theoretic characterization of the pv_chunk class of synchronizing primitives. SIAM J. Comput. **6**(1), 88–108 (1977). http://dx.doi.org/10.1137/0206008
9. Jacobs, D.P., Trevisan, V., Tura, F.: Computing the characteristic polynomial of threshold graphs. J. Graph Algorithms Appl. **18**(5), 709–719 (2014)
10. Keller-Gehrig, W.: Fast algorithms for the characteristic polynomial. Theor. Comput. Sci. **36**(2,3), 309–317 (1985)
11. Mahadev, N.V.R., Peled, U.N.: Threshold Graphs and Related Topics. Annals of Discrete Mathematics. Elsevier Science Publishers B.V. (North Holland), Amsterdam-Lausanne-New York-Oxford-Shannon-Tokyo (1995)
12. Mohar, B.: Computing the characteristic polynomial of a tree. J. Math. Chem. **3**(4), 403–406 (1989)
13. Pernet, C., Storjohann, A.: Faster algorithms for the characteristic polynomial. In: Brown, C.W. (ed.) Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation, University of Waterloo, Waterloo, Ontario, Canada, 29 July–1 August 2007, pp. 307–314. ACM Press, pub-ACM:adr (2007)
14. Schönhage, A.: Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coeficients. In: Calmet, J. (ed.) EUROCAM 1982. LNCS, vol. 144, pp. 3–15. Springer, Heidelberg (1982)
15. Schönhage, A., Strassen, V.: Schnelle Multiplikation grosser Zahlen. Computing **7**, 281–292 (1971)
16. Tinhofer, G., Schreck, H.: Computing the characteristic polynomial of a tree. Computing **35**(2), 113–125 (1985)