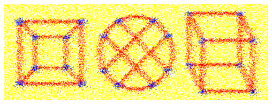
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 18, no. 5, pp. 709–719 (2014)
DOI: 10.7155/jgaa.00342

Computing the Characteristic Polynomial of Threshold Graphs

David P. Jacobs[1] *Vilmar Trevisan*[2] *Fernando Tura*[3]

[1]School of Computing, Clemson University Clemson, SC 29634 USA
[2]Instituto de Matemática, UFRGS, 91509–900 Porto Alegre, RS, Brazil
[3]Departamento de Matemática, UFSM, 97105–900 Santa Maria, RS, Brazil

Abstract

We present an algorithm for constructing the characteristic polynomial of a threshold graph's adjacency matrix. The algorithm is based on a diagonalization procedure that is easy to describe. It can be implemented using $O(n)$ space and with running time $O(n^2)$.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
December 2013	October 2014	November 2014	December 2014	December 2014

Published:
December 2014

Article type:
Regular paper

Communicated by:
Martin Fürer

E-mail addresses: dpj@clemson.edu (David P. Jacobs) trevisan@mat.ufrgs.br (Vilmar Trevisan) ftura@smail.ufsm.br (Fernando Tura)

1 Introduction

Given $G = (V, E)$, an undirected graph with vertices $V = (v_1, \ldots, v_n)$ and edge set E, its *adjacency matrix* $A(G) = [a_{ij}]$ is the $n \times n$ 0–1 matrix for which $a_{ij} = 1$ if and only if v_i is adjacent to v_j (that is, there is an edge between v_i and v_j). Let x be an indeterminate. The *characteristic polynomial* of G is usually defined as $\rho_G(x) = \det(xI_n - A)$. Some authors define it as $\det(A - xI_n)$, however these polynomials differ by the factor $(-1)^n$. Their roots are called the *eigenvalues* of G.

This paper is concerned with *threshold graphs*, introduced by Chvátal and Hammer [3] and Henderson and Zalcstein [5] in 1977. They are an important class of graphs because of their numerous applications in diverse areas which include computer science and psychology [10]. Threshold graphs can be characterized in several ways. One way of obtaining a threshold graph is through an iterative process which starts with an isolated vertex, and where, at each step, either a new *isolated* vertex is added, or a *dominating* vertex is added. We may represent a threshold graph G on n vertices using a binary sequence $(b_1, \ldots, b_n)$. Here $b_i = 0$ if vertex v_i was added as an isolated vertex, and $b_i = 1$ if v_i was added as a dominating vertex. This representation has been called a *creation sequence*. In our representation b_1 is always zero. If $n \geq 2$, G is connected if and only if $b_n = 1$. In constructing an adjacency matrix, we order the vertices in the same way they are given in their creation sequence. Figure 1 shows the adjacency matrix of the threshold graph represented by $(0, 1, 0, 1, 1)$.

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Figure 1: Adjacency matrix of threshold graph.

In [12] Sciriha and Farrugia discussed combinatorial and spectral properties of threshold graphs. Threshold graphs were studied in [1, 13, 14] under the name *nested split* graphs. In [8] the authors identified those threshold graphs having the smallest eigenvalue among all threshold graphs of order n, and in [9] showed that no threshold graph has an eigenvalue in the real interal $(-1, 0)$.

Algorithms and formulas for the characteristic polynomial of trees have been studied in [7, 6, 11] and by M. Fürer, who in [4], presented an $O(n \log^2 n)$ algorithm for computing the characteristic polynomial of a tree on n vertices.

The good performance for tree algorithms is somewhat expected since the matrices associated are sparse. For general graphs there is no known algorithm for computing its characteristic polynomial that performs better than algorithms for general matrices, which can be obtained in $O(n^3)$ [15].

Since adjacency matrices of threshold graphs can be dense, obtaining an efficient algorithm is a greater challenge. In this paper, taking advantage of

the structure of the adjacency matrix, we present efficient algorithms for constructing the characteristic polynomial $\rho_G(x)$ where G is a threshold graph. We can assume that G is connected for otherwise $\rho_G(x) = x^k \rho_{G'}(x)$, where G' is a connected component.

Our characteristic polynomial algorithm is based on the diagonalization algorithm for threshold graphs presented in [8], and these ideas are reviewed in Section 2. In Section 3 we present an algorithm for constructing the characteristic polynomial of a threshold graph that runs in time $O(n^2 \log n)$ and space $O(n)$. We illustrate our algorithm in Section 4. In Section 5 we discuss a method used in [12], based on equitable partitions, for computing the characteristic polynomial of threshold graphs. Finally, in Section 6, we sketch an interpolation approach that has time $O(n^2)$ and space $O(n)$.

2 Diagonalization

Recall that two matrices R and S are *congruent* if there exists a nonsingular matrix P such that $R = P^T S P$. An important tool used in [8] was an algorithm for constructing a *diagonal* matrix D congruent to $B_x = A + xI$, where A is the adjacency matrix of a threshold graph, and x is an arbitrary scalar. Our algorithm only adds scalar multiples of a row (column) to another row (column), i.e. performs elementary type III operations, and so $\det(P) = 1$. The algorithm is shown in Figure 2. The diagonal elements are stored in the array d, and the graph's initial representation is stored in b.

Algorithm `Diagonalize` works bottom up. For a graph of order n, it makes $n - 1$ steps. Each diagonal element, except the first and last, participates in two iterations. During each iteration, the assignment to d_m produces a final diagonal element. On the last iteration, when $m = 2$, the assignment to d_{m-1} also produces a final diagonal element at the top.

Note when $b_m = 0$, the algorithm does nothing and moves to the next step. Also note that the values in b can change. In each iteration, the algorithm executes one of the five subcases. It should be noted that **subcase 1a** and **subcase 2b** are the normal cases, and the other three subcases represent singularities. Executing **subcase 1b** requires $x = 1$, executing **subcase 2a** requires $x = 0$, and executing **subcase 1c** requires $\alpha + x = 2$. The following theorem, whose proof appears in [8], asserts the correctness of the algorithm.

Theorem 1 *For inputs G and x, where G is a threshold graph with adjacency matrix A, algorithm* `Diagonalize` *computes a diagonal matrix D, which is congruent to $A + xI$.*

3 Symbolic algorithm

The characteristic polynomial is obtained by computing $\det(xI_n - A)$, where x is an indeterminate. This can be done by diagonalizing $A - xI_n$, multiplying the diagonal elements, and then adjusting the sign. Algorithm `Diagonalize`

```
Algorithm Diagonalize(G, x)
   initialize d_i ← x, for all i
   for  m = n to 2
      α ← d_m
      if b_{m-1} = 1 and b_m = 1
         if α + x ≠ 2                    //subcase 1a
            d_{m-1} ← (αx − 1)/(α + x − 2)
            d_m ← α + x − 2
         else if x = 1                   //subcase 1b
            d_{m-1} ← 1
            d_m ← 0
         else                            //subcase 1c
            d_{m-1} ← 1
            d_m ← −(1 − x)^2
            b_{m-1} ← 0
      else if b_{m-1} = 0 and b_m = 1
         if x = 0                        //subcase 2a
            d_{m-1} ← 1
            d_m ← −1
         else                            //subcase 2b
            d_{m-1} ← α − 1/x
            d_m ← x
            b_{m-1} ← 1
   end loop
```

Figure 2: Diagonalizing $A(G) + xI$.

will diagonalize $A + xI$ over any field F, where $x \in F$. Thus we may use `Diagonalize`$(G, -x)$ to compute $\det(A - xI_n)$. However we are no longer working over $\mathbb{R}$, but rather the quotient field $\mathbb{R}(x)$ of $\mathbb{R}[x]$. Over this field, the algorithm can be simplified.

Lemma 1 *During the execution of* `Diagonalize`$(G, -x)$*, it is impossible for the algorithm to enter* **subcase 1b** *or* **subcase 2a**.

Proof: Since x is an indeterminate, $x \neq 0, 1$.

During the execution of `Diagonalize`$(G, -x)$ there is a sequence of n elements of $\mathbb{R}(x)$, calculated right to left

$$\alpha_{n-1}(x), \ldots, \alpha_1(x), \alpha_0(x) = -x$$

that are temporarily assigned to the diagonal. We call it the *α-sequence*. Except for the final value α_{n-1}, each gets overwritten. When the algorithm executes **subcase 1a**, we have:

$$\alpha_{i+1}(x) = \frac{-x\alpha_i(x) - 1}{\alpha_i(x) - x - 2} \tag{1}$$

When the algorithm executes **subcase 2b**,

$$\alpha_{i+1}(x) = \alpha_i(x) + \frac{1}{x} \tag{2}$$

Lemma 2 *Assume* `Diagonalize`$(G, -x)$ *executes only* **subcase 1a** *and* **subcase 2b** *for its first j steps. Then for each $i \leq j$, $\alpha_i(x) = \frac{p(x)}{q(x)}$, where $p(x)$ and $q(x)$ are polynomials such that*

1. $\deg(p) = \deg(q) + 1$;
2. *$p(x)$ has a negative leading coefficient;*
3. *$q(x)$ has a positive leading coefficient.*

Proof: By induction on i. When $i = 0$, we have $\alpha_0(x) = \frac{-x}{1}$. So assume α_i can be written as $\frac{p(x)}{q(x)}$ having these three properties, and consider α_{i+1}. There are two cases, according to whether **subcase 1a** or **subcase 2b** occured in the last iteration.

subcase 1a Then (1) holds, and we have

$$\alpha_{i+1}(x) = \frac{-x\frac{p(x)}{q(x)} - 1}{\frac{p(x)}{q(x)} - x - 2} = \frac{xp(x) + q(x)}{xq(x) - p(x) + 2q(x)}$$

Using the induction assumption, we see that the degree and leading coefficient properties are preserved.

subcase 2b Then (2) holds, and we have $\alpha_{i+1}(x) = \alpha_i(x) + \frac{1}{x} = \frac{xp(x)+q(x)}{xq(x)}$. Using the induction assumption, we see that the degree and leading coefficient properties are also preserved.

This completes the proof.

Lemma 3 *During the execution of* `Diagonalize`$(G, -x)$*, it is impossible for the algorithm to enter* **subcase 1c**.

Proof: Suppose, by contradiction, the algorithm enters **subcase 1c**. Then it must be the case, that for some i, $\alpha_i(x) = 2 + x$. Assume this is the first such i. By Lemma 2

$$\lim_{x\to\infty} \alpha_i(x) = \lim_{x\to\infty} \frac{p(x)}{q(x)} = -\infty.$$

But

$$\lim_{x\to\infty} 2 + x = \infty.$$

This is a contradiction.

It follows from Lemma 1 and Lemma 3 that `Diagonalize`$(G, -x)$ can only execute **subcase 1a** or **subcase 2b**. The simplified algorithm is shown in Figure 3, and there is no modification of the b_i required. After diagonalization, the determinant is computed by multiplying the diagonal terms and adjusting the sign. We have:

```
Algorithm CharPoly(G)
   initialize d_i ← −x, for all i
   for  m = n to 2
      α(x) ← d_m
      if b_{m−1} = 1
            d_{m−1} ← −(x·α(x)+1)/(α(x)−x−2)
            d_m ← α(x) − x − 2
      else if b_{m−1} = 0
            d_{m−1} ← α(x) + 1/x
            d_m ← −x
      end if
   end loop
   ρ(x) = (−1)^n ∏_{i=1}^{n} d_i
```

Figure 3: Characteristic Polynomial of Threshold Graph G.

Theorem 2 *Algorithm* `CharPoly` *computes the characteristic polynomial of a threshold graph* G.

Algorithm `CharPoly` makes $O(n)$ polynomial arithmetic operations. As the proof of Lemma 2 shows, those operations inside the loop are simple and each can be done in linear time. The final product involves multiplying n rational functions. Using fast polynomial arithmetic, each multiplication will take $O(n \log n)$, so the running time is $O(n^2 \log n)$. By constructing the partial product inside the loop, only $O(n)$ space is required.

4 Examples

Example 1 We apply Algorithm `CharPoly` to the threshold graph represented by $(0, 0, 1, 1)$. After initialization, there will be three steps. Since $b_3 = 1$ and $\alpha(x) = -x$, in the first step the assignments

$$\begin{aligned} d_3 &\leftarrow -\frac{x(-x)+1}{-2x-2} \\ d_4 &\leftarrow -2x-2. \end{aligned}$$

are made. Since $b_2 = 0$ and $\alpha(x) = -\frac{x(-x)+1}{-2x-2}$, in the second step we have:

$$\begin{aligned} d_2 &\leftarrow -\frac{x(-x)+1}{-2x-2} + \frac{1}{x} \\ d_3 &\leftarrow -x. \end{aligned}$$

Since $b_1 = 0$, and $\alpha(x) = -\frac{x(-x)+1}{-2x-2} + \frac{1}{x}$, in the final step we have:

$$\begin{aligned} d_1 &\leftarrow -\frac{x(-x)+1}{-2x-2} + \frac{1}{x} + \frac{1}{x} \\ d_2 &\leftarrow -x. \end{aligned}$$

The following table shows the steps of algorithm

b_i	d_i
0	$-x$
0	$-x$
1	$-x$
1	$-x$

initial

b_i	d_i
0	$-x$
0	$-x$
1	$-\frac{x(-x)+1}{-2x-2}$
1	$-2x-2$

step 1

b_i	d_i
0	$-x$
0	$-\frac{x(-x)+1}{-2x-2}+\frac{1}{x}$
1	$-x$
1	$-2x-2$

step 2

b_i	d_i
0	$-\frac{x(-x)+1}{-2x-2}+\frac{2}{x}$
0	$-x$
1	$-x$
1	$-2x-2$

step 3

The characteristic polynomial is $(-\frac{x(-x)+1}{-2x-2}+\frac{2}{x})(-x)(-x)(-2x-2)$ or

$$x^4 - 5x^2 - 4x.$$

Example 2 We implemented our algorithm in Maple, and computed the characteristic polynomial of the threshold graph of order 16 having creation sequence $(0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1)$, depicted in Figure 4. In the figure, the vertices $v_1,\ldots,v_{16}$ are arranged counterclockwise with v_1 at the top. The algorithm quickly computed

$$x^{16} - 64\,x^{14} - 280\,x^{13} - 252\,x^{12} + 784\,x^{11} + 1708\,x^{10} + 156\,x^9 - 1930\,x^8$$
$$-832\,x^7 + 992\,x^6 + 408\,x^5 - 336\,x^4 - 40\,x^3 + 62\,x^2 - 14\,x + 1.$$

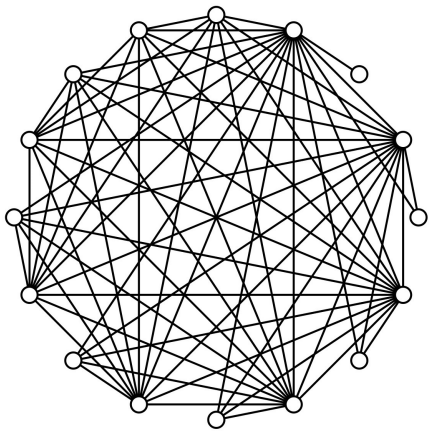

Figure 4: Threshold graph $(0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1)$

5 Equitable partition approach

We mention that there is an alternate approach for computing the characteristic polynomial of a threshold graph G, that uses a reduction procedure, and takes into account the multiplicity of the eigenvalues 0 and -1. We refer the reader to [2, 9, 12, 13, 14] for details, and summarize the results important to our discussion. Let G be a threshold graph with creation sequence

$$\mathbf{b} = (b_1, b_2, \ldots, b_n) = 0^{s_1}1^{t_1}\ldots 0^{s_k}1^{t_k},$$

where $b_1 = 0$ and $b_n = 1$. Then

1. the multiplicity of 0 is the number of substrings 00 in $\mathbf{b}$.
2. the multiplicity of -1 is given by:

$$\begin{cases} \sum_{i=1}^{k}(t_i - 1) & \text{if } s_1 > 1 \\ 1 + \sum_{i=1}^{k}(t_i - 1) & \text{if } s_1 = 1. \end{cases}$$

After removing the terms x and $(x + 1)$, we construct an equitable partition matrix whose characteristic polynomial is the remaining factor.

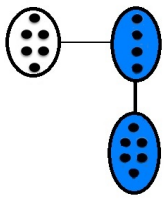

Figure 5: Threshold graph $(0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1)$

As an example, consider the threshold graph G of order 16 represented by $(0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1) = 01^50^61^4$ and depicted in Figure 5, where the blue cells represent cliques and the white cell represents the independent set. Using the formulas above, we see that the multiplicity of zero is 5, and the multiplicity of -1 is 8. We form the equitable partition matrix

$$\begin{bmatrix} 3 & 6 & 6 \\ 4 & 5 & 0 \\ 4 & 0 & 0 \end{bmatrix}.$$

Computing the characteristic polynomial of this smaller matrix, we see that the characteristic polynomial of G is

$$x^5(x + 1)^8(x^3 - 8x^2 - 33x + 120).$$

This method is useful when the size of the equitable partition matrix is relatively small, that is, when the graph has high multiplicities of 0 and -1.

However, in this case, because the equitable partition matrix does not have a special structure, general tools must be used to find its characteristic polynomial. This reduction does not seem useful for graphs having low multiplicity of 0 and -1. For example, in the graph of Figure 4 the reduction would produce an equitable partition matrix that was 15×15.

The method described in this paper is an efficient algorithm that can compute the characteristic polynomial of any threshold graph.

6 Interpolation algorithm

It is theoretically possible to compute the characteristic polynomial in time $O(n^2)$. We sketch the approach. On input G, we generate $n+1$ arbitrary distinct values $x_j \in \mathbb{R}$, $j = 0, \ldots, n$. For each x_j, we call `Diagonalize`(G, x_j) to obtain a diagonal d. Computing $y_j = (-1)^n \prod_{i=1}^{n} d_i$ gives a point (x_j, y_j) on the graph. We then apply interpolation to the points $\{(x_0, y_0), \ldots, (x_n, y_n)\}$. It is easy to see that each pair (x_i, y_i) takes $O(n)$ to construct and so it will take $O(n^2)$ to generate the set. Finally, it is well-known that interpolation can be done in time $O(n^2)$ and space $O(n)$.

Acknowledgements

This work is supported by Science without Borders, under CNPq grant 400122/2014-6. Vilmar Trevisan is partially supported by CNPq (Proc. 305583/2012-3 and 481551/2012-3) and by CAPES grant PROBRAL 408/13. Fernando Tura acknowledges the support of CAPES Grant 0283/12-6.

References

[1] M. Andelić, C. M. da Fonseca, S. K. Simić, and D. V. Tošić. Some further bounds for the Q-index of nested split graphs. *J. Math. Sci.*, 182(2):193–199, 2012. Translated from Sovrem. Mat. Prilozh., Vol. 71, 2011. doi:10.1007/s10958-012-0739-x.

[2] R. B. Bapat. On the adjacency matrix of a threshold graph. *Linear Algebra Appl.*, 439(10):3008–3015, 2013. doi:10.1016/j.laa.2013.08.007.

[3] V. Chvátal and P. L. Hammer. Aggregation of inequalities in integer programming. In *Studies in integer programming (Proc. Workshop, Bonn, 1975)*, pages 145–162. Ann. of Discrete Math., Vol. 1.

[4] M. Fürer. Efficient computation of the characteristic polynomial of a tree and related tasks. *Algorithmica*, 68(3):626–642, 2014. doi:10.1007/s00453-012-9688-5.

[5] P. B. Henderson and Y. Zalcstein. A graph-theoretic characterization of the $\mathrm{PV}_{\mathrm{chunk}}$ class of synchronizing primitives. *SIAM J. Comput.*, 6(1):88–108, 1977. doi:10.1137/0206008.

[6] D. P. Jacobs, C. M. S. Machado, and V. Trevisan. An $O(n^2)$ algorithm for the characteristic polynomial of a tree. *J. Combin. Math. Combin. Comput.*, 54:213–221, 2005.

[7] D. P. Jacobs and V. Trevisan. Constructing the characteristic polynomial of a tree's adjacency matrix. In *Proceedings of the Twenty-ninth Southeastern International Conference on Combinatorics, Graph Theory and Computing (Boca Raton, 1998)*, volume 134, pages 139–145, 1998.

[8] D. P. Jacobs, V. Trevisan, and F. Tura. Eigenvalue location in threshold graphs. *Linear Algebra Appl.*, 439(10):2762–2773, 2013. doi:10.1016/j.laa.2013.07.030.

[9] D. P. Jacobs, V. Trevisan, and F. Tura. Eigenvalues and energy in threshold graphs. *Linear Algebra Appl.*, 465:412–425, 2015. doi:10.1016/j.laa.2014.09.043.

[10] N. V. R. Mahadev and U. N. Peled. *Threshold graphs and related topics*, volume 56 of *Annals of Discrete Mathematics*. North-Holland Publishing Co., Amsterdam, 1995.

[11] M. Robbiano and V. Trevisan. Applications of recurrence relations for the characteristic polynomials of Bethe trees. *Comput. Math. Appl.*, 59(9):3039–3044, 2010. doi:10.1016/j.camwa.2010.02.023.

[12] I. Sciriha and S. Farrugia. On the spectrum of threshold graphs. *ISRN Discrete Mathematics*, 2011. doi:10.5402/2011/108509.

[13] S. K. Simić, F. Belardo, E. M. Li Marzi, and D. V. Tošić. Connected graphs of fixed order and size with maximal index: some spectral bounds. *Linear Algebra Appl.*, 432(9):2361–2372, 2010. doi:10.1016/j.laa.2009.06.043.

[14] Z. Stanić. On nested split graphs whose second largest eigenvalue is less than 1. *Linear Algebra Appl.*, 430(8-9):2200–2211, 2009. doi:10.1016/j.laa.2008.11.026.

[15] J. H. Wilkinson. *The algebraic eigenvalue problem.* Monographs on Numerical Analysis. The Clarendon Press, Oxford University Press, New York, 1988. Oxford Science Publications.