An $O(n^2)$ Algorithm for the Characteristic Polynomial of a Tree

David P. Jacobs
Dept. of Computer Science, Clemson University
Clemson, SC 29634-0974 USA dpj@cs.clemson.edu

Catia M. S. Machado
FURG - Departamento de Matemática
96201-900 Rio Grande, RS, Brasil mmacmsm@super.furg.br

Vilmar Trevisan
UFRGS–Instituto de Matemática
91509–900 Porto Alegre, RS, Brasil trevisan@mat.ufrgs.br

April 11, 2005

Abstract

We describe an algorithm, that uses $O(n)$ arithmetic operations, for computing the determinant of the matrix $M = (A + \alpha I)$, where A is the adjacency matrix of an order n tree. Combining this algorithm with interpolation, we derive a simple algorithm requiring $O(n^2)$ arithmetic operations, to find the characteristic polynomial of the adjacency matrix of any tree. We apply our algorithm and recompute a 22-degree characteristic polynomial, which had been incorrectly reported in the quantum chemistry literature.
keywords: tree, adjacency matrix, characteristic polynomial.
AMS subject classifcation: 05C05, 05C50, 05C85, 15A15 .

1 Introduction

Recall that the *characteristic polynomial* of an $n \times n$ matrix M is the monic degree-n polynomial

$$p(\lambda) = \det(M - \lambda I_n), \tag{1}$$

and its roots in $\mathbf{C}$ are called the *eigenvalues* of M. Even when restricted to $0-1$ matrices, eigenvalues and eigenvectors can have surprising applications. Recently Wilf observed in [15] that determining the importance of web pages, something done by search engines, can be viewed as an eigenvector problem, and is related to early papers on ranking [9, 14].

Let $G = (V, E)$ be an undirected graph with vertices $V = (v_1, \ldots, v_n)$ and edge set E. We assume edges occur only between pairs of distinct vertices, and between any pair of vertices there is at most one edge. The *adjacency matrix* $A = [a_{ij}]$ of G 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). Real symmetric matrices, such as a adjacency matrices, are Hermitian and known to have all real eigenvalues [10].

In chemistry, the characteristic polynomial is called the secular polynomial, and has been used in Hückel theory [13] and quantum chemistry [11]. When graphs are used to represent molecules, the characteristic polynomial of the graph is related to certain thermodynamic properties of the molecule [2]. Two different molecules having the same characteristic polynomial will have similar thermodynamic properties.

There are several methods to compute the characteristic polynomial $p(\lambda)$. We wish to obtain the exact integer coefficients of $p(\lambda)$, and not their approximation using a numerical procedure. Procedures returning exact values are called *algebraic* or *symbolic*. Wilkinson ([16], p. 411) presents one such algorithm that computes the characteristic polynomial of any $n \times n$ matrix using only $O(n^3)$ scalar multiplications.

In this paper we consider the problem for the class of adjacency matrices of trees (connected, acyclic graphs). The characteristic polynomial of a tree T, denoted $p_T(\lambda)$, is the characteristic polynomial of its adjacency matrix.

One reduction method [3, 6] for computing $p_T(\lambda)$ works as follows. Let x_1 be a leaf of T, and x_2 its neighbor. Let T' and T'' denote the induced graphs obtained by deleting $\{x_1\}$ and $\{x_1, x_2\}$, respectively. Then

$$p_T = \lambda \cdot p_{T'} - p_{T''}.$$

Although simple, this method takes exponential time since a problem of size n is being replaced by problems of size $n-1$ and $n-2$.

In [2], Balasubramanian outlines a reduction method for computing $p_T(\lambda)$, based on ideas of Godsil and McKay [5] and Schwenk [12]. The method appears correct, but no algorithmic analysis is given. In [4, 7] matrix methods were described that operated directly on the tree. The procedure was refined in [8], using techniques to simplify the polynomial arithmetic. The resulting algorithm requires $O(n^2 \log(n))$ operations to compute the characteristic polynomial of an n-vertex tree. The purpose of this note is to describe an algorithm that computes the characteristic polynomial of a tree's adjacency matrix in $O(n^2)$ operations.

2 Computing the Determinant

Given an n-vertex tree T, the characteristic polynomial of its adjacency matrix A is a degree n polynomial

$$p(\lambda) = a_0 + a_1\lambda + \cdots + \lambda^n. \tag{2}$$

Our algorithm obtains the coefficients a_i by first computing $\det(A + I\alpha_i) = p(-\alpha_i)$ for sufficiently many α_i, and then interpolating the points $(-\alpha_i, p(-\alpha_i))$. In this section we will explain how to compute $\det(A + I\alpha_i)$ in $O(n)$ scalar operations.

Given a matrix M, a method sometimes used for obtaining its determinant is to apply Gaussian eliminations (i.e. operations in which a multiple of a row is added to another row beneath it), hoping to obtain an upper triangular matrix U. Columns are processed left-to-right, and nonzero entries below the diagonal are eliminated. As each Gaussian elimination keeps the determinant invariant, the final upper triangular matrix satisfies $\det U = \det M$.

For T a tree and A its adjacency matrix, if we use the Gaussian transformation approach described above to compute

$$\det M = \det(A + \alpha I),$$

the ordering of the vertices will effect the amount of fill-in created by the row operations. We can avoid fill-in by first selecting a root of the tree and labeling it v_n. We then order the vertices $V = (v_1, \ldots, v_n)$ so that if v_j is a parent of v_i then $i < j$. The vertex order is crucial in preventing fill-in. Because of our vertex order, row operations always represent a child's row acting on a parent's row. Beside the entry being eliminated, the only other entry affected is the diagonal entry of the parent. Figure 1 illustrates a subtree and its corresponding sub-matrix.

Since our Gaussian transformations do not produce additional fill-in, it is not necessary to store all of U, but only its diagonal entries m_{jj}, which are transformed according to the rule $m_{jj} \leftarrow m_{jj} - \sum \frac{1}{m_{ii}}$, where the sum ranges over the children v_i of v_j. As long as none of the m_{ii} are zero, the algorithm works and, in fact, can be performed directly on the tree, each node storing its diagonal value.

As an example, consider the tree of Figure 2 and its neighborhood matrix $M = A + I$. Initially all the nodes (diagonal entries) are assigned the

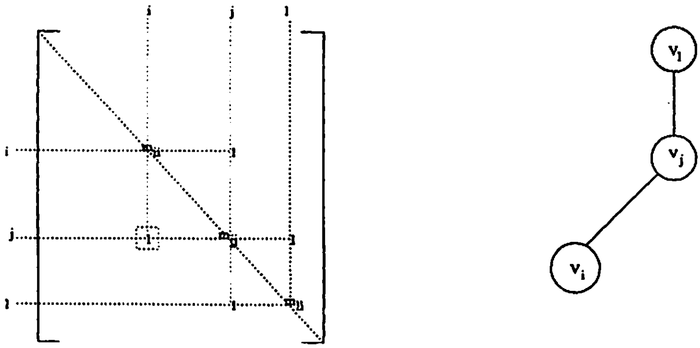

Figure 1: Sub-matrix and corresponding subtree

value $\alpha = 1$. We process the vertices bottom-up, the leaves remaining unchanged. Processing a vertex v means that the 1's, below the main diagonal, representing the edges between v and its children, are being eliminated by the diagonal elements of the children. The resulting diagonal elements are shown in Figure 2. The value of $\det M$ is the product of the node values, which is -1 in the example.

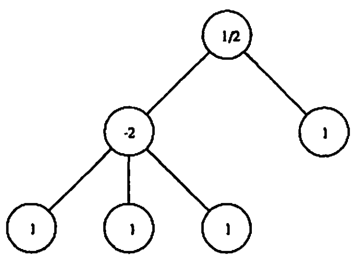

Figure 2: Diagonal of the upper-triangularized $A + I$

Consider now the case when exactly one of the children has a zero value in the diagonal. This corresponds to the situation depicted in the matrix of Figure 3, where the vertex v_j has a child v_i with $m_{ii} = 0$. The following elementary row operations are performed. Row i is replaced by the sum of row i and j. Row j now is replaced by row j minus row i. That is,

$$\begin{aligned} \text{row}(i) &\longleftarrow \text{row}(i) + \text{row}(j) \\ \text{row}(j) &\longleftarrow \text{row}(j) - \text{row}(i) \end{aligned}$$

We note that all the elements of row j are now zero except for $m_{jj} = -1$,

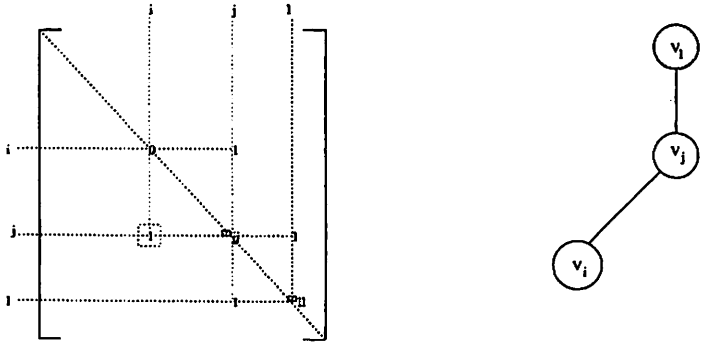

Figure 3: Vertex v_j and child v_i with $m_{ii} = 0$.

and this value may be used to annihilate the m_{lj}, if the vertex v_j has a parent v_l. This operation does alter m_{ll}.

In the ith-row we observe that $a_{ii} = 1$. It is important to notice that some m_{ik}, for $k < i$, might have the value 1 after the row operations. However, in this case, v_k and v_i are siblings and we use m_{kk} to annihilate m_{ik} without altering $m_{ii} = 1$, since $m_{ki} = 0$.

We illustrate the final appearance of the matrix in Figure 4. The actual algorithm does not execute these operations entirely. Rather, it merely records the resulting diagonal values for the parent (-1) and child (1). Since $m_{jl} = 0$, v_j does not alter the diagonal of its parent v_l. We represent this by the deletion of the edge between v_l and v_j. This causes v_l to be treated as leaf when it is processed later.

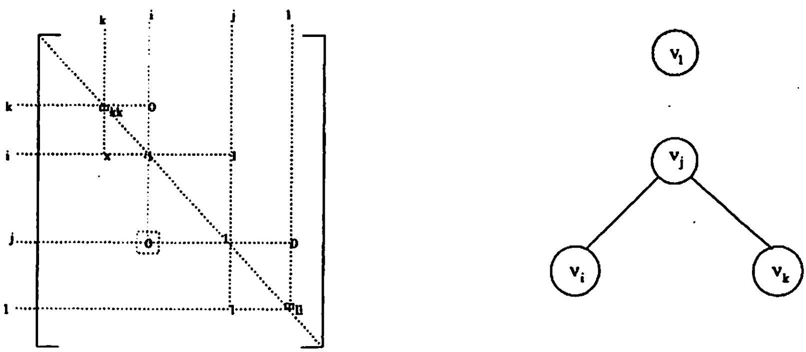

Figure 4: Sub-matrix after the row operations.

If two or more children v_i and $v_{i'}$ have value 0, then it is easy to see that both rows i and i' will be equal and, therefore, the determinant is zero. Figure 5 describes the algorithm performed directly on the tree. Since processing a vertex takes a fixed number of scalar operations and each vertex is processed once, we have

Theorem 1 *The algorithm of Figure 5 computes* $\det(A + \alpha I)$ *in* $O(n)$ *scalar operations.*

The algorithm assumes that the input is a rooted tree with nodes ordered bottom-up, and where each node contains pointers to both its parent and children.

> Initialize $a(v) := \alpha$ to each vertex v.
> **process** the vertices bottom-up as follows:
>   **if** v is a leaf **then**
>   do nothing.
>   **else if** v has more than one child with value 0, **then**
>   **return** 0
>   **else if** v has exactly one child w with value 0 **then**
> $a(v) := -1$
> $a(w) := 1$
>   if v has a parent z, then remove the edge vz
>   **else**
> $a(v) := \alpha - \sum \frac{1}{a(c_i)}$, summing over all children.
> **end loop**
> **return** $\prod a(v)$

Figure 5: Algorithm to compute $\det(A + \alpha I)$ for tree T.

As an example, consider the tree shown on the left of Figure 6. We wish to compute the determinant of $M = A + I$. The tree on the left shows the initialization of the algorithm, that is, all the diagonal elements (vertices) have the value $\alpha = 1$. The second graph of Figure 6 shows the forest that results from executing the algorithm. Note that the determinant, the product of all values, is zero.

We note that an algorithm to compute $\det(A+\alpha I)$ in $O(n)$ was obtained in [4]. The new algorithm we suggest is simpler, and its correctness is easier to prove.

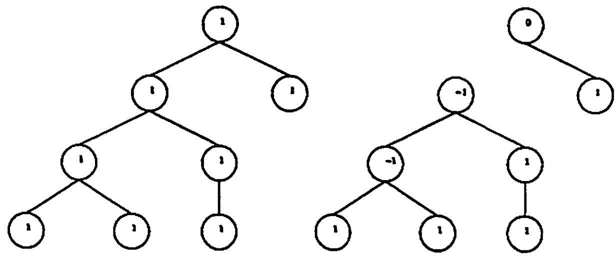

Figure 6: Tree with 8 vertices

3 Computing the Characteristic Polynomial

Our algorithm to compute the characteristic polynomial of a tree may now be summarized as

a) Choose $n+1$ distinct scalars $\alpha_0, \ldots \alpha_n$.

b) Compute $\det(A + I\alpha_i) = p(-\alpha_i)$ for $i = 0, \ldots, n$, using method above.

c) Interpolate the points $(-\alpha_i, p(-\alpha_i))$.

Theorem 2 *The characteristic polynomial of a tree can be computed in $O(n^2)$ scalar operations.*

Proof: Since we need $n+1$ values of α, the total cost for the evaluation is $O(n^2)$. As interpolation of the $n+1$ points $(\alpha, p(\alpha))$ can be done in $O(n \log^2 n)$ (see [1], p. 299), the total cost of the algorithm is $O(n^2)$. □

It is known that for trees T, $p_T(\lambda)$ is of the form $\lambda^k s(\lambda)$, where $s(\lambda)$ is a symmetric polynomial. Therefore, the polynomial $p(-x)$ is either identical to either $p(x)$ or $-p(x)$. So in practice, one needs to evaluate the polynomial for only about $\frac{n}{2}$ α's. From a practical standpoint, for most initial choices of α, nodes will have nonzero children, and so one is only computing the expression $\alpha - \sum \frac{1}{a(c_i)}$. If a computation produces a node with a value zero, then one can simply try a different α.

4 Application

We used our algorithm to compute the characteristic polynomial p of the 22-vertex tree shown in Figure 7, a polynomial considered in the chemistry literature [2], but apparently computed incorrectly. Using Maple, we wrote a function that computes $p(\alpha)$, for an arbitrary α according to the algorithm

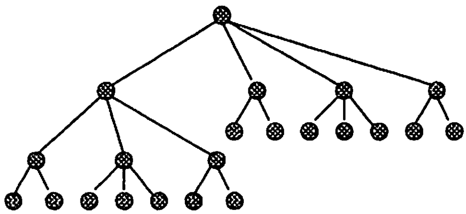

Figure 7: A 22-vertex tree.

in Figure 5, and generated twenty-three pairs $(\alpha, p(\alpha))$ that are shown in Figure 8. Finally, we used Maple's `ratinterp` function to perform the interpolation, obtaining $p(\lambda) =$

$$\lambda^{22} - 21\lambda^{20} + 174\lambda^{18} - 737\lambda^{16} + 1708\lambda^{14} - 2104\lambda^{12} + 1168\lambda^{10} - 144\lambda^{8}.$$

Our entire computation took less than a second. One can check that if eq. (2.12) in [2] been correctly calculated, it would have agreed with our polynomial above.

α	$p(\alpha)$	α	$p(\alpha)$	α	$p(\alpha)$
± 2	12288	± 3	$3^{12}\ 1813$	± 4	$2^{20}\ 3546375$
$\pm\sqrt{3}$	243	$\pm\frac{1}{2}$	$\frac{7^2\, 13533}{2^{22}}$	$\pm\frac{1}{3}$	$-\frac{17^2\, 628307}{3^{22}}$
$\pm\frac{5}{2}$	$-\frac{5^8\, 17^2\, 135339}{2^{22}}$	$\pm\frac{3}{5}$	$\frac{3^{10}\, 41^2\, 29094861}{5^{22}}$	0	0
± 1	45	$\pm\sqrt{2}$	0	$\pm\frac{3}{2}$	$\frac{3^{10}\, 285}{2^{22}}$

Figure 8: Interpolation pairs.

Finally, we note that while our characteristic polynomial algorithm requires only $O(n^2)$ arithmetic operations, its bit-complexity may be greater because of the growth of its operands. Such an analysis would be worth further study.

References

[1] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] K. Balasubramanian, Spectra of chemical trees, International Journal of Quantum Chemistry 21 (1982), 581–590.

[3] D. Cvetković, M. Doob, and H. Sachs, *Spectra in Graphs*, Academic Press, New York, 1980.

[4] G.H. Fricke, S.T. Hedetniemi, D.P. Jacobs, and V. Trevisan, Reducing the adjacency matrix of a tree, Electronic Journal of Linear Algebra 1 (1996), 34–44.

[5] C.D. Godsil and B.D. McKay, A new graph product and its spectrum, Bulletin of the Australian Mathematical Society, 18 (1978) 21-28.

[6] F. Harary, C. King, A. Mowshowitz, and R. Read, Cospectral graphs and digraphs, Bull. London Math. Soc. 3 (1971) 321–328.

[7] D.P. Jacobs and V. Trevisan, The determinant of a tree's neighborhood matrix, Linear Algebra and Its Applications 256 (1997), 235-249.

[8] D.P. Jacobs and V. Trevisan, Constructing the characteristic polynomial of a tree's adjacency matrix, Congressus Numerantium 139 (1998), 139-145.

[9] M.G. Kendall, Further contributions to the theory of paired contributions, Biometrics 11 (1955) 43.

[10] M. Marcus and H. Minc, *Introduction to Linear Algebra*, Macmillan, New York, 1965.

[11] D.H. Rouvray, The topological matrix in quantum chemistry, in *Chemical Applications of Graph Theory*, A.T. Balaban, ed. Academic Press, New York, 1976.

[12] A.J. Schwenk, Computing the characteristic polynomial of a graph, Proc. Capital Conf. on Graph Theory and Combinatorics, Lecture Notes in Math., A. Dold and B. Eckmann eds., Springer-Verlag, New York, 406 (1974) 153-172.

[13] N. Trinajstić, Graph theory and molecular orbitals, in *Chemical Graph Theory*, D. Bonchev and D.H. Rouvray, eds. Abacus Press/Gordon & Breach, New York, 1991.

[14] T.H. Wei, The algebraic foundations of ranking theory, Cambridge University Press, London, 1952.

[15] H. Wilf, Searching the web with eigenvectors, manuscript.

[16] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.