communications in mathematical
match
and in computer chemistry

no. 40, October 1999

ISSN 0340-6253

MATCDY (40) 273-278 (1999)

Novel Approaches to Exact Coefficients of Acyclic Polynomials

Gordon G. Cash[1] and William C. Herndon[2]

[1] U. S. Environmental Protection Agency, Office of Pollution Prevention and Toxics, Risk Assessment Division (7403), 401 M Street, S. W., Washington, DC 20460
[2] Department of Chemistry, University of Texas at El Paso, El Paso, TX 79968

Abstract

Published methods for computing coefficients for acyclic (matching) polynomials of chemical graphs suffer from various shortcomings, namely, they are too slow for use with many large chemical graphs of theoretical interest, they produce only approximate coefficients of unspecified accuracy, they require computer hardware resources not widely available, or authors have chosen to make only results, not the methods themselves, available. Here, several new methods are presented, programmed in Mathematica®, all of which produce exact coefficients for acyclic polynomials on readily available computers. These methods also illustrate various algebraic approaches to obtaining the polynomial which may prove useful in other applications.

1 Introduction

The principal use of the acyclic or matching polynomial of a chemical graph has been its definition as the secular (characteristic) polynomial for the hypothetical reference structure used in calculating topological resonance energies of unsaturated hydrocarbon pi electron species [1-3]. Several applications to fullerene carbon clusters have appeared in the recent theoretical chemistry literature [4-11]. However, a lack of convenient and efficient computation procedures hampers considering many very large polycyclic molecular π systems of theoretical interest. The same circumstances may also account for slow development of additional practical applications for the matching polynomial, e.g., the use of the coefficient terms of the polynomial as topological indices in structure/property and structure/activity studies [12-14].

In general, obtaining the matching polynomial for polycyclic graphs is a combinatorial problem requiring a large number of computational steps. Herndon [15], and independently Babić and Ori [6], showed that the number of terms that must be computed and summed to

make up the polynomial can be minimized by selecting from the graph a contiguous set of edges, the removal of which renders the graph acyclic. Later, Babić, *et al.* [8] published results of a method that is apparently based on a quite different approach. This method is very fast, and graphs of up to 70 vertices are tractable. However, in these publications only results were given; the method itself was not described.

Very recently, Salvador *et al.* [16] published two additional methods, following the work of Rosenfeld and Gutman [17]. As in Babić *et al.* [8], several results are given for one of the methods, with only an outline of the actual computations. A URL is provided, however, for downloading an executable file. In this method, as with most of the other methods, precision of results is limited by the accuracy of the computer's floating-point registers. These authors do, however, claim this method to be about as fast as that outlined by Babić *et al.* Their other method consists of a Mathematica® procedure, therefore giving exact results, but requires large amounts of computer RAM to avoid extensive swapping to disk. This is surely the result of the procedure's use of Mathematica® as a symbol processor, taking many partial derivatives of equations that contain numerically undefined symbols. In our hands, some other 60-vertex graphs would not run at all, even with 64 megabytes of RAM. Finally, Balakrishnaraja and Venuvanalingam [11] devised a method based on an artificial intelligence algorithm, for which they claim considerable speed, but there appear to be no further published development of their method.

In discussing the relative merits of various computational methods, it should be borne in mind that there is no *a priori* reason for different methods to scale in the same way with graph size, and the best method for one problem may not be best for another. The procedures presented here are all written in Mathematica®, and so provide exact polynomial coefficients for graphs of all sizes. In addition, the procedures make only very limited use of Mathematica® as a symbol processor, and so make no excessive demands on computer memory. This second feature would also make it easier to translate the programs into Fortran, Pascal, or C, subject to the condition that none of these has the built-in ability of Mathematica® to retain all digits in integer calculations.

2 Methods

All computations of acyclic polynomials in this paper utilize the relationship that the acyclic polynomial of a graph, P(G), is equal to the acyclic polynomial of that graph with one edge removed, P(G-e), minus that for the same graph with that edge, its two defining vertices, and all other edges attached thereto removed, $P(G\text{-}v_1,v_2)$ [18], and also the fact that, for a graph that contains no cycles, the acyclic polynomial is identical to the characteristic polynomial. Thus, the acyclic polynomial of a polycyclic graph can be built up by adding and subtracting characteristic polynomials of various acyclic fragments. There are many published methods for computing characteristic polynomials of graphs. We have programmed the most promising of these in Mathematica® and compared their ease and speed of use. Those results will be published separately [19].

As a baseline matching polynomial method, we wrote a program that uses as input an adjacency matrix and a user-defined path of contiguous edges, the removal of all of which leaves an acyclic fragment. This program removes edge sets in a stepwise fashion that eventually treats all relevant substructures while minimizing the number of computational steps to go from one to the next, *i. e.*, by adding or subtracting only one bond where possible. All fragments are further decomposed to acyclic substructures, and the characteristic

polynomials of these are computed by a Mathematica® implementation of the Le Verrier-Faddeev-Frame method [20]. This procedure runs adequately on a mid-range PC (233MHz), but its execution time increases rapidly with graph size, no doubt because of the very large combinatorial increase. This is the same situation described by Balasubramanian [9,10]. We also wrote a version that demands a user-defined set of edges but also works if the edge set is not a contiguous path. Using coronene ($C_{24}H_{12}$) as a test case, computation time for a maximally disjoint edge set (with no vertex common to any two edges) was three times greater than that for a contiguous-path edge set.

We also tested a series of programs based on automatically removing edges with what we term the "greatest extended connectivity" instead of those comprising a pre-defined set. The concept of extended bond connectivity is rooted in that of vertex extended connectivity as used by Herndon [21,22]. The premise here is that, if one looks not just at those vertices to which a vertex is directly connected, but also those to which it is connected by two-edge paths, three-edge paths, *etc.*, one or a few vertices will be seen to be more "connected" than the others, even in highly interconnected structures like fullerenes. For technical reasons [21] one uses the augmented adjacency matrix, which is the sum of the normal adjacency matrix and the identity matrix of appropriate size, to compute vertex extended connectivity. In the computation, the augmented adjacency matrix is iteratively postmultiplied by a column vector until one element of the vector is larger than all the others, or until the count of largest numbers does not change, *i. e.*, $\mathbf{v}_{n+1} = \mathbf{A}\cdot\mathbf{v}_n$. The elements of the initial vector, $\mathbf{v}_1$, are all ones.

This concept is easily extended to edges by defining edge extended connectivity at each iteration as the sum of the extended connectivities of the two vertices that define the edge. In our implementation, iteration continues until one bond is more "connected" than all the others or until the number of edges with equal extended connectivities does not change. The edge so selected is the one removed from the structure in the next step of generating an acyclic structure.

None of our extended connectivity procedures uses the Le Verrier-Faddeev-Frame method, nor indeed computes any characteristic polynomials at all. Decomposition of the graph continues to subgraphs consisting only of isolated vertices, simple cycles, and linear chains, and the characteristic polynomials for these are looked up in a table. (The characteristic polynomial of a cycle, C_n, is equal to that of a linear chain of the same size minus that of the linear chain two units smaller, *i. e.*, $P(C_n) = P(L_n) - P(L_{n-2})$.) The characteristic polynomial of n isolated vertices is just x^n, and that of a graph consisting of several disjoint pieces is the product of the characteristic polynomials of the pieces.

It is necessary to recalculate the edge extended connectivities after the removal of each edge because removal of one edge changes the extended connectivities of all the others. In practice, we found that computing the edge extended connectivities only once and using those results throughout the computation led to a large increase in computation time, although of course the correct answer was eventually obtained. The edge extended connectivity procedure at this stage of development, however, still did not match the performance of our baseline method that utilized a pre-defined continuous path of edges.

We thought to further improve performance by adding trees to the look-up table. Characteristic polynomials for all trees with 10 or fewer vertices were published long ago [23]. Challenges in this approach were (1) how to uniquely identify the trees, (2) how to recognize trees in adjacency matrices of subgraphs consisting of several disjoint pieces while excluding branched cycles, and (3) how to do this in a computationally efficient manner. In

the actual experiments, we added to the look-up table all trees (not containing cycles) with nine or fewer vertices.

In addressing the first problem, we found that the sum of the edge extended connectivities after the fourth iteration was a unique identifier for trees of this size with only two exceptions, *viz.*, 3,3-dimethylpentane is the same as 2,4-dimethylhexane, and 2,3-dimethylhexane is the same as 3-methyloctane. These cases are easily dealt with because the trees are of different sizes. The identifier does distinguish all pairs of isospectral trees in the table. Computing the sums after five iterations instead of four may resolve these ambiguities, too. We did not actually try that experiment, but it would be a logical step if larger numbers of duplicates appeared when larger trees were added. The product of the edge extended connectivities provided a unique identifier, but the integers involved were so large that a significant penalty in execution time would have resulted from their use.

The second problem was somewhat less tractable. The solution we found consisted of keeping a running list of vertices accounted for as isolated vertices, parts of simple cycles and linear chains, and parts of trees already dealt with. To identify a new fragment in the adjacency matrix of a subgraph, we found the lowest-numbered vertex not already accounted for, then constructed a vector consisting of all zeroes except for a one in the position of that lowest-numbered vertex. Then, we iteratively computed additional vectors by $\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{A}\cdot\mathbf{v}_n$, where this **A** is the normal (not augmented) adjacency matrix of the subgraph. When the number of zeroes in **v** stops decreasing, then **v** has nonzero elements at the positions of all the vertices that belong to the new fragment. By examining the count of nonzero elements and the relevant row sums from the adjacency matrix, it is easy to determine whether the fragment can be looked up in the table. If the entire subgraph consists of such fragments, then its characteristic polynomial is computed as the product of the appropriate entries in the look-up tables; if not, the subgraph is decomposed further.

The third problem was dealt with by the exceptional ability of Mathematica® to manipulate vectors, and matrices. Mathematica® treats vectors as lists of numbers and matrices as lists of (row) vectors, and it has many built-in functions for performing various operations on those objects and determining the nature of their elements. With coronene as the test case, this method was still about 40% slower than the continuous-path method. With a C_{34} fullerene, however, it was about 33% *faster* than the continuous-path method. As we stated above, there is no guarantee that different algorithms for computing the acyclic polynomial will scale in the same way with the size of the graph.

Several further improvements could probably be made on this method. Acyclic polynomials of small graphs with branched cycles and polycycles could be added to the look-up tables. Simple trees with 10 vertices could be included (although this would double the number of trees in the look-up table). A more computationally efficient unique identifier for the trees could probably be found. Whether the methods discussed here ever become the most efficient for computing acyclic polynomials or not, we believe the greatest edge extended connectivity algorithm offers a novel approach to analyzing graphs that may prove useful in other contexts as well.

3 Conclusions

Several novel methods for computing the acyclic polynomial of a graph were programmed in Mathematica® and tested. So far, the most efficient of these for larger graphs is one based on a novel concept of decomposing the graph according to greatest edge extended

connectivity. This concept is a logical outgrowth of the previously elucidated vertex extended connectivity for uniquely identifying graphs [21,22]. To our knowledge, the concept of edge extended connectivity is new, and this descriptor may find uses in other areas in the future.

Acknowledgment

W. C. Herndon gratefully acknowledges the financial support of the Welch Foundation of Houston, Texas, and of the University of Texas at El Paso Center for Environmental Resource Management (E. P. A. Superfund Research Program).

Disclaimer

This document has been reviewed by the Office of Pollution Prevention and Toxics, U. S. EPA, and approved for publication. Approval does not signify that the contents necessarily reflect the views and policies of the Agency, nor does the mention of any trade names or commercial products constitute endorsement or recommendation for use.

References

3. Trinajstić, N., *Chemical Graph Theory*, Vol. II, CRC Press, Boca Raton, Fla. (1983), Chpt. 1.
4. Aihara, J. A New Definition of Dewar-Type Resonance Energies. *J. Am. Chem. Soc.* **1976,** *98*, 2750-2758.
5. Gutman, I.; Milun, M.; Trinajstić, N. Graph Theory and Molecular Orbitals, Part 19: Nonparametric Resonance Energies of Arbitrary Conjugated Systems. *J. Am. Chem. Soc.* **1977**, *99*, 1692-1704.
6. Aihara, J.; Hosoya, H. Spherical Aromaticity of Buckminsterfullerene. *Bull. Chem. Soc. Japan.* **1988**, *61*, 2657-2659.
5. Manoharan, M.; Balakrishnarajan, M. M.; Venuvanalingam, P.; Balasubramanian, K. Topological Resonance Energy Predictions of the Stability of Fullerene Clusters. *Chem. Phys. Lett.* **1994**, *222*, 95-100.
6. Babić, D.; Ori, O. Matching Polynomial and Topological Resonance Energy of C_{70}. *Chem. Phys. Lett.* **1995**, *234*, 240-244.
7. Aihara, J.; Babic, D.; Gutman, I. Matching Spectra of Fullerenes. *MATCH* **1996**, *33*, 7-16.
8. Babic, D.; Brinkmann, G.; Dress, A. Topological Resonance Energy of Fullerenes. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 920-923.
9. Balasubramanian, K. Exhaustive Generation and Analytical Expressions of Matching Polynomials of Fullerenes $C_{20\text{-}50}$. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 421-427.
10. Balasubramanian, K. Matching Polynomial of Fullerene Clusters. *Chem. Phys. Lett.* **1993**, *201*, 306-314.
11. Balakrishnaraja, M. M.; Venuvanalingam, P. General Method for the Computation of Matching Polynomials of Graphs. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 1122-1126.
12. Hosoya, H. Topological Index. A Newly Proposed Quantity Characterizing the Topological Nature of Structural Isomers of Saturated Hydrocarbons. *Bull. Chem. Soc. Japan.* **1971**, *44*, 2332-2339.

13. Hosaya, H.; Gotoh, G.; Murakimi, M.; Ikeda, S. Topological Index and Thermodynamic Properties. 5. How Can We Explain the Topological Dependency of Thermodynamic Properties of Alkanes With the Topology of Graphs? *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 192-196, and references cited therein.

14. Hosoya, H. Matching and Symmetry of Graphs. *Comput. Math. Appls.* **1986**, *12B*, 271-290.

15. Herndon, W. C.; Radhakrishnan, T. P.; Živković, T. P. Characteristic and Matching Polynomials of Chemical Graphs. *Chem. Phys. Lett.* **1988**, *152*, 233-237.

16. Salvador, J. M.; Hernandez, A.; Beltran, A.; Duran, R.; Mactutis, A. Fast Partial-Differential Synthesis of the Matching Polynomial of $C_{72\text{-}100}$. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 1105-1110.

17. Rosenfeld, V. R.; Gutman, I. A Novel Approach to Graph Polynomials. *MATCH* **1989**, *24*, 191-199.

18. Heilbronner, E. Das Kompositions-Prinzip: Eine anschauliche Methode zur elektronen-theoretischen Behandling nicht oder niedrig symmetrischer Molekeln im Rahmen der Mo-Theorie. *Helv. Chim. Acta* **1953**, *36*, 170-188.

19. Cash, G. G. A Simple Program for Computing Characteristic Polynomials with Mathematica®. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, accepted for publication.

20. Trinajstić, N., *Chemical Graph Theory*, 2nd Ed., CRC Press, Boca Raton, Fla. (1992), pp. 78-79.

21. Herndon, W. C. Canonical Labeling and Linear Notation for Chemical Graphs. *Chemical Applications of Topology and Graph Theory*, Elsevier, Amsterdam, (1983), pp. 231-242.

22. Herndon, W. C.; Leonard, J. E. Canonical Numbering, Stereochemical Descriptors, and Unique Notations for Polyhedral Clusters. Chemical Graphs. *Inorganic Chemistry* **1983**, *22*, 554-557.

23. Cvetkovic, D. M.; Doob, M.; Sachs, H. *Spectra of Graphs*, Academic Press (1980), New York, pp. 277-291.