MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **75** (2016) 703-714

ISSN 0340 - 6253

A Linear–Time Algorithm for the Hosoya Index of an Arbitrary Tree*

Jingyuan Zhang†, Xufeng Chen, Weigang Sun

School of Science, Hangzhou Dianzi University, Hangzhou 310018, P. R. China
jyzhang@hdu.edu.cn, chenxf@hdu.edu.cn, wgsun@hdu.edu.cn

(Received October 2, 2015)

Abstract

In this paper, we propose a novel method to calculate the Hosoya index of a tree by associating a vertex with a weight. Compared to the existing methods that include calculating the sum of the absolute values of all coefficients of the characteristic polynomial or computing the determinant of a tree's matrix, the complexity of computability of our method is lower. Based on the proposed method and the data structure of labeled trees, we further provide a linear–time algorithm to demonstrate the obtained results.

1 Introduction

The Hosoya index [1], proposed by Hosoya in 1971, is a typical example of graph invariants used in computational chemistry for quantifying the behavior of molecular structure, and it has been proved as a fundamental concept in correlations with boiling points [2], entropies [3], heat of vaporization [4], as well as for coding of chemical structures [5]. Till now, it has been used in a graph-based molecular descriptor [6]. A detailed survey for the Hosoya index has been given in [7–12].

In particular, the Hosoya index of a tree has attracted considerable attention in the past decades. For instance, Gutman *et al.* obtained a product formula in terms of

*The authors wish to acknowledge the contributions of the anonymous referees for pointing out many improvements to this paper. This research is supported by the Natural Science Foundation of China (Nos. 61203155 and 11402226) and the Zhejiang Provincial Natural Science Foundation of China (Nos. LY13F030016 and LY16A010014).

†The Corresponding author.

the eigenvalues [13]; Hou used the permanent method to compute the Hosoya index of a tree [14]; Gutman related caterpillar trees to the Kekulé structures in benzenoid molecules [15], and recently Hosoya and Gutman determined the Hosoya index of these trees [16]; Hudelson observed that each rooted tree corresponds to a unique tree expression that is evaluated as a rational number (not necessarily in lowest terms), whose numerator is equal to the Hosoya index of the entire tree [17].

Compared with the $\#P$-completeness for Hosoya index of even planar graphs [18], calculations of the Hosoya index of a tree of order n are denoted as the sum of the absolute values of the coefficients of the characteristic polynomial [19–21], where the complexity of computability is $O(nlog^2 n)$ [22]. It is also expressed as a determinant [23–25] with the complexity $O(n^{2+\epsilon})$ [26], where $\epsilon > 0$. In addition, the Hosoya index of a graph is the evaluation of the generating matching polynomial, and it can be calculated in linear time for a graph of bounded tree-width(see Theorem 32 in [27]). Therefore, we propose a novel method to calculate this index of a tree and provide an independent, elementary proof that the Hosoya index of a tree can be computed in linear time. We further present a linear–time algorithm based on this method and the data structure of labeled trees and verify the obtained results by a numerical example.

2 A formula for calculating the Hosoya index

Let G be a simple graph of order n with the vertex set $V(G)$ and the edge set $E(G)$. The Hosoya index $Z(G)$ is defined as the total number of independent edge sets of G, where two edges of G are *independent* if they have no vertex in common. In graph-theoretical terminology, $Z(G)$ is the number of all matchings of G, i.e. $Z(G) = \sum_{k\geq 0} m(G,k)$, where $m(G,k)$ is the number of k independent edges of G. By convention, $m(G,0)$ is one.

In order to derive our formula, we begin with this relationship [23, 24]

$$Z(G) = \det\left(I_n + A(G^o)\right), \tag{1}$$

where the graph G has no cycles of even length, I_n is the unit matrix of order n, G^o is an arbitrary orientation of G. $A(G^o) = (a_{kl})_{n\times n}$ is the skew adjacency matrix defined by $a_{kl} = 1$ if $(v_k, v_l) \in E(G^o)$, $a_{kl} = -1$ if $(v_l, v_k) \in E(G^o)$; otherwise $a_{kl} = 0$.

Let T be a tree with the vertex set $V(T) = \{v_1, \ldots, v_n\}$. If a vertex v_i is a leaf (a vertex of degree one) and v_iv_j is an edge, then (1) reads as

$$\det\begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & w_{ii} & \cdots & a_{ij} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & -a_{ij} & \cdots & w_{jj} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix},$$

where $w_{kk} = 1$, $1 \leq k \leq n$, $a_{ij} = 1$ or -1, and $a_{ik} = 0$ with $k \in \{1, \ldots, n\} - \{i, j\}$. Calculating the above determinant gives

$$Z(T) = \det\begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & w_{ii} & \cdots & a_{ij} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & \cdots & w_{jj} + 1/w_{ii} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} = w_{ii}\det(C), \tag{2}$$

where C is a matrix of order $n-1$ obtained from the matrix in (2) by deleting the ith row and the ith column. We further expand $Z(T)$ until the order of the corresponding matrix is reduced to one.

It is noted that the expansion of the determinant in (2) has a graphical interpretation. Define T as a rooted tree by associating every vertex v with a weight $\alpha(v)$, where the initial weights are all set one. Firstly, find a leaf $u_k, k = 1, \ldots, (n-1)$ in T, then delete u_k from T and contribute $1/\alpha(u_k)$ to the weight of its parent vertex. Repeat this process until the only root is left, see Fig. 1. In fact, the weights $\alpha(v)$ correspond to the diagonal elements w_{ii} in (2). On the other hand, the weights $\alpha(v)$ are also obtained recursively as follows: if v is a leaf of T, then $\alpha(v) = 1$; if v is not a leaf of T, then $\alpha(v) = 1 + \sum_{u \in S} \frac{1}{\alpha(u)}$, where S is the set of children of v.

Based on the process of calculations of $\alpha(v)$, we obtain a formula on calculating the Hosoya index of a tree.

Theorem 2.1. *Given the weights $\alpha(v)$ for each vertex v of T as above, then $Z(T) = \prod_{v \in T} \alpha(v)$.*

Remark 2.2. In Theorem 2.1, the tree could be rooted in an arbitrary way and the starting leaf is not restricted. Compared to the existing methods [13, 14, 17, 21], this method is intuitive and simple, and the computational complexity is lower.

Remark 2.3. Another proof of Theorem 2.1 is given below. A quantity $\beta(v) = \frac{m(v)}{m^-(v)}$ for every vertex v of the rooted tree T is recursively defined as follows: let T' be a copy of T.

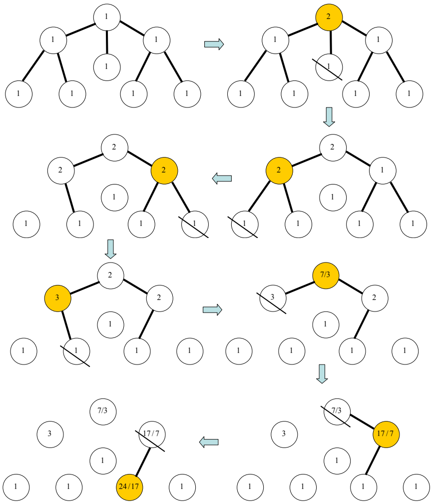

Figure 1: calculating the weights $\alpha(v)$. $Z(T_8) = 1 \times 1 \times 1 \times 1 \times 3 \times \frac{7}{3} \times \frac{17}{7} \times \frac{24}{17} = 24$.

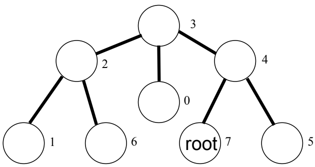

Figure 2: A labeled tree T_8 with 8 vertices labeled from 0 to 7.

(I) If T' is a single-vertex tree with root r, then $\beta(r) = m(r) = m^-(r) = 1$; otherwise, $\beta(r) = \frac{m(r)}{m^-(r)}$, where $m(r)$ counts all matchings of T', and $m^-(r)$ counts all matchings which do not contain the root r.

(II) Replace T' with each non-empty subtree of T' and go back to (I).

As a result, for a leaf v of T, $\beta(v) = 1$; otherwise, for the non-leaf vertex v, $\beta(v)$ reads as

$$\beta(v) = \frac{m(v)}{m^-(v)} = \frac{\prod_{u \in S} m(u) + \sum_{u \in S} m^-(u) \prod_{w \in S - \{u\}} m(w)}{\prod_{u \in S} m(u)} = 1 + \sum_{u \in S} \frac{1}{\beta(u)},$$

where S is the set of children of v. Hence, $\beta(v) = \alpha(v)$. Finally, $\prod_{v \in T} \alpha(v) = \prod_{v \in T} \beta(v) = m(r_0)$, where r_0 is the root of T. Obviously, $m(r_0)$ is the Hosoya index of T, and counts all matchings of T.

3 A linear–time algorithm for the Hosoya index

In this section, using the data structure of labeled trees, we provide a linear–time algorithm to calculate $Z(T)$ and verify the obtained method in the preceding section.

Lemma 3.1. *([28]) Given a list of $(n-1)$ edges of a labeled tree T whose vertices are labeled from 0 to $(n-1)$, there exists a linear time algorithm (see Algorithm 1) on T for Prüfer coding.*

Algorithm 1 Generating the Prüfer code of a labeled tree T [28].

Input:
A list of $(n-1)$ edges of T whose vertices are labeled from 0 to $(n-1)$.
Output:
The Prüfer code $(t_1, t_2, \cdots, t_{n-2})$ of T.

Build the degree array $d[\cdot]$ and parent array $f[\cdot]$ from T by the depth-first search method;
for $v = 0$ to $n-1$ **do**
  if $d[v] = 1$ **then**
    break;
  end if
end for
$index \leftarrow v$;
for $j = 0$ to $n-3$ **do**
  $u \leftarrow f[v]$;
  $t_{j+1} \leftarrow u$;
  $d[u] \leftarrow d[u] - 1$;
  if $u < index$ and $d[u] = 1$ **then**
    $v \leftarrow u$;
  else
    for $v = index + 1$ to $n-1$ **do**
      if $d[v] = 1$ **then**
        break;
      end if
    end for
    $index \leftarrow v$;
  end if
end for
return $(t_1, t_2, \cdots, t_{n-2})$.

Lemma 3.2. *([29]) Any vertex v of T occurs $d(v) - 1$ times in the Prüfer code $(t_1, t_2, \cdots, t_{n-2})$.*

From Lemma 3.2, when the Prüfer code of a labeled tree with n vertices is input, there exists a linear time algorithm (see Algorithm 2) to generate the degree array $d[\cdot]$ and parent array $f[\cdot]$ (see Algorithm 3).

Algorithm 2 Generating the degree array $d[\cdot]$ of a labeled tree T.

Input:
A Prüfer code $(t_1, t_2, \cdots, t_{n-2})$ of T.
Output:
The degree array $d[\cdot]$.

for $v = 0$ to $n-1$ **do**
    $d[v] \leftarrow 1$;
end for
for $j = 0$ to $(n-3)$ **do**
    $v \leftarrow t_{j+1}$
    $d[v] \leftarrow d[v] + 1$;
end for
return $d[\cdot]$.

Algorithm 3 Producing the parent array $f[\cdot]$ of a labeled tree T.

Input:
A Prüfer code $(t_1, t_2, \cdots, t_{n-2})$ of T.
Output:
The parent array $f[\cdot]$.

$f[n-1] \leftarrow -1$;
$t_{n-1} \leftarrow (n-1)$;
for $v = 0$ to $n-1$ **do**
    if $d[v] = 1$ **then**
        break;
    end if
    $index \leftarrow v$;
end for
for $j = 0$ to $n-2$ **do**
    $u \leftarrow t_{j+1}$;
    $d[u] \leftarrow d[u] - 1$;
    $f[v] \leftarrow u$;
    if $u < index$ and $d[u] = 1$ **then**
        $v \leftarrow u$;
    else
        for $v = index + 1$ to $n-1$ **do**
            if $d[v] = 1$ **then**
                break;
            end if
        end for
        $index \leftarrow v$
    end if
end for
return $f[\cdot]$.

Theorem 3.3. *Given a list of $(n-1)$ edges of a labeled tree whose vertices are labeled from 0 to $(n-1)$, there exists a linear time algorithm(see Algorithm 4) to compute the*

Algorithm 4 Calculating the Hosoya index of a labeled tree T.

Input:

A list of $(n-1)$ edges of a labeled tree T whose vertices are labeled from 0 to $(n-1)$.

Output:

The Hosoya index Z of T.

1: $root \leftarrow n-1$;
2: Recall Algorithm 1 to build the Prüfer code of T;
3: Recall Algorithm 2 to generate the degree array $d[\cdot]$ from the Prüfer code of T;
4: Recall Algorithm 3 to produce the parent array $f[\cdot]$ from the Prüfer code of T;
5: $index \leftarrow x \leftarrow \min\{-1 < k < root : d[k] = 1\}$;
6: For each vertex v in T, set the weight $\alpha[v] \leftarrow 1$;
7: $Z \leftarrow 1$;
8: **for** $j = 0$ to $n-2$ **do**
9:     $y \leftarrow f[x]$;
10:     $Z \leftarrow Z * \alpha[x]$;
11:     $\alpha[y] \leftarrow \alpha[y] + \frac{1}{\alpha[x]}$;
12:     $d[y] \leftarrow d[y] - 1$;
13:     **if** $y < index$ and $d[y] = 1$ **then**
14:         $x \leftarrow y$;
15:     **else**
16:         $index \leftarrow x \leftarrow \min\{index < k < root : d[k] = 1\}$;
17:     **end if**
18: **end for**
19: $Z \leftarrow Z * \alpha[root]$;
20: **return** Z.

Hosoya index of the considered tree.

Proof. Algorithm 4 provides the Hosoya index Z of a labeled tree T as an output and receives a list of its $(n-1)$ edges as an input. From Theorem 2.1, the root vertex is the only left vertex after the successive operations of deleting a leaf. In Algorithm 4, we select a vertex with the maximal label $(n-1)$ as the root, actually the root of T may be an arbitrary vertex of T. This is a key point to enable the algorithm to be linear. In general, we can find the vertex with the smallest label in $O(n \log n)$ time by using the heap $\min\{index < k < root : d[k] = 1\}$, where the variable *index* is a cursor of degree array $d[\cdot]$. From the existing results [28], we observe that some vertices are immediately deleted from the heap before they are inserted into the heap. This type of vertices are easily treated without the heap operation. As a result, the remained heap operation can be performed by a linear scan of $d[\cdot]$. From line 16, we see that the cursor *index* moves from one leaf to the next leaf, and goes through $d[\cdot]$ from left to right only once. So the time of line 16 is $O(n)$ time. Furthermore, the parent array $f[\cdot]$ and degree array $d[\cdot]$ are built with only $O(n)$ preprocessing time by the depth-first search method [30]. Thus, the time complexity of Algorithm 4 is $O(n)$ time, i.e., a linear time.

Corollary 3.4. *Given the Prüfer code of a labeled tree with n vertices as an input, there exists a linear–time algorithm to compute the Hosoya index of this tree.*

4 A example

We choose a labeled tree T_8 (see Fig. 2) to demonstrate Algorithm 4. The *root* is vertex 7. The algorithm accepts the edge list {0,3}, {2,3}, {3,4}, {2,6}, {1,2}, {4,7}, {4,5} of T_8 as an input. Table 1 shows $f[\cdot]$, $d[\cdot]$ and weight array $\alpha[\cdot]$ with initial values. At the initial state, the *index* is equal to the subscript 0. Firstly, the leaf 0 is deleted. Then, the degree $d[3]$ of vertex 3 decreases by 1 and the weight $\alpha[3]$ of vertex 3 increases by $1/1 = 1$. Since $d[3] > 1$, the vertex 3 is not a leaf, the program moves *index* to the next leaf by scanning $d[\cdot]$ and turns to the subscript 1. Now the status of T_8 is provided in Table 2. Next, the vertex 1 is deleted and the status of T_8 with updating $\alpha[2]$ and $d[2]$ is in Table 3. Repeating the above mentioned process until the root is left, we obtain the corresponding states of T_8, see Tables 4, 5, 6. Finally, we obtain $Z = \prod_{v=0}^{7} \alpha[v] = 24$.

Table 1: The arrays $f[\cdot], \alpha[\cdot], d[\cdot]$ with initial values.

j	0	1	2	3	4	5	6	7
$f[j]$	3	2	3	4	7	4	2	-1
$\alpha[j]$	1	1	1	1	1	1	1	1
$d[j]$	1	1	3	3	3	1	1	1
index	↑							

Table 2: After leaf 0 is deleted

j	[0]	1	2	3	4	5	6	7
$f[j]$	3	2	3	4	7	4	2	-1
$\alpha[j]$	1	1	1	**2**	1	1	1	1
$d[j]$	1	1	3	**2**	3	1	1	1
index		↑						

Table 3: After leaf 1 is deleted

j	[0]	[1]	2	3	4	5	6	7
$f[j]$	3	2	3	4	7	4	2	-1
$\alpha[j]$	1	1	**2**	2	1	1	1	1
$d[j]$	1	1	**2**	2	3	1	1	1
index						$\uparrow$		

Table 4: After leaf 5 is deleted

j	[0]	[1]	2	3	4	[5]	6	7
$f[j]$	3	2	3	4	7	4	2	-1
$\alpha[j]$	1	1	2	2	**2**	1	1	1
$d[j]$	1	1	2	2	**2**	1	1	1
index							$\uparrow$	

Table 5: After leaf 6 is deleted

j	[0]	[1]	2	3	4	[5]	[6]	7
$f[j]$	3	2	3	4	7	4	2	-1
$\alpha[j]$	1	1	**3**	2	2	1	1	1
$d[j]$	1	1	**1**	2	2	1	1	1
index							$\uparrow$	

Table 6: After the leaves 2, 3, and 4 are deleted.

j	[0]	[1]	[2]	[3]	[4]	[5]	[6]	7
$f[j]$	3	2	3	4	7	4	2	-1
$\alpha[j]$	1	1	3	$\frac{7}{3}$	$\frac{17}{7}$	1	1	$\frac{24}{17}$
$d[j]$	1	1	1	1	1	1	1	1
index							$\uparrow$	

Remark 4.1. The troublesome aspect of Algorithm 4 is that the arithmetic produces fractions, which can be overcome by using a vector trick. Denote $\alpha[v] = \frac{p[v]}{q[v]}$ by the vector $(p[v], q[v])$. Then, the value of $\alpha[y] + \frac{1}{\alpha[x]}$ in line 11 is rewritten as the vector $(p[y]p[x] + q[y]q[x], q[y]p[x])$. Here the improved algorithms are omitted.

References

[1] H. Hosoya, a newly proposed quantity characterizing the topological nature of structural isomers of saturated hydrocarbons, *Bull. Chem. Soc. Jpn.* **44** (1971) 2332–2339.

[2] H. Hosoya, K. Kawasaki, K. Mizutani, Topological index and thermodynamic properties. I. Empirical rules on the boiling point of saturated hydrocarbons, *Bull. Chem. Soc. Jpn.* **45** (1972) 3415–3421.

[3] H. Narumi, H. Hosoya, Topological index and thermodynamic properties. II. Analysis of the topological factors on the absolute entropy of acyclic saturated hydrocarbons, *Bull. Chem. Soc. Jpn.* **53** (1980) 1228–1237.

[4] H. Hosoya, M. Gotoh, M. Murakami, S. Ikeda, 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.* **39** (1999) 192–196.

[5] R. E. Merrifield, H. E. Simmons, *Topological Methods in Chemistry*, Wiley, New York, 1989.

[6] M. Randić, J. Zupan, On interpretation of well-known topological indices, *J. Chem. Inf. Comput. Sci.* **41** (2001) 550–560.

[7] O. Chan, I. Gutman, T. K. Lam, R. Merris, Algebraic connections between topological indices, *J. Chem. Inf. Comput. Sci.* **38** (1998) 62–65.

[8] I. Gutman, O. E. Polansky, *Mathematical Concepts in Organic Chemistry*, Springer, Berlin, 1986.

[9] I. Gutman, Acyclic conjugated molecules, trees and their energies, *J. Math. Chem.* **29** (1987) 123–143.

[10] S. J. Cyvin, I. Gutman, *Kekulé Structures in Benzenoid Hydrocarbons*, Springer, Berlin, 1988.

[11] H. Hosoya, The topological index Z before and after 1971, *Int. El. J. Mol. Des.* **1** (2002) 428–442.

[12] S. Wagner, I. Gutman, Maxima and minima of the Hosoya index and the Merrifield–Simmons index, *Acta. Appl. Math.* **112** (2010) 323–346.

[13] I. Gutman, Z. Marković, S. Marković, A simple method for the approximate calculation of Hosoya's index, *Chem. Phys. Lett.* **134** (1987) 139–142.

[14] Y. Hou, On acyclic systems with minimal Hosoya index, *Discr. Appl. Math.* **119** (2002) 251–257.

[15] I. Gutman, Topological properties of benzenoid systems, *Theor. Chim. Acta* **45** (1977) 309–315.

[16] H. Hosoya, I. Gutman, Kekulé structures of hexagonal chains–some unusual connections, *J. Math. Chem.* **44** (2008) 559–568.

[17] M. Hudelson, Vertex topological indices and tree expressions, generalizations of continued fractions, *J. Math. Chem.* **47** (2010) 219–228.

[18] M. Jerrum, Two–dimensional monomer–dimer systems are computationally intractable, *J. Stat. Phys.* **48** (1987) 121–134.

[19] A. Mowshowitz, The characteristic polynomial of a graph, *J. Comb. Theory B* **12** (1972) 177–193.

[20] M. Ahmadi, H. Dastkhzer, On the Hosoya index of trees, *J. Optoelectron. Adv. Mat.* **13** (2011) 1122–1125.

[21] A. Anuradha, R. Balakrishnan, W. So, Skew spectra of graphs without even cycles, *Lin. Algebra Appl.* **444** (2014) 67–80.

[22] M. Fürer, Efficient computation of the characteristic polynomial of a tree and related tasks, *Algorithmica* **68** (2014) 626–642.

[23] E. J. Farrell, S. A. Wahid, D-graphs, I. An introduction to graphs whose matching polynomials are determinants of matrices, *Bull. Inst. Combin. Appl.* **15** (1995) 81–86.

[24] W. Yan, Y. Yeh, F. Zhang, On the matching polynomials of graphs with small number of cycles of even length, *Int. J. Quantum Chem.* **105** (2005) 124–130.

[25] D. Deford, *An application of the permanent–determinant method: computing the Z-index of tree*, http://www.math.wsu.edu/TRS/2013-2.pdf.

[26] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison–Wesley, Reading, 1974.

[27] B. Courcelle, J. A. Makowsky, U. Rotics, On the fixed parameter complexity of graph enumeration problems definable in monadic second order logic, *Discr. Appl. Math.* **108** (2001) 23–52.

[28] X. Wang, L. Wang, Y. Wu, An optimal algorithm for Prufer codes, *J. Soft. Engin. Appl.* **2** (2009) 111–115.

[29] J. A. Bondy, U. S. R. Murty, *Graph Theory with Applications*, North-Holland, 1976.

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, 2001.