**MATCH**
*Communications in Mathematical and in Computer Chemistry*

MATCH Commun. Math. Comput. Chem. **78** (2017) 665-674

ISSN 0340 - 6253

# On Some Algorithms for Computing Topological Indices of Chemical Graphs

**Aleksandar Ilić**
*Facebook Inc, Menlo Park, CA, USA*
e-mail: `aleksandari@gmail.com`

**Milovan Ilić**
*Metropolitan University, Belgrade, Serbia*
e-mail: `ilic.milovan@gmail.com`

(Received December 1, 2016)

**Abstract**

In this note, we present some improvements on the recently proposed algorithms for computing certain topological indices of chemical graphs. In particular, we design simpler recursive algorithm for computing Hosoya index of trees and unicyclic graphs, present better time complexity algorithm for computing Wiener polarity index of chemical graphs, reference more efficient algorithm for computing the Merrifield-Simmons index of graphs, and compute Balaban index of trees in linear time using depth first search algorithm.

## 1 Introduction

In theoretical chemistry molecular structure descriptors (also called topological indices) are used for modeling physico-chemical, pharmacologic, toxicologic, biological and other properties of chemical compounds [12, 23]. There exist several types of such indices, especially those based on distances and independent sets of vertices and edges. The Wiener [10] and Hosoya [16] index are arguably the best known global indices that describe entire molecule.

In this note, we continue analysis of the algorithms for computing certain chemical topological indices of trees and chemical graphs. It is very important to have efficient algorithms for computing such descriptors, as there are a lot of research studies on deriving closed formulas for special classes of chemical graphs: for example dendrimers [2], polyphenyl chains [24], fullerenes and hexagonal systems [4], large molecules [13], etc.

Let $G$ be a simple connected graph of order $n$ with the vertex set $V(G)$ and the edge set $E(G)$. The distance between the vertices $v$ and $u$, denoted by $d(v, u)$, is defined as the length of a shortest path connecting $v$ and $u$. Two edges of $G$ are independent if they have no vertex in common, and similarly two vertices of $G$ are independent if they are not adjacent. A $k$-independent set of edges (vertices) is a set of $k$ mutually independent edges (vertices). Let $deg(v)$ denote the degree of the vertex $v$. A rooted tree is a tree that has one vertex distinguished as a root, where every vertex besides the root has a unique parent and every vertex is associated with a list of child vertices (this list can be empty).

A chemical graph is a labeled graph whose vertices correspond to the atoms of the compound and edges correspond to chemical bonds. In most applications, such graphs have vertex degree $\leq 4$.

The paper is organized as follows. In Section 2, we design a simpler recursive algorithm for computing Hosoya index of trees and unicyclic graphs, thus improving the results from [26]. In Section 3, we present better time complexity algorithm for computing Wiener polarity index of chemical graphs, thus improving the results from [11]. In Section 4, we reference more efficient algorithm for computing Merrifield-Simmons index of graphs by counting the cliques in the complement graph instead of independent sets, thus improving the results from [1]. Finally, we present a linear recursive algorithm for computing Merrifield-Simmons and Balaban index of trees in Section 5 based on the modification of depth first search graph traversal.

These results can be simply generalized to weighted trees and some other distance-based indices or polynomials.

## 2 Computing Hosoya index of trees and unicyclic graphs

The Hosoya index $Z(G)$ is defined as the total number of independent edge sets of $G$. In graph-theoretical terminology, $Z(G)$ is the number of all matchings of $G$, i.e.

$$Z(G) = \sum_{k \geq 0} m_k(G),$$

where $m_k(G)$ is the number of $k$ independent edges of $G$. By convention, $m_0(G)$ is one.

The Hosoya index [15], proposed by Hosoya in 1971, is a typical example of a graph invariant 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, entropies, heat of vaporization, as well as for coding of chemical structures (see [16]).

Recently, Zhang, Chen and Sun in [26] introduced a novel method to calculate the Hosoya index of a tree by associating a vertex with a weight and using the data structure of labeled trees with Prüfer's code. That algorithm involved multiple steps and complex implementation and analysis.

Here we present a simple recursive linear solution based on depth first search of the rooted tree [7, 18]. The main idea is to keep two arrays:

- $h[v]$, that represents the total number of matchings under the subtree rooted at $v$
- $hw[v]$, that represents the total number of matchings under the subtree rooted at $v$ that do not have the root vertex $v$ paired.

We are also going to use the following two simple results for Hosoya index. For an arbitrary edge $e = uv$ of $G$, it holds

$$Z(G) = Z(G - uv) + Z(G - \{u, v\}), \tag{1}$$

where $G - uv$ is a graph obtained from $G$ by removing the edge $uv$, and $G - \{u, v\}$ is a graph obtained from $G$ by removing the vertices $u$ and $v$ together with corresponding edges that are adjacent to them.

If $G$ is a graph with connected components $G_1, G_2, \ldots, G_k$, then

$$Z(G) = Z(G_1) \cdot Z(G_2) \cdot \ldots \cdot Z(G_k). \tag{2}$$

The terminating condition of the recursion is when the algorithm reaches the leaf vertex of degree one, and we set the default values in that case (note that $m_0(G) = 1$). Otherwise, we just iterate through the child vertices and recursively compute the values $h$ and $hw$. From Equation (2) the value $hw[v]$ is simply the product of the values $h[u]$ for all child vertices, while from Equation (1) for $h[v]$ we need to fix one edge and take any combination of the remaining independent subtrees. We can pick any vertex as root and call the function $DFS1(root)$ to compute the Hosoya index. The pseudo-code of this approach is implemented in Algorithm 1.

In practice, this algorithm is much faster than the one proposed in [26] because of multiple iterations of the input tree. Note also that we do not need the second for loop, if we can compute the sum $h[u]/hw[u]$ with enough precision.

**Algorithm 1:** Compute Hosoya index of the rooted tree

```
Function DFS1(vertex $v$)
    Data: Tree with child links
    Result: The final solution is $h[root]$
    if $v$ is a leaf then
        $h[v] = 1$;
        $hw[v] = 1$;
    end
    else
        $prod = 1$;
        for every child $u$ of $v$ do
            $DFS1(u)$;
            $prod = prod * h[u]$;
        end
        $hw[v] = prod$;
        $h[v] = hw[v]$;
        for every child $u$ of $v$ do
            $h[v] = h[v] + (prod/h[u]) * hw[u]$;
        end
    end
```

Algorithm 1 can be easily generalized to compute the Hosoya index of unicyclic and bicyclic graphs in $O(n)$ time - by reducing the problem to trees. For unicyclic graph, fix one edge $e = uv$ from the unique cycle. For $k$ independent set of edges we can apply either Equation (1) or (2), i.e.

(i) take the edge edge $e = uv$ in the matching, in which case we decompose the unicyclic graph into multiple trees by removing the vertices $u$ and $v$, or

(ii) do not take the edge $e$ in the matching, in which case the remaining graph is a tree.

Both of the above problems can be solved in linear time, and this completes the algorithm for computing Hosoya index of unicyclic graphs. For bicyclic graphs, we just need to pick two edges whose removal will make the remaining graph a tree - which can again be solved in $O(n)$ time.

## 3 Computing Wiener polarity index of chemical graphs

The Wiener polarity index of a graph $G$ is defined as the number of unordered pairs of vertices $\{u, v\}$ of $G$ such that the shortest distance $d(u, v)$ between $u$ and $v$ is 3,

$$W_P(G) = |\{\{u, v\} \mid d(u, v) = 3,\ u, v \in V\}|.$$

Hosoya [16] found a physico-chemical interpretation of $W_P$, and recently there are multiple research on mathematical properties of this index (see [20] and references therein).

Du, Li and Shi in [11] constructed a linear algorithm for computing the Wiener polarity index of trees. Furthermore, they presented an algorithm which computes the index $W_P(G)$ for any given connected graph $G$ on $n$ vertices in time $O(M(n))$, where $M(n)$ denotes the time necessary to multiply two $n \times n$ matrices of small integers - which is currently known to be $O(n^{2.376})$.

Ilić and Ilić in [20] defined the generalized Wiener polarity index as the number of unordered pairs of vertices $\{u, v\}$ of $G$ such that the shortest distance $d(u, v)$ between $u$ and $v$ is $k$,

$$W_k(G) = |\{\{u, v\} \mid d(u, v) = k,\ u, v \in V\}|.$$

For $k = 3$, we exactly get the Wiener polarity index. Furthermore, the same authors presented a linear algorithm for calculating this index for trees, thus generalizing results from [11].

Yuster and Zwick in [25] proposed a new algorithm that multiplies two matrices $A$ and $B$ using $O(m^{0.7}n^{1.2} + n^{2+o(1)})$ algebraic operations (i.e., multiplications, additions and subtractions), where $m$ denotes the number of non-zero values (or equivalently the number of edges in the adjacency matrix). The naive matrix multiplication algorithm, on the other hand, may need to perform $O(mn)$ operations to accomplish the same task.

For $m \leq n^{1.14}$, which is the case for chemical graphs given that their vertex degree is less than or equal to 4, this algorithm performs an almost optimal number of only $n^{2+o(1)}$ operations (as $1.14 \cdot 0.7 + 1.2 < 2$). For $m \leq n^{1.68}$, this algorithm is also faster than the best known matrix multiplication algorithm for dense matrices, thus being superior alternative for computing the Wiener polarity index in this case as well.

## 4 Computing Merrifield–Simmons index of graphs and trees

Denote by $i_k(G)$ the number of the $k$-independent vertex sets of $G$. The Merrifield-Simmons index [21] of a molecular graph $G$ is defined as the total number of the independent vertex sets,

$$i(G) = \sum_{k=0}^{n} i_k(G).$$

Ahmadi and Alimorad Dastkhezr in [1] presented an algorithm for calculating the number of $k$-independent sets of graph $G$ using its adjacency matrix and then obtained the Merrifield-Simmons index by summing over all subsets sizes.

Clique is a complete subgraph of a graph $G$. The complement of a graph $G$ is a graph $\overline{G}$ on the same vertex set $V(G)$, such that two distinct vertices of $\overline{G}$ are adjacent if and only if they are not adjacent in $G$. Computing the number of independent sets of size $k$ is equivalent to computing the number of cliques of size $k$ in the complement graph $\overline{G}$.

Bron and Kerbosch [6] described a recursive algorithm to compute all cliques in linear time (relative to the number of cliques), and for more details on maximal clique problems see [5]. It is still widely used and referred to as one of the fastest algorithms according to recent survey by Stix [22]. Therefore, the simple algorithm proposed in [1] is inferior compared to these state-of-the-art algorithms and should not be used in practice.

Similarly as in Section 2, we can construct a recursive linear algorithm to compute Merrifield-Simmons index of trees based on depth first search. For that we define the following two arrays:

- $i[v]$, that represents the total number of independent sets under the subtree rooted at $v$, and
- $iw[v]$, that represents the total number of independent sets under the subtree rooted at $v$ that do not include the root vertex $v$.

Note that Algorithms 1 and 2 can be simply generalized to compute Hosoya (matching) and independence polynomials [14], by storing the array of coefficients.

**Algorithm 2:** Compute Merrifield-Simmons index of the rooted tree

```
Function DFS2(vertex v)
    Data: Tree $T$ with child links
    Result: The final solution is $i[root]$
    $prod\_i = 1$;
    $prod\_iw = 1$;
    for every child $u$ of $v$ do
        $DFS2(u)$;
        $prod\_i = prod\_i * i[u]$;
        $prod\_iw = prod\_iw * iw[u]$;
    end
    $iw[v] = prod\_i$;
    $i[v] = iw[v] + prod\_iw$;
```

## 5 Computing Balaban index of trees

Let $D(v)$ denotes the summation of all distances between a fixed vertex $v$ and all other vertices of $G$. Naive algorithm for computing $D(v)$ for all vertices runs in $O(nm)$, using breadth first search from every vertex [7].

The Balaban index is a topological index introduced by Balaban more than 30 years ago [3]. It is defined as

$$J(G) = \frac{m}{\mu + 1} \sum_{uv \in E} \frac{1}{\sqrt{D(u)D(v)}},$$

where $m$ is the number of edges of $G$, $n$ is the number of vertices and $\mu = m - n + 1$ is called the cyclomatic number of $G$. The Balaban index is one of the widely used topological indices for QSAR and QSPR studies, and appears to be a very useful molecular descriptor with attractive properties [2,8].

Here we describe a linear algorithm for computing Balaban index of trees, composed of two parts. In the first recursion, for every vertex $v$ we compute two arrays:

- $c[v]$, that stores the number of vertices under the subtree rooted at $v$ (including $v$), and
- $D[v]$, that stores the sum of distances from the root $v$ to all vertices under $v$.

After the first part, $D[root]$ contains exactly the sum of all distances from the root to all other vertices and $c[root] = n$. In the second recursion, for every edge $e = vu$ where $u$ is a child of $v$ – we will adjust and compute $D[u]$ from the values $D[v]$ and $c[u]$. Namely, the distances from $u$ to all vertices under $u$ are decreased by one and for all the other vertices are increased by one. The terminating condition of the recursion is implicitly handled for leaves, given that they do not have any child vertices, and for loop will not be executed. The full implementation is described in Algorithm 3 and 4, and the final value of $J$ should be multiplied by the factor of $\frac{m}{m-n+2}$, which is equal to $n-1$ for trees.

Also note that using this method one can compute other variants of this index, like sum Balaban index [9], eccentric distance sum [19] or eccentric connectivity index [17]. Furthermore, the algorithm can be simply generalized to weighted trees.

**Algorithm 3:** Compute the arrays $c$ and $D$

```
Function DFS3(vertex v)
    Data: Tree $T$ with child links
    Result: The arrays $c$ and $D$
    $c[v] = 1$;
    $D[v] = 0$;
    for every child $u$ of $v$ do
        $DFS3(u)$;
        $c[v] = c[v] + c[u]$;
        $D[v] = D[v] + D[u] + c[u]$;
    end
```

**Algorithm 4:** Compute Balaban index of the rooted tree

```
Function DFS4(vertex v)
    Data: Tree $T$ with child links, and arrays $c$ and $D$
    Result: The final solution is stored in the global variable $J$
    for every child $u$ of $v$ do
        $D[u] = D[v] - c[u] + (n - c[u])$;
        $J = J + 1/sqrt(D[v] * D[u])$;
        $DFS4(u)$;
    end
```

## References

[1] M. B. Ahmadi, H. Alimorad Dastkhezr, An algorithm for computing the Merrifield–Simmons index, *MATCH Commun. Math. Comput. Chem.* **71** (2014) 355–359.

[2] A. R. Ashrafi, H. Shabani, M. V. Diudea, Balaban index of dendrimers, *MATCH Commun. Math. Comput. Chem.* **69** (2013) 151–158.

[3] A. T. Balaban, Highly discriminating distance–based topological index, *Chem. Phys. Lett.* **89** (1982) 399–404.

[4] A. Behmaram, H. Yousefi–Azari, A. R. Ashrafi, Wiener polarity index of fullerenes and hexagonal systems, *Appl. Math. Lett.* **25** (2012) 1510–1513.

[5] I. M. Bomze, M. Budinich, P. M. Pardalos, M. Pelillo, The maximum clique problem, in: D. Z. Du, P. M. Pardalos (Eds.), *Handbook of Combinatorial Optimization*, Kluwer, Boston, 1999.

[6] C. Bron, J. Kerbosch, Algorithm 457: Finding all cliques of an undirected graph, *Commun. ACM* **16** (1973) 575–577.

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

[8] H. Deng, On the Balaban index of trees, *MATCH Commun. Math. Comput. Chem.* **66** (2011) 253–260.

[9] H. Deng, On the sum–Balaban index, *MATCH Commun. Math. Comput. Chem.* **66** (2011) 273–284.

[10] A. A. Dobrynin, R. C. Entringer, I. Gutman, Wiener index of trees: theory and applications, *Acta Appl. Math.* **66** (2001) 211–249.

[11] W. Du, X. Li, Y. Shi, Algorithms and extremal problem on Wiener polarity index, *MATCH Commun. Math. Comput. Chem.* **62** (2009) 235–244.

[12] I. Gutman, O. E. Polansky, *Mathematical Concepts in Organic Chemistry*, Springer–Verlag, Berlin, 1988.

[13] I. Gutman, On the Hosoya index of very large molecules, *MATCH Commun. Math. Comput. Chem.* **23** (1988) 95–103.

[14] I. Gutman, S. Klavžar, M. Petkovšek, P. Žigert, On Hosoya polynomials of benzenoid graphs, *MATCH Commun. Math. Comput. Chem.* **43** (2001) 49–66.

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

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

[17] A. Ilić, Eccentric connectivity index, in: I. Gutman, B. Furtula (Eds.), *Novel Molecular Structure Descriptors – Theory and Applications II*, Univ. Kragujevac, Kragujevac, 2010.

[18] A. Ilić, S. Klavžar, D. Stevanović, Calculating the degree distance of partial Hamming graphs, *MATCH Commun. Math. Comput. Chem.* **63** (2010) 411–424.

[19] A. Ilić, G. H. Yu, L. H. Feng, On the eccentric distance sum of graphs, *J. Math. Anal. Appl.* **381** (2011) 590–600.

[20] A. Ilić, M. Ilić, Generalizations of Wiener polarity index and terminal Wiener index, *Graphs Comb.* **29** (2013) 1403–1416.

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

[22] V. Stix, Finding all maximal cliques in dynamic graphs, *Comput. Opt. Appl.* **27** (2004) 173–186.

[23] R. Todeschini, V. Consonni, *Handbook of Molecular Descriptors*, Wiley–VCH, Weinheim, 2002.

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

[25] R. Yuster, U. Zwick, Fast sparse matrix multiplication, *ACM Trans. Alg.* **1** (2005) 2–13.

[26] J. Zhang, X. Chen, W. Sun, A linear time algorithm for the Hosoya index of an arbitrary tree, *MATCH Commun. Math. Comput. Chem.* **75** (2016) 703–714.