ISSN 2007-9737

Extremal Instances of Polyphenylene Dendrimers for the Merrifield-Simmons Index

Guillermo De Ita Luna, Michelle Guerra-Marín*, Juan Pintor-Michimani, Cybele Neves-Moutinho

Benemérita Universidad Autónoma de Puebla,
Mexico

{guillermo,deita}@correo.buap.mx, {michelle.guerram,juan.pintorm}@alumno.buap.mx,
cyneves@gmail.com

Abstract. Computing topological indices for molecular graphs is a key element in computational chemistry, particularly when analyzing the structural and functional properties of chemical compounds. Among these indices, the Merrifield-Simmons (M-S) index, defined as the total number of independent vertex sets in a molecular graph, provides valuable information on the compartmentalization, stability, and connectivity of a molecular graph. These properties are essential when dendrimer structures are used in nanotechnology, drug delivery systems, and advanced material design. Dendrimers, especially polyphenylene dendrimers (PPDs), are highly branched macromolecules known for their robustness, shape persistence, and ability to encapsulate and release therapeutic agents in a controlled manner. This paper recognizes the patterns for extreme topologies associated with the M-S index on dendrimer graph molecules, analyzing its structural features that maximize or minimize this topological invariant. This study shows how graph theory and recurrence relations can be used to design efficient counting algorithms, benefiting both the pattern recognition area and practical applications in molecular design, drug delivery, and nanomaterials engineering.

Keywords. Merrifield–Simmons index, dendrimers, polyphenylene dendrimers, topological indices, independent sets, efficient algorithms.

1 Introduction

Since their initial successful studies in the 1980s, dendrimers have attracted significant attention. They are polymers characterized by repeating units emerging from a focal point, possessing a large number of exposed terminal functionalities on the surface [13]. Polyphenylene dendrimers (PPDs) are highly branched, differing from classical polymers due to their step-by-step synthesis, which ensures a precisely defined and reproducible 3D architecture [10]. This highly controlled structure makes them exceptionally stable and useful, for instance, in vaccine delivery, where they enhance immune response and efficacy [4].

Graph theory provides essential tools to chemists, such as topological descriptors, by representing molecules as molecular graphs. In mathematical chemistry, topological indices are fundamental for analyzing the physical properties of chemical compounds, establishing the correlation between molecular topology and physicochemical properties instrumental in Quantitative Structure-Activity Relationships (QSAR) and Quantitative Structure-Property Relationships (QSPR) [16, 3].

A topological index is a real number associated with the structural graph of a molecule, independent of its visual representation [1]. The Merrifield-Simmons (M-S) Index, $i(G)$, defined as the total number of independent sets of a graph G [15], is a key descriptor. By quantifying independent sets, the M-S index reveals node "disconnectedness," which is relevant to properties like high drug encapsulation and controlled release capacity [14].

The mathematical properties of the M-S index have been extensively studied [15], although

its computational application in QSPR/QSAR for complex structures still presents challenges. Computing topological indices involves designing specialized algorithms to predict molecular properties without direct experimentation [11].

Basic techniques developed for computing the M-S index use computation on the adjacency matrix of the molecular graph [1], or to apply recurrence relations and Branching and Pruning algorithms. These methods decompose a graph G based on the vertex reduction rule: $i(G) = i(G-v)+i(G-N[v])$, where v is a vertex and $N[v]$ is its closed neighborhood. While mathematically sound, these general recursive methods can face significant computational complexity which is often exponential, making them inefficient for large graphs [9, 8, 16, 11, 5].

To overcome the application of exponential algorithms, the state of the art has focused on two main strategies: first, developing highly optimized formulas and recurrence relations for graphs with predictable symmetry, such as trees [12], benzenoid systems [6, 17], and specific linear chain structures; and second, devising tailored algorithms for particular molecular classes like certain types of dendrimers [2]. Recent research also explores hybrid approaches combining graph theory with machine learning for property prediction [7].

A clear gap exists between the complexity of PPD graph structures and the efficiency of existing computational methods. For instance, in [8] an algorithm for computing the M-S index on polygonal grids is introduced. However, this proposal has an exponenial time behaviour. On the other hand, dendrimer structures (PPDs) have a highly hierarchical and self-similar branching topology, thus, the application of general decomposition graph techniques to PPDs fails to leverage the unique structural symmetries of dendrimers, resulting in the high computational cost associated with general recursion.

Computational modeling has become an essential tool in almost every stage of drug discovery. In special, pattern recognition tools have been developed to recognize extreme topologies into the area of molecular modeling. A relevant issue in pattern recognition is the identification of instances of a problem that produce its most extreme cases. In the context of calculating the number of independent sets in a graph G, and particularly for graphs that represent the molecular structures of dendrimers, to recognize these extreme patterns allows us to identify dendrimer topologies that can demonstrate optimized behaviors with the same number of phenylenes.

The motivation for this work is to overcome the limitations of these general and structure-specific (e.g., branch and bound) methods by proposing an efficient algorithm that exploits the intrinsic hierarchical symmetry of the PPDs. The main contribution of this paper is, by applying our algorithmic proposal to compute $i(G)$, to recognize extremal topologies in dendrimer structures, demonstrating which dendrimer topologies yield extreme values (maximum and minimum) for the number of independent sets of the molecular dendrimer graph ($i(G)$), when a new phenylene is inserted into the dendrimer structure and enabling more accurate and scalable QSAR/QSPR analysis for this important class of macromolecules.

2 Notation and Preliminaries

In this article, we assume that $G = (V, E)$ is a connected undirected simple graph with a vertex set V and an edge set E. Two vertices v and w are called *adjacent* if there is an edge $\{v, w\} \in E$ connecting them. Sometimes, the shorthand notation vw is used to denote the edge $\{v, w\} \in E$

The *neighborhood* of a vertex $x \in V$, denoted $N(x)$, is defined as the set $N(x) = \{y \in V : \{x, y\} \in E\}$. The *closed neighborhood* of x, denoted $N[x]$, is given by $N(x) \cup \{x\}$. We denote the *cardinality* of a set A by $|A|$. The degree of a vertex x, denoted $\delta(x)$, is $|N(x)|$, and the maximum degree of G is $\Delta(G) = \max\{\delta(x) : x \in V\}$. A vertex v is called *pendant* if $\delta(v) = 1$, and an edge $e = \{x, y\}$ is *pendant* if either x or y is a pendant vertex.

A path between two vertices v and w, denoted as P_{vw} or simply as P_n, is a sequence of edges:

$$v_0v_1, v_1v_2, \ldots, v_{n-1}v_n.$$

such that $v = v_0$, $v_n = w$, and $v_k v_{k+1} \in E$ for $0 \leq k < n$. The length of the path is the number of edges in it. A simple path is a path in which all vertices $v_0, v_1, \ldots, v_n$ are distinct. A simple cycle is a simple non-empty path in which the first and last vertices are identical.

A connected graph is a graph with a path between any pair of vertices. An acyclic graph is a graph that does not contain cycles. Connected acyclic graphs are called trees, and it is straightforward to infer that a tree has a unique path connecting any pair of its vertices. We denote by P_{n-1}, C_n, and T_n the simple path, the simple cycle, and the tree, respectively, each containing n vertices.

A subset $S \subseteq V$ is called independent if, for every $u, v \in S$, we have $uv \notin E$. $I(G)$ denotes the set of all independent sets of G. For a vertex $v \in V$, we denote by $I_v(G) = \{S \in I(G) : v \in S\}$ the set of independent sets that contain v, and by $I_{-v}(G) = \{S \in I(G) : v \notin S\}$ those independent sets that do not contain v.

The corresponding counting problem on independent sets, denoted by $i(G)$, involves counting the number of independent sets of a graph G. Computing $i(G)$ is a $\#P$-complete problem for graphs G with $\Delta(G) \geq 3$ [9].

3 Counting Independent Sets on Basic Graph Topologies

In the field of chemistry, graph theory provides many useful tools, such as topological indices. The "charge" of a vertex $v \in V$ is defined as the pair $(\alpha_v, \beta_v)_G$, where $\alpha_v = |I_{-v}(G)|$ is the number of independent sets of G where v does not appear, and $\beta_v = |I_v(G)|$ represents the number of independent sets in G where v does appear.

For example, for a simple path P_n, the set of vertices is $V = \{v_1, v_2, \ldots, v_{n+1}\}$, and each pair of consecutive vertices v_i and v_{i+1} is connected by an edge $e_i = \{v_i, v_{i+1}\}$. We construct a family of subgraphs $f_i = \{G_i\}$, where each $G_i = (V_i, E_i)$ contains only the first i vertices of V.

Each vertex $v_j \in V_i$ has associated its charge $(\alpha_j, \beta_j)_{G_i}$, with $\alpha_j = |I_{-v_j}(G_i)|$ and $\beta_j = |I_{v_j}(G_i)|$. Thus, the number of independent sets in G_i, denoted $i(G_i)$, is given by $\alpha_j + \beta_j$. The first pair (α_1, β_1) is defined as $(1, 1)$ in subgraph $G_1 = \{v_1\}$, since $I(G_1) = \{\emptyset, \{v_1\}\}$. To calculate the next pair $(\alpha_{i+1}, \beta_{i+1})$ given (α_i, β_i), the edge $\{v_i, v_{i+1}\}$ and vertex v_{i+1} are visited, while the following Fibonacci recurrence is applied:

$$\alpha_{i+1} = \alpha_i + \beta_i, \quad \beta_{i+1} = \alpha_i. \tag{1}$$

This recurrence is known as the Fibonacci counting rule. By traversing P_n sequentially, the last pair (α_n, β_n) will be (F_{n+1}, F_n), where F_n is the n-th Fibonacci number. Then, the number of independent sets in P_n is $i(P_n) = F_{n+1} + F_n = F_{n+2}$.

The charge $(\alpha_i, \beta_i)_H$ of a vertex v_i can change depending on the subgraph H considered. To compute the number of independent sets along any path in a graph G, we use computing threads or simple threads. A thread is a sequence of pairs (α_i, β_i), $i = 0, \ldots, n$, used to calculate the number of independent sets in a path $P_n : v_0, v_1, \ldots, v_n$, where each pair (α_i, β_i) is associated with each vertex v_i, $i = 0, \ldots, n$ of the path.

3.1 Computing the Charges in a Tree Graph

Consider a tree graph; $T = (V, E)$ with a rooted vertex $v_r \in V$. In this structure, the vertices of degree one are the leaves. Meanwhile, every internal node has a degree greater than one. Assuming $n = |V(T)|$, we traverse T in a post-order search. We define a subtree $T_i = (V_i, E_i)$, with $V_i \subset V$ and $E_i \subset E$ $i = 1, \ldots, n$, formed by the i vertices that begin from one of the leaves in postorder traversing. Each vertex $v \in V_i$ is associated with a charge (α_i, β_i) defined as $\alpha_i = |I_{-v}(T_i)|$ and $\beta_i = |I_v(T_i)|$. Therefore, $i(T_i) = \alpha_i + \beta_i$.

For each leaf vertex v, we have $(\alpha_v, \beta_v) = (1, 1)$, since at the beginning of the counting, each leaf vertex defines the beginning of a path and $I(\{v\}) = \{\emptyset, \{v\}\}$. Any new pair $(\alpha_{i+1}, \beta_{i+1})$ is built from the previous one using the Fibonacci recurrence (1). For nodes $v_i \in V(T_n)$ with multiple children, the Hadamard product among $(\alpha_{i_j}, \beta_{i_j})$ for $j = 1, \ldots, k$ is computed to obtain (α_i, β_i). For example, the Hadamard product between the

two charges $(\alpha_{i_1}, \beta_{i_1})$ and $(\alpha_{i_2}, \beta_{i_2})$ is given by the following product rule:

$$(\alpha_{i_1}, \beta_{i_1}) \cdot (\alpha_{i_2}, \beta_{i_2}) = (\alpha_{i_1} \cdot \alpha_{i_2}, \beta_{i_1} \cdot \beta_{i_2}). \quad (2)$$

The algorithm for computing $i(T)$ is as follows: when traversing T in post-order, for a leaf node v, assign $(\alpha_v, \beta_v) = (1, 1)$; if v is the root, return $\alpha_v + \beta_v$; otherwise, for child nodes $u_1, u_2, \ldots, u_k$, calculate $\alpha_v = \prod_{j=1}^{k} \alpha_{uj}$ and $\beta_v = \prod_{j=1}^{k} \beta_{uj}$.

The time complexity for computing $i(T)$ comes for the time complexity of traversing T through a post-order search. Then, the computation of $i(T)$ has a time of $O(n)$, where n is the number of vertices in the tree.

3.2 Computing the Charges in a Cycle

Let $C_n = (V, E)$ be a simple cycle with $n = |V| = m = |E|$. In a simple cycle, each vertex has degree two. When the n-th vertex is adjacent to the first vertex, P_{n-1} forms a cycle C_n, and the number of subsets with no two adjacent elements corresponds to the n-th Lucas number, $i(C_n) = L_n = F_{n+1} + F_{n-1}$.

This relationship arises from decomposing the cycle C_n as $P_{n-1} \cup \{c_m\}$, where P_{n-1} is a path of n vertices and $c_m = \{v_n, v_1\}$ is the frond edge.

For each vertex $v_i \in V$, we associate (α_i, β_i) where $\alpha_i = |I_{-v_i}(G)|$ and $\beta_i = |I_{v_i}(G)|$. The first pair is $(1, 1)$ for the induced subgraph $G_1 = \{v_1\}$.

To eliminate conflicting sets where both v_1 and v_n are included, we use two threads: the main thread L_p computes $i(P_{n-1})$ while the secondary thread L_C computes the number of independent sets type: $\{S \in I(G) : v_1 \in S \wedge v_n \in S\}$. Starting with $(\alpha'_1, \beta'_1) = (0, 1)$, as it is illustrated in Figure 1. If only the main thread is active, then all new cycle request to build a new subordinated thread, in this way, only two threads are needed to compute the charges in a simple cycle. But in general, when any new cycle in a walking on the graph is found, the number of active threads has to be duplicated.

For example, expressing $i(C_n)$ in terms of Fibonacci numbers yields $(\alpha'_n, \beta'_n) = (F_{n-1}, F_{n-2})$, resulting in $|S \in I(G') : v_1 \in S \wedge v_n \in S| = 0 + \beta'_n = F_{n-2}$. Thus, the final computation gives $i(C_n) = F_{n+1} + F_{n-1}$, confirming the identity of the n-th Lucas number.

The subtracted rule is used to count independent sets on simple cycles, and it is performed when a frond edge is visited. The rule operates in two phases. First, when visiting vertex v of a frond edge $\{v, w\}$, each active thread associated with v duplicates, creating new subordinate threads with initial values $(0, \beta_v)$ pointing back to their master thread. Second, when the traversing reaches vertex w, then the subtracted rule is applied between the charges in the master and subordinated thread. Assuming for example, a charge (α_w, β_w) for w in the main thread, and a charge of $(\alpha_{vw}, \beta_{vw})$ for w in the subordinated thread, then in order to process the frond edge $\{v, w\}$, the charge for w is updated as:

$$(\alpha_w, \beta_w) = (\alpha_w, \beta_w - \beta_{vw}). \quad (3)$$

Furthermore, the control of the traversing is maintained in the vertex w since the vertex v has already visited. The application of the subtracted rule reduces the number of active threads by closing all subordinate threads with subindex vw. Figure 1 shows a phenylene and the application of the subtracted rule. The following steps describe the clockwise walk and the application of the counting subtracted rule.

In Figure 1, at the end of the clockwise walk, the main thread obtains as final charge $i(h_i) = 13 + 5 = 18$, which is the total number of independent sets of a phenylene. In a dendrimer, there are three types of edges: the edges that form a phenylene, one of which is recognized as a frond edge, and the connectivity edge that connects two different phenylenes.

In [6, 5] there are some applications of the previous tree counting rules; Fibonacci, product, and subtracted rule to compute the M-S index on benzenoid molecular graphs and on grid graphs. In this work, we introduce how to apply those counting rules for computing the M-S index on dendrimer molecular graphs. However, for grid and benzenoid graphs the resulting methods have an exponential time complexity. Meanwhile, our proposal has a linear time complexity for dendrimer molecules.

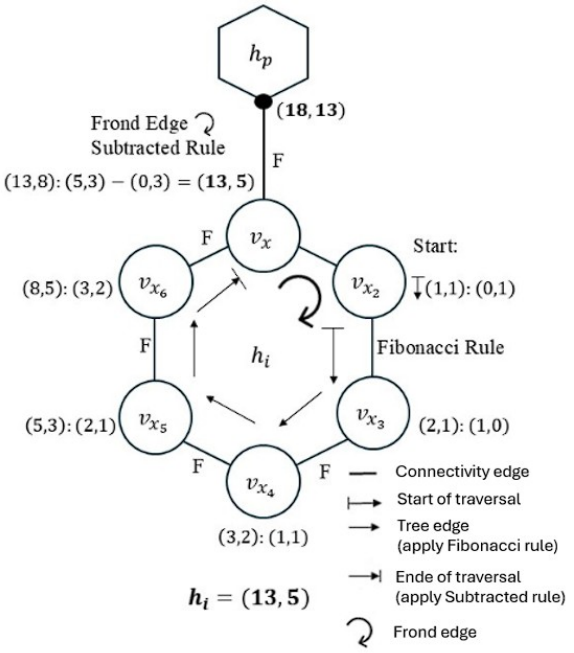

Fig. 1. Application of the subtracted rule on a phenylene

4 Graph Theory Applications: Case Studies on Maximum and Minimum Scenarios

In graph theory, Fibonacci numbers and their properties are used to model structures and relationships between nodes and edges. The table shown in Table 1 demonstrates the decomposition of integers n into sums of integer terms $a + b$, and their corresponding products $a \times b$, which have implications for combinations of paths and connections within a graph.

The integers in the table are decomposed into sums of two terms. This decomposition has applications in graph paths, where each sum combination, such as (a, b), can be interpreted as a connection between two nodes via an edge, where the values of a and b represent distances or weights in the graph. For example, the number 6 is decomposed into sum combinations $(5, 1)$, $(4, 2)$, and $(3, 3)$, indicating that there are different paths between nodes summing to 6 as the total distance or weight.

Next, multiplying each pair of summands $a \times b$ results in the products, which are crucial for modeling weighted connections in the graph. The product between the summands a and b represents the interaction or combined weight of two components in the graph. In other words, these products can model how nodes connected by an edge interact based on their values. For instance, for $n = 6$, the multiplications $5 \times 1 = 5$, $4 \times 2 = 8$, and $3 \times 3 = 9$ represent different "weights" of the possible connections between nodes in a graph, depending on the combinations of distances or relationships between them.

4.1 Graph Visualization of Decompositions

In graphs, the sum decompositions correspond to the possible paths between two nodes, while the products represent the total cost of traversing those paths. Therefore, Table 1 and Figure 1 help understand how the Fibonacci products (and their decompositions) allow the calculation of the total weight of connections and paths in a graph, which can be optimized or analyzed to find the best route or connection between nodes.

In conclusion, the decomposition into sums and products of numbers in the table not only has a pure mathematical interpretation, but it can also be directly applied to graph theory. By understanding how sums and products interact within graphs, we can model and analyze connections between nodes more accurately, optimizing paths or studying the structure of relationships within complex networks.

Following Table 1, Figure 1 illustrates a graphical representation of the decomposed sums and products within a path graph. Each vertex in the graph is labeled with the corresponding pairs from the decomposition table, and the edges between them represent the calculated products of the sums.

For instance, the path graph starts with a new vertex v, followed by a series of nodes with weights determined by the products of the sum pairs, such as $5 \times 3 = 15$ and $8 \times 5 = 40$, showing how the interactions between nodes (edges) reflect the weighted distances or costs within the graph structure.

Table 1. Decomposition into sums and products of integers with Min and Max indicators

Number n	**Sum Combinations**	**Product Results**		
		Min		Max
5	(4,1), (3,2)	4*1 = 4,		3*2 = 6
6	(5,1), (4,2), (3,3)	5*1 = 5,	4*2 = 8,	3*3 = 9
7	(6,1), (5,2), (4,3)	6*1 = 6,	5*2 = 10,	4*3 = 12
8	(7,1), (6,2), (5,3), (4,4)	7*1 = 7,	6*2 = 12, 5*3 = 15,	4*4 = 16
9	(8,1), (7,2), (6,3), (5,4)	8*1 = 8,	7*2 = 14, 6*3 = 18,	5*4 = 20
10	(9,1), (8,2), (7,3), (6,4)	9*1 = 9,	8*2 = 16, 7*3 = 21,	6*4 = 24
11	(10,1), (9,2), (8,3), (7,4)	10*1 = 10,	9*2 = 18, 8*3 = 24,	7*4 = 28
12	(11,1), (10,2), (9,3), (8,4)	11*1 = 11,	10*2 = 20, 9*3 = 27,	8*4 = 32
13	(12,1), (11,2), (10,3), (9,4)	12*1 = 12,	11*2 = 22, 10*3 = 30,	9*4 = 36
14	(13,1), (12,2), (11,3), (10,4)	13*1 = 13,	12*2 = 24, 11*3 = 33,	10*4 = 40

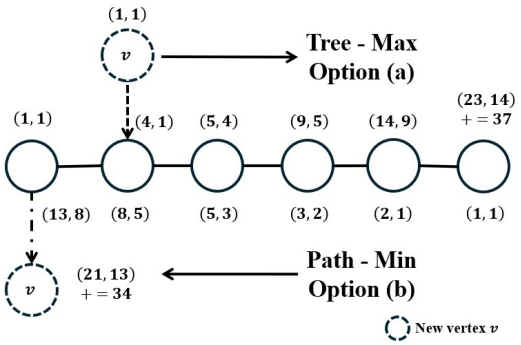

Fig. 2. Graph visualization of sum and product decompositions in a path graph, where edges represent the weighted connections between nodes

In next section, we will show how symmetric graph serves as a foundational example of regular graph structures, where the symmetry simplifies the analysis of graph properties such as connectivity, optimization, and pathfinding.

4.2 Case 1: Regular (Symmetric) Graphs

Figure 3 shows a symmetric graph that contains two distinct trees, which demonstrate the regularity and symmetry inherent in the structure. The two trees in the graph can be described as follows:

— Tree 1: This tree has 4 vertices and is fully connected with each vertex having exactly 2 edges. The structure of this tree mirrors a cycle, where each vertex is connected symmetrically to its neighbors. The product of the sums for these vertices, as explained earlier, represents the weights of the edges. This is a simple cyclic structure often seen in symmetric graphs, where each connection is equally weighted.

— Tree 2: This second tree also consists of 4 vertices, but it is structured in a linear fashion rather than cyclic. The vertices are connected in sequence, and the graph displays a regular branching of nodes, where each vertex is directly connected to other two vertices. This tree, like the first, reflects symmetry in its connections, but its layout emphasizes a linear path with evenly distributed weights across all edges.

Both trees in the graph highlight how symmetry can be applied to model paths in a graph, with each edge representing the interaction between connected nodes. The regularity of the graph ensures that each vertex is connected in a balanced and predictable manner, allowing for optimization and analysis of potential paths.

4.3 Case 2: Irregular (Asymmetric) Graphs

In contrast to symmetric graphs, Figure 4 presents an example of an irregular (asymmetric) graph, where the structure and connections between the vertices do not follow any symmetric pattern. The lack of symmetry introduces more complexity

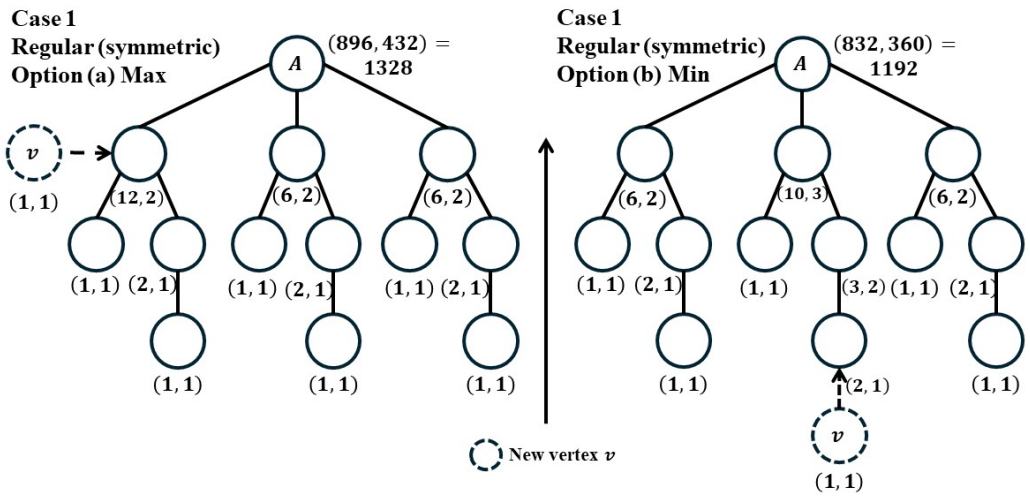

Fig. 3. Symmetric graph showing two trees with weighted edges. The symmetry is maintained across the entire structure, and each edge weight is derived from the product of sums

in the analysis, as the graph does not exhibit predictable patterns of connection. The graph's structure changes significantly depending on how new vertices are added.

— Tree 1: The first tree in the asymmetric graph is made up of 4 vertices arranged in a linear fashion. The edges between these vertices are not uniform in weight, and each vertex has a different degree of connectivity with its neighbors. This tree is irregular because the edges between the nodes are not symmetric; some nodes may have more or fewer connections than others, resulting in an unbalanced structure.

— Tree 2: The second tree consists of a more complex arrangement where the vertices are connected in a non-linear fashion, resulting in an unbalanced branching pattern. Adding a new vertex v to this tree can create further asymmetry. For example, if the new vertex is connected to a vertex that already has a large number of neighbors, it could increase the overall connectivity of the tree, making it denser. This tree shows how irregular graphs can have varying edge weights and node degrees, which contribute to their complex, unpredictable nature.

The placement of vertex v in an asymmetric graph can cause significant changes in the graph's structure, especially in irregular graphs. These graphs do not follow a simple pattern, and the new vertex can either increase the irregularity of the graph or introduce a more complex pattern of connections.

This example highlights the fundamental differences between regular and irregular graphs. While regular graphs maintain a predictable structure, irregular graphs can be more difficult to analyze due to their lack of symmetry, leading to a wide range of possible configurations and edge weights.

5 Methodology

There are various methods to compute $i(G)$ when G is part of a restricted set of simple

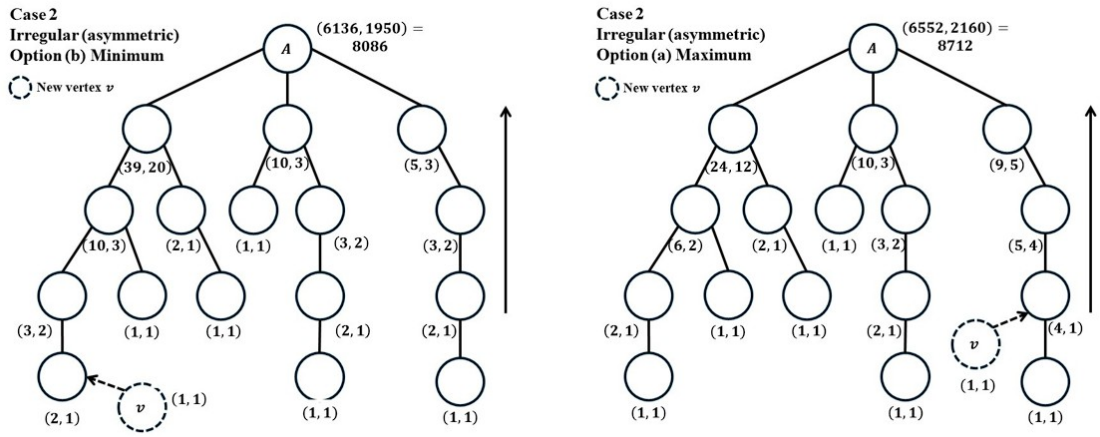

Fig. 4. Irregular (asymmetric) graph showing weighted edges, where the structure and connections are not symmetric. The addition of vertex v further complicates the graph

graph topologies [5, 2]. For example, in [8] an algorithm for computing $i(G)$ is presented, being G a hexagonal mesh, and where the general decomposition vertex rule $i(G) = i(G - v) + i(G–N[v])$ is applied, with the cost of obtaining an exponential time algorithm. However, in this study, the topology of dendrimer graphs eliminates the need for this decomposition vertex rule.

Furthermore, the tree-like structure of a polyphenylene dendrimer G allows for the development of a polynomial time method for computing $i(G)$.

In this proposal, the primary focus is on calculating the charge (α_v, β_v) for each vertex v in the graph during a post-order search on G, assuming that G models a dendrimic compound. Thus, the charge is an auxiliary temporal pair to determine the number of independent sets in G in the position of v.

In our method, different counting rules are performed for computing the charge of vertex v, primarily depending on the topology of the subgraph $N[v]$ at the time v is visited during the post-order search on G. The main counting rules to process any subgraph are based in recognizing if the visited edge is a tree edge or a frond edge.

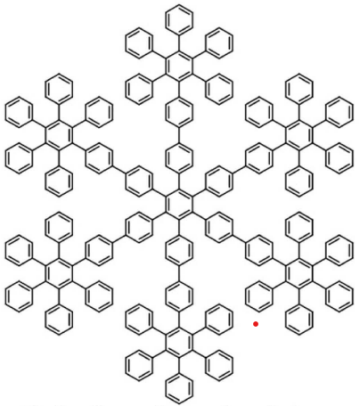

Fig. 5. A Polyphenylene Dendrimer [18]

5.1 Computing the Merrifield-Simmons Index on Polyphenylene Dendrimers

A polyphenylene dendrimer features a tree-like structure in which conventional nodes are replaced by phenylene hexagons. Thus, each vertex of a simple tree is expanded into a hexagonal cycle, producing a highly branched aromatic macromolecule.

The topology of a polyphenylene dendrimer is determined by a central hexagonal core from which

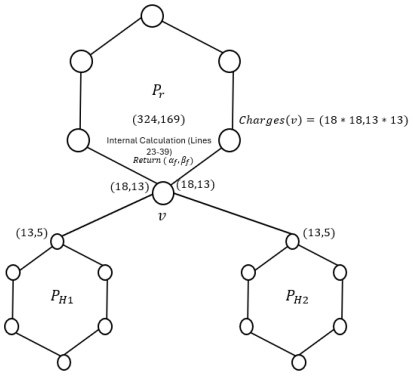

Fig. 6. Execution trace of Algorithm 1 on a basic dendrimer, demonstrating charge calculation and propagation

branched phenylene units radiate. This hierarchical arrangement, organized in generations, increases structural complexity, surface area, and potential for molecular interactions. Connectivity between phenylene units is maintained by covalent bonds, and the inherent hexagonal symmetry ensures a uniform spatial distribution, properties that are essential for reactivity, self-assembly, and applications in nanotechnology.

Topological indices, such as the Merrifield-Simmons (M-S) index, capture the branching complexity and connectivity of such dendrimers, providing a quantitative tool for comparison and design.

Our algorithm to compute the M-S index of a polyphenylene dendrimer is based on a post-order search strategy. This classical traversal guarantees that all child phenylenes are processed before their parent phenylene, enabling the systematic transfer of local information (charges) towards the root. A global description of this procedure is presented in Algorithm 1.

To illustrate the execution of Algorithm 1, consider the simple dendrimer in Figure 6, composed of a root phenylene (P_{r}) connected to two leaf phenylenes (P_{H1} and P_{H2}) at the same vertex. The execution begins in the `ComputeMSIndex` function, which calls `ComputeCharge` on the root P_{r} (line 3). This main call first invokes `ComputeCharge` on its children (line 13).

Step 1: Processing Leaves ($P_{\text{H1}}, P_{\text{H2}}$). As P_{H1} is a leaf, its `ForAll` loop (lines 12–21) is skipped. The function executes the "Calculation" phase (lines 23–39) and returns its fundamental charge, $(13, 5)$, as shown in the figure. The same process occurs for P_{H2}.

Step 2: Processing Root (P_{r}). Execution returns to P_{r} to execute its `ForAll` loop (lines 12–21). It first processes the $(13, 5)$ charge from P_{H1}. The Fibonacci Rule is applied (line 14), converting it to $t_{\text{charge}} = (18, 13)$, as visualized in the figure. It then processes the $(13, 5)$ charge from P_{H2}, which is also transferred as $(18, 13)$. Since both charges arrive at the same vertex v, the Product Rule (line 17) is activated. The accumulated charge in $Charges[v]$ becomes $(18 \times 18, 13 \times 13) = (324, 169)$.

Step 3: Final Result. After accumulating the $(324, 169)$ charge, P_{r} executes its own "Internal Calculation" phase (lines 23–39), as noted in the figure. The resulting final charge $charge_f$ is returned to `ComputeMSIndex` (line 3), which finally sums its components $(\alpha_f + \beta_f)$ to obtain the total M-S index (line 4).

For the time-complexity analysis of our proposal, let us review each step to be performed in the algorithm. In step 1, even though a constant charge of (13,5) is transferred from a leaf phenylene to its parent phenylene, the process of identifying the connectivity edge takes linear time on the number of edges of each phenylene parent.

The clockwise traversal conducted on each phenylene, along with the post-order search performed on the tree structure of the polyphenylene dendrimer graph, both exhibit linear time complexity concerning the number of edges.

The charges for the vertices of a phenylene are computed using only two computing threads. As a result, the time complexity of traversing the vertices of a phenylene is multiplied by a factor of two. Thus, this traversal maintains a linear time complexity, scaled by a constant factor of two, and remains within linear time complexity. As a result, Algorithm 1 has a linear time complexity with respect to the number of edges in the polyphenylene dendrimer graph.

Algorithm 1 A method for computing the M-S index on dendrimers

Require: Dendrimer D
Ensure: The Merrifield-Simmons Index

function COMPUTEMSINDEX(D)
  $P_{\text{root}} \leftarrow$ GetRootPhenylene(D)
  $(\alpha_f, \beta_f) \leftarrow$ ComputeCharge(P_{root})
  return $\alpha_f + \beta_f$
end function

function COMPUTECHARGE(P)
  // Post-order: Accumulate from children
  for all vertex v **in** P **do**
    $Charges[v] \leftarrow (1, 1)$
  end for

  for all P_{child} connected **to** P **at** v **do**
    $(a_c, b_c) \leftarrow$ ComputeCharge(P_{child})
    $t_{\text{charge}} \leftarrow (a_c + b_c, a_c)$ ▷ *Fibonacci rule*
    $(a_{\text{curr}}, b_{\text{curr}}) \leftarrow Charges[v]$
    $Charges[v] \leftarrow (a_{\text{curr}} \times t_{\text{charge}}.\alpha,$
      $b_{\text{curr}} \times t_{\text{charge}}.\beta)$ ▷ *Product rule*
  end for

  // Calculation: Traverse the cycle
  $v_{\text{start}} \leftarrow$ GetStartVertex(P)
  $v_{\text{frond}} \leftarrow$ GetPrevVertex(v_{start})
  $Lp[v_{\text{start}}] \leftarrow Charges[v_{\text{start}}]$
  $Ls[v_{\text{start}}] \leftarrow (0, Lp[v_{\text{start}}].\beta)$
  $v_{\text{curr}} \leftarrow$ GetNextVertex(v_{start})

  while $v_{\text{curr}} \neq v_{\text{start}}$ **do**
    $v_{\text{prev}} \leftarrow$ GetPrevVertex(v_{curr})
    $Lp[v_{\text{curr}}] \leftarrow (Lp[v_{\text{prev}}].\alpha + Lp[v_{\text{prev}}].\beta,$
      $Lp[v_{\text{prev}}].\alpha)$
    $Ls[v_{\text{curr}}] \leftarrow (Ls[v_{\text{prev}}].\alpha + Ls[v_{\text{prev}}].\beta,$
      $Ls[v_{\text{prev}}].\alpha)$
    $v_{\text{curr}} \leftarrow$ GetNextVertex(v_{curr})
  end while

  $charge_f \leftarrow (Lp[v_{\text{frond}}].\alpha,$
    $Lp[v_{\text{frond}}].\beta - Ls[v_{\text{start}}].\beta)$
  return $charge_f$ ▷ *Subtracted rule*
end function

6 Conclusion

The study of topological indices, particularly the Merrifield-Simmons (M-S) index, has emerged as a valuable tool in the analysis and characterization of molecular graphs. The M-S index represents the number of independent vertex sets within a molecular graph that provides crucial insights into a molecule's "disconnectedness" which in turn reflects its stability, potential for structural optimization, and suitability for applications requiring controlled interactions or encapsulation.

Within the scope of our research, our algorithmic proposal is a novel method for computing the M-S index on dendrimers. It is also an efficient proposal that has a linear time complexity on the number of edges involved in such dendrimer compounds.

Moreover, the application of advanced computational methods, including Fibonacci recurrence-based algorithms for counting independent sets, has significantly reduced the complexity of calculating the M-S index, thereby broadening its applicability to larger and more complex molecular structures. This research has shown the extremal topologies associated with dendrimer tree structures. We identify, with respect to the M-S index, the maximum as the minimum topologies when a new phenylene is added to an existing dendrimer, although it has an irregular configuration. This integration would not only facilitate the analysis of large molecular databases but also accelerate the discovery and design of new compounds with optimized properties.

Our study will allow to extend the scope of the method for computing the M-S index to consider variants of dendrimers, for example when the Diels-Alder cycloaddition process is considered, where a set of phenylenes with shared bonds is formed in the core of the dendrimer, which is one of the future works to be investigated.

References

1. **Ahmadi, M., Dastkhezr, H. A. (2014).** An algorithm for computing the merrifield–simmons index. MATCH Commun. Math. Comput. Chem, Vol. 71, pp. 355–359.

2. **Ahmadi, M., Seif, M. (2010).** The merrifield-simmons index of an infinite class of dendrimers. Digest J Nano mater Bios, Vol. 5, pp. 335–338.

3. **Bokhary, S. A. U. H., Imran, M., Manzoor, S. (2016).** On molecular topological properties of dendrimers. Canadian Journal of Chemistry, Vol. 94, No. 2, pp. 120–125.

4. **Chowdhury, S., Toth, I., Stephenson, R. J. (2022).** Dendrimers in vaccine delivery: Recent progress and advances. Biomaterials, Vol. 280, pp. 121303.

5. **De Ita Luna, G., Bello, P., Marcial-Romero, R. (2024).** Counting rules for computing the number of independent sets of a grid graph. Mathematics, Vol. 12, No. 6, pp. 922.

6. **De Ita Luna, G., Bello López, P., Contreras, M. (2023).** A method for computing the merrifield-simmons index on benzenoid systems. MATCH Communications in Mathematical and in Computer Chemistry, Vol. 89, No. 1, pp. 245–270. DOI: 10.46793/match.89-1.245I.

7. **Estrada, E., Uriarte (2001).** Recent advances on the role of topological indices in drug discovery research. Current Medicinal Chemistry, Vol. 8, No. 13, pp. 1573–1588. DOI: 10.2174/0929867013371923.

8. **González-Vázquez, H., López-Ramírez, C., Bello-López, P., De Ita Luna, G. (2024).** Branching and pruning algorithm for computing the merrifield-simmons index on polygonal grids. Computación y Sistemas, Vol. 28, No. 4, pp. 2171–2182.

9. **Greenhill, C. (2000).** The complexity of counting colourings and independent sets in sparse graphs and hypergraphs. computational complexity, Vol. 9, pp. 52–72.

10. **Hammer, B. A., Müllen, K. (2018).** Expanding the limits of synthetic macromolecular chemistry through polyphenylene dendrimers. Journal of Nanoparticle Research, Vol. 20, No. 10, pp. 262.

11. **Ilic, A., Milovan Ilic, M. (2017).** On some algorithms for computing topological indices of chemical graphs. MATCH Communications in Mathematical and in Computer Chemistry, Vol. 78, pp. 665–674.

12. **Li, X., Zhao, H., Gutman, I. (2005).** On the merrifield-simmons index of trees. MATCH Commun. Math. Comput. Chem, Vol. 54, No. 2, pp. 389–402.

13. **Lyu, Z., Ding, L., Huang, A.-T., Kao, C.-L., Peng, L. (2019).** Poly (amidoamine) dendrimers: Covalent and supramolecular synthesis. Materials Today Chemistry, Vol. 13, pp. 34–48.

14. **Merrifield, R. E., Simmons, H. E. (1980).** The structures of molecular topological spaces. Theoretica chimica acta, Vol. 55, pp. 55–75.

15. **Merrifield, R. E., Simmons, H. E. (1981).** Enumeration of structure-sensitive graphical subsets: Theory. Proceedings of the National Academy of Sciences, Vol. 78, No. 2, pp. 692–695.

16. **Movahedi, F., Akhbari, M. H., Hasni, R. (2024).** Computing the hosoya index of some nanostar dendrimers. Malaysian Journal of Fundamental and Applied Sciences, Vol. 20, No. 3, pp. 588–596.

17. **Oz, M. S., Cangul, I. N. (2022).** Computing the merrifield-simmons indices of benzenoid chains and double benzenoid chains. Journal of Applied Mathematics and Computing, Vol. 68, No. 5, pp. 3263–3293.

18. **Pérez-Ferreiro, M., M. Abelairas, A., Criado, A., Gómez, I. J., Mosquera, J. (2023).** Dendrimers: exploring their wide structural variety and applications. Polymers, Vol. 15, No. 22, pp. 4369.

Article received on 27/08/2025; accepted on 25/11/2025.
**Corresponding author is Michelle Guerra-Marín.*