MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **89** (2023) 245–270
ISSN: 0340–6253
doi: 10.46793/match.89-1.245I

A Method for Computing the Merrifield–Simmons Index on Benzenoid Systems

Guillermo De Ita[*], **Pedro Bello**, **Meliza Contreras**

Benemérita Universidad Autónoma de Puebla
Edif. CCO1, 14 Sur and San Baltazar, C.P. 72592, Ciudad Universitaria, Puebla, México
deitaluna63@gmail.com, pedro.bello@correo.buap.mx,
meliza.contreras@correo.buap.mx

(Received June 16, 2022)

Abstract

We present a method for computing the Merrifield-Simmons index on some basic graphs. For example, our proposal works on paths, simple cycles, trees, and we show that the method can be extended to process benzenoid systems $H_{r,t}$ with a total of $r \cdot t$ hexagons. In the case of benzenoid systems, our method consists of performing a linear-time Hamiltonian walking on an isomorphic hexagonal grid from $H_{r,t}$, at the same time that the number of independent sets are being computed incrementally.

Our method improves the asymptotic behavior of the time-complexity with respect to the transfer matrix method, which is the classic method for computing the Merrifield-Simmons index on grid graphs, even for benzenoid systems.

1 Introduction

Merrifield and Simmons showed the correlation between the number of independent sets of G, denoted by $i(G)$, and the boiling points of the

[*]Corresponding author.

molecular graph represented by G [18]. This is one of the main reasons why the number of independent sets of a graph G, in the area of mathematical chemistry, is called the Merrifield-Simmons index (M-S) of G. However, in the area of graph theory, $i(G)$ is called the Fibonacci number of G.

The Merrifield-Simmons index is a significant topological index of the structural chemistry of the molecular graph G [8,25]. A topological index is a map from the set of chemical compounds represented by molecular graphs to the set of real numbers. Many topological indices are closely correlated with some physico-chemical characteristics of the underlying compounds. The Merrifield-Simmons index is one of the topological indices, whose mathematical properties have been studied in some detail in [17,18,22,25]. The M-S index and the Hosoya index are some of the most popular topological indices in chemistry.

Calkin [3] calculates the number of independent sets on grid graphs using the *transfer matrix method.* Euler [11] showed different generating functions for counting maximal independent sets for different sizes of a grid graph. In [14] is presented an extension of the *transfer matrix method* to count the number of satisfying assignments of Boolean 2-CNF.

There is a large volume of literature devoted to count structures in a grid graph, e.g., spanning trees, Hamiltonian cycles, independent sets, acyclic orientations, k-coloring, and so on [3,11,12,14]. Applications of the counting objects on grids also include for instance tiling and efficient coding schemes in data storage [23].

The recognition of structural patterns lying on a graph G has been helpful to design efficient algorithms for computing $i(G)$. For example, the linear-time Okamoto's algorithm [19] computes $i(G)$, when G is a chordal graph and, when the decomposition of G in its clique tree gives the possibility of applying dynamic programming in an efficient way. Another case, is the Zhao's algorithm for computing $i(G)$ on regular graphs [26]. In [7], it is shown a method based on the use of macros to compute $i(T_n)$ when T_n is a polygonal tree graph. However, for some kind of graphical patterns G, there is not an efficient known method to compute $i(G)$.

The transfer matrix method has been adapted for computing the M-S index on different class of benzenoid systems [16,27]. However, this class of structures has required the application of multi-step transfer matrices for computing the Merrifield-Simmons index by increasing in exponential way the complexity time of the method.

In the work from Oz et al. [20, 21] a Merrifield-Simmons vector defined on a path is introduced in order to compute the M-S index of a benzenoid chain or a double benzenoid chain via the product of six different matrices and the M-S vector. Oz et al. [20] claims that it does not exist a practical way for computing the Merrifield-Simmons index of any double hexagonal (benzenoid) chain. Thus, they present an extensive matrix numerical method for this task. Contrary to that affirmation, we present here a practical graph-based method to compute the M-S index on any regular benzenoid system, including a chain, double chain, or for any number of rows (and columns) of a regular benzenoid system.

We introduce a novel method for computing the Merrifield-Simmons index on benzenoid systems that compared to apply the transfer matrix method, reduces significantly the complexity time of the computation.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph with a set of vertices V and a set of edges E. It is assumed that G is a simple graph, that is, it does not have loops nor parallel edges. Sometimes, we denote an edge $\{u, v\} \in E$ in abbreviated form as uv. The *neighborhood* of $x \in V$ is the set $N(x) = \{y \in V : xy \in E\}$, and its *closed neighborhood* is $N(x) \cup \{x\}$, which is denoted by $N[x]$. The degree of a vertex x in the graph G, denoted by $\delta_G(x)$, is $|N(x)|$. The degree of the graph G is $\Delta(G) = \max\{\delta_G(x) : x \in V\}$.

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

A connected graph is a graph where for any pair of vertices there is a path connecting them. An acyclic graph is a graph that does not contain cycles. The connected acyclic graphs are called *trees*, and it is not difficult to infer that in a tree there is a unique path connecting any two pair of its vertices. We denote by P_{n-1}, C_n, and T_n to the simple path, the simple cycle, and the tree, respectively, all of them containing n vertices.

A subset $S \subseteq V$ is called independent, when every $u, v \in S$ implies that $uv \notin E$. $I(G)$ denotes the set of all independent sets of G. Let $v \in V(G)$, we denote as $I_v(G) = \{S \in I(G) : v \in S\}$ and $I_{-v}(G) = \{S \in I(G) : v \notin S\}$. The corresponding counting problem on independent sets, denoted by $i(G)$, consists of counting the number of independent sets of a graph G. Computing $i(G)$ is a $\sharp$P-complete problem for graphs G, where $\Delta(G) \geq 3$. Let $G = (V, E)$ be a molecular graph. We denote $i(G, k)$ as the number of ways in which k mutually independent vertices can be selected in G. By definition, $i(G, 0) = 1$ for all graphs, and $i(G, 1) = |V(G)|$. Furthermore, $i(G) = \sum_{k \geq 0} n(G, k)$ is the *Merrifield-Simmons index* (M-S) of G, which is the exact number of independent sets of G.

Counting problems are not only mathematically interesting; they also arise in many applications. Regarding hard counting problems, the computation of the number of independent sets of a graph G has been a key in determining the frontier between efficient counting and intractable counting algorithms. Vadhan [24] showed that counting the number of independent sets in graphs of maximum degree 4 is #P-complete. Greenhill [13] refined the previous result showing that counting the number of independent sets on graphs of degree 3 or on 3-regular graphs is also #P-complete.

3 Counting independent sets on basic graphs

Given a connected graph $G = (V, E)$, we call *the sign of a vertex v in G* to the pair $(\alpha_v, \beta_v)_G$, where $\alpha_v = |I_{-v}(G)|$, which means that α_i is the number of subsets in $I(G)$ where v_i does not appear. Meanwhile, $\beta_v = |I_v(G)|$ conveys the number of subsets in $I(G)$, where v_i appears.

Let $P_n = G = (V, E)$ be a simple path, i.e. $V = \{1, 2, ..., n, n+1\}$ and there exists an edge $e_i = \{i, i+1\}, i = 1, \ldots, n$, for each pair of sequential vertices. We build the family $f_i = \{G_i\}, i = 1, \ldots, n+1$ where $G_i = (V_i, E_i)$ is the induced graph of G formed by just the first i vertices of V.

We associate to each vertex $v_i \in V(G_i)$ the pair $(\alpha_i, \beta_i)_{G_i}$, where $\alpha_i = |I_{-v_i}(G_i)|$ and $\beta_i = |I_{v_i}(G_i)|$. Thus, $i(G_i) = \alpha_i + \beta_i$.

The first pair (α_1, β_1) is $(1, 1)$ since on the induced subgraph $G_1 = \{v_1\}$, $I(G_1) = \{\emptyset, \{v_1\}\}$. If we know the value of (α_i, β_i) for any $i < n$, and as the next induced subgraph G_{i+1} is built from G_i adding the vertex v_{i+1} and the edge $\{v_i, v_{i+1}\}$, it is not hard to see that the pair $(\alpha_{i+1}, \beta_{i+1})$ is built from (α_i, β_i) by applying the following Fibonacci recurrence equation:

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

The previous rule is known as the application of a Fibonacci recurrence, and it will be denoted by the symbol $\rightarrow$. The series $(\alpha_i$,$\beta_i)$ built from recurrence (1), lead to compute $i(G_i), i = 1, \ldots, n+1$. Then, the computation of $i(G)$ is based on the incremental calculation of $i(G_i), i = 1, \ldots, n+1$. Traversing the path P_n in linear way, the last pair (α_n, β_n) will be (F_{n+1}, F_n) where F_n is the $nth-$Fibonacci number. Then, $i(P_n) = F_{n+1} + F_n = F_{n+2}$.

Notice that (α_i, β_i) represents the values $(|I_{-v_i}(P_i)|, |I_{v_i}(P_i)|), i < n$, such pairs will be denoted as $(\alpha_i, \beta_i)_{P_i}$ with $i < n$. We call $(\alpha_i, \beta_i)_H$ as the *sign* of the vertex v_i on the subgraph H. Notice that the sign of v_i can change according to the subgraph H that is considered.

In order to process the number of independent sets on any path in a graph G, we will use *computing threads* or just *threads*. A computing thread is a sequence of pairs $(\alpha_i, \beta_i), i = 0, \ldots, n$ used for computing the number of independent sets on a path $P_n : v_0, v_1, \ldots, v_n$, and where each (α_i, β_i) is associated to each vertex $v_i, i = 0, \ldots, n$ of the path.

3.1 Counting independent sets on trees

Let $T = (V, E)$ be a tree rooted at a vertex $v_r \in V$. The vertices in a tree with degree equal to one are called leaves or pendant nodes, while the non roots vertices of degree greater than one are called internal nodes of the tree. We traverse T in post-order. Let $\mathcal{F}_i = \{T_i\}, i = 1, \ldots, n$, where $\mathcal{F}_i$ is a family of induced subgraphs. Furthermore, $T_i = (V_i, E_i), V_i \subset V, E_i \subset E$, and each V_i is built as $V_i = \{v_1, \dots v_i\}$ of V. We associate to each vertex $v_i \in V$ its sign (α_i, β_i) with $\alpha_i = |I_{-v_i}(T_i)|$, $\beta_i = |I_{v_i}(T_i)|$. And therefore, $i(T_i) = \alpha_i + \beta_i$.

For all pendant vertex v of T, (α_v, β_v) is $(1, 1)$ since for the induced subgraph $T_v = \{v\}$, $I(T_v) = \{\emptyset, \{v\}\}$. Any new pair $(\alpha_{i+1}, \beta_{i+1})$ is built from the previous one by the application of a Fibonacci recurrence (1).

When a node $v_i \in V(T_n)$ has more than one child, then the Hadamard product among the $(\alpha_{i_j}, \beta_{i_j}), j = 1, \dots, k$ is formed in order to obtain (α_i, β_i). The following algorithm shows how to compute $i(T)$ for a tree T.

Algorithm 1 Linear_Tree(T)

```
Require: A tree T
Ensure: i(T)
  Traversing $T$ in post-order, and when a node $v \in T$ is left, assign:
  if $v$ is a leaf node in $T$ then
    $(\alpha_v, \beta_v) = (1, 1)$
  else if $v$ is the root node of $T$ then
    return $\alpha_v + \beta_v$
  else if $u_1, u_2, ..., u_k$ are the child nodes of $v$, as we have already visited
  all child nodes, then each pair $(\alpha_{u_j}, \beta_{u_j})$ $j = 1, ..., k$ has been determined
  based on recurrence (1) then
    Let $\alpha_v = \prod_{j=1}^{k} \alpha_{v_j}$ and $\beta_v = \prod_{j=1}^{k} \beta_{v_j}$.
  end if
```

The computation of $i(T)$ is done while T is traversing in post-order by the algorithm 1. Algorithm 1 returns the number of independent sets of a rooted tree T in time of order $O(n + m)$, which is the necessary time for traversing T in post-order.

3.2 Counting independent sets on cycles

Let $C_n = (V, E)$ be a simple cycle, $|V| = n = |E| = m$, i.e. every vertex in C_n has degree two. If the n-th element in $\{1, 2, ..., n\}$ is adjacent to the first vertex, then P_{n-1} turns into a simple cycle C_n, and the number of subsets (including the empty set) with no two adjacent elements is characterized by the n-th Lucas number. Therefore, $i(C_n) = L_n = F_{n+1} + F_{n-1}$ [22].

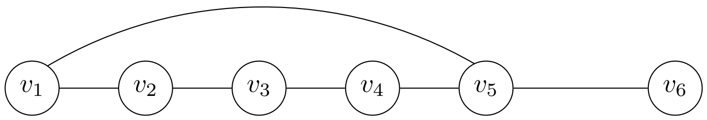
This last equality comes from decompose the cycle C_n as: $P_{n-1} \cup \{c_m\}$, where $P_{n-1} = (V, E')$, $E' = \{c_1, ..., c_{m-1}\}$. P_{n-1} is a path of n vertices, and $c_m = \{v_n, v_1\}$ is called the *back edge* of the cycle. We denote by $\hookrightarrow$ to the back edge that is processed by the counting procedure of $i(C_n)$.

Let $\mathcal{F}_i = \{G_i\}, i = 1, \ldots, n$ be a family graph, where $G_i = (V_i, E_i)$ is the induced graph of P_n formed by just the first i vertices of V. We consider the association of the sign (α_i, β_i) to each vertex $v_i \in V$, where $\alpha_i = |I_{-v_i}(G)|$ and $\beta_i = |I_{v_i}(G)|$. Therefore, $i(G_i) = \alpha_i + \beta_i$.

The first pair (α_1, β_1) is $(1, 1)$ since for the induced subgraph $G_1 = \{v_1\}$, $I(G_1) = \{\emptyset, \{v_1\}\}$, and every new pair $(\alpha_{i+1}, \beta_{i+1})$ is built from the previous one by the application of a Fibonacci recurrence (1).

Note that every independent set in G is an independent set in C_n, except for the sets $S \in I(G)$ where $v_1 \in S$ and $v_m \in S$. In order to eliminate those conflicting sets, we use two computing threads, one is the main thread L_p used to compute $i(P_{n-1})$. The other secondary thread, denoted by L_C, is used to compute $|\{S \in I(G) : v_1 \in S \wedge v_n \in S\}|$. The secondary thread begins with $(\alpha'_1, \beta'_1) = (0, 1)$, in order to consider only the independent sets of $I(G)$ where v_1 appears.

By expressing the computation of $i(C_n)$ in terms of Fibonacci numbers, we have $(\alpha'_1,\beta'_1) = (0, 1) = (F_0, F_1) \to (\alpha'_2, \beta'_2) = (1, 0) = (F_1, F_0) \to (\alpha'_3, \beta'_3) = (1, 1) = (F_2, F_1), \ldots, (\alpha'_n, \beta'_n) = (F_{n-1}, F_{n-2})$, and the value for the final pair is $(0, F_{n-2})$. Therefore, $|\{S \in I(G') : v_1 \in S \wedge v_n \in S\}| = 0 + \beta'_n = F_{n-2}$. Then, the last pair associated to the computation of $i(C_n)$ is $(F_{n+1}, F_n - F_{n-2}) = (F_{n+1}, F_{n-1})$. Then, $i(C_n) = F_{n+1} + F_{n-1}$, obtaining a well-known identity, the n-th Lucas number.



$L_p(\alpha_i, \beta_i) : (1, 1) \to (2, 1) \to (3, 2) \to (5, 3) \hookrightarrow (8, 5) - (0, 2) = (8, 3) \to (11, 8)$
$L_C(\alpha'_i, \beta'_1) : (0, 1) \to (1, 0) \to (1, 1) \to (2, 1) \to (3, 2) \hookrightarrow (0, 2)X$

Figure 1. Counting Independent Sets on the graph $C_5 \cup \{\{v_5, v_6\}\}$.

Notice that when a back edge $\{v_n, v_1\}$ is processed, the control on the main computing thread is kept at the vertex v_n of the back edge.

This allows us to continue the processing of the remaining graph from v_n (without changing the control to v_1), as it is illustrated in Figure 1. After processing a back edge of a cycle C, the associated computing thread of the cycle L_C is closed, as it is shown by the symbol X in Figure 1. Then, $i(G)$ can be computed according to each edge that is recognized when a walking is applied on G.

4 Grid and benzenoid graphs

A grid graph of size $m \times n$ is a graph $G_{m,n}$ with vertex set $V = \{(i,j) : 1 \leq i \leq m, 1 \leq j \leq n\}$ and edge set $E = \{\{(j,i),(j+1,i)\} \cup \{((j,i),(j,i+1)\} \mid 1 \leq j < m, 1 \leq i \leq n\}$.

We consider a grid graph $G_{m,n}$ as the union of n paths, each path with m vertices. The path $P_m^i(Fig.\ 2), 1 \leq i \leq n$ have a vertex set $V(P_m^i) = \{(1,i),(2,i),\cdots,(m,i)\}$ and edge set $E(P_m^i) = \{((j,i),(j+1,i)) \mid 1 \leq j < m\}$. The vertices of the grid graph $V(G_{m,n}) = \cup_{i=1}^{n} V(P_m^i)$ and the edges of the grid graph $E(G_{m,n}) = \{((j,i),(j,i+1)) \mid 1 \leq i < n, 1 \leq j \leq m\} \cup (\cup_{i=1}^{n} E(P_m^i))$.

The computation of the M-S index on grid structures is closely related to the ''hard-square model", which is used in statistical physics and, of particular interest is the so-called hard-square entropy constant [2]. It has several applications in statistical physics [2,10,24], e.g. for the computation in the Potts and hardcore lattice gas model and the problem of counting q-particle Widom-Rowlinson configurations in graphs, where $q > 2$.

The classical method for computing the M-S index on grid graphs is based on the transfer matrix method [3, 11]. The transfer matrix method consist of building an initial matrix of F_{m+2} rows and F_{m+2} columns that are indexed by $(m+1)$-vectors of zeros and ones. Let us consider S be an independent set of $G_{m,n}$. Let $\mathcal{C}_m$ be the set of all $(m+1)$-vectors $\mathbf{v}$ of $0's$ and $1's$ without two consecutive $1's$, in which a 1 indicates that the vertex is in S, and a 0 indicates that the vertex is not in S. The number of these vectors is F_{m+2}, the $m+2$-th Fibonacci number. Let T_m be an $F_{m+2} \times F_{m+2}$ symmetric matrix of $0's$ and $1's$ whose rows and columns are indexed by the vectors of $\mathcal{C}_m$.

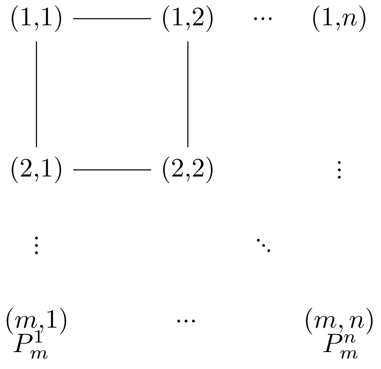


Figure 2. Path identification over $G_{m,n}$.

The condition that vectors $\mathbf{u}, \mathbf{v}$ in $\mathcal{C}_m$ are a possible consecutive pair of columns in an independent set of $G_{m,n}$ is simply that they meet the condition of having no 1's in common position, i.e., that $\mathbf{u} \cdot \mathbf{v} = 0$ in the sense of the usual dot product of vectors over the real numbers. Then, the entry of T_m in position $(\mathbf{u}, \mathbf{v})$ is 1, if the vectors $\mathbf{u}, \mathbf{v}$ are orthogonal; otherwise, it is 0. T_m is called the transfer matrix for $G_{m,n}$. Notice that T_m has $F_{m+2} \cdot F_{m+2}$ inputs.

The M-S index of a grid graph $G_{m,n}$ is obtained by the sum of all entries of the n-th power matrix T_m^n, i.e., $i(G_{m,n}) = \mathbf{1}^t T_m^n \mathbf{1}$, where $\mathbf{1}$ is the (F_{m+2})-vector whose entries are all $1's$.

Only the construction of the initial matrix T_m involves the order of $O((F_{m+2})^2 \cdot (m+1))$ dot products of $m+1$-vectors over the real numbers. Afterward, the computation of T_m^n implicates an order of $O((F_{m+2})^{3n})$ multiplications among integers, or an order of $O(((1.618)^{(m+2)\cdot(3n)})$ if we consider the asymptotic behavior of the Fibonacci numbers, and the approximation to the golden ratio of (1.618). This last complexity could be reduced to $O((1.618)^{(m+2)\cdot(2.81n)})$ integer multiplications, when the Strassen matrix-multiplication algorithm is applied. In any case, the application of the transfer matrix method for computing $i(G_{m,n})$ has a time-complexity of $O((F_{m+2})^2 \cdot (m+1) + (F_{m+2})^{2.81n})$ integer multiplications that results in an exponential upper bound on both dimensions m and n

of the grid $G_{m,n}$.

4.1 Benzenoid systems

A benzenoid system, denoted by $H_{r,t}$, is a subset (with 1-connected interior) of a regular tiling of the plane by hexagonal tiles. To each benzenoid system, we can assign a graph by taking the vertices of hexagons as the vertices, and the sides of hexagons as the edges of the graph. The resulting simple, plane, and bipartite graph is called a benzenoid graph.

Benzenoid systems are of great importance for theoretical chemistry, because they are the natural graph representation of benzenoid hydrocarbons. Another line of researching is the modelation of graphene sheets, that is one of the main element of certain carbon allotropes, through benzenoid systems. They are used also to describe phase transitions for physical systems [9, 27].

All faces of a benzenoid graph, except the unbounded face, are hexagons. The vertices lying on the border of the non-hexagonal face of a benzenoid graph are called external; other vertices, if any, are called internal. A benzenoid graph without internal vertices is called catacondensed. Pericondensed benzenoids have internal vertices. If no hexagon in a catacondensed benzenoid is adjacent to three other hexagons, then the benzenoid is a chain [9].

We will consider pericondensed benzenoid systems $H_{r,t}$ formed by $r \cdot t$ congruent regular hexagons arranged in r rows, each row of t hexagons. Each row is shifted for half of a hexagon from the following row; an example is illustrated in Figure 3. The arrangement is such that two hexagons either share an edge, or they are completely disjoint. $H_{r,t}$ has $r \cdot t$ hexagonal faces.

The first step in our procedure for computing $i(H_{r,t})$ is to embed the pericondensed benzenoid systems $H_{r,t}$ in a regular hexagonal grid HG, where instead of the traditional squares in the grid, we consider hexagons, as it is shown in Figure 4.

Both graphs, the benzenoid system $H_{r,t}$ and the regular hexagonal grid HG are isomorphic graphs, and they have the same number of hexagons. But while the number of columns is regular in $H_{r,t}$, the number of columns in HG has $2t + 1$ or $2t + 2$ vertices. We denote the number of rows in HG

by m, and the number of columns of the first row by n. Then, $HG_{m,n}$ is a hexagonal grid of m rows, where each row of $HG_{m,n}$ has n or $n+1$ columns. The first row in $HG_{m,n}$ has n vertices, but row 2 and 3 have $n+1$ vertices. The row 4 returns to have n vertices, and the following two rows have $n+1$ vertices, and so forth.

The number of hexagons is the same on each row of $HG_{m,n}$, in such a way that the hexagons in $H_{r,t}$ are one-to-one related with the hexagons in $HG_{m,n}$, as it is shown in Figure 3 and Figure 4. The embedding of a benzenoid system $H_{r,t}$ in a regular hexagonal grid $HG_{m,n}$ can be done in linear time on $r \cdot t$, since linear-time algorithms that build an embedding of a planar graph into a grid exist [4,6]. Therefore, we can represent the edges of $HG_{m,n}$ as straight-line segments.

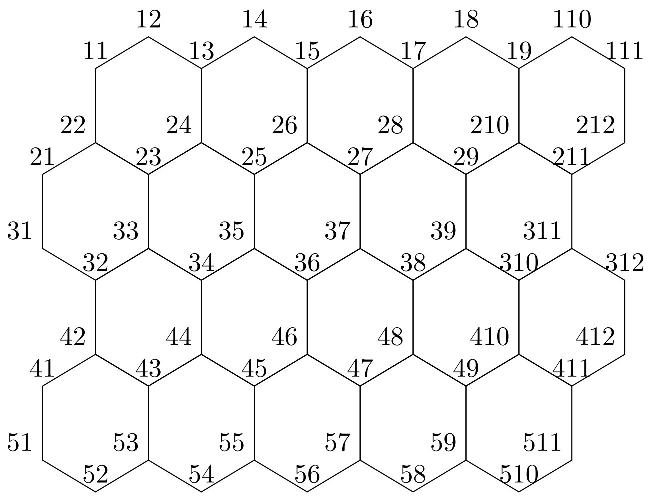


Figure 3. A benzenoid system.

Similar to a grid graph, we identify the vertices in $HG_{m,n}$ by the coordinates (i,j) that indicate the position of the vertex in the row $i, i = 1,\dots,m$ and the column $j, j = 1,\dots,n+1$. A vector column of $HG_{m,n}$ has m vertices, while a vector row of $HG_{m,n}$ has n or $n+1$ vertices. Notice that $HG_{m,n}$ has only vertices of degree two or three. The edges in $HG_{m,n}$ are of two types; horizontal edges with vertices $\{(i,j),(i,j+1)\}, i = 1,\dots,m, j =$

$1,\ldots,n$. The remaining edges in $HG_{m,n}$ are called vertical edges. A vertical edge in $HG_{m,n}$ has vertices $\{(i,j),(i+1,j)\}$, or $\{(i,j),(i+1,j+1)\}$, or $\{(i,j),(i+1,j-1)\}$, according to the row i that is being considered.

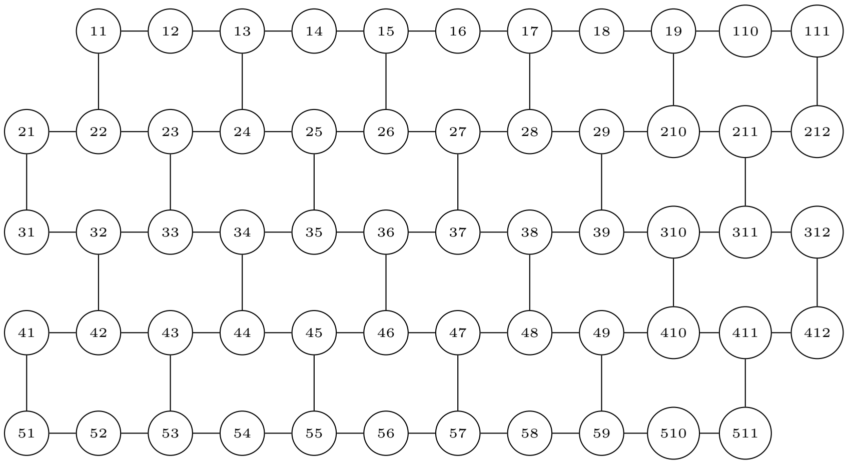


Figure 4. The grid graph representation of the benzenoid system.

The adaptation of the transfer matrix method for the computation of $i(H_{r,t})$, where $H_{r,t}$ is a benzenoid system, requires the increase in exponential factors with respect to the performed operations when the M-S index is computed on grid graphs. For example, Zhang Z. [27] has to build a matrix M, where one of its dimensions is of order 3^m for $2m$-vectors of 0's and 1's, for computing the M-S index in a particular benzenoid system. It requires the order of $O(3^{2.81mn})$ basic operations for computing the M-S index in such structures.

We show in the following sections how to compute $i(HG_{m,n})$, where $HG_{m,n}$ is the embedded graph obtained from a benzenoid system $H_{r,t}$. This computation does not require as many exponential number of basic operations as the transfer matrix method does.

5 Computation of the Merrifield-Simmons index on hexagonal grids

A relevant element in the computation of $i(HG_{m,n})$ is the construction of a Hamiltonian walking that trails by every vertex of $HG_{m,n}$ exactly once. We use a table $T_{k,l}$ of k rows and l column in order to store the partial calculus in the computation of $i(HG_{m,n})$, while the walking is being performed on $HG_{m,n}$.

The maximum number for k in the table $T_{k,l}$ corresponds to the maximum number of computing threads that must be active in any moment during the computation of $i(HG_{m,n})$. On the other hand, the number of columns l corresponds to the number of total vertices that are visited, which in this case is the total number of vertices in $HG_{m,n}$. Thus, $l = c{\cdot}r{\cdot}t$, where c is a constant factor.

For the vertices of degree 2, the walking visits both edges. Meanwhile, for the vertices of degree three, two of its edges are visited by the walking and the third edge is recognized as a back edge, but this class of edges does not change the direction of the walking. During the walking on $HG_{m,n}$, when a vertex v is visited and where v indicates the beginning of a back edge $\{v, w\}$, then as many new threads are created as active lines with a value $\beta_v > 0$ that exists on $T_{k,l}$.

For example, let us consider that $\{v, w\}$ is a back edge, and the vertex v is visited by the walking. If a thread L_i is active with a pair $(\alpha_v, \beta_v)_i$ associated to L_i in the column v, and where $\beta_v > 0$, then a new computing thread L_{v_i} is created. The first pair associated to the thread L_{v_i} is $(0, \beta_v)_{v\_i}$, which is stored in a cell of $T_{k,l}$ in the positions: row v_i and column v. The value i is a pointer that indicates the row i that corresponds with the thread L_i in the table $T_{k,l}$. In this way, each new back edge increases the number of active threads.

On the other hand, the threads associated to a cycle with back edge $\{v, w\}$, which were created when v was visited, they are closed after the walking has visited w. When w is visited, the pair $(\alpha_w, \beta_w)_v$ associated to the thread L_{v_ap} is used to form the pair $(0, \beta_w)_v$ that is substracted to the current pair in the thread L_{ap}, i.e. $(\alpha_w, \beta_w)_{ap} = (\alpha_w, \beta_w)_{ap} - (0, \beta_w)_v$.

Afterwards, all thread containing the term v as prefix or postfix in its label is closed.

When the threads are closed, it is possible to move rows on $T_{k,l}$ in order to keep only the active threads on $T_{k,l}$. As some rows on $T_{k,l}$ are moved, it would be necessary to update the pointers used on the labels of the threads. Nonetheless, it adjusts the value for k in order to indicate the maximum number of active threads that are requested in any moment during the computation of $i(HG_{m,n})$.

The number of rows k on $T_{k,l}$ is mainly determined by the maximum number of active threads during the whole computation of $i(HG_{m,n})$. Thus, k is a dynamic value that changes according to the back edges that are processed, and its maximum value corresponds to the maximum number of open cycles during the walking on $HG_{m,n}$.

5.1 Walkings on the hexagonal grid

In this section, we present different ways to traverse through the hexagonal grid $HG_{m,n}$. The walkings will be represented by dashed lines in the figures of the hexagonal grids. The first walking on $HG_{m,n}$ is through rows. While the walking on $HG_{m,n}$ is performed, the cells in $T_{k,l}$ are filled. The edges of $HG_{m,n}$ are processed in two forms: via recurrence equations (when horizontal edges or vertices of degree two are visited), or as back edges (such edges only correspond to vertical edges).

On the other hand, we can visit the vertices on $HG_{m,n}$ through columns. In some cases, a series of Hamiltonian walkings on $HG_{m,n}$ are built. And in this case, when different threads meet in a same vertex, the threads are processed via the Hadamard product of its respective sign-pairs, as it is done when the childs of a father node are processed.

5.2 A walking by rows on the hexagonal grid

First, we present a Hamiltonian trail by rows on $HG_{m,n}$, as it is shown by the dashed lines in Figure 5. This walking starts in the vertex $(1, 1)$, and after, it proceeds to visit all vertex on row 1 of $HG_{m,n}$, with horizontal movements and on the right. The walking moves from $(1, j)$ to $(1, j +$

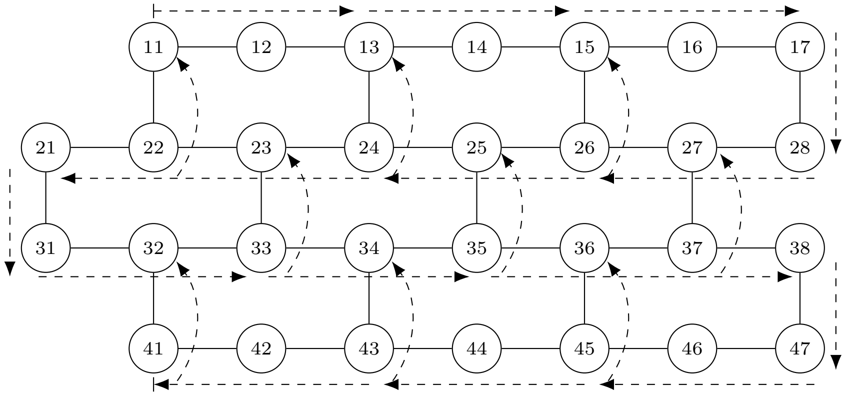


Figure 5. A walking by rows of the hexagonal grid

$1), j = 1, \ldots, n-1$ in incremental way, and at the same time, a Fibonacci recurrence is applied on the sign-pairs of the active threads in $T_{k,l}$.

The walking visits all the vertices in row 1 until arriving to the vertex $(1, n)$. During this walking the vertices $(1, j)$ with adjacent vertex $(2, j + 1), j = 1, \ldots, n-1$ indicate the beginning of a new cycle. Each beginning of a cycle duplicates the number of active threads L_{ap} in $T_{k,l}$ where $\beta_{1,j} > 0$. The new threads are labeled as L_{1,j_ap}, where $1, j$ indicates the vertex where the cycle is started and ap is a pointer to the active thread L_{ap} of $T_{k,l}$.

After visiting the first row of $HG_{m,n}$, the walking moves from $(1, n)$ to $(2, n + 1)$, and at the same time, a Fibonacci recurrence is applied on the sign-pairs of the active threads. Now, the walking visits all the vertices on the row 2 with horizontal movements and on the left. This means that the walking moves from $(2, j)$ to $(2, j - 1), j = n + 1, \ldots, 2$, until arriving to the vertex $(2, 1)$. As in the case of the first row, the vertices $(2, j)$ with neighbor vertex $(3, j)$ for $j = 1, \ldots, n$, indicate the beginning of new cycles.

Nonetheless, from the second row and forth on, we also find vertices that indicate the end of open cycles, i.e vertices $(2, j)$ with neighbor vertex $(1, j - 1)$. In order to close a cycle (initiated in $(1, j - 1)$), we have to process a back edge and then cancel all the threads where the particle

$1, j-1$ appears in any part of their label. Notice that during the walking on $HG_{m,n}$, the back edges are recognized and processed on $T_{k,l}$, but they do not change the direction of the trail.

From vertex $(2,1)$, the walking visits the vertex $(3,1)$ and after, it visits all the vertices in row 3 of $HG_{m,n}$, with movement to the right from $(3,j)$ to $(3,j+1), j=1,\dots,n$. Again, when the walking finds the vertex with vertical incident edges, then such edges are processed as the beginning of cycles (for the edges type $\{(3,j),(4,j-1)\}$, or as the end of open cycles (for the edges type $\{(3,j),(2,j)\}$).

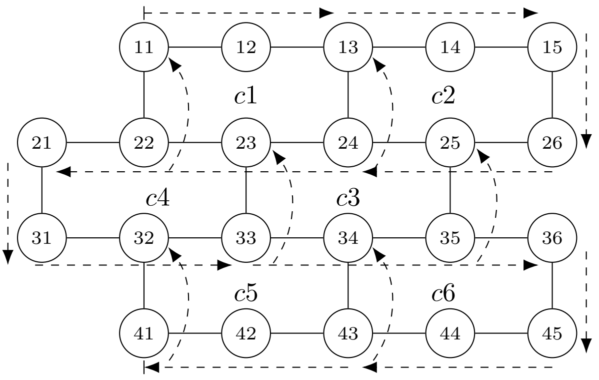


Figure 6. A small hexagonal grid graph $HG_{3,2}$

Notice that in general, the horizontal edges are processed by the Fibonacci recurrence (1). Meanwhile, the vertical edges indicate the beginning of new cycles, the end of open cycles, or one movement of the trail in order to change rows.

In this way, the walking visits all the vertices in $HG_{m,n}$, with only horizontal movements in each row. The walking moves to the right for odd rows and to the left for even rows. Also, the walking only performs one vertical movement for changing row.

With purposes of illustration, we present the walking by rows of $HG_{3,2}$ in Figure 6, and its corresponding table $T_{k,l}$ from Table 1 to Table 3.

Table 1. Initial partial calculus of $i(HG_{3,2})$

Threads |	11	12	13	14	15	26	25	$24 \circlearrowleft 13$	23
Lp	(1,1)	(2,1)	(3,2)	(5,3)	(8,5)	(13,8)	(21,13)	(34,21)	
								$\underline{-(0,6)}$	
								(34,15)	(49,34)
$c1$	(0,1)	(1,0)	(1,1)	(2,1)	(3,2)	(5,3)	(8,5)	(13,8)	
								$\underline{-(0,3)}$	
								(13,5)	(18,13)
$c2$			(0,2)	(2,0)	(2,2)	(4,2)	(6,4)	$(10,6)_X$	
$c2c1$			(0,1)	(1,0)	(1,1)	(2,1)	(3,2)	$(5,3)_X$	
$c3$							(0,13)	(13,0)	(13,13)
$c3c1$							(0,5)	(5,0)	(5,5)
$c3c2$							(0,4)	$(4,0)_X$	
$c3c2c1$							(0,2)	$(2,0)_X$	
$c4$									(0,34)
$c4c1$									(0,13)
$c4c3$									(0,13)
$c4c3c1$									(0,5)

5.3 Time-complexity analysis for the computation of the M-S index on benzenoid systems

The Hamiltonian walking by rows is not cyclical, since it does not return to the vertex from which it departed. Each movement of the walking allows us to visit a vertex, then the walking performs the same number of movements as vertices that exist in $HG_{m,n}$. Thus, the walking has a linear-time complexity on the number of total vertices in $HG_{m,n}$ that is a constant factor of the number of hexagons in $H_{r,t}$, which is the order $O(r \cdot t)$. In addition, the number of vertices in $H_{r,t}$ is the same number as the requested columns in $T_{k,l}$; therefore, $l = c \cdot r \cdot t$, where c is a constant factor.

On the other hand, the number of rows k in $T_{k,l}$ is a dynamic value that increases when the beginning of a new cycle is found during the walking. Meanwhile, the number of open cycles reduces its value when a back edge is processed. The maximum value for k corresponds to the maximum number of open cycles during the walking on $HG_{m,n}$.

The computation of $i(HG_{m,n})$ starts with two threads: L_p - the main thread, and $c1$ - the thread subordinated to the cycle that starts in the

Table 2. The continuation of the calculus of $i(HG_{3,2})$

Threads	$22 \circlearrowleft 11$	21	31	32	$33 \circlearrowleft 23$	34	$35 \circlearrowleft 25$
Lp	(83,49)						
	-(0,18)						
	(83,31)	(114,83)	(197,114)	(311,197)	(508,311)		
					-(0,102)		
					(508,209)	(717,508)	(1225,717)
							-(0,209)
							(1225,508)
*c*1	$(31,18)_X$						
*c*3	(26,13)						
	-(0,5)						
	(26,8)	(34,26)	(60,34)	(94,60)	(154,94)		
					-(0,39)		
					(154,55)	(209,154)	$(363,209)_X$
*c*3*c*1	$(10,5)_X$						
*c*4	(34,0)	(34,34)	(68,34)	(102,68)	$(170,102)_X$		
*c*4*c*1	$(13,0)_X$						
*c*4*c*3	(13,0)	(13,13)	(26,13)	(39,26)	$(65,39)_X$		
*c*4*c*3*c*1	$(5,0)_X$						
*c*5				(0,197)	(197,0)	(197,197)	(394,197)
							-(0,60)
							(394,137)
*c*5*c*3				(0,60)	(60,0)	(60,60)	$(120,60)_X$
*c*5*c*4				(0,68)	$(68,0)_X$		
*c*5*c*4*c*3				(0,26)	$(26,0)_X$		
*c*6						(0,508)	(508,0)
*c*6*c*3						(0,154)	$(154,0)_X$
*c*6*c*5						(0,197)	(197,0)
*c*6*c*5*c*3						(0,60)	$(60,0)_X$

vertex $(1,1)$. Each time that a vertex $(1,j)$ is visited, with $j < n$ and j odd, such vertex indicates the beginning of a new cycle. There are a total of t vertices by row with a neighbor in the following row. As each beginning of cycle implicates the duplication of the number of active threads, then there are 2^t active threads when the first row of $HG_{m,n}$ has been traversed. Before closing the cycles, the vertex $(2,n)$ also marks the beginning of a new cycle. Therefore, the maximum number of active threads before closing cycles is 2^{t+1}.

The first cycle that has to be closed is found when the vertex $(2,n-1)$ is visited, and its starting vertex is $(1,n-2)$. As $(1,n-2)$ was the penultimate open cycle, then there are 2^t threads whose labels have included the particle $1,n-2$. Those 2^t threads are closed after the vertex $(2,n-1)$ has been visited, remaining then 2^t active threads on the table $T_{k,l}$.

Table 3. Final computation of $i(HG_{3,2}) = 18465 + 7570$

Threads	36	45	44	43 $\circlearrowleft$ 34	42	41 $\circlearrowleft$ 32
Lp	(1733,1225)	(2958,1733)	(4691,2958)	(7649,4691)		
				-(0,1524)		
				(7649,3167)	(10816,7649)	(18465,10816)
						-(0,3246)
						(18465,7570)
$c5$	(531,394)	(925,531)	(1456,925)	(2381,1456)		
				-(0,591)		
				(2381,865)	(3246,2381)	$(5630,3246)_X$
$c6$	(508,508)	(1016,508)	(1524,1016)	$(2540,1524)_X$		
$c6c5$	(197,197)	(394,197)	(591,394)	$(985,591)_X$		

During the traversing on the row 2, each time that a vertex marking the beginning of a new cycle is found, then the following vertex that has to be visited indicates that a back edge is found. Then, the number of active threads alternates from 2^t to 2^{t+1}, when a new cycle is found. Meanwhile, the number of active threads changes from 2^{t+1} to 2^t active threads, when a vertex with an incident back edge is found. This behavior on the number of active threads is kept until arriving to the last row of $HG_{m,n}$, where the walking visits only vertices that indicate to close the open cycles.

When the walking visits the vertices of the last row, the number of threads is reduced until keeping only the main thread L_p. Thus, the maximum value that k achieves is 2^{t+1}, which corresponds to the maximum number of threads requested to process $t+1$ open cycles. The table $T_{k,l}$ used during the computation of $i(HG_{m,n})$, request 2^{t+1} rows and $c \cdot t \cdot r$ columns, with c a constant factor.

All the remaining computations, such as the application of the Fibonacci recurrence for horizontal edges, the marking of the beginning or end of a cycle, as well as the subtraction of pairs of signs, can be done as calls to basic operations. However, during the computation of the M-S index on benzenoid systems, it could be common to involve the product among big integers. Therefore, instead of basic operations, we should consider increasing in a polynomial factor the complexity of the computations.

The walking by rows, and the search for terms on the labels of the threads, are both procedures that can be done in linear time on its sizes, which are of order $O(r \cdot t)$ and $O(2^{t+1})$, respectively. Similarly, complexity time of $O(2^{t+1})$ is spent in order to move threads and avoid gaps in the table $T_{k,l}$.

Therefore, the total complexity time for filling the cells of $T_{k,l}$ requests the order of $O(2^{t+1} \cdot t \cdot r)$ basic operations, which is the same as the time-complexity of our procedure for computing $i(H_{r,t})$. The time-complexity of our proposal for computing the M-S index is dramatically inferior to the time-complexity that the transfer matrix method requires on benzenoid systems. It is even inferior to the time-complexity that the transfer matrix requires for computing the M-S index on grid graphs.

5.4 Traversing by columns on the hexagonal grid

In this section, we present a series of Hamiltonian trails on $HG_{m,n}$ represented by dashed lines in Figure 7. For a walking by columns, the movements of the walking alternate between one horizontal movement and one vertical movement in order to visit the first two vertices of each row. The walking visits the vertices $(i, 1)$ and $(i, 2), i = 1, \ldots, m$ with a horizontal movement. Afterwards, with a vertical movement, it changes from row.

The first two vertices of each row are visited downwards. Subsequently, the walking visits the third and fourth vertices of each row (the walking visits the vertices $(i, 3)$ and $(i, 4), i = m, \ldots, 1$) upwards. In general, the walking alternates between downward and upward directions, while each hexagonal column of the grid $HG_{m,n}$ is visited.

While the trail visits the vertices $(i, 3)$ and $(i, 4), i = m, \ldots, 1$ upwards, it also recognizes the edges $\{(i, 2), (i, 3)\}, i = m-1, \ldots, 1$ as back edges that indicate to close open cycles. There are m back edges for each hexagonal column in $HG_{m,n}$, and each back edge is an incident edge of a vertex of degree three. Moreover, during the trail of a hexagonal column, the walking finds m vertices that indicate the beginning of cycles.

For this trail, there are different starting points that they should converge at common vertices. One starting point is the vertex $(1, 2)$ in order to continue visiting the first two vertices of each row of $HG_{m,n}$. Another starting point is the vertex $(m, 4)$ whose incident edge $\{(m, 4), (m, 3)\}$ is processed as child edge from its father vertex $(m, 3)$. Thus, the edges $\{(m, 4), (m, 3)\}$ and $\{(m, 3), (m, 2)\}$ are considered as child tree edges from

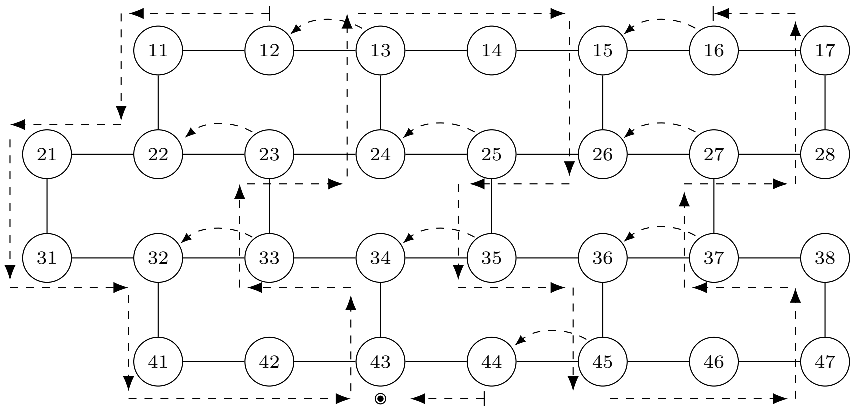


Figure 7. A walking by columns on the hexagonal grid

$(m, 3)$. When the vertex $(m, 3)$ is visited, the Hadamard product is performed on the associated signs-pairs of its child vertices.

In general, each hexagonal column of $HG_{m,n}$ (with exception of the last hexagonal column) has associated one starting point, whose incident edges have to be processed as tree edges (applying the Hadamard product between the associated signs-pairs of two tree edges). This trail has as many starting vertex as $(t-1)$ hexagonal columns in $HG_{m,n}$.

Two threads are built and associated to each starting vertex. When the two threads converge at its father vertex (the following adjacent vertex in the walking), then the number of active threads in $T_{k,l}$ is duplicated. This is done since all of them result on the multiplication of its signs with the pair $(2, 1)$, and the other group is the result of the multiplication of its signs with the pair $(1, 0)$. The trail finishes when all the hexagonal columns of $HG_{m,n}$ have been visited. Thus, we can exploit the fact of having different starting points in order to build a general walking on irregular benzenoid systems.

5.5 Time-complexity analysis for the trail by columns

The walking by columns on $HG_{m,n}$ finds $(m-1)$ starting cycles in the vertices $(2, i), i = 1, \ldots, m-1$, and the other vertex that duplicates the number of active threads is $(3, m)$. In total, the walking found m vertices on each column that indicate the beginning of cycles.

The first three vertices that indicate starting cycles implicate that the number of active threads has to be duplicated until row 3, then there are $2^3 = 8$ active threads in $T_{k,l}$. Afterwards, the growth on the number of active threads alternates between a Fibonacci behavior and the duplication of the number of threads. For example, for the row 4 there are $12 = 8 + 4$ threads; for the row 5, there are $12 \cdot 2$ threads; for the row 6, there are $24 + 12 = 36$ threads, and so on.

Thus, the number of threads in $T_{k,l}$ follows the numeric sequence: $2, 4, 8, 12, 24, 36, 62, 98, \ldots$ that is defined for the recurrence $T(i) = T(i - 1) * 2$ for odd i, and $T(i) = T(i-1) + T(i-2)$ for even i. Moreover, they have the initial conditions: $T(1) = 2, T(2) = 4, T(3) = 8, T(4) = 12$.

Starting from the second column and onwards, the walking finds a back edge in each row, from the row $m - 1$ until the row 1. However, for each row, when the walking finds a back edge, then the following vertex marks the beginning of a new cycle. Thus, the number of threads that are canceled by the processing of a back edge is subsequently recovered in the following vertex that indicates the beginning of a cycle.

The asymptotic behavior on the number of threads, for this walking by columns, follows the alternation between the recurrence $T(i) = 2 \cdot T(i-1)$, and $T(i+1) = T(i) + T(i-1)$. But the Fibonacci growth of the latter equation is upper bounded by the growth of $T(i) = 2 \cdot T(i - 1)$, thus, $T(i+1) = T(i) + T(i-1) < T(i) + T(i) = 2 \cdot T(i)$.

Furthermore, the number of active threads alternates from 2^m to 2^{m-1} and the 2^{m-1} to 2^m threads during the traversing of the $(r - 1)$ columns. And only during the traversing of the last column, the walking finds only back edges that reduces the number of active threads until keeping active only the main thread L_p. Therefore, the number of active threads is upper bounded by $O(2^{r+1}) = O(2^m)$ in regards to the benzenoid system $H_{r,t}$.

We have similar conditions on the basic operations that the algorithm performs on the cell of the table $T_{k,l}$. For example, the application of the Fibonacci recurrence (1), the identification of the beginning and end of cycles, as well as the subtracted pairs or multiply two pairs by the

Hadamard product, all of those operations can be done in a constant time of computation. Meanwhile, the walking by columns and the look for terms on the labels of the threads are both operations that can be done in linear time on its sizes, which are of order $O(r \cdot t)$ and $O(2^{r+1})$, respectively. A complexity time of $O(2^{r+1})$ is spent in order to move threads and avoid gaps in the table $T_{k,l}$.

Therefore, the total complexity time for filling the cells of $T_{k,l}$ should require the order of $O(2^{r+1} \cdot t \cdot r)$ basic operations. This is similar to the time-complexity of our procedure for computing $i(HG_{m,n})$ when a walking by rows is performed.

Thus, the walking by rows and the walking by columns on $HG_{m,n}$ provide similar asymptotic behaviors. We have an asymptotic behavior of $O(2^{r+1} \cdot (r \cdot t))$ when the walking is by columns, or an asymptotic behavior of $O(2^{t+1} \cdot (r \cdot t))$ when the walking is by rows. It is preferable to select the walking by columns if $r < t$, otherwise the walking by rows is better. Therefore, our procedure for computing $i(H_{r,t})$, where $H_{r,t}$ is a regular benzenoid system, has a complexity time upper bounded by $O(2^{min\{r,t\}+1} \cdot (r \cdot t))$ basic operations.

The time-complexity of our proposal has an exponential character only on one of the two dimensions of the benzenoid system $H_{r,t}$ for computing its M-S index. This is contrary to the transfer matrix method, which has an exponential growth on both dimensions of the system. Moreover, it is even inferior to the time-complexity that the transfer matrix requires for computing the M-S index on grid graphs.

6 Conclusions

We present a novel method for computing the Merrifield-Simmons index on regular benzenoid systems $H_{r,t}$ of r rows and t hexagonal columns. Our method consists of performing a linear-time Hamiltonian walking on an isomorphic hexagonal grid of $H_{r,t}$, while the number of independent sets is also computed incrementally.

The complexity time of our proposal for computing the M-S index of a benzenoid system $H_{r,t}$ has an exponential character only on one of the its two dimensions. This is contrary to the transfer matrix method, which has an exponential growth on both dimensions of the system. Similarly, our proposal has an inferior upper bounded of its time complexity with respect to the transfer matrix method, when the M-S index is computed on grid graphs.

References

[1] O. Angelsmark, P. Jonsson, Improved Algorithms for Counting Solutions in Constraint Satisfaction Problems, in: F. Rossi (Eds.), *Proceedings of CP'03*, Springer, 2003, pp. 81–95.

[2] R. J. Baxter, Planar lattice gases with nearest-neighbor exclusion, *Ann. Comb.* **3** (1999) 191–203.

[3] N. Calkin,H. Wilf, The number of independent sets in a grid graph, *SIAM J. Discr. Math.* **11** (1998) 54–60.

[4] M. Chrobak, T. H. Payne, A linear-time algorithm for drawing a planar graph on a grid, *Inf. Proc. Lett.* **54** (1995) 241–246.

[5] V. Dahllöf, P. Jonsonn, M. Wahlström, Counting models for 2SAT and 3SAT formulae, *J. Theor. Comput. Sci.* **332** (2005) 265–291.

[6] H. De Fraysseix, J. Pach, R. Pollack, How to draw a planar graph on a grid, *Combinatorica* **10** (1990) 41–51.

[7] G. De Ita , J. R. Marcial-Romero, P. Bello, M. Contreras, Linear-time algorithm for computing the Merrifield-Simmons index on polygonal trees, *MATCH Commun. Math. Comput. Chem.* **79** (2018) 55–78.

[8] H. Deng, The smallest Merrifield-Simmons index of $(n, n+1)$-graphs, *Math. Comput. Model.* **49** (2009) 320–326.

[9] T. Došlić, I. Zubac, Saturation number of benzenoid graphs, *MATCH Commun. Math. Comput. Chem.* **73** (2015) 491–500.

[10] M. Dyer, C. Greenhill, Corrigendum: The complexity of counting graph homomorphism, *Rand. Struct. Alg.* **25** (2004) 346–352.

[11] R. Euler, The Fibonacci number of a grid graph and a new class of integer sequences, *J. Integer Seq.* **8** (2005) 1–16.

[12] M. Golin, Y. Cho Leung, Y. Wang, X. Yong, Counting structures in grid graphs, cylinders and tori using transfer matrices: Survey and new results, in: C. Demetrescu, R. Sedgewick, R. Tamassia (Eds), *Proceedings of ALENEX/ANALCO 2005*, SIAM, 2005, pp. 250–258.

[13] C. Greenhill, The complexity of counting colourings and independent sets in sparse graphs and hypergraphs, *Comput. Complex.* **9** (2000) 52–72.

[14] C. Guillen, A. Lopez, G. De Ita, Computing #2-sat of grids, grid-cylinders and grid-tori boolean formulas, in: M. Gavanelli, T. Mancini (Eds), *Proceedings of RCRA-2008*, CEUR-WS (2008) 152–167.

[15] C. Hoede, X. Li, Clique polynomials and independent set polynomials of graphs, *Discr. Math.* **125** (1994) 219–228.

[16] D. G. Klein, G. E. Hite, T. G. Schmalz, Transfer-Matrix method for subgraph enumeration: Applications to polypyrene fusenes, *J. Comput. Chem.* **7** (1986) 443–456.

[17] X. Li, H. Zhao, I. Gutman, On the Merrifield-Simmons index of trees, *MATCH Commun. Math. Comput. Chem.* **54** (2005) 389–402.

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

[19] Y. Okamoto, T. Uno, R. Uehara, Counting the number of independent Sets in chordal graphs, *Lect. Notes Comput. Sci.* **3787** (2005) 433–444.

[20] M. S. Oz, I. N. Cangul, Computing the Merrifield-Simmons indices of benzenoid chains and double benzenoid chains, *J. Appl. Math. Comput.*, in press, doi: 10.1007/s12190-021-01659-x.

[21] M. S. Oz, I. N. Cangul, Computing the Hosoya and the Merrifield-Simmons indices of two special benzenoid systems, *Iranian J. Math. Chem.* **12** (2021) 161–174.

[22] H. Prodinger, R. F. Tichy, Fibonacci numbers of graphs, *Fibonacci Quart.* **20** (1982) 16–21.

[23] R. M. Roth, P. H. Siegel, J. K. Wolf, Efficient coding schemes for the hard-square model, *IEEE Trans. Inf. Theory* **47** (2001) 1166–1176.

[24] P. S. Vadhan, The complexity of counting in sparse, regular, and planar graphs, *SIAM J. Comput.* **31** (2001) 398–427.

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

[26] Y. Zhao, The number of independent sets in a regular graph, *Comb. Prob. Comput.* **19** (2010) 315–320.

[27] Z. Zhang, Merrifield-Simmons index and its entropy of the 4-8-8 lattice, *J. Stat. Phys.* **154** (2014) 1113–1123.