mathematics

MDPI

Communication

Figures of Graph Partitioning by Counting, Sequence and Layer Matrices

Mihaela Aurelia Tomescu [1], Lorentz Jäntschi [2,3,*] and Doina Iulia Rotaru [4]

[1] Department of Mathematics-Informatics, University of Petrosani, 332006 Petrosani, Romania; MihaelaTomescu@upet.ro
[2] Department of Physics and Chemistry, Technical University of Cluj-Napoca, 400641 Cluj-Napoca, Romania
[3] Chemical Doctoral School, Babes-Bolyai University, 400028 Cluj-Napoca, Romania
[4] Department of Conservative Dentistry, "Iuliu Hatieganu" Medicine and Pharmacy University, 400349 Cluj-Napoca, Romania; doina.rotaru@umfcluj.ro
* Correspondence: lorentz.jantschi@gmail.com or lorentz.jantschi@chem.utcluj.ro or lorentz.jantschi@ubbcluj.ro; Tel.: +40-264-401-775

Abstract: A series of counting, sequence and layer matrices are considered precursors of classifiers capable of providing the partitions of the vertices of graphs. Classifiers are given to provide different degrees of distinctiveness for the vertices of the graphs. Any partition can be represented with colors. Following this fundamental idea, it was proposed to color the graphs according to the partitions of the graph vertices. Two alternative cases were identified: when the order of the sets in the partition is relevant (the sets are distinguished by their positions) and when the order of the sets in the partition is not relevant (the sets are not distinguished by their positions). The two isomers of C_{28} fullerenes were colored to test the ability of classifiers to generate different partitions and colorings, thereby providing a useful visual tool for scientists working on the functionalization of various highly symmetrical chemical structures.

Keywords: graph partitioning; counting matrices; sequence matrices; layer matrices; molecular topology; molecular similarity; molecular symmetry

MSC: 05C30; 05C60; 05C92

Citation: Tomescu, M.A.; Jäntschi, L.; Rotaru, D.I. Figures of Graph Partitioning by Counting, Sequence and Layer Matrices. *Mathematics* **2021**, *9*, 1419. https://doi.org/10.3390/math9121419

Academic Editor: Adolfo Ballester-Bolinches

Received: 16 May 2021
Accepted: 16 June 2021
Published: 18 June 2021

1. Introduction

Graph theory is at the foundations of the Internet [1]; parallel [2] and distributed [3] computing; molecular topology [4] and dynamics [5]; and energy [6], electric [7] and electronic [8] circuit design.

In some cases, operating with connected undirected unweighted graphs [9] provides all necessary information, whereas in other cases more specificity is needed [10]. On the other hand, coloring graphs (of the vertices [11], edges [12] and planes [13]) can provide useful visual information.

In this study, the case of unweighted, undirected connected graphs is considered.

Some basic concepts about graphs are given here.

Let $G = (V, E)$ be an unweighted, undirected connected graph (with V, the set of vertices, and E, the set of edges). Then E is an subset ($\subseteq$) of $V \times V$. Usually the vertices are indexed (numerically, starting from 1) so that if there are n vertices ($|V| = n$) then their numbering gives $\{1, 2, \ldots, n\}$ as their representation in the informational space. It should be noted that for a given graph G of n vertices there are exactly $n!$ possibilities of numbering the vertices and (unfortunately) the same number of isomorphisms induced by numbering. This is why the search for graph invariants (an graph invariant is a property calculated from a graph that remains unchanged when the numbering changes) is one of the most important issues addressed when comparing graphs. From the edges (or the

vertices) the next construction is *chains* of connected edges (or vertices). If such a chain allows revisitation of edges and vertices then it is called a walk. If only revisiting of vertices is allowed, then it is called a trail; and finally, if edges and vertices appears only once in the chain, then it is called a path (defined by edges ($e_i \in E$): $P = e_1 ... e_k$ with $e_i \cap e_{i+1} \neq \{\}$ for $1 \leq i < k$; or defined by vertices ($v_i \in V$): $P = v_1, \ldots, v_{k+1}$ with $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k$; a path with k edges has $k + 1$ vertices). Paths are an important concept in graph theory, because the topological distance metric ($d_{i,j}$ as the length of the shortest path between two vertices i and j) and the graph diameter (the longest distance in the graph; Algorithm A3 provided in Appendix B) are built on them.

Related Research

A literature survey showed vertex partitioning and graph coloring to be of growing interest. Cutting a graph into smaller pieces is one of the fundamental algorithmic operations; partitioning large graphs is often an important subproblem for complexity reduction or parallelization [14]. With (or without [15]) nonnegative weights on vertices, in [16,17] the balanced connected k-partition problem was addressed, which is known to be NP-hard. In the same context the minimum gap graph partitioning problem was formulated, as addressed in [18]. Partition strategies on resource description framework graphs have been studied in [19] and in [20]. Graph contraction (creating of a graph minor [21]) is used in some specific graph-related problems [22]. When parallel motif discovery is employed on complex networks [23], graph partitioning divides the network for efficient parallelization (to an approximately equal number of vertices to parts). In this context, the graph partitioning problem is NP-complete [24], and there are available strategies based on spectral [25] (eigenproblem in [26]), combinatorial [27], geometric [28] and multi-level [29] heuristics. Partitioning of the graph vertices leads to recognition the of 2-subcolorable [30], bipartite [31], cluster [32], dominable [33], monopolar [34], r-partite [35], split [36], unipolar [37], trapezoid [38] and graphical algorithms (etc.) working efficiently with special classes of graphs that have been devised (for monopolar and 2-subcolorable in [30]; for unipolar and generalized split in [39]; for partitioning a big graph into k sub-graphs in [40,41]; for graph that does not contain an induced subgraph, a claw in [42]). For an extended survey on finding sets of related vertices in graphs clustering, the reader should go to [43]. As other recent studies have shown, vertex coloring in graphs may solve a series of real problems. To tackle these problems, different coloring schemes have been proposed: the scheme based on distances in [44], the scheme based on templates in [45], the scheme based on adjacencies in [46], the scheme based on heuristics in [47] and the scheme based on pseudo-randomness (with constrains, Grundy and color-dominating) in [48]. The properties of the colorings have been studied in [49] and the counting of distinguishing (symmetry breaking) colorings with k colors in [50]. One should notice that all Zagreb indices and their relatives [51] are useless for any topological isomers of fullerene, in which any vertex has a degree of 3 (in the related notation, $d_v = d_w = 3$). Sequence matrices appeared first in a study by Frank Harary (American mathematician, specialized in graph theory, widely recognized as one of the ''fathers" of modern graph theory) regarding the distribution of phonemes [52], which has been since proven useful in solving scheduling problems [53], in connection with pipelines in [54], and for route discovery in [55]. A layer matrix, a term initially used for tables expressing stratification by age in biological populations [56], was introduced into graph theory by Andrey A. Dobrynin (see [57–59]). Most of the studies involving the use of the layer matrix to differentiate between topological isomers were lead by Dobrynin [60,61] and Mircea V. Diudea [62], but other researchers also found uses for the layer matrices in their studies (see [63,64]). Haruro Hosoya were the first to introduce a counting polynomial (Z-counting polynomial in [65]; see general review on counting polynomials in chemistry in [66]) to characterize a graph, and George Pólya were the first to introduce the counting polynomial into graph theory to count the topological isomers [67]. Counting matrices are the expanded forms of counting polynomials [68], since some distance-related properties

can be expressed in the polynomial form, with coefficients calculable from the matrices (see [69,70]; for isomer-counting matrices, see [71]).

One set of works is especially related to the current study, since layer matrices were involved in the analysis of fullerenes: In [72], vertices were partitioned into classes of equivalence and ordered according to their centrality indexes, computed on layer matrices of vertex properties. In [73], the prediction of stability of C_{40} fullerenes was derived from two indices (of complexity and of centrocomplexity) calculated on the layer matrix of valences.

The use of the counting, sequence and layer matrices and some proposed modifications and extensions to generate different partitions on graphs are given and illustrated. Finally, these partitions were used for for getting visual representations of them. As an application of molecular topology, two isomers of C_{28} fullerene were subjected to atom partitioning, and it was of interest to obtain alternative groups of atoms.

To the best of the authors' knowledge, this communication is the first systematic approach of graph coloring based on a pool of partitions. To some extent, the sequence and layer matrices involved here were previously reviewed in [9], and the coloring of vertices based on counting matrices was previously reported in [12].

2. Graphs and Their Representation

An indexed numbered graph can be kept in the informational space as a list of the edges (pairs of integers), finally accompanied, for convenience, by the number of vertices. This type of representation is a powerful one (convenient in terms of the small amount of memory required for representation, as well as its fast processing; see Appendix A). However, in some cases a better equipped algebraic structure is preferred: a matrix representation. From this point of view, a graph can be represented by an adjacency matrix: $[Ad]$ (or $[AdV]$) for adjacency of the vertices, $[AdE]$ for adjacency of the edges, or even $[AdVE]$ a adjacency of vertices with edges. First two ($[AdV]$ and $[AdE]$) are square and symmetric matrices of size equal with the number of vertices ($[AdV]$), and number of the edges ($[AdE]$), respectively, while $[AdVE]$ is a rectangular matrix of the size of the number of vertices by the number of edges.

The more convenient representation (than a rectangular one) is a square matrix—it can be raised to a power, two of such matrices can be multiplied, etc. The number of walks between two vertices is found in the powers of the adjacency matrix (of the vertices). More importantly, this it is the most commonly used matrix base representation of a graph.

Let us take an example of a graph (the one in Figure 1) to be used to introduce the following concepts.

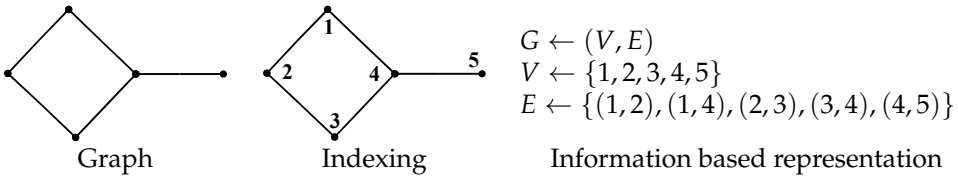
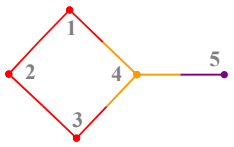


Figure 1. A simple connected undirected unweighted graph.

The adjacency matrix ($[Ad]$) contains information about adjacencies (Figure 2). If two vertices (either i and j) are connected by an edge ($(i, j) \in E$), then the corresponding elements in the matrix ($Ad_{i,j}$) are set to 1; otherwise, they are set to 0 (Algorithm A1 in Appendix B).

$[Ad]$	1	2	3	4	5	Σ
1	0	1	0	1	0	2
2	1	0	1	0	0	2
3	0	1	0	1	0	2
4	1	0	1	0	1	3
5	0	0	0	1	0	1



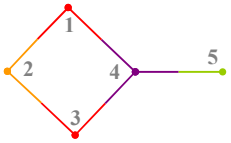
$$Ad_{i,j} \leftarrow \begin{cases} 1, & (i,j) \in E \\ 0, & (i,j) \notin E \end{cases}$$

Figure 2. Adjacency matrix on Figure 1.

The matrix given in Figure 2 contains an additional column Σ that collects the valences (connections; number of adjacent vertices) for each vertex and it can be used to discriminate the vertices (the vertices are represented with different colors accordingly in Figure 2), being thus a first example of a criterion that can be used to create a vertices partition (for $\Sigma_j Ad_{i,j}$ the partition is in three groups: {1, 2, 3}, {4}, {5}).

The distance matrix ($[Di]$) contains information about distances (Figure 3). For any two vertices, the corresponding elements in the matrix ($Di_{i,j}$) are set to the value of the distance between them (Algorithm A2 in Appendix B).

$[Di]$	1	2	3	4	5	Σ
1	0	1	2	1	2	6
2	1	0	1	2	3	7
3	2	1	0	1	2	6
4	1	2	1	0	1	5
5	2	3	2	1	0	8



$Di_{i,j} \leftarrow d_{i,j}$
(d: the topological distance)

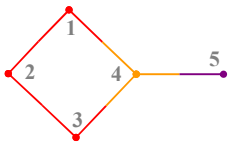
Figure 3. Distance matrix on Figure 1.

The additional column (Σ) collects the sum of the distances (at all vertices) for each vertex and can also be used to discriminate vertices (the vertices are represented with different colors accordingly in Figure 3) and as a criterion that can be used to create a vertices partition (for $\Sigma_j D_{i,j}$ the partition is in four groups: $\{4\}, \{1,3\}, \{2\}, \{5\}$—here, the groups are ordered according to the ascending value of $\Sigma_j D_{i,j}$).

For undirected graphs $[Ad]$ and $[Di]$ are always symmetrical.

Another square matrix that collects the properties of graphs is Szeged matrix ($[Szd]$, Figure 4), in which each entry counts the number of vertices closer to the vertex with the same index with the line, than to the vertex with the same index with the column (Algorithm A9 in Appendix B). Example: since $d_{1,1} < d_{1,2}$, $d_{1,2} > d_{2,2}$, $d_{1,3} > d_{2,3}$, $d_{1,4} < d_{2,4}$, $d_{1,5} < d_{2,5}$ (see $[Di]$ in Figure 3), results that 1, 4 and 5 are closer to 1 than to 2 and 2 and 3 are closer to 2 than to 1 (and $Szd_{1,2} = 3$ and $Szd_{2,1} = 2$).

$[Szd]$	1	2	3	4	5	Σ
1	0	3	1	2	2	8
2	2	0	2	1	3	8
3	1	3	0	2	2	8
4	3	2	3	0	4	12
5	1	2	1	1	0	5



$$Szd_{i,j} \leftarrow |\{k | d_{i,k} < d_{j,k}\}|$$

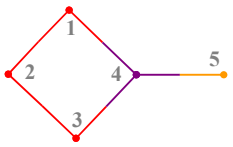
Figure 4. walk matrix on Figure 1.

$[Szd]$ is an important example, since unlike $[Ad]$ and $[Di]$ is unsymmetrical.

3. Counting Matrices

A counting matrix (let us call it $[CM]$) it is obtained from a vertex pair based square matrix (let us say $[MA]$) by counting distinct values for each vertex ($CM_{i,k} \leftarrow |\{MA_{i,j} | MA_{i,j} = k\}|$; Algorithm A11 in Appendix B). Counting $[Ad]$, $[Di]$, and $[Szd]$ are given in Figures 5–7.

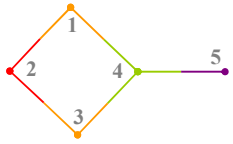
$[Adc]$	0	1	"."
1	3	2	3.2
2	3	2	3.2
3	3	2	3.2
4	2	3	2.3
5	4	1	4.1



$$Adc_{i,k} \leftarrow |\{Ad_{i,j}|Ad_{i,j} = k\}|$$

Figure 5. $[Adc]$ (counting for $[Ad]$) on Figure 1.

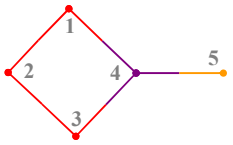
$[Dic]$	0	1	2	3	"."
1	1	2	2	0	1.2.2.0
2	1	2	1	1	1.2.1.1
3	1	2	2	0	1.2.2.0
4	1	3	1	0	1.3.1.0
5	1	1	2	1	1.1.2.1



$$Dic_{i,k} \leftarrow |\{Di_{i,j}|Di_{i,j} = k\}|$$

Figure 6. $[Dic]$ (counting for $[Di]$) on Figure 1.

$[Szc]$	0	1	2	3	4	"."
1	1	1	2	1	0	1.1.2.1.0
2	1	1	2	1	0	1.1.2.1.0
3	1	1	2	1	0	1.1.2.1.0
4	1	0	1	2	1	1.0.1.2.1
5	1	3	1	0	0	1.3.1.0.0



$$Szc_{i,k} \leftarrow |\{Szd_{i,j}|Szd_{i,j} = k\}|$$

Figure 7. [Szc] (counting for $[Szd]$) on Figure 1.

Counting matrices (by their definition) are always asymmetric. Another important property of the counting matrices is that always the sum of the elements is the same for any vertex (see columns Σ in Figures 2–4). Another classifier is introduced and is useful here. We will call it dot classifier ("." column in Figures 5–7). The numerical ordering (of the values given in the Σ columns in Figures 2–4) can be replaced with lexicographic ordering (as for the values given in the "." column in Figures 5–7).

4. Collecting Sets of Vertices

An important step forward to generalize (on the one hand) and simplify (on the other hand) sequencing and layering (to be defined) is to collect sets of vertices that meet certain criteria instead of their count (Algorithm A10 in Appendix B). This procedure slightly changes the previous one: from $CM_{i,k} = |\{MA_{i,j}|MA_{i,j} = k\}|$ to $LM_{i,k} = \{MA_{i,j}|MA_{i,j} = k\}$. As a result, Tables 1–3 contain the layers ($[LA0]$, $[LD0]$, $[LS0]$) associated with the counts from $[Adc]$, $[Dic]$ and $[Szc]$.

Collecting (instead of counting) defines layers natively (see Tables 1–3). Once obtained, the layers can easily be exploited to build other layer matrices.

Table 1. Adjacency layers ($LA0_{i,k} \leftarrow \{j|Ad_{i,j} = k\}$) for Figure 1.

$[\boldsymbol{LA0}]$	**0**	**1**
1	{1, 3, 5}	{2, 4}
2	{2, 4, 5}	{1, 3}
3	{1, 3, 5}	{2, 4}
4	{2, 4}	{1, 3, 5}
5	{1, 2, 3, 5}	{4}

Table 2. Distance layers ($LD0_{i,k} \leftarrow \{j|Di_{i,j} = k\}$) for Figure 1.

$[LD0]$	**0**	**1**	**2**	**3**
1	{1}	{2, 4}	{3, 5}	{}
2	{2}	{1, 3}	{4}	{5}
3	{3}	{2, 4}	{1, 5}	{}
4	{4}	{1, 3, 5}	{2}	{}
5	{5}	{4}	{1, 3}	{2}

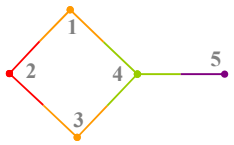
Table 3. Szeged layers ($LS0_{i,k} \leftarrow \{j|Szd_{i,j} = k\}$) for Figure 1.

$[LS0]$	**0**	**1**	**2**	**3**	**4**
1	{1}	{3}	{4, 5}	{2}	{}
2	{2}	{4}	{1, 3}	{5}	{}
3	{3}	{1}	{4, 5}	{2}	{}
4	{4}	{}	{2}	{1, 3}	{5}
5	{5}	{1, 3, 4}	{2}	{}	{}

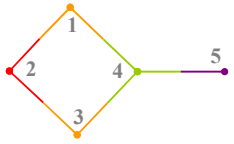
5. Layer Matrices

The first reported layer matrix was for distance [59] and is the same as distance counting ($[LD1] \leftarrow [Dic]$ from Figure 6; $LD1 \leftarrow Dic$ in Algorithm A18 in Appendix B). In general, a layer matrix collects (as Σ) a property for all vertices belonging to the layer (for example for entry for vertex 1 and layer 2 of the Szeged layers in Figure 1 given in Table 3, a layer matrix will apply a sum of a property to $\{4, 5\}$ as being the set of all vertices that belong to the layer). As the power of discrimination of any topological descriptor is limited (and for layer matrices as well), other layer matrices has been proposed to better take into account for branching, edges, and their sum (matrices B, E and S in [74]; $[LD2]$, $[LD3]$, and $[LD4]$ in Figures 8–10 below; Algorithms A4–A6 in Appendix B).

$[LD1]$	0	1	2	3	"."
1	1	2	2	0	1.2.2.0
2	1	2	1	1	1.2.1.1
3	1	2	2	0	1.2.2.0
4	1	3	1	0	1.3.1.0
5	1	1	2	1	1.1.2.1
$[LD2]$	0	1	2	3	"."
1	2	5	3	0	2.5.3.0
2	2	4	3	1	2.4.3.1
3	2	5	3	0	2.5.3.0
4	3	5	2	0	3.5.2.0
5	1	3	4	2	1.3.4.2



$LD1_{i,k} = Dic_{i,k}$
$(LD1_{i,k} \leftarrow Dic_{i,k})$



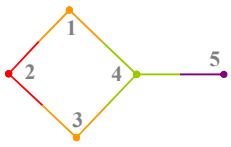
$LD2_{i,k} \leftarrow$
$\Sigma_j Adc_{j,1}$ for $j \in LD0_{i,k}$

Figure 8. $[LD2]$ on Figure 1.

Note that column 0 of $[LD2]$ is identical to column 1 of $[LD1]$, which is an important constructive property, which reappears later at walks degrees. For example, in Figure 8 $LD2_{1,1}$ is 5 (= $Adc_{2,1} + Adc_{4,1}$) since $LD0_{1,1} = \{2, 4\}$ (see $[LD0]$ in Table 2) and $Adc_{2,1} = 2$ and $Adc_{4,1} = 3$ (see $[Adc]$ in Figure 5).

$[LD3]$ counts distinct edges incident with the vertices in $[LD0]$, without counting any edge that has already been counted in a previous layer. The counting for edges and the counting for adjacent vertices are the same (Figure 9). For example, in Figure 9, since $LD0_{1,1} = \{2, 4\}$ from all edges (5; all with endpoints in 2 or 4) there remains only 3 not counted previously ((1, 2) and (1, 4) counted for $LD3_{1,0}$) to be counted for $LD3_{1,1}$.

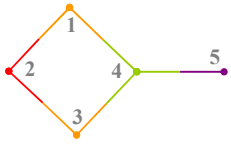
$[LD3]$	0	1	2	3	"."
1	2	3	0	0	2.3.0.0
2	2	2	1	0	2.2.1.0
3	2	3	0	0	2.3.0.0
4	3	2	0	0	3.2.0.0
5	1	2	2	0	1.2.2.0



$LD3_{i,k} \leftarrow |\{(u,v)\}|$,
where
$u \in LD0_{i,k}$
$Ad_{u,v} = 1$
$v \notin LD0_{i,q} \quad \forall q < k$

Figure 9. $[LD3]$ on Figure 1.

$[LD4]$	0	1	2	3	"."
1	4	8	3	0	4.8.3.0
2	4	6	4	1	4.6.4.1
3	4	8	3	0	4.8.3.0
4	6	7	2	0	6.7.2.0
5	2	5	6	2	2.5.6.2



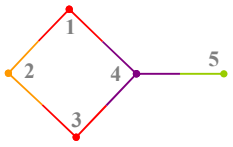
$LD4_{i,k} \leftarrow LD2_{i,k} + LD3_{i,k}$

Figure 10. $[LD4]$ on Figure 1.

Even if matrices $[LD2]$ to $[LD4]$ have been developed to avoid the degeneracy of $[LD1]$ ([74]), this behavior cannot be avoided ($[LD1]$ to $[LD4]$ generates same partition for vertices, see Figures 8–10) for simple graphs, such as the one in Figure 1.

As can be seen in Figures 8–10, different from the counting matrices (see Figures 5–7), all layer matrices have the same number of layers because of the counts from 0 to the diameter of the graph. Another layer matrix introduced is one of distance sums (R matrix in [75]; $[LD5]$ in Figure 11; see Algorithm A7 in Appendix B), which, once again, results naturally from $[LD0]$. As an example, $LD5_{1,0} \leftarrow 6 \leftarrow \Sigma_j Di_{1,j}$ ($LD0_{1,0} = \{1\}$).

$[LD5]$	0	1	2	3	"."
1	6	12	14	0	6.12.14.0
2	7	12	5	8	7.12.05.8
3	6	12	14	0	6.12.14.0
4	5	20	7	0	5.20.07.0
5	8	5	12	7	8.05.12.7



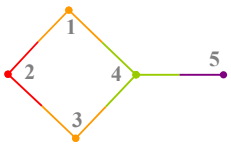
$Di_{i,\cdot} \leftarrow \Sigma_j Di_{i,j}$
$LD5_{i,k} \leftarrow \Sigma_u Di_{u,\cdot}$,
$u \in LD0_{i,k}$

Figure 11. $[LD5]$ on Figure 1.

6. Sequence Matrices

Vertices similarity analysis can be involved beyond layers. One strategy is to build edge sequences. According to [76], a sequence matrix is a collection of walks (of increasing elongation) starting from each of the vertices to all the others, in opposition to a layer matrix collecting the properties of vertices u located in concentric shells (layers). The walks degrees (and their layers) are derived from rising to powers (up to the diameter of the graph) of the adjacency matrix and the subsequent collecting of the traces. Alternatively, the calculation of the layers of walk degrees of increasing length can be shortened using, along with the adjacency matrix ($[Ad]$ from Figure 2), the layers of distance ($[LD0]$ from Table 2), in an iterative algorithm (see Algorithm A8 in Appendix B; see Figures 12–14, as well as Figures 12–14, which contain the resulted matrices $[LW1]$ to $[LW3]$ for Figure 1). For instance, $LW1_{1,1} \leftarrow 2 + 3 \leftarrow LW1_{2,0} + LW1_{4,0}$, $LW2_{1,1} \leftarrow 4 + 5 \leftarrow LW2_{2,0} + LW2_{4,0}$ and $LW3_{1,1} \leftarrow 10 + 13 \leftarrow LW3_{2,0} + LW3_{4,0}$ since $LD0_{1,1} = \{2, 4\}$.

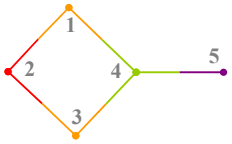
$[LW1]$	0	1	2	3	"."
1	2	5	3	0	2.5.3.0
2	2	4	3	1	2.4.3.1
3	2	5	3	0	2.5.3.0
4	3	5	2	0	3.5.2.0
5	1	3	4	2	1.3.4.2



$LW1_{i,0} \leftarrow Adc_{i,1}$
$LW1_{i,k} \leftarrow \Sigma_u LW1_{u,0}$,
$u \in LD0_{i,k}, \quad k > 0$

Figure 12. $[LW1]$ on Figure 1.

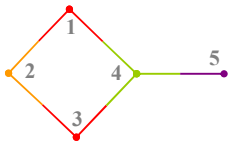
[*LW*2]	0	1	2	3	"."
1	5	9	8	0	5.09.08.0
2	4	10	5	3	4.10.05.3
3	5	9	8	0	5.09.08.0
4	5	13	4	0	5.13.04.0
5	3	5	10	4	3.05.10.4



$$LW2_{i,0} \leftarrow LW1_{i,1}$$
$$LW2_{i,k} \leftarrow \Sigma_u LW2_{u,0},$$
$$u \in LD0_{i,k}, \quad k > 0$$

Figure 13. [*LW*2] on Figure 1.

[*LW*3]	0	1	2	3	"."
1	9	23	14	0	09.23.14.00
2	10	18	13	5	10.18.13.05
3	9	23	14	0	09.23.14.00
4	13	23	10	0	13.23.10.00
5	5	13	18	10	05.13.18.10



$$LW3_{i,0} \leftarrow LW2_{i,1}$$
$$LW3_{i,k} \leftarrow \Sigma_u LW3_{u,0},$$
$$u \in LD0_{i,k}, \quad k > 0$$

Figure 14. [*LW*3] on Figure 1.

Walks degrees (and their layers) are connected with the Wiener index, the half sum of all entries in [*Di*] and the sum of all distances [77]. On the first layer (layer 0), [*LW*1] collects the valencies of the vertices ($LW1_{i,0} \leftarrow Adc_{i,1}$), while any other higher degree collects on the first (0) layer, which the previous one collected on the second (1) layer ($LW(e)_{i,0} \leftarrow LW(e-1)_{i,1}$), while the rest of the layers are built purely on the distance layers ([*LD*0] from Table 2): $LWe_{i,k} \leftarrow \Sigma_u LWe_{u,0}$ for $u \in LD0_{i,k}$. The calculation given by [$LW(e)$] is equivalent to the walk degrees by iterative summation over all neighbors, as Morgan proposed through their extended connectivity, *ECs*, to provide a unique representation for chemical structures [78].

7. Paths and Cycles

As mentioned in the beginning, when implying the distances (in graphs), one cannot escape from introducing paths. Unfortunately, to compute all paths in a graph is an NP (non-polynomial) hard problem [79] (its complexity increases in an non-polynomial manner) and it quickly goes 'out of memory' for any medium sized graph. For instance, when listing all paths for an isomer of C_{28} fullerene (see below) the output alone contains over 1.5 million lines. Therefore, there may be a real interest for a shortened version of them, for example listing only paths less than or equal to the diameter of the graph (among those are the distance paths; Algorithm A12 in Appendix B). Such a procedure is significantly faster, and its complexity is limited by the diameter of the graph. The result (the paths list) can be further processed, and collected in a matrix form for each pair of vertices (Algorithm A14 in Appendix B); the result is labeled [*SP*0] and is listed in Table 4 for Figure 1).

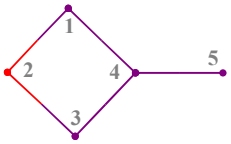
Table 4. Diameter limited paths for Figure 1.

[*SP*0]	1	2	3	4	5
1	{}	{1 2}	{1 4 3, 1 2 3}	{1 4}	{1 4 5}
2	{2 1}	{}	{2 3}	{2 3 4, 2 1 4}	{2 1 4 5, 2 3 4 5}
3	{3 4 1, 3 2 1}	{3 2}	{}	{3 4}	{3 4 5}
4	{4 1}	{4 3 2, 4 1 2}	{4 3}	{}	{4 5}
5	{5 4 1}	{5 4 1 2, 5 4 3 2}	{5 4 3}	{5 4}	{}

$SP0_{i,j}$ is the set of paths between i (the line index in Table 4) and j (the column index in Table 4) that connects the vertices with the smallest set of edges. In general [*SP*0] is a complex multi-path structure that contains all distance paths between pairs of vertices. As can be seen, for instance between 1 and 3 (either of $SP0_{1,3}$ and $SP0_{3,1}$) in Table 4, the two shortest paths connecting those two vertices are 1 4 3 and 1 2 3. The simplest operation on the set of paths is counting of the paths, and the result is given as [*SP*1] (Figure 15), while

the other operation can be counting the vertices (both in Algorithm A16 in Appendix B), and the result is given as $[SP2]$ (Figure 16).

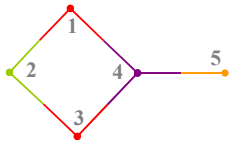
$[SP1]$	1	2	3	4	5	Σ
1	0	1	2	1	1	5
2	1	0	1	2	2	6
3	2	1	0	1	1	5
4	1	2	1	0	1	5
5	1	2	1	1	0	5



$$SP1_{i,j} \leftarrow |SP0_{i,j}|$$

Figure 15. $[SP1]$ on Figure 1.

$[SP2]$	1	2	3	4	5	Σ
1	0	2	6	2	3	13
2	2	0	2	6	8	18
3	6	2	0	2	3	13
4	2	6	2	0	2	12
5	3	8	3	2	0	16



$$SP2_{i,j} \leftarrow \Sigma_p|p|,$$
$$p \in SP0_{i,j}$$
$$(i \rightarrow v_1, \ldots, v_k \leftarrow j \in V)$$
$$p \leftarrow v_1, \ldots, v_k \Rightarrow |p| = k$$

Figure 16. $[SP2]$ on Figure 1.

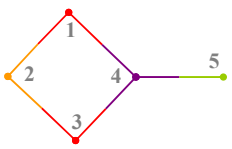
Layers on/from the sequences of paths can be generated too. The procedure is immediate and operates on the sequences already collected in $[SP0]$ (see Table 4; both in Algorithm A14 in Appendix B) and the result is given as $[LP0]$ (see Table 5).

Table 5. Layers of diameter limited paths for Figure 1.

$[LP0]$	0	1	2	3
1	{}	{1 2, 1 4}	{1 4 3, 1 4 5, 1 2 3}	{}
2	{}	{2 1, 2 3}	{2 3 4, 2 1 4}	{2 1 4 5, 2 3 4 5}
3	{}	{3 2, 3 4}	{3 4 5, 3 4 1, 3 2 1}	{}
4	{}	{4 1, 4 3, 4 5}	{4 3 2, 4 1 2}	{}
5	{}	{5 4}	{5 4 3, 5 4 1}	{5 4 1 2, 5 4 3 2}

$LP0_{i,j}$ is the set of paths starting from i (the line index in Table 5) and having a number of k (the column index in Table 5) edges connecting the vertices with the smallest set of edges. In general, $[LP0]$ is a complex multi-path structure containing all distance paths between the pairs of vertices ($LP0_{i,k} \leftarrow p|p \in SP0_{i,\cdot}, |p| = max(k-1, 0)$, k from $p \leftarrow v_1 \ldots v_k \Rightarrow |p| = k$). As can be seen in Table 5, for Figure 1, paths starting from 1 and having two edges ($LP0_{1,2}$), are three shortest paths fitting the description (1 4 3, 1 4 5, and 1 2 3); two of them begin and end in the same vertices (1 4 3 and 1 2 3). The simplest operation on it is counting of the paths, and the result is given as $[LP1]$ (Figure 17), while the other operation can be counting the vertices (both in Algorithm A16 in Appendix B), and the result is given as $[LP2]$ (Figure 18).

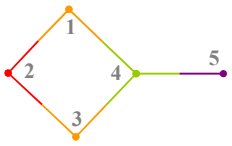
$[LP1]$	0	1	2	3	"."
1	0	2	3	0	0.2.3.0
2	0	2	2	2	0.2.2.2
3	0	2	3	0	0.2.3.0
4	0	3	2	0	0.3.2.0
5	0	1	2	2	0.1.2.2



$$LP1_{i,k} = |LP0_{i,k}|$$

Figure 17. $[LP1]$ on Figure 1.

$[LP2]$	0	1	2	3	"."
1	0	4	9	0	0.4.9.0
2	0	4	6	8	0.4.6.8
3	0	4	9	0	0.4.9.0
4	0	6	6	0	0.6.6.0
5	0	2	6	8	0.2.6.8



$$LP2_{i,j} \leftarrow \Sigma_p|p|,$$
$$p \in LP0_{i,j}$$
$$(i \rightarrow v_1, \ldots, v_k \in V)$$
$$p \leftarrow v_1, \ldots, v_k \Rightarrow |p| = k$$

Figure 18. $[LP2]$ on Figure 1.

Once generated, another immediate (from generating paths) result of the (limited by diameter) paths (Table 4) is generating (small, diameter limited) cycles (Algorithm A13 in Appendix B). A small cycle can be seen as being built from two distance paths at which we may need to add an edge to enclose a cycle. Please note that this assertion (that a cycle is build up from two distance paths) is not true in general, we may regard the cycle's build-up in this way as no bigger than double the diameter. The result of applying this procedure for Figure 1 is shown in Table 6 and their layers in Table 7 (Figure 1 have only one cycle and it is built up on two distance paths).

Table 6. Cycles no bigger than the double of the diameter for Figure 1.

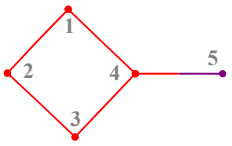
$[SC0]$	**1**	**2**	**3**	**4**	**5**
1	{}	{1 2 3 4}	{1 2 3 4}	{1 2 3 4}	{}
2	{1 2 3 4}	{}	{1 2 3 4}	{1 2 3 4}	{}
3	{1 2 3 4}	{1 2 3 4}	{}	{1 2 3 4}	{}
4	{1 2 3 4}	{1 2 3 4}	{1 2 3 4}	{}	{}
5	{}	{}	{}	{}	{}

Table 7. Layers of the cycles no bigger than the double of the diameter for Figure 1.

$[LC0]$	**3**	**4**
1	{}	{1 2 3 4}
2	{}	{1 2 3 4}
3	{}	{1 2 3 4}
4	{}	{1 2 3 4}
5	{}	{}

Similarly with the paths, two sequence and two layer matrices can be constructed on the sets of the cycles (Algorithm A15 in Appendix B for the sets of sequences and layers; Algorithm A16 in Appendix B for the sequence and layer matrices). Therefore, since $[SC0]$ is similar in structure (both containing as entries lists of paths) with $[SP0]$ and $[LC0]$ similar in structure with $[LP0]$, then the calculation for $[SC1]$, $[SC2]$, $[LC1]$, and $[LC2]$ is similar with for $[SP1]$, $[SP2]$, $[LP1]$, and $[LP2]$, respectively, (Tables 6 and 7 vs. Tables 4 and 5; Figures 19–22 vs. Figures 15–18).

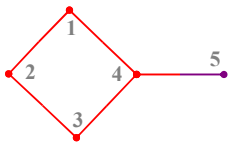
$[SC1]$	1	2	3	4	5	Σ
1	0	1	1	1	0	3
2	1	0	1	1	0	3
3	1	1	0	1	0	3
4	1	1	1	0	0	3
5	0	0	0	0	0	0



$$SC1_{i,j} \leftarrow |SC0_{i,j}|$$

Figure 19. $[SC1]$ on Figure 1.

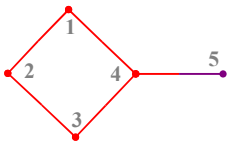
$[SC2]$	1	2	3	4	5	Σ
1	0	4	4	4	0	12
2	4	0	4	4	0	12
3	4	4	0	4	0	12
4	4	4	4	0	0	12
5	0	0	0	0	0	0



$SC2_{i,j} \leftarrow \Sigma_c|c|,$
$c \in SC0_{i,j}$
$(i \rightarrow v_1, \ldots, v_k \leftarrow j \in V)$
$c \leftarrow v_1, \ldots, v_k \Rightarrow |c| = k$

Figure 20. $[SC2]$ on Figure 1.

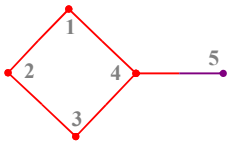
$[LC1]$	3	4	"."
1	0	1	0.1
2	0	1	0.1
3	0	1	0.1
4	0	1	0.1
5	0	0	0.0



$LC1_{i,k} = |LC0_{i,k}|$

Figure 21. $[LP1]$ on Figure 1.

$[LC2]$	3	4	"."
1	0	4	0.4
2	0	4	0.4
3	0	4	0.4
4	0	4	0.4
5	0	0	0.0



$LC2_{i,j} \leftarrow \Sigma_c|c|,$
$c \in LC0_{i,j}$
$(i \rightarrow v_1, \ldots, v_k \in V)$
$c \leftarrow v_1, \ldots, v_k \Rightarrow |c| = k$

Figure 22. $[LP2]$ on Figure 1.

A distinctiveness between paths and cycles is that one cannot have cycles listed with less than three vertices, and as an effect, their layers start from 3 (Figures 21 and 22).

As a natural extension of generating sequences and sets of sequences for paths (Table 4) and cycles (Table 6), but also as a natural extension of generating the adjacency, distance and Szeged layers (Tables 1–3), another upgraded structure (upgraded from Table 3 and Figure 4) results—Szeged sets (or actually, sets of connected vertices or, in other words, fragments; Algorithm A17 in Appendix B) from collecting vertices instead of counting them (as $[Szd]$ do, Figure 4). The result is given in Table 8.

Table 8. Szeged fragments for Figure 1.

$[Szs]$	**1**	**2**	**3**	**4**	**5**
1	{}	{1, 4, 5}	{1}	{1, 2}	{1, 2}
2	{2, 3}	{}	{1, 2}	{2}	{1, 2, 3}
3	{3}	{3, 4, 5}	{}	{2, 3}	{2, 3}
4	{3, 4, 5}	{4, 5}	{1, 4, 5}	{}	{1, 2, 3, 4}
5	{5}	{4, 5}	{5}	{5}	{}

As mentioned above at $[Szd]$ (Figure 4), an important characteristic of $[Szs]$ too (Table 8) is its asymmetry.

8. Distinct Partitions Coloring of Vertices

As can be seen in the illustrations given above, either the Σ operator for square symmetrical matrices cumulating properties for pairs of vertices and either "." operator for layer matrices are able to produce different partitions of the vertices in the graphs.

Shifting from numbers (here all integer, thus from ordinal scale) or integer sequences (separated with ".", sortable, thus from an induced order scale) to colors, it hardly makes sense to keep the order relationship alive (someone may argue that the wavelength is an ordering operator, but is out of the scope of its use here).

Besides, when dealing with categories (multinominal, multinomial scales) in most of the cases, it is more important to keep the undistinctivness alive. Let us take here an

example: Table 9 lists the side by side the values of Σ operator on $[Szd]$ against the values of "." operator on $[Szc]$. Coloring of the vertices has been made (for any of the Figures 2–22) using the colors from the {Violet ■, Red ■, Light orange ■, Lime ■, Sea green ■, Aqua ■, Light blue ■} ordered set based on the operator induced partition sets.

Table 9. Two orders for the same partition of Figure 1 vertices.

Vertices	$\Sigma_j Szd_{i,j}$	$"."_k Szc_{i,k}$
1	8	1.1.2.1.0
2	8	1.1.2.1.0
3	8	1.1.2.1.0
4	12	1.0.1.2.1
5	5	1.3.1.0.0

In some cases, it is of interest to discriminate among the two cases (see the two side by side colorings in Figure 23)—when the order of the sets in the partition is relevant, but of importance is also listing only distinct partitions when the order of the sets in the partition is not relevant. In this later case falls the example given in Table 9 and Figure 23—because actually both operators provide the same groups of vertices: ({1, 2, 3}, {4}, {5}).

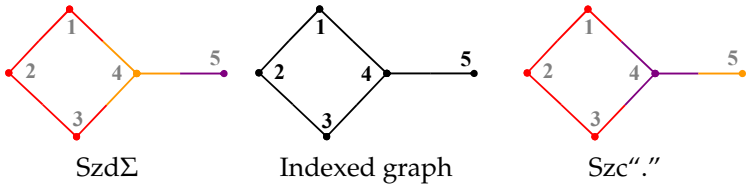


Figure 23. Same partition, different coloring of vertices for Figure 1 (see Table 9).

A supplementary treatment of the information is required to alleviate this distinctiveness. Either way, it is a matter of deciding if the order of the groups is relevant or not (distinctiveness vs. undistinctiveness). Accounting for both, two different groups of classifiers it results. Table 10 gives the results of classifiers for Figure 1 side by side.

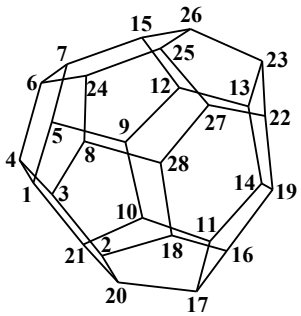
Table 10. Different partitions of vertices for Figure 1.

Order of the Vertices Sets Is Relevant	Classifier (from $[Matrix]$)	Order of the Vertices Sets Is Not Relevant
{5}, {1, 2, 3}, {4} {4}, {1, 2, 3}, {5}	Ad, Szd Adc, Szc	{1, 2, 3}, {4}, {5}
{4}, {1, 3}, {2}, {5}	Di, LD5	{1, 3}, {2}, {4}, {5}
{5}, {2}, {1, 3}, {4}	Dic, LD1, LD2, LD3, LD4, LW1, LW2, LP1, LP2	
{5}, {1, 3}, {2}, {4}	LW3	
{4}, {1,3}, {5}, {2}	SP2	
{1, 3, 4, 5}, {2}	SP1	{1, 3, 4, 5}, {2}
{5}, {1, 2, 3, 4}	SC1, SC2, LC1, LC2	{1, 2, 3, 4}, {5}

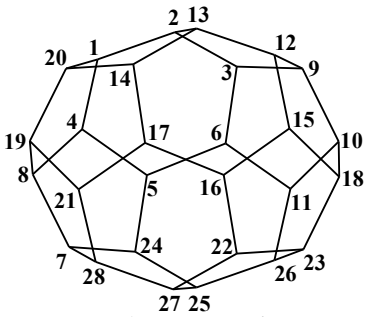
Different classifications (Table 10) are always of interest in chemistry for instance in identification of new reaction pathways [80].

9. Case Study for Isomers of C_{28} Fullerene

Fullerene is defined to have only cycles of 5 and 6 and each atom vertex to always have three neighbors (see Figure 24). Functionalization of fullerenes is of great interest for green energy [81], drug design [82], and even dentistry [83].



$C_{28} - T_d$ (isomer with T_d symmetry)



$C_{28} - D_2$ (isomer with D_2 symmetry)

Figure 24. The isomers of C_{28} fullerene.

Table 11 gives the result of the grouping partition analysis on $C_{28} - D_2$, while Table 12 gives the result of the grouping partition analysis on $C_{28} - T_d$ (images in Figures 25 and 26 for $C_{28} - D_2$ and in Figure 27 for $C_{28} - T_d$). The classifiers discussed above seem perfectly fit for this task (like Table 10 contains a summary for Figure 1 as the exemplified case).

Table 11. Different partitions of vertices for $C_{28} - D_2$.

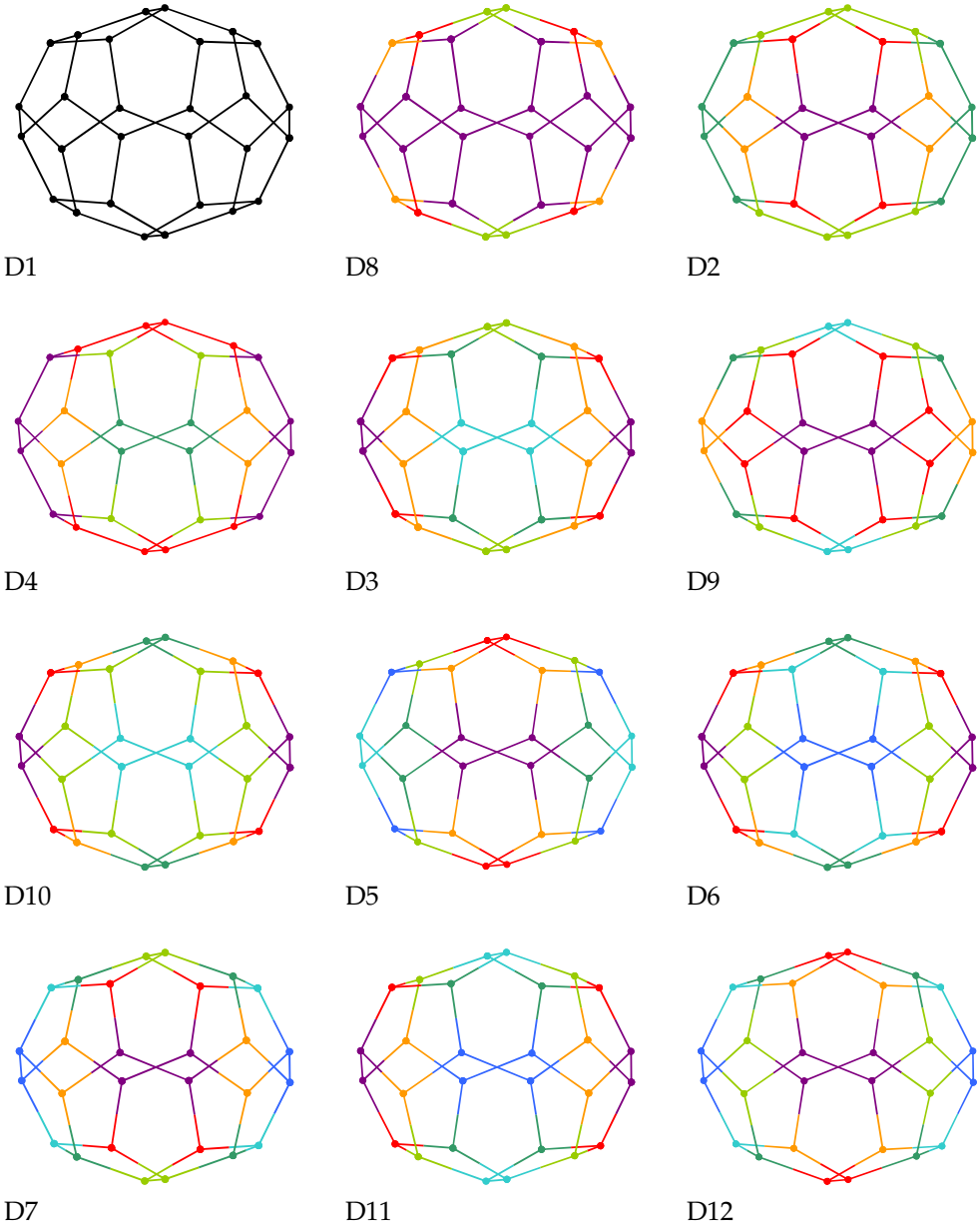
Order Is Relevant	Classifiers	Order Is Not Relevant
D1	Ad, Adc	U1
D2	Di	U2
D4	Dic, LD1, LD2, LW1, LW2, LW3, LW4, LW5, LW6	
D3	Szd	U3
D5	Szc	U4
D6	LD3, LD4	
D7	LD5	
D11	SC1, SC2	
D12	LC1, LC2	
D8	SP1	U5
D9	SP2	U6
D10	LP1, LP2	

D1–D12 partitions depicted in Figure 25. U1–U6 partitions depicted in Figure 26.

Table 12. Different partitions of vertices for $C_{28} - T_d$.

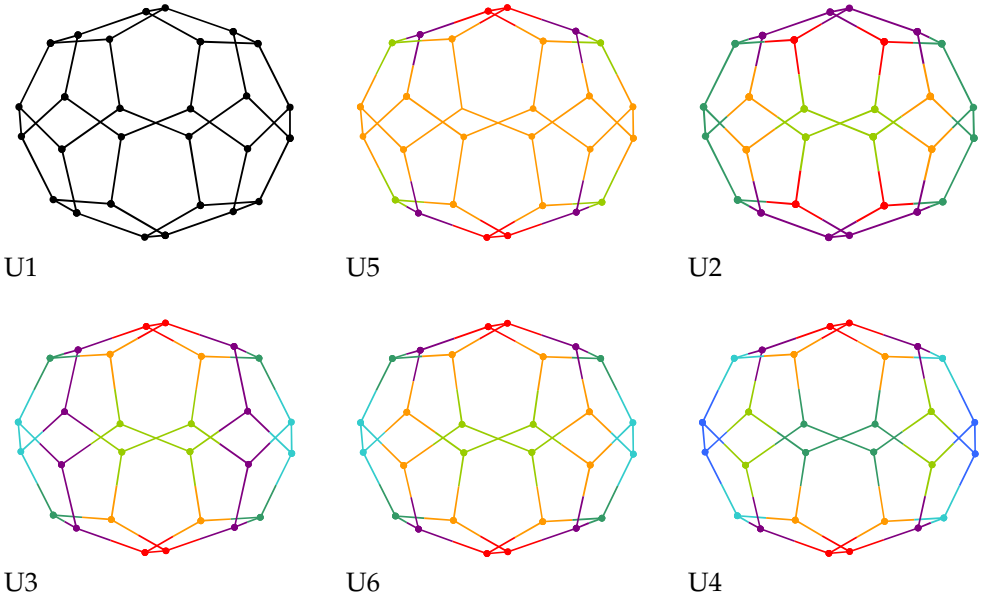
Order Is Relevant	Classifiers	Order Is Not Relevant
D1	Ad, Adc	U1
D2	Di	U2
D3	Dic, LD1, LD2, LW1, LW2, LW3, LW4, LW5, LW6, Szd	U3
D4	Szc, LD5, SP1, SP2, SC1, SC2, LC1, LC2	
D5	LD3, LD4, LP1, LP2	

D1–D5 and U1–U3 partitions depicted in Figure 26.



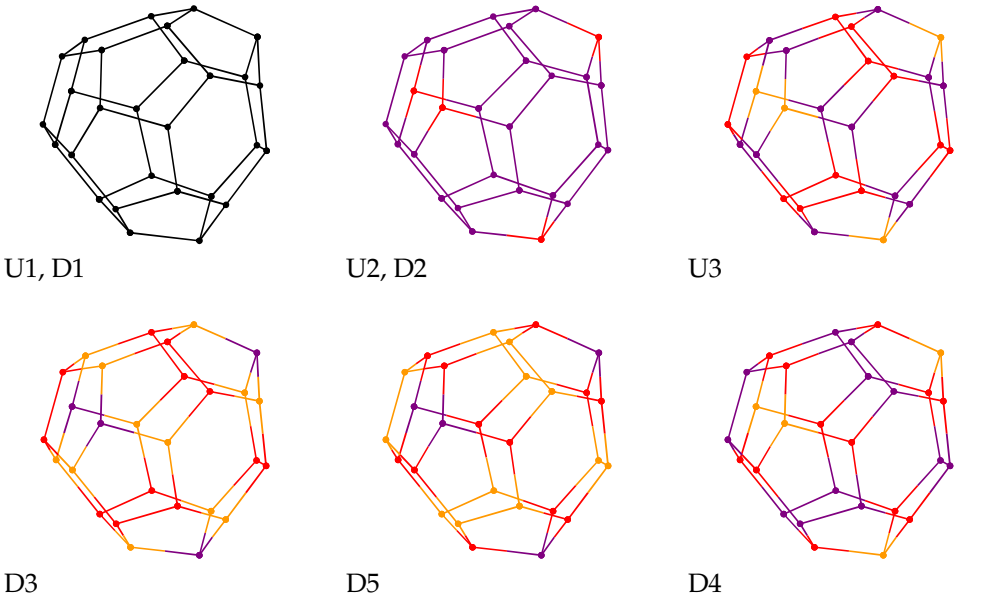
D1: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28}
D8: {3, 4, 5, 6, 8, 10, 11, 14, 15, 16, 17, 18, 19, 21, 22, 24}, {1, 12, 26, 28}, {7, 9, 20, 23}, {2, 13, 25, 27}
D2: {5, 6, 16, 17}, {3, 14, 22, 24}, {4, 11, 15, 21}, {1, 2, 12, 13, 25, 26, 27, 28}, {7, 8, 9, 10, 18, 19, 20, 23}
D4: {7, 8, 9, 10, 18, 19, 20, 23}, {1, 2, 12, 13, 25, 26, 27, 28}, {4, 11, 15, 21}, {3, 14, 22, 24}, {5, 6, 16, 17}
D3: {8, 10, 18, 19}, {7, 9, 20, 23}, {1, 4, 11, 12, 15, 21, 26, 28}, {2, 13, 25, 27}, {3, 14, 22, 24}, {5, 6, 16, 17}
D9: {5, 6, 16, 17}, {3, 4, 11, 14, 15, 21, 22, 24}, {8, 10, 18, 19}, {1, 12, 26, 28}, {7, 9, 20, 23}, {2, 13, 25, 27}
D10: {8, 10, 18, 19}, {7, 9, 20, 23}, {1, 12, 26, 28}, {3, 4, 11, 14, 15, 21, 22, 24}, {2, 13, 25, 27}, {5, 6, 16, 17}
D5: {5, 6, 16, 17}, {{2, 13, 25, 27}, {3, 14, 22, 24}, {1, 12, 26, 28}, {4, 11, 15, 21}, {8, 10, 18, 19}, {7, 9, 20, 23}
D6: {8, 10, 18, 19}, {7, 9, 20, 23}, {1, 12, 26, 28}, {4, 11, 15, 21}, {2, 13, 25, 27}, {3, 14, 22, 24}, {5, 6, 16, 17}
D7: {5, 6, 16, 17}, {3, 14, 22, 24}, {4, 11, 15, 21}, {2, 13, 25, 27}, {1, 12, 26, 28}, {7, 9, 20, 23}, {8, 10, 18, 19}
D11: {8, 10, 18, 19}, {7, 9, 20, 23}, {4, 11, 15, 21}, {1, 12, 26, 28}, {3, 14, 22, 24}, {2, 13, 25, 27}, {5, 6, 16, 17}
D12: {5, 6, 16, 17}, {2, 13, 25, 27}, {3, 14, 22, 24}, {4, 11, 15, 21}, {1, 12, 26, 28}, {7, 9, 20, 23}, {8, 10, 18, 19}

Figure 25. Distinguishable partitions on $C_{28} - D_2$ from Figure 24.



U1: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28}
U5: {1, 12, 26, 28}, {2, 13, 25, 27}, {3, 4, 5, 6, 8, 10, 11, 14, 15, 16, 17, 18, 19, 21, 22, 24}, {7, 9, 20, 23}
U2: {1, 2, 12, 13, 25, 26, 27, 28}, {3, 14, 22, 24}, {4, 11, 15, 21}, {5, 6, 16, 17}, {7, 8, 9, 10, 18, 19, 20, 23}
U3: {1, 4, 11, 12, 15, 21, 26, 28}, {2, 13, 25, 27}, {3, 14, 22, 24}, {5, 6, 16, 17}, {7, 9, 20, 23}, {8, 10, 18, 19}
U6: {1, 12, 26, 28}, {2, 13, 25, 27}, {3, 4, 11, 14, 15, 21, 22, 24}, {5, 6, 16, 17}, {7, 9, 20, 23}, {8, 10, 18, 19}
U4: {1, 12, 26, 28}, {2, 13, 25, 27}, {3, 14, 22, 24}, {4, 11, 15, 21}, {5, 6, 16, 17}, {7, 9, 20, 23}, {8, 10, 18, 19}

Figure 26. Undistinguishable partitions on $C_{28} - D_2$ from Figure 24.



U1, D1: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28}
U2, D2: {1, 2, 3, 4, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28}, {5, 8, 17, 23}
U3: {1, 3, 7, 9, 11, 13, 16, 20, 22, 24, 26, 28}, {2, 4, 6, 10, 12, 14, 15, 18, 19, 21, 25, 27}, {5, 8, 17, 23}
D3: {5, 8, 17, 23}, {2, 4, 6, 10, 12, 14, 15, 18, 19, 21, 25, 27}, {1, 3, 7, 9, 11, 13, 16, 20, 22, 24, 26, 28}
D5: {5, 8, 17, 23}, {1, 3, 7, 9, 11, 13, 16, 20, 22, 24, 26, 28}, {2, 4, 6, 10, 12, 14, 15, 18, 19, 21, 25, 27}
D4: {2, 4, 6, 10, 12, 14, 15, 18, 19, 21, 25, 27}, {1, 3, 7, 9, 11, 13, 16, 20, 22, 24, 26, 28}, {5, 8, 17, 23}

Figure 27. Partitions on $C_{28} - T_d$ from Figure 24.

Regarding the pairing (U2, D2) appearing for $C_{28} - T_d$ (Figure 27), this pairing does not appear for $C_{28} - D_2$ (Figure 25 vs. Figure 26) suggesting that its occurrence is again due to the increased symmetry of $C_{28} - T_d$ than $C_{28} - D_2$.

Fullerenes are structures [84] with a high symmetry, stabilized by resonance, in which the difference between different positions (atoms) are very small and are of interest for their reactivity and functionalization [85]. Following this idea, of interest is identifying, visually if possible, different equivalent positions in the structures. The counting, sequence and layer matrices just do this.

As expected, with the increasing symmetry, the possibilities of distinguishing between the vertices (here atoms) are diminished. Thus, if the selected classifiers make 12 distinct classifications for $C_{28} - D_2$ (from which 6 in which order of the vertices sets are not relevant), only 5 were created for the more symmetrical $C_{28} - T_d$ congener (actually 4 considering that *Ad* and *Adc* do not distinguish between the vertices) from which only 3 (actually 2 considering that *Ad* and *Adc* do not distinguish between the vertices) patterns the vertices in sets in which the order of the vertices sets is not relevant.

Taking into account two relatively large structures (the C_{28} fullerene isomers), it allow us to point out once again that the collections of cycles $[SC0]$ (and $[LC0]$) do not represent all cycles or the smallest cycles, but only a particular set of them—the ones no bigger than the diameter. In general, many cycles are omitted by joining together two distance paths. For example, the diameter of $C_{28} - D_2$ is 6 (and a simple check on its distance matrix will proof this), a distance (or shortest) path of length 6 (in it) is 1 2 3 9 10 18 23, and it is used to build more than a cycle (to be exact, three cycles of length 12): 1 2 3 9 10 18 23 22 16 17 14 20, 1 2 3 9 10 18 23 26 11 6 5 4, and 1 2 3 9 10 18 23 26 25 24 5 4. However, $C_{28} - D_2$ obviously contains cycles larger than 12. Actually it contains 840 distinct Hamiltonian cycles (having all 28 vertices in it), from which exactly 3 containing the distance path 1 2 3 9 10 18 23 (1 2 3 9 10 18 23 22 27 25 26 11 6 5 24 7 28 21 17 16 15 12 13 14 20 19 8 4, 1 2 3 9 10 18 23 22 27 28 7 24 25 26 11 6 5 4 8 19 21 17 16 15 12 13 14 20, and 1 2 3 9 10 18 23 22 27 28 21 17 16 15 12 13 14 20 19 8 7 24 25 26 11 6 5 4). Nevertheless, [SC0] and [LC0] (and all subsequent matrices) contains only cycles no larger than the double of the diameter, which in the case of $C_{28} - D_2$ is 12.

10. Conclusions

Sequence and layer matrices are introduced accompanied with an example. Some of their extensions are given as well. These matrices were introduced to discriminate among graph's vertices on one hand, and to create different degrees of distinctiveness for the graph's vertices on the other hand. Following this foundational idea, graphs were colored according to the partitions of the graph's vertices. Two alternate cases have been identified: when the order of the sets in the partition of the vertices is relevant (the sets are distinguishable by their position), and the other when the order of the sets in the partition of the vertices is not relevant (the sets are indistinguishable by their position). The analysis employed on C_{28} fullerene isomers shows that the classifiers are useful to generate a good number of different partitions and may be very helpful for scientists working in applied sciences, for functionalization of different highly symmetrical chemical structures.

Author Contributions: Conceptualization, L.J.; methodology, L.J.; software, L.J.; validation, M.A.T. and D.I.R.; formal analysis, M.A.T.; investigation, D.I.R.; resources, D.I.R.; data curation, M.A.T.; writing—original draft preparation, L.J.; writing—review and editing, M.A.T.; visualization, D.I.R.; supervision, L.J.; project administration, L.J.; funding acquisition, L.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Technical University of Cluj-Napoca open access publication grant.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All results reported in this study are supported by the data provided.

Acknowledgments: Dedicated to the memory of Mircea V. Diudea (b. 11 Nov. 1950; d. 25 Jun. 2019).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Sample Availability: Drawings of the graphs as molecules are available from corresponding author.

Appendix A. Molecular Graphs and Their Representation

A lot of software and file formats are available and are used in chemistry. For instance, PubChem exports ASNT, JSON, SDF, and XML files, from which SDF is probably most compacted one (the best size:information ratio), all storing human readable information.

The format of a SDF file is as follows:

- First three lines—reserved for compound identification
- On fourth line: the number of atoms, the number of bonds, followed by a series of (8) reserved values (numeric and string)
- An block of lines describing on each line one atom: the cartezian coordinates (x, y, and z), the symbol of the atom and a series of (12) reserved fields (numeric)
- An block of lines describing on each line one bond: two numbers acting as indices for the atoms and a third number indicating the bond order, followed by a series of (4) reserved fields (numeric)

Other common file, PDB, have the following format:

- First two lines—reserved for compound identification
- An block of lines describing on each line one atom: type of the fragment, index (numeric), symbol of the atom (1–3 characters), two other columns followed by the cartezian coordinates (x, y, and z)
- An block of lines describing on each line the topology for one atom: atom index followed by the indices of the atoms connected with it

A more compacted format, HIN, gives on the same line both the topology and the geometry for each atom:

- in between mol *number* and endmol *number* on each line one atom having on the second column the atom index, on the fourth the atom symbol, from column 8 to 11 the cartesian coordinates, on column 12 the number of bonds followed (starting with column 13) by each bond on two columns each (atom index, bond order)

Appendix B. Algorithms

For the soley purpose of molecular topology of interest is the chopped reduced structure containing only the heavy atoms (heavy than Hydrogen), and for those atoms typically is collected the list of bonds (connectivities). The description of the algorithms providing molecular topology tools starts from a list of entries describing for each atom (now vertex) its bonds (now connections, edges). Let us consider that we already have keept the molecule as a graph in memory in a tabular form as in Table A1.

Table A1. Primary molecular topology information.

tv	**0**	**1**	**...**	
1	$c_{1,0}$	$c_{1,1}$	...	$c_{1,c_{1,0}}$
...	...	...	...	...
n	$c_{n,0}$	$c_{n,1}$	...	$c_{n,c_{n,0}}$

n: number of vertices ($n \geq 2$); tv: connectivity table.

First step is to obtain the vertex adjacency matrix (Algorithm A1), and then the rest of the matrices (see Algorithm A18).

Algorithm A1 Set adjacency matrix.

Input: n, c //n←**n**, c←**tv** (see Table A1)

procedure SET_ADM(n, &c, &a)
For ($i \leftarrow 1, \ldots, n$)
for ($j \leftarrow 1, \ldots, n$) $a[i][j] \leftarrow 0$
for ($j \leftarrow 1, \ldots, c[i][0]$) $a[i][c[i][j]] \leftarrow 1$
EndFor
end procedure

Output: a //a→**Ad**, Ad—the adjacency matrix (n × n)

Algorithm A2 Set distance matrix.

Input: n, a //n←**n**, a←**Ad**

procedure SET_DIM(n, &a, &d)
$d2 \leftarrow n * n$
For ($i \leftarrow 1, \ldots, n$) For ($j \leftarrow 1, \ldots, n$)
If ($(i \neq j)$ and $(a[i][j] = 0)$)
$d[i][j] \leftarrow d2$
Else
$d[i][j] \leftarrow a[i][j]$
EndIf
EndFor EndFor
For ($k \leftarrow 1, \ldots, n$) For ($i \leftarrow 1, \ldots, n$) For ($j \leftarrow 1, \ldots, n$)
$d2 \leftarrow d[i][k] + d[k][j]$; if ($d[i][j] > d2$) $d[i][j] \leftarrow d2$
EndFor EndFor EndFor
end procedure

Output: d //d→**Di**, Di—the distance matrix (n × n)

Algorithm A3 Set graph diameter.

Input: n, d //n←**n**, d←**Di**

procedure SET_DIA(n, &d, &e)
$e \leftarrow 0$
For ($i \leftarrow 1, \ldots, n$) For ($j \leftarrow 1, \ldots, n$)
if ($e < d[i][j]$) $e \leftarrow d[i][j]$
EndFor EndFor
end procedure

Output: e //e→**d**, d—the diameter (longest distance)

Algorithm A4 Set LD2 matrix.

Input: n, e, f, g //n←**n**, e←**d**, f←**LD0**, g←**Dic**

procedure SET_LD2(n, e, &f, &g, &h)
$h \leftarrow []$
For ($i \leftarrow 1, \ldots, n$) For ($j \leftarrow 0, \ldots, e$)
$h[i][j] \leftarrow 0$; for ($k \leftarrow 0$ .. COUNT($f[i][j]$)) $h[i][j] \leftarrow h[i][j] + g[f[i][j][k]][1]$
EndFor EndFor
end procedure

Output: h //h→**LD2**, the LD2 matrix

Algorithm A5 Set LD3 matrix.

Input: n, e, $vs.$, a, f //n←**n**, v←**tv**, a←**Ad**, e←**d**, f←**LD0**

procedure SET_LD3(n, e, &v, &a, &f, &r)
$r \leftarrow []$; $g \leftarrow []$; for ($i \leftarrow 1, \ldots, n$) $g[i] \leftarrow a$
For ($i \leftarrow 1, \ldots, n$) For ($k \leftarrow 0, \ldots, e$)
$r[i][k] \leftarrow 0$; $b \leftarrow$COUNT($f[i][k]$)-1
For ($j \leftarrow 0, \ldots, b$) For ($l \leftarrow 1, \ldots, v[f[i][k][j]][0]$)
$u \leftarrow f[i][k][j]$; $w \leftarrow v[u][l]$
If ($f[i][u][w] = 1$) //(u,w) adjacency is used here
$r[i][k] \leftarrow r[i][k] + 1$; $f[i][u][w] = 0$; $f[i][w][u] = 0$
EndIf
EndFor EndFor
EndFor EndFor
end procedure

Output: r //r→**LD3**, the LD3 matrix

Algorithm A6 Set LD4 matrix.

Input: n, e, b, c //n←**n**, e←**d**; b←**LD2**, c←**LD3**

procedure SET_LD4(n, e, &b, &c, &s)
$s \leftarrow []$; for ($i \leftarrow 1, \ldots, n$) for ($k \leftarrow 0, \ldots, e$) $s[i][k] \leftarrow b[i][k] + c[i][k]$
end procedure

Output: s //s→**LD4**, the LD4 matrix

Algorithm A7 Set LD5 matrix.

Input: n, e, d, f //n←**n**, a←**Ad**, e←**d**, g←**LD0**

procedure SET_LD5(n, e, &d, &f, &g, &h)
$s \leftarrow []$; for ($i \leftarrow 1, \ldots, n$) $s[i] \leftarrow$ ARRAY_SUM($d[i]$)
For ($i \leftarrow 1, \ldots, n$) For ($k \leftarrow 0, \ldots, e$)
$h[i][k] \leftarrow 0$; $b \leftarrow$ COUNT($g[i][k]$)-1
for ($j \leftarrow 0, \ldots, b$) $h[i][k] \leftarrow h[i][k] + s[g[i][k][j]]$
EndFor EndFor
end procedure

Output: h //h→**LD5**, the LD5 matrix

Algorithm A8 Set LkW matrix.

Input: n, e, a, g //n←**n**, a←**Ad**, e←**d**, g←**LD0**

procedure SET_LKW(n, e, &a, &g, &h)
$h \leftarrow []$; $h[1] \leftarrow []$; for ($i \leftarrow 1, \ldots, n$) $h[1][i][0] \leftarrow$ ARRAY_SUM($a[i]$)
For ($i \leftarrow 1, \ldots, n$) For ($k \leftarrow 1, \ldots, e$)
$g[1][i][k] \leftarrow 0$; $b \leftarrow$COUNT($g[i][k]$)-1
for ($j \leftarrow 0, \ldots, b$) $h[1][i][k] \leftarrow h[1][i][k] + h[1][g[i][k][j]][0]$
EndFor EndFor
For ($l \leftarrow 2, \ldots, e$)
$h[l] \leftarrow []$; for ($i \leftarrow 1, \ldots, n$) $h[l][i][0] \leftarrow h[l-1][i][1]$
For ($i \leftarrow 1, \ldots, n$) For ($k \leftarrow 1, \ldots, e$)
$g[l][i][k] \leftarrow 0$; $b \leftarrow$COUNT($g[i][k]$)-1
for ($j \leftarrow 0, \ldots, b$) $h[l][i][k] \leftarrow h[l][i][k] + h[l][g[i][k][j]][0]$
EndFor EndFor
EndFor
end procedure

Output: h //h→**LW**, LW the L(1)W..L(e)W matrices

Algorithm A9 Set Szeged matrix.

Input: n, s //n←**n**, s←**Szs**

procedure SET_SZD(n, &s, &d)
for ($i \leftarrow 1, \ldots, n$) for ($j \leftarrow 1, \ldots, n$) $d[i][j] \leftarrow$ COUNT($s[i][j]$)
end procedure

Output: d //d→**Szd**, Szd—the Szeged matrix

Algorithm A10 Set layer sets.

Input: n, e, m //n←**n**, e←**d**, m←**Ad**, **Di**,**Szd** (m—any vertex pair based matrix)

procedure SET_LS(n, e, &m, &s)
For ($i \leftarrow 1, \ldots, n$) For ($j \leftarrow 0, \ldots, e$)
$s[i][j] \leftarrow []$; for ($k \leftarrow 1, \ldots, n$) if ($d[i][k] = j$) $s[i][j][] \leftarrow k$
EndFor EndFor
end procedure

Output: s //s→**LA0**,**LD0**,**LS0** (s—the layer set of the matrix m)

Algorithm A11 Set counting matrix.

Input: n, m //n←**n**, m←**Ad**,**Di**,**Szd** (any $n \times n$ matrix)

procedure SET_CNT(n, &m, &c)
$vs. \leftarrow []$
For ($i \leftarrow 1, \ldots, n$) For ($j \leftarrow 1, \ldots, n$) If (IN_ARRAY($m[i][j], v$))
else
$v[] \leftarrow m[i][j]$
EndIf EndFor EndFor
SORT($vs.$); $o \leftarrow$COUNT($vs.$)-1
For ($i \leftarrow 1, \ldots, n$)
for ($j \leftarrow 0, \ldots, o$) $c[i][v[j]] \leftarrow 0$; for ($j \leftarrow 1, \ldots, n$) $c[i][m[i][j]] \leftarrow c[i][m[i][j]] + 1$
EndFor
end procedure

Output: c //c→**Adc**,**Dic**,**Szc** (counting matrix of m)

Algorithm A12 Set distances of paths list.

Input: n, c, d, e //n←**n**, c←**tv**; d←**Di**; e←**d**

procedure SET_DIP(n, e, &c, &d, &p)
$p[1] \leftarrow []$
For ($i \leftarrow 1, \ldots, n$) For ($j \leftarrow 1, \ldots, n$) If ($d[i][j] = 1$)
$w \leftarrow []$; $w[] \leftarrow i$; $w[] \leftarrow j$; $p[1] \leftarrow w$ //all edges are all paths of length 1
EndIf EndFor EndFor
For ($k \leftarrow 2, \ldots, e$)
$p[k] \leftarrow []$; $f \leftarrow$ COUNT($(p[k-1])$-1
For ($i \leftarrow 0, \ldots, f$)
$g0 \leftarrow p[k-1][i][0]$; $g1 \leftarrow p[k-1][i][k-1]$
For ($j \leftarrow 1, \ldots, c[g1][0]$)
$g2 \leftarrow c[g1][j]$
If ($d[g0][g2] = k$)
$h \leftarrow p[k-1][i]$; $h[] \leftarrow g2$; $p[k][] \leftarrow h$
EndIf
EndFor
EndFor
EndFor
end procedure

Output: p //p→**path**, path—the distance paths

Algorithm A13 Set cycles list.

Input: n, a, e, p //n←**n**, a←**Ad**, e←**d**, p←**path**

procedure A_CMP(&x, &y)
$nx \leftarrow$ COUNT(x); $ny \leftarrow$ COUNT(y); $nz \leftarrow$ MIN(nx, ny)
For ($i \leftarrow 1, \ldots, nz$)
if($x[i] < y[i]$) return(-1); if($y[i] < x[i]$) return(+1)
EndFor
if($nx < ny$) return(-1); if($ny < nx$) return(+1); return(0)
end procedure //auxiliary procedure
procedure A_SRC(&a, &v, l, h)
$m \leftarrow (int)((l + h)/2)$; $r \leftarrow$ A_CMP($v, a[m]$); if ($r = 0$) return(-1)
IF ($h < l$)
if ($r > 0$) return($m + 1$) else return(m)
EndIf
if ($r > 0$) return(A_SRC($a, v, m + 1, h$)); return(A_SRC($a, v, l, m - 1$))
end procedure //auxiliary procedure
procedure A_INS(&a, &v)
$n \leftarrow$ COUNT(a); if ($n \geq 0$) $k \leftarrow$ A_SRC($a, v, 0, n$) else $k \leftarrow 0$
If ($k \geq 0$)
If ($k = 0$)
ARRAY_UNSHIFT($a, vs.$)
else
If ($k = n + 1$)
ARRAY_PUSH($a, vs.$)
else
$r \leftarrow$ ARRAY_SLICE($a, 0, k$); $s \leftarrow$ ARRAY_SLICE(a, k); $t \leftarrow$ ARRAY($vs.$)
$a \leftarrow$ ARRAY_MERGE(r, t, s)
EndIf
EndIf
EndIf
end procedure
procedure SET_CYC(n, e, &a, &p, &c)
$x = []$; for ($i \leftarrow 1, \ldots, n$) $x[i] \leftarrow []$
For ($k \leftarrow 1, \ldots, e$) For ($l \leftarrow 0$ .. (COUNT($p[k]$)-1))
$q \leftarrow p[k][l]$; $i \leftarrow q[0]$; $x[i][] \leftarrow q$
EndFor EndFor
For ($i \leftarrow 1, \ldots, n$) For ($ki \leftarrow 0$ .. (COUNT($x[i]$)-1))
$r \leftarrow x[i][ki]$; $nr \leftarrow$ COUNT(r)-1; $b \leftarrow []$; for ($j \leftarrow 1, \ldots, n$) $b[j] \leftarrow 0$
for ($j \leftarrow 1, \ldots, nr$) $b[r[j]] \leftarrow b[r[j]] + 1$
For ($kj \leftarrow 0, \ldots, (ki - 1)$)
$s \leftarrow x[i][kj]$; $ns \leftarrow$ COUNT(s)-1
$f \leftarrow b$; for ($j \leftarrow 1, \ldots, ns$) $f[s[j]] \leftarrow f[s[j]] + 1$
$d \leftarrow 0$; for ($j \leftarrow 1, \ldots, n$) if ($f[j] > 1$) $d \leftarrow 1$; if ($d = 1$) continue
If ($a[r[nr]][s[ns]] = 1$)
ARRAY_SHIFT(s); $q \leftarrow$ ARRAY_REVERSE(s); $w \leftarrow$ ARRAY_MERGE(r, q)
$wm \leftarrow$ MIN(w); $wk \leftarrow$ ARRAY_SEARCH(wm, w); $wn \leftarrow$ COUNT(w)
$wj \leftarrow wk - 1$; $wi \leftarrow wk + 1$; $vs. \leftarrow []$
If ($w[(wj + wn)\ \%\ wn] < w[(wi)\ \%\ wn]$)
for ($j \leftarrow 1, \ldots, wn$) $v[] \leftarrow w[(wk + wn - j)\ \%\ wn]$
Else
for ($j \leftarrow 1, \ldots, wn$) $v[] \leftarrow w[(wk + j)\ \%\ wn]$
EndIf //% is the modulo operator
A_INS($c, vs.$) //insert the cycle vs. in the list of cycles c
EndIf
EndFor
EndFor EndFor
end procedure

Output: c //c→**cycle**, cycle—the ones no longer than the double of the diameter

Algorithm A14 Set distance path sequence and layers.

Input: n, e, p //n←**n**, e←**d**; p←**path**

procedure SET_PSL(n, e, &p, &s, &l)
$s \leftarrow []$; for ($i \leftarrow 1, \ldots, n$) for ($j \leftarrow 1, \ldots, n$) $s[i][j] \leftarrow []$
$l \leftarrow []$; for ($i \leftarrow 1, \ldots, n$) for ($j \leftarrow 0, \ldots, e$) $l[i][j] \leftarrow []$
For ($j \leftarrow 1, \ldots, e$) For ($i \leftarrow 0$ .. COUNT($p[j]$))
$vs. \leftarrow p[j][i]$; $f \leftarrow$ COUNT($vs.$); $u \leftarrow v[0]$; $w \leftarrow v[f-1]$
$s[u][w][] \leftarrow vs.$; $l[u][j] \leftarrow vs.$
EndFor EndFor
end procedure

Output: s, l //s→**SP0**, l→**LP0**—the distance paths sequence and layers

Algorithm A15 Set cycle sequence and layers.

Input: n, c //n←**n**, c←**cycle**

procedure SET_CSL(n, &c, &s, &l)
$e \leftarrow$ COUNT(c)-1; $m \leftarrow 0$; for ($i \leftarrow 0, \ldots, e$) if ($m <$ COUNT($c[i]$)) $m \leftarrow$ COUNT($c[i]$)
$s \leftarrow []$; for ($i \leftarrow 1, \ldots, n$) for ($j \leftarrow 1, \ldots, n$) $s[i][j] \leftarrow []$
$l \leftarrow []$; for ($i \leftarrow 1, \ldots, n$) for ($j \leftarrow 3, \ldots, m$) $l[i][j] \leftarrow []$
For ($j \leftarrow 0, \ldots, e$)
$vs. \leftarrow c[j]$; $f \leftarrow$ COUNT($vs.$)-1; for ($i \leftarrow 0, \ldots, f$) $l[v[i]][f+1][] \leftarrow vs.$
For ($i \leftarrow 0, \ldots, (f-1)$) For ($k \leftarrow (i+1), \ldots, f$)
$s[i][k][] \leftarrow vs.$; $s[k][i][] \leftarrow vs.$
EndFor EndFor
EndFor
end procedure

Output: s, l //s→**SC0**, l→**LC0**—the cycles sequence and layers

Algorithm A16 Count and sum for paths and cycles.

Input: n, $vs.$ //n←**n**; v←**SP0**,**LP0**,**SC0**, **LC0**

procedure CS_PC(n, &v, &c, &s)
$u \leftarrow$ ARRAY_KEYS($vs.$); $m \leftarrow$ COUNT(u)-1
For ($i \leftarrow 1, \ldots, n$) For ($j \leftarrow 0, \ldots, m$)
$c[i][u[j]] \leftarrow 0$; $s[i][u[j]] \leftarrow 0$; $t \leftarrow v[i][u[j]]$
For ($k \leftarrow 0$ .. COUNT(t)-1)
$c[i][u[j]] \leftarrow c[i][u[j]] + 1$; $s[i][u[j]] \leftarrow s[i][u[j]] +$ COUNT($t[k]$)
EndFor
EndFor EndFor
end procedure

Output: s, c //c→**SP1**,**LP1**,**SC1**,**LC1** (counts), s→**SP2**,**LP2**,**SC2**,**LC2** (sums)

Algorithm A17 Set Szeged sets.

Input: n, d //n←**n**, d←**Di**

procedure SET_SZS(n, &d, &s)
for ($i \leftarrow 1, \ldots, n$) for ($j \leftarrow 1, \ldots, n$) $s[i][j] \leftarrow []$ //[] is the void array
For ($i \leftarrow 1, \ldots, (n-1)$) For ($j \leftarrow (i+1), \ldots, n$) For ($k \leftarrow 1, \ldots, n$)
if($d[i][k] < d[j][k]$) $s[i][j][] \leftarrow k$; if($d[j][k] < d[i][k]$) $s[j][i][] \leftarrow k$
EndFor EndFor EndFor
end procedure

Output: s //s→**Szs**, Szs—the Szeged sets

The matrices from paper were obtained with an implementation of the above given algorithms. The main program call those algorithms as subroutines (see Algorithm A18).

Algorithm A18 Use of the above algorithms.

SET_ADM(n, tv, Ad)
SET_DIM(n, Ad, Di)
SET_DIA(n, Di, d)
SET_SZS(n, Di, Szs)
SET_SZD(n, Szs, Szd)
SET_LS($n, 1, Ad, LA0$); SET_LS($n, d, Di, LD0$); SET_LS($n, n-1, Szd, LS0$)
SET_DIP($n, d, tv, Di, path$)
SET_CYC($n, d, Ad, path, cycle$)
SET_PSL($n, d, path, SP0, LP0$)
SET_CSL($n, cycle, SC0, LC0$)
CS_PC($n, SP0, SP1, SP2$); CS_PC($n, LP0, LP1, LP2$)
CS_PC($n, SC0, SC1, SC2$); CS_PC($n, LC0, LC1, LC2$)
SET_CNT(n, Ad, Adc); SET_CNT(n, Di, Dic); SET_CNT(n, Szd, Szc)
$LD1 \leftarrow Dic$
SET_LD2($n, d, Dic, LD0, LD2$)
SET_LD3($n, d, tv, Ad, LD0, LD2$)
SET_LD4($n, d, LD2, LD3, LD4$)
SET_LD5($n, d, Di, LD0, LD5$)
SET_LKW($n, d, Ad, LD0, LW$)
//main program

References

1. Latif, S.; Afzaal, H.; Zafar, N.A. Modelling of Graph-Based Smart Parking System Using Internet of Things. In Proceedings of the 2018 International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan, 17–19 December 2018; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
2. Kaveh, A. Decomposition for Parallel Computing: Graph Theory Methods. In *Computational Structural Analysis and Finite Element Methods*; Springer: Cham, Switzerland, 2013; pp. 341–376. [CrossRef]
3. Bermond, J.-C.; Delorme, C.; Quisquater, J.-J. Strategies for interconnection networks: Some methods from graph theory. *J. Parallel Distrib. Comput.* **1986**, *3*, 433–449. [CrossRef]
4. Mallion, R.B.; Rouvray, D.H. Molecular topology and the Aufbau principle. *Mol. Phys.* **1978**, *36*, 125–128. [CrossRef]
5. Choi, J.-H.; Lee, H.; Choi, H.R.; Cho, M. Graph Theory and Ion and Molecular Aggregation in Aqueous Solutions. *Annu. Rev. Phys. Chem.* **2018**, *69*, 125–149. [CrossRef] [PubMed]
6. Wang, R.; Wu, J.; Qian, Z.; Lin, Z.; He, X. A Graph Theory Based Energy Routing Algorithm in Energy Local Area Network. *IEEE Trans. Ind. Inform.* **2017**, *13*, 3275–3285. [CrossRef]
7. Toscano, L.; Stella, S.; Milotti, E. Using graph theory for automated electric circuit solving. *Eur. J. Phys.* **2015**, *36*, 035015. [CrossRef]
8. Ustun, T.S.; Ayyubi, S. Automated Network Topology Extraction Based on Graph Theory for Distributed Microgrid Protection in Dynamic Power Systems. *Electronics* **2019**, *8*, 655. [CrossRef]
9. Diudea, M.-V.; Gutman, I.; Jäntschi, L. *Molecular Topology*, 1st ed.; Nova Science: New York, NY, USA, 2001; pp. 1–328.
10. Joiţa, D.-M.; Jäntschi, L. Extending the Characteristic Polynomial for Characterization of C20 Fullerene Congeners. *Mathematics* **2017**, *5*, 84. [CrossRef]
11. Jäntschi, L. Graph Theory. 2. Vertex Descriptors and Graph Coloring. *Leonardo Electron. J. Pract. Technol.* **2002**, *1*, 37–52.
12. Jäntschi, L. Graph Theory. 1. Fragmentation of Structural Graphs. *Leonardo Electron. J. Pract. Technol.* **2002**, *1*, 19–36.
13. Ballico, E.; Favacchio, G.; Guardo, E.; Milazzo, L.; Thomas, A.C. Steiner Configurations Ideals: Containment and Colouring. *Mathematics* **2021**, *9*, 210. [CrossRef]
14. Buluc, A.; Meyerhenke, H.; Safro, I.; Sanders, P.; Schulz, C. Recent Advances in Graph Partitioning. *Lect. Notes Comput. Sci.* **2013**, *9220*, 117–158. [CrossRef]
15. Choi, D.; Han, J.; Lim, J.; Han, J.; Bok, K.; Yoo, J. Dynamic Graph Partitioning Scheme for Supporting Load Balancing in Distributed Graph Environments. *IEEE Access* **2021**, *9*, 65254–65265. [CrossRef]
16. Miyazawa, F.K.; Moura, P.F.S.; Ota, M.J.; Wakabayashi, Y. Partitioning a graph into balanced connected classes: Formulations, separation and experiments. *Eur. J. Oper. Res.* **2021**, *293*, 826–836. [CrossRef]
17. Miyazawa, F. K.; Moura, P.F.S.; Ota, M.J.; Wakabayashi, Y. Cut and flow formulations for the balanced connected k-partition problem. *Lect. Notes Comput. Sci.* **2021**, *12176*, 128–139. [CrossRef]
18. Bruglieri, M.; Cordone, R. Metaheuristics for the Minimum Gap Graph Partitioning Problem. *Comput. Oper. Res.* **2021**, *132*, 105301. [CrossRef]

19. Bok, K.; Kim, J.; Yoo, J. Dynamic Partitioning Supporting Load Balancing for Distributed RDF Graph Stores. *Symmetry* **2019**, *11*, 926. [CrossRef]
20. Zheng, Z.-Y.; Wang, C.-Y.; Ding, Y.; Li, L.; Li, D. Research on partitioning algorithm based on RDF graph. *Concurr. Comput. Pract. Exper.* **2021**, *33*, E5612. [CrossRef]
21. Wagner, K. Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.* **1937**, *114*, 570–590. [CrossRef]
22. Bodlaender, H.L. A Tourist Guide through Treewidth. *Acta Cybern.* **1993**, *11*, 1–21.
23. Ateskan, E.R.; Erciyes, K.; Dalkilic, M.E. Parallelization of network motif discovery using star contraction. *Parallel Comput.* **2021**, *101*, 102734. [CrossRef]
24. Berry, J.W.; Goldberg, M.K. Path optimization for graph partitioning problems. *Discret. Appl. Math.* **1999**, *90*, 27–50. [CrossRef]
25. Pothen, A.; Simon, H.D.; Liou, K.-P. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.* **1990**, *11*, 430–452. [CrossRef]
26. Jäntschi, L. The Eigenproblem Translated for Alignment of Molecules. *Symmetry* **2019**, *11*, 1027. [CrossRef]
27. Gupta, A. Fast and effective algorithms for graph partitioning and sparse-matrix ordering. *IBM J. Res. Dev.* **1997**, *41*, 171–183. [CrossRef]
28. Gilbert J.R.; Miller G.L.; Teng S.-H. Geometric mesh partitioning: Implementation and experiments. *SIAM J. Sci. Comput.* **1998**, *19*, 2091–2110. [CrossRef]
29. Karypis G.; Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.* **1998**, *20*, 359–392. [CrossRef]
30. Kanj, I.; Komusiewicz, C.; Sorge, M.; van Leeuwen, E.J. Parameterized algorithms for recognizing monopolar and 2-subcolorable graphs. *J. Comput. Syst. Sci.* **2018**, *92*, 22–47. [CrossRef]
31. Ahmadi, R.; Nami, S. Investigation of entanglement entropy in cyclic bipartite graphs using computer software. *Pramana J. Phys.* **2021**, *95*, 39. [CrossRef]
32. Hosseinian, S.; Butenko, S. Polyhedral properties of the induced cluster subgraphs. Discrete *Appl. Math.* **2021**, *297*, 80–96. [CrossRef]
33. Nakamura, K.; Araki, T. Partitioning vertices into in- and out-dominating sets in digraphs. *Discret. Appl. Math.* **2020**, *285*, 43–54. [CrossRef]
34. Barbato, M.; Bezzi, D. Monopolar graphs: Complexity of computing classical graph parameters. *Discret. Appl. Math.* **2021**, *291*, 277–285. [CrossRef]
35. Zoltán F.; Jiang, T.; Kostochka, A.; Mubayi, D.; Verstraëte, J. Partitioning ordered hypergraphs. *J. Comb. Theory A* **2021**, *177*, 105300. [CrossRef]
36. Hellmann, T. Pairwise stable networks in homogeneous societies with weak link externalities. *Eur. J. Oper. Res.* **2021**, *291*, 1164–1179. [CrossRef]
37. McDiarmid, C.; Yolov, N. Recognition of unipolar and generalised split graphs. *Algorithms* **2015**, *8*, 46–59. [CrossRef]
38. Golumbic, M.C.; Lewenstein M. New results on induced matchings. *Discret. Appl. Math.* **2000**, *101*, 157–165. [CrossRef]
39. Eschen, E.M.; Wang, X. Algorithms for unipolar and generalized split graphs. *Discret. Appl. Math.* **2014**, *162*, 195–201. [CrossRef]
40. Adoni, W.Y.H.; Nahhal, T.; Krichen, M.; El byed, A.; Assayad, I. DHPV: A distributed algorithm for large-scale graph partitioning. *J. Big Data* **2020**, *7*, 76. [CrossRef] [PubMed]
41. Zhang, T.; Gao, Y.; Zheng, B.; Chen, L.; Wen, S.; Guo, W. Towards distributed node similarity search on graphs. *World Wide Web* **2020**, *23*, 3025–3053. [CrossRef]
42. Schaudt, O. On weighted efficient total domination. *J. Discret. Alg.* **2012**, *10*, 61–69. [CrossRef]
43. Schaeffer, S.E. Graph clustering. *Comput. Sci. Rev.* **2007**, *1*, 27–64. [CrossRef]
44. de Freitas, R.; Dias, B.; Maculan, N. Szwarcfiter, J. On distance graph coloring problems. *Int. Trans. Oper. Res.* **2021**, *28*, 1213–1241. [CrossRef]
45. Slamin, S.; Adiwijaya, N.O.; Hasan, M.A.; Dafik, D.; Wijaya, K. Local Super Antimagic Total Labeling for Vertex Coloring of Graphs. *Symmetry* **2020**, *12*, 1843. [CrossRef]
46. Šurimová, M.; Lužar, B.; Madaras, T. Adynamic coloring of graphs. *Discret. Appl. Math.* **2020**, *284*, 224–233. [CrossRef]
47. Dokeroglu, T.; Sevinc, E. Memetic Teaching–Learning-Based Optimization algorithms for large graph coloring problems. *Eng. Appl. Artif. Intell.* **2021**, *102*, 104282. [CrossRef]
48. Zaker, M. A New Vertex Coloring Heuristic and Corresponding Chromatic Number. *Algorithmica* **2020**, *82*, 2395–2414. [CrossRef]
49. Lehner, F.; Smith, S.M. On symmetries of edge and vertex colourings of graphs. *Discret. Math.* **2020**, *343*, 111959. [CrossRef]
50. Ahmadi, B.; Alinaghipour, F.; Shekarriz, M.H. Number of distinguishing colorings and partitions. *Discret. Math.* **2020**, *343*, 111984. [CrossRef]
51. Raza, Z.; Essa K.; Sukaiti, M. M-Polynomial and Degree Based Topological Indices of Some Nanostructures. *Symmetry* **2020**, *12*, 831. [CrossRef]
52. Harary, F.; Paper, H.H. Toward a General Calculus of Phonemic Distribution. *Language* **1957**, *33*, 143. [CrossRef]
53. Giffler, B.; Thompson, G. L. Algorithms for Solving Production-Scheduling Problems. *Oper. Res.* **1960**, *8*, 487–503. [CrossRef]
54. Györi, I.; Joó, G. Computer-aided acoustic analysis of reciprocating compressor pipeline systems. *Eng. Comput.* **1987**, *3*, 21–33. [CrossRef]

55. Lee S.; Liu Z.; Kim C. An Agent Using Matrix for Backward Path Search on MANET. *Lect. Notes Comput. Sci.* **2008**, *4953*, 203–211. [CrossRef]
56. Hayward, C.L. Biotic Communities of the Wasatch Chaparral, Utah. *Ecol. Monogr.* **1948**, *18*, 473–506. [CrossRef]
57. Dobrynin, A.A. Graphs with coincident complete matrices of layers. *Vychisl. Sist.* **1987**, *119*, 3–12.
58. Dobrynin, A.A. Graphs with coincident chain-like matrices of layers. *Vychisl. Sist.* **1987**, *119*, 13–33.
59. Skorobogatov, V.A.; Dobrynin, A.A. Metric analysis of graphs. *MATCH-Commun. Math. Chem.* **1988**, *23*, 105–151.
60. Dobrynin, A.A. Regular graphs having the same path layer matrix. *J. Graph. Theory* **1990**, *14*, 141–148. [CrossRef]
61. Dobrynin, A.A. Cubic graphs with 62 vertices having the same path layer matrix. *J. Graph. Theory* **1993**, *17*, 1–4. [CrossRef]
62. Diudea, M.V.; Horvath, D.; Kacsó, I.E.; Minailiuc, O.M.; Parv, B. Molecular topology. VIII: Centricities in molecular graphs. The mollen algorithm. *J. Math. Chem.* **1992**, *11*, 259–270. [CrossRef]
63. Yuansheng, Y.; Xiaohui, L.; Zhiqiang, C.; Weiming, L. 4-Regular Graphs without Cut-Vertices having the Same Path Layer Matrix. *J. Graph. Theory* **2003**, *44*, 304–311. [CrossRef]
64. Lungu, C.N. C-C chemokine receptor type 3 inhibitors: Bioactivity prediction using local vertex invariants based on thermal conductivity layer matrix. *Stud. Univ. Babes-Bolyai Chem.* **2018**, *63* , 177–188. [CrossRef]
65. Hosoya H. Topological Index. A Newly Proposed Quantity Characterizing the Topological Nature of Structural Isomers of Saturated Hydrocarbons. *Bull. Chem. Soc. Jpn.* **1971**, *44*, 2332–2339. [CrossRef]
66. Hosoya, H. On some counting polynomials in chemistry. *Discret. Appl. Math.* **1988**, *19*, 239–257. [CrossRef]
67. Pólya G. Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen. *Acta Math.* **1937**, *68*, 145–254. [CrossRef]
68. Diudea, M.V. Omega polynomial in twisted/chiral polyhex tori. *J. Math. Chem.* **2009**, *45*, 309–315. [CrossRef]
69. Diudea, M.V. Hosoya Polynomial in Tori. *MATCH-Commun. Math. Chem.* **2002**, *45*, 109–122.
70. Müller, J. On the multiplicity-free actions of the sporadic simple groups. *J. Algebra* **2008**, *320*, 910–926. [CrossRef]
71. Fujita, S. Symmetry-itemized enumeration of cubane derivatives as three-dimensional entities by the fixed-point matrix method of the USCI approach. *Bull. Chem. Soc. Jpn.* **2011**, *84*, 1192–1207. [CrossRef]
72. Ştefu, M.V.; Pârvan-Moldovan, A.; Kooperazan-Moftakhar, F.; Diudea, M.V. Topological symmetry of C60-related multi-shell clusters. *MATCH-Commun. Math. Chem.* **2015**, *74*, 273–284.
73. Dinca M.F.; Ciger, S.; Ştefu, M.V.; Gherman F.; Miklos, K.; Nagy, C.L.; Ursu, O.; Diudea, M.V. Stability prediction in C40 fullerenes. *Carpathian J. Math.* **2004**, *20*, 211–221.
74. Diudea, M.V.; Minailiuc, O.M.; Balaban, A.T. Molecular topology. IV. Regressive vertex degrees (new graph invariants) and derived topological indices. *J. Comput. Chem.* **1991**, *12*, 527–535. [CrossRef]
75. Balaban, A.T.; Diudea, M.V. Real Number Vertex Invariants: Regressive Distance Sums and Related Topological Indices. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 421–428. [CrossRef]
76. Diudea M.V. Molecular Topology. 16. Layer Matrixes in Molecular Graphs. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 1064–1071. [CrossRef]
77. Wiener, H. Structural Determination of Paraffin Boiling Point. *J. Am. Chem. Soc.* **1947**, *69*, 17–20. [CrossRef]
78. Morgan, H. The Generation of a Unique Machine Description for Chemical Structures. A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, *5*, 107–113. [CrossRef]
79. Knuth, D. Postscript about NP-hard problems. *ACM SIGACT News* **1974**, *6*, 15–16. [CrossRef]
80. Pan, Y.; Morisako, S.; Aoyagi, S.; Sasamori, T. Generation of Bis(ferrocenyl)silylenes from Siliranes. *Molecules* **2020**, *25*, 5917. [CrossRef]
81. Thompson, B.C.; Fréchet, J.M.J. Polymer-fullerene composite solar cells. *Angew. Chem. Int. Ed.* **2008**, *47*, 58–77. [CrossRef] [PubMed]
82. Al-Jumaili, A.; Alancherry, S.; Bazaka, K.; Jacob, M.V. Review on the Antimicrobial Properties of Carbon Nanostructures. *Materials* **2017**, *10*, 1066. [CrossRef]
83. De Stefani, A.; Bruno, G.; Preo, G.; Gracco, A. Application of Nanotechnology in Orthodontic Materials: A State-of-the-Art Review. *Dent. J.* **2020**, *8*, 126. [CrossRef] [PubMed]
84. Došlić, T. All Pairs of Pentagons in Leapfrog Fullerenes Are Nice. *Mathematics* **2020**, *8*, 2135. [CrossRef]
85. Da Ros, T.; Prato, M. Medicinal chemistry with fullerenes and fullerene derivatives. *Chem. Commun.* **1999**, 663–669. [CrossRef]