# Symmetry of Chemical Structures: A Novel Method of Graph Automorphism Group Determination

S. Bohanec* and M. Perdih

National Institute of Chemistry, Hajdrihova 19, 61115 Ljubljana, Slovenia

Received April 20, 1993*

Chemical structure may be described in terms of graph theory. Symmetry properties of a graph are expressed by its automorphism group. A novel method for deducing the automorphism group is presented. The method is fast, general, irredundant, and exhaustive. Its major advantage is that it avoids the possibility of combinatorial explosion in symmetric graphs by the induction of order into graphs. All generated permutations are members of the automorphism group. Some suggestions of possible applications in chemistry are also given.

## 1. INTRODUCTION

Knowing the symmetry properties of chemical structures plays an important role in the interpretation of their NMR and IR spectra or X-ray diffraction patterns. Often it is not necessary to know geometry of the structures. In most cases it is sufficient to know only the constitution of the molecule. Therefore, the molecular structure can be described in terms of graphs; atoms are interpreted as vertices, and bonds are interpreted as edges.

There are numerous algorithms using this approach, such as the algorithms for computer-generated stereoisomer lists of structures,[1,2] for substructure search, and for the elimination of identical structures. The last two algorithms were implemented in many systems (CARBON,[3] UPGEN,[4] GENSTR,[5-7] GENMAS[6,7]) developed in our laboratory.

Symmetry properties of each graph are derived from its automorphism group.[8,9] The automorphism group of the graph is a group of automorphism permutations (automorphisms) that transform the graph into the identical graph, i.e., automorphic graph. Automorphic graphs are obtained by the permutation of one or more constitutionally equivalent vertices.

In the graph the constitutionally equivalent vertices can be separated into sets of vertices, called *orbits*. Members of each orbit cannot be distinguished by any graph-theoretical property. Such a partition of vertices into orbits, i.e., automorphism partition, can be obtained by the application of one of the algorithms which are also used for canonical numbering of vertices of graphs.[10-14]

Several algorithms for automorphism group construction are known. For example, the algorithm of Razinger et al.,[15] which constructs *all* permutations of vertices within the orbits and checks if they are automorphisms or not. The main problem of this algorithm is the combinatorial explosion, which can arise especially in highly symmetric graphs with many constitutionally equivalent vertices.

Another approach is given by Balasubramanian.[16] It uses pruning tree algorithm and "wreath product" formalism.[17]

The main idea of the method, presented in this paper, is the construction of the automorphism group by selecting only the permutations of constitutionally equivalent vertices in which no edges between vertices are broken. The number of these permutations is small enough to avoid the combinatorial explosion. The result of these permutations is by definition the automorphic graph, and the applied permutations are automorphism permutations, called automorphisms.

In contrast to the algorithm given by Razinger et al.,[15] no additional checking of permutations is necessary, because we know they are automorphisms.

To determine which of the constitutionally equivalent vertices can be permuted without breaking any edge, i.e., on which the automorphisms can be performed, a new type of hierarchical ordering of graphs is introduced. The main advantage of the presented method is that it is faster compared to the mentioned one. It is also exhaustive and efficient. As such it can be a useful part of many computer programs requiring spectrum simulation and/or spectrum interpretation, computer-enhanced structure elucidation, structure or substructure searches, etc.

The presented algorithm uses as input graphs with vertices, which are already partitioned into automorphism orbits by the algorithm developed by Shelly and Munk.[12]

This paper is organized as follows. In the next section the transformation of molecular graphs into a quasi-tree form is described. In the section that follows a simple example of the automorphism group with the detailed description of different vertex permutations is presented. Section 4 gives the algorithm. The number of permutations needed to obtain the automorphism group and the number of all automorphisms that are members of the automorphism group are shown in section 5. In section 6 our method is tested on many different symmetric molecular graphs and is compared to the method given by Balasubramanian.[16,17] Some parallels between this mathematically derived and established method and our method are also given to obtain an indirect proof of the legality of our method. In the last section some characteristics and some applications of our method in chemistry are presented.

## 2. MOLECULAR GRAPHS AS QUASI-TREES

Chemical structure can be represented by a graph and with some modifications also by a tree. The modifications are necessary because the chemical structures frequently contains cycles and multiple bonds, while a tree is defined as a connected acyclic graph in which every two vertices are joined by a unique path.[8]

Computational treatment of the trees is easier compared to the treatment of arbitrary graphs. The transformation of graphs of chemical structures into trees induces an order into the graphs: the hierarchy of atoms and their neighborhoods are exactly defined.

*Abstract published in *Advance ACS Abstracts*, August 15, 1993.

0095-2338/93/1633-0719$04.00/0 

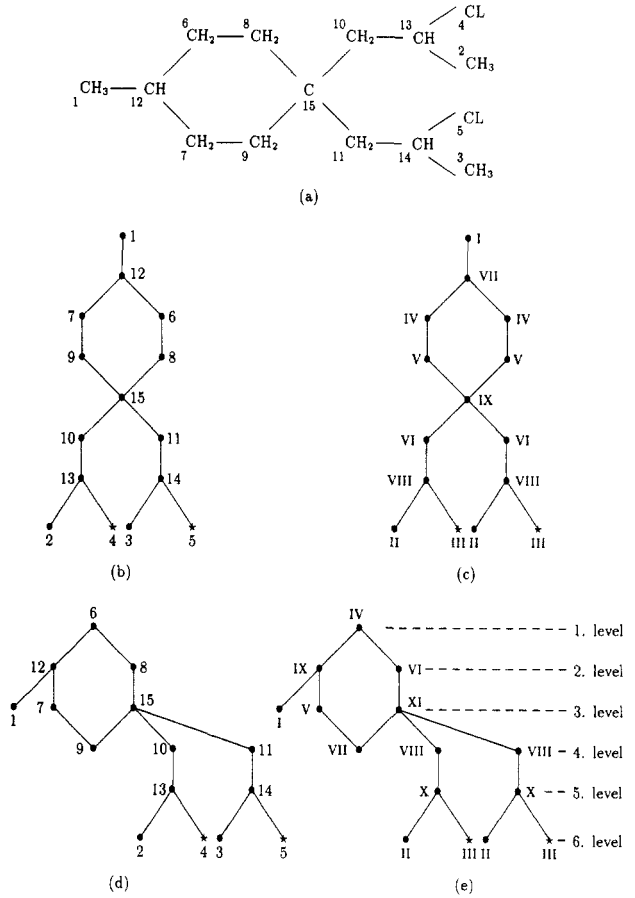


**Figure 1.** (a) Chemical structural formula, its graph (b) with canonical sequence numbers of vertices and (c) with sequence numbers of automorphism orbits, and its transformation into the quasi-tree (d) with sequence numbers of vertices and (e) with sequence numbers of equivalence classes. The sequence numbers of vertices are marked with Arabic numbers, while the sequence numbers of orbits and equivalence classes are numbered with Roman numbers. Carbon and chlorine atoms are marked with "●" and "☆", respectively. Hydrogens are omitted.

One of the algorithms for transforming graphs into trees introduces additional vertices ("duplicate atoms") to rearrange multiple bonds and cycles into the tree form.[18] There are at least two disadvantages which may appear during the transformation. First, in some cases some constitutional differences of the structure can be lost, and such tree is not a unique representation of the structure.[19] As the consequence, two different chemical structures can be transformed into identical trees. And second, the tree of a structure with a lot of cross-bonding vertices can be very complicated and unclear. To avoid such situations, a new type of hierarchical ordering, the so called *irregular* or *quasi-tree*, is used in our algorithm.

On the contrary to the common trees known for the graph theory, the quasi-trees contain also multiple bonds (edges) and/or cycles if the original chemical structures are unsaturated and/or cyclic. An example of such structure and its graph is shown in Figure 1. The non-hydrogen atoms of this structure (Figure 1a) are canonically numbered.

The transformation of a graph into the quasi-tree can be described as the hanging of the graph by one vertex which represents the root of the tree. Such "hanging transformation" of the cyclic graph (Figure 1) is shown in Figure 1d.

Vertices, permuted by the automorphisms, can be easily determined from the quasi-tree. One can see that only these permutations of vertices, (1) which are on the same level, (2) which are constitutionally equivalent, regarding the new ordering of the vertices in the quasi tree, and (3) which are connected to the same vertex placed one level higher, are the automorphism permutations. For example, such vertices are vertices 10 and 11 (Figure 1d). For more detailed description of vertices used for automorphisms, some additional definitions of their places in the quasi-tree will be given.

The root is placed in the first, i.e., the highest quasi-tree's level. The neighbors of the root are in the second level, the neighbor's neighbors except the root are in the third, etc. The vertices in the same level constitute a *generation* of vertices.[19] The minimal distances from the root to the vertices in the same level are equal.

In trees, the successors and the predecessors of any vertex can be defined. In general, a *successor* of a vertex is a vertex which is placed *one level lower* in the tree than the vertex being observed. As the opposite, a *predecessor* of a vertex is a vertex placed *one level higher* in the tree than the vertex being observed. Therefore, to find the vertices for permutations, it is enough to check all the successors of each vertex in the quasi-tree. All successors of each vertex that are constitutionally equivalent should be marked to be used for further permutations.

For quasi-trees not containing cycles or containing only cycles consisting of an even number of vertices, the situation is clear; for each vertex its successors can be unambiguously defined. For quasi-trees containing also cycles consisting of an odd number of vertices, a special situation occurs. When such cycle is "hanged" by the vertex, two the most distant vertices from the hanged one are placed in the same level and are connected. They can be either successors or predecessors to each other. They cannot be permuted with their real successors, i.e., with the vertices laying one level lower in the quasi-tree; consequently, we classify them as predecessors. According to this we should modify the definition of a predecessor: a predecessor must be placed one level higher or *in the same level* as the vertex being observed. Therefore, on the contrary to proper trees, in quasi-trees a vertex can have more than one predecessor. The root has no predecessor, while a leaf has no successor.

The *descendants* of the vertex *i* are it successors, these successors' successors, and so on to the lowest level containing vertices, i.e., leafs. All descendants of one vertex constitute a *subtree* (a branch of the tree) where this vertex is the root (head) of the subtree.

The transformation of the graph into the quasi-tree induces an additional order on the vertices within the orbits. A similar effect is obtained if the root vertex is separated from its orbit into a separate class. After that, all vertices are further partitioned into classes. The vertices that are members of the same class have the same path lengths between each of them and the root and are members of the same orbit. These classes are called the equivalence classes.

The successors of each vertex of the quasi-tree, starting from the root, are grouped into permutation classes. In the same permutation class are successors of the vertex, members of the same equivalence class. The successors of each vertex are separated into that many permutation classes as many different equivalence classes they are members of. For example, from Figure 1d,e it can be seen that vertex 15 has three successors: vertices 9–11. The last two are equivalent, and they are members of the equivalence class VIII (Figure 1d,e). Therefore, the successors of vertex 15 are separated into two permutation classes; the vertex 9 is in the first and vertices 10 and 11 are in the second equivalence class.

Vertices with two successors are vertices 6 and 12–14 (Figure 1d,e). The successors of each of these vertices are all in

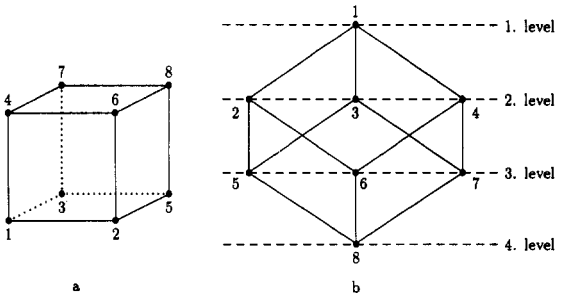


**Figure 2.** (a) The graph of cubane and (b) its transformation into the quasi-tree.

different permutation classes, because they are different. The vertices with one successor always have exactly one permutation class. Since the leafs have no successors, they have also no permutation classes.

By limiting the number of neighbors of each vertex to four, a permutation class can have at most three members, unless it is determined for the root vertex, which can have at most four members.

For the given graph, the partition of vertices into nine orbits and the partition of vertices of the quasi-tree into eleven equivalence classes is shown in Figure 1. Construction of permutation classes of all vertices gives only one permutation class with two members. Members of this class are vertices 10 and 11, which are successors of vertex 15. They are members of the equivalence class VIII. All other permutation classes contain one member only.

According to the definitions of equivalence and permutation classes, orbits, and subtrees, the vertices, which are in the same permutation class, have equivalent descendants, and, therefore, they are heads of equivalent subtrees. The automorphism permutations are obtained by the permutations of such subtrees.

In quasi-trees the permutations of subtrees are sometimes limited because of the presence of cycles, which can link some subtrees. Such linked subtrees cannot be permuted independently. This type of permutations is called "cycle dependent permutation" of subtrees, in contrast to the "independent permutation". In general, the cycle dependent permutations in quasi-trees are caused by vertices which have two or more predecessors that link two or more equivalent subtrees.

An example of an independent permutation is the permutation of subtrees with heads 10 and 11 in Figure 1d. The cycle dependent permutation can be described on the graph of cubane, Figure 2. In the quasi-tree of cubane each vertex in the second level is linked to the vertex in the third level and vice versa. Consequently any permutation of vertices in the second level causes also the permutation of vertices in the third level and vice versa. For example, by the permutation of vertices 2 and 3 the vertices in the third level 6 and 7, which are the successors of vertex 4, are also permuted and vice versa. In both cases the same automorphism is found. To avoid such duplications, the vertices, which have already been dependently permuted, should not be permuted again.

## 3. VERTEX PERMUTATIONS AND AUTOMORPHISM GROUP

Before the description of the algorithm and the theoretical derivation of the number of permutations, a simple example of the automorphism group, its representation, and the basic idea of how to get the automorphism group will be given.

In Figure 3 is a set of automorphic graphs of the structure

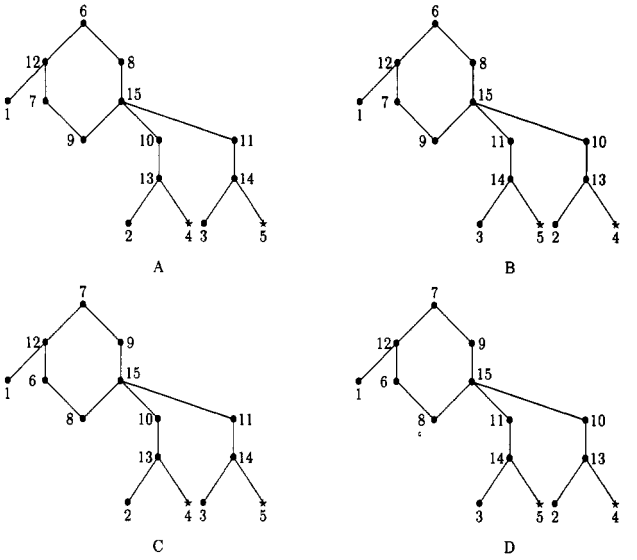


**Figure 3.** Quasi-trees of the molecular graph from Figure 1.

from Figure 1a. This set consists of four automorphic graphs. For the sake of easier comparison all graphs are written in quasi-tree forms. The first two quasi-trees, A and B, have vertex 6 in the root, while quasi-trees C and D have vertex 7 in the root. From Figure 1c one can see that vertices 6 in 7 are both members of orbit IV. The orbit being selected is an orbit which members must be all placed in the root of the quasi-trees to obtain the whole automorphism group. In general, any orbit can be selected, because the obtained automorphism group is always the same. The size of the selected orbit has the effect on the number of permutations that should be made and also on the execution time needed to obtain the automorphism group. Therefore, the selection of the orbit is the most critical step in the algorithm for automorphism group constructing. The effects of the orbit selection on the execution time will be described at the end of this section.

Quasi-tree A is obtained by the transformation of the original graph. It is called the *basic quasi-tree*. In the root of the basic quasi-tree is always the vertex from the selected orbit with the smallest sequence number.

Quasi-tree B is obtained by the permutation of the equivalent subtrees, successors of vertices 10 and 11, respectively, of the basic quasi-tree. This permutation and the identity form the set of automorphism permutations of the basic quasi-tree called the *basic automorphisms*. One of the basic automorphisms is always the *identity*. The set of basic automorphisms is a subgroup of the graph automorphism group.

In this example, there are two basic automorphisms. In general, all possible automorphism permutations obtained by the permutation of the equivalent subtrees of the basic quasi-tree form the set of basic automorphisms, $P_b$. The cardinality of this set, $|P_b|$, equals the number of all basic automorphisms in the set, i.e., the number of all possible automorphism permutations allowed in the basic quasi-tree. The automorphism permutations are permutations that give the quasi-trees equivalent to the basic one.

Quasi-tree C is obtained by the transformation of the original graph the same way as quasi-tree A. But, in this case, the next vertex of the selected orbit is placed in the root. Because of the constitutional equivalence of root vertices 6 and 7, quasi-trees A and C are equivalent too. Comparing both quasi-trees, A and C, one can find out that only the constitutionally equivalent vertices have changed their positions in the quasi-

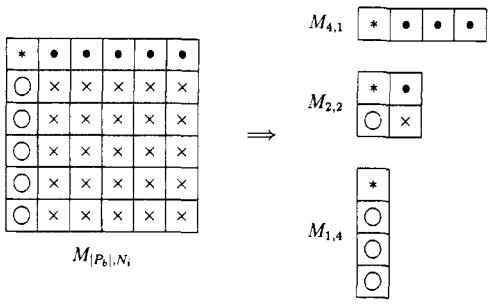


**Figure 4.** Schematic representation of the automorphism group.

tree; i.e., they have been permuted. In the given example the pairs of vertices (6, 7) and (8, 9) have been permuted.

The transformation of the basic quasi-tree to the quasi-tree with other constitutionally equivalent vertex in the root is called the *root-to-root transformation*. There are as many root-to-root transformations as there are vertices in the selected orbit. These root-to-root transformations are also automorphism permutations. On the contrary to the basic automorphisms, they are called the *root-to-root automorphisms*. They form the set of root-to-root automorphisms. In general, there are $N_i$ root-to-root automorphisms, where $N_i$ is the number of vertices in the selected $i$th orbit. Since the first root-to-root automorphism is the identify, additional $N_i - 1$ root-to-root automorphisms have to be derived.

The permutation that gives quasi-tree D from the basic quasi-tree is a "mixture" of two permutations, the one which transforms the basic quasi-tree into quasi-tree B and the one which transforms the basic quasi-tree into quasi-tree C. Mathematically this mixture is obtained by the product of the two previously mentioned permutations. In general, the automorphisms are obtained by the Cartesian product of the set of root-to-root automorphisms with the set of basic automorphisms.

The whole automorphism group for the structure in Figure 1a consists of the following four automorphisms:

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | 3 | 2 | 5 | 4 | 6 | 7 | 8 | 9 | 11 | 10 | 12 | 14 | 13 | 15 |
| 1 | 2 | 3 | 4 | 5 | 7 | 6 | 9 | 8 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | 3 | 2 | 5 | 4 | 7 | 6 | 9 | 8 | 11 | 10 | 12 | 14 | 13 | 15 |

In general, the whole automorphism group can be determined when the set of $|P_b|$ basic automorphisms and the set of $N_i$ quasi-trees are obtained. The number of all automorphism permutations $|P|$ of input graph is

$$|P| = N_i|P_b| \tag{1}$$

The automorphism group $P$ can be represented as the matrix $\mathbf{M}_{|P_b|,N_i}$ shown in Figure 4. Each element of this matrix corresponds to one automorphism. The elements are marked with different signs, which depend on the way that the particular automorphism is obtained. There are four different types of automorphisms: identity (denoted with *), basic automorphisms, (denoted with ●), root-to-root automorphisms (denoted with ○), and automorphisms obtained by the Cartesian product (denoted with ×). Therefore, the automorphism group given in Figure 3 can be represented with the matrix $\mathbf{M}_{2,2}$. It includes the following four automorphisms: one identity that gives quasi-tree A, one basic automorphism that gives quasi-tree B, one root-to-root automorphism that gives quasi-tree C, and one automorphism obtained by the Cartesian product (quasi-tree D). In general the automorphism group represented with the matrix $\mathbf{M}_{|P_b|,N_i}$ consists of the following $|P_b|N_i$ automorphisms: one identity, $|P_b| - 1$ basic automorphisms, $N_i - 1$ root-to-root automorphisms, and $(|P_b| - 1)(N_i - 1)$ automorphisms obtained by the Cartesian product. Execution times needed to obtain the automorphism depend on the type of the automorphism. Execution times are in the following relation:

$$t_* \doteq t_\circ > t_\bullet \gg t_\times$$

The procedures for obtaining the identity and root-to-root automorphisms are the most time consuming, because in both cases the whole quasi-tree should be constructed and the permutation classes should be selected. To obtain the basic automorphisms, only parts of the basic quasi-tree, i.e., the vertices in the permuted subtrees, should be renumbered. Therefore, the procedure for obtaining a basic automorphism (except the identity) is less time consuming compared to the procedures for obtaining identity or root-to-root automorphisms. The time needed for the product of two permutations can be neglected in relation to other execution times, since the product is a single step operation.

In general, an automorphism group with cardinality four can be obtained in three different ways, as shown in Figure 4, right. These three possibilities can be represented with three different matrices: $\mathbf{M}_{4,1}$, $\mathbf{M}_{2,2}$, and $\mathbf{M}_{1,4}$.

The automorphism group of the graph from Figure 1 can be obtained on two different ways of three possible. If the orbit II, III, IV, V, VI, or VIII with two members (Figure 1b,c) is selected, the way that the automorphism group is obtained can be represented with the matrix $\mathbf{M}_{2,2}$. Otherwise, if the orbit I, VII, or IX with one member (vertex 1, 12, or 15) is selected, the generated automorphism group can be represented with the matrix $\mathbf{M}_{4,1}$. There is no possibility to obtain the automorphism group represented with the matrix $\mathbf{M}_{1,4}$, since the graph has no orbit with four members.

According to the given execution times, the automorphism group represented with matrix $\mathbf{M}_{2,2}$ is obtained faster than the group $\mathbf{M}_{4,1}$. Therefore, to obtain the automorphism group of the given graph as fast as possible, the orbit with two vertices should be selected.

This is also the reason why vertices 6 and 7 are chosen for the root instead of the vertex 15, although at first sight the choice of the vertex 15 seems more reasonable. In general, to obtain the automorphism group in the fastest way as many as possible automorphisms should be obtained by the Cartesian product. To achieve this, the number of basic permutations (including identity) $|P_b|$ should be similar to the number of vertices that are placed in the root, $N_i$. Therefore, the form of the representation matrix $\mathbf{M}_{|P_b|,N_i}$ should be more like a square than like a rectangular.

## 4. ALGORITHM

The input data for the novel algorithm are the chemical structure written in the form of connection table[3,4,6,7] with atoms already partitioned into orbits and canonically numbered. The connection table gives only the connectivities between the atoms within the structure without any information about the bond lengths or angles between them.

In order to derive the automorphism group of any chemical structure represented by a graph, the following steps should be made: (1) choice of the root vertex, (2) transformation of the graph into the quasi-tree and classification of vertices into permutation classes, (3) determination of basic automorphisms, (4) determination of root-to-root automorphisms, and (5) Cartesian product of the set of root-to-root automorphisms with the set of basic automorphisms.

**4.1. Choice of the Root Vertex.** In principle, it is unimportant the members of which orbit are placed in the root, because whatever the choice is, the automorphism group obtained is always the same. Nevertheless, the choice has the effect on the time needed for the determination of the automorphism group. As already described in the previous section, the number of basic permutations $|P_b|$ should be similar to the number of vertices that are placed in the root, $N_i$. To satisfy this condition, the vertices that belong to an orbit of medium size are usually placed in the root. In the graph in Figure 1 such orbits are orbits with two vertices: (2,3), (4,5), (6,7), (8,9), (10,11), and (13,14). There are differences in the number of neighbors that each of these vertices has; vertices 2–5 have one neighbor, vertices 6–11 have two neighbors, and vertices 13 and 14 have three neighbors. Only to make things clearly defined, without any effects on the execution times or on the results, the following rule was established: when there are several orbits containing the same number of vertices, the vertices with two neighbors have priority over the vertices with three neighbors, these have the priority over the vertices with one neighbor, and these have the priority over the vertices with four neighbors. If necessary, further differentiation is made according to the sequence numbers of vertices. Following these rules the orbit with vertices 6 and 7 is selected.

**4.2. Transformation of Graph into Quasi-Tree and Classification of Vertices into Permutation Classes.** After the vertex for the root has been chosen, the graph is transformed into the quasi-tree. Constitutionally equivalent vertices previously ordered into orbits are now additionally ordered into equivalence and permutation classes. The permutation classes with more than one member are detected and selected for permutations.

**4.3. Determination of Basic Automorphisms.** When the basic quasi-tree of the graph is obtained and the vertices are classified into permutation classes, all basic automorphisms can be derived. There are as many basic automorphisms as there are different possible combinations of permutations of vertices within all permutation classes.

For basic quasi-tree A, shown in Figure 3, only two basic automorphism are possible, because the basic quasi-tree has only one permutation class with two vertices. This is the permutation class with vertices 10 and 11. One of the permutations is identity, while the other one transforms basic quasi-tree A to quasi-tree B.

**4.4. Determination of Root-to-Root Automorphisms.** When all basic automorphisms are determined, all other $N_i - 1$ vertices from the $i$th selected orbit are placed in the root. So, $N_i - 1$ quasi-trees with similar symmetry properties as the basic one are made. For all these quasi-trees root-to-root transformations are obtained by mapping the equally placed vertices of the basic quasi-tree to each of $N_i - 1$ quasi-trees. In this way $N_i - 1$ root-to-root automorphisms are obtained.

In Figure 3 the root-to-root transformation of basic quasi-tree A results in quasi-tree C with root vertex 7. Therefore, vertex 6 is mapped to vertex 7 and vice versa and vertex 8 is mapped to vertex 9 and vice versa, while all other vertices are mapped to themselves.

**4.5. Cartesian Product of Set of Root-to-Root Automorphisms with Set of Basic Automorphisms.** When all basic and root-to-root automorphisms are derived, the whole group of automorphisms is obtained by the Cartesian product of the set of root-to-root automorphisms with the set of basic automorphisms.

In Figure 3 quasi-tree D is obtained from the basic quasi-tree by the permutation which is the product of two permutations: the first transforms basic quasi-tree A to quasi-tree B and the second transforms basic quasi-tree A to quasi-tree C.

## 5. NUMBER OF PERMUTATIONS

**5.1. Basic Equations.** A complete graph permutation group consists of $N!$ permutations, where $N$ is the number of vertices of the graph. The number of automorphism permutations is usually smaller and depends on the graph symmetry.

The algorithm for automorphism group constructing introduced by Razinger et al.[15] permutes the vertices within the orbits only. Permutations of the whole graph are constructed from all possible interorbit combinations of intraorbit permutations. By the classification of vertices into orbits the number of possible candidates for the automorphism group, $N!$, is reduced to $|P_0|$. The number of automorphism permutation candidates $|P_0|$ can be calculated as

$$|P_0| = \prod_{i=1}^{m} N_i! \tag{2}$$

where $m$ is the number of orbits and $N_i$ is the number of vertices in the $i$th orbit.

The number of all vertices $N$ in the graph is the sum of all members of all orbits:

$$N = \sum_{i=1}^{m} N_i \tag{3}$$

All $|P_0|$ permutations are checked if they transform the graph into itself or not. Permutations that satisfy this criterion are members of the automorphism group. Frequently only a few permutations among these $|P_0|$ permutations are automorphisms.

By the transformation of the graph into a quasi-tree and the additional ordering of the vertices into permutation classes, the number of all basic automorphisms $|P_b|$ is

$$|P_b| = \prod_{j=1}^{N} \prod_{k=1}^{p_j} n_{jk}! \tag{4}$$

where $N$ is the number of all vertices of the quasi-tree, $p_j$ is the number of all permutation classes in which the successors of the $j$th vertex are partitioned, and $n_{jk}$ is the number of successors are defined below.

The values of $n_{jk}$, for the $j$th vertex and its $k$th permutation class, can be the following:

$$n_{jk} = \begin{cases} 0, & \text{if there are no equivalent successors} \\ 0, & \text{if the successors are dependently permuted} \\ 2, & \text{if there are two equivalent successors} \\ 3, & \text{if there are three equivalent successors} \\ & \text{etc.} \end{cases}$$

If some cycle dependent permutations are possible in the quasi-tree, some parameters $n_{jk}$ equal 0. This is because the $j$th vertex has equivalent successors in its $k$th permutation class, and these equivalent successors can be dependently permuted by the permutation of vertices in higher level of the quasi-tree, compared to the position of the $j$th vertex in the quasi-tree.

On the contrary to $|P_0|$ potential permutations (eq 2), these $|P_b|$ basic permutations do not need any additional checking if they are automorphisms or not, because they are automorphisms by definition.

Considering the eq 1 and the construction algorithm, it follows that for the construction of the automorphism group

represented by the matrix $\mathbf{M}_{|P_b|,N_i}$, the number of all needed permutations, $|P_n|$, which must be calculated to obtain all data necessary for constructing the automorphism group, is

$$|P_n| = (N_i - 1) + |P_b| \quad (5)$$

**5.2. Calculation of $|P_0|$, $|P_b|$, $|P|$, and $|P_n|$.** As an example, the numbers of permutations $|P_0|$, $|P_b|$, $|P|$, and $|P_n|$ will be calculated for the graph in Figure 1.

The vertices of the graph in Figure 1b are classified into nine orbits with 1, 2, 2, 2, 2, 2, 1, 2, and 1 members as shown in Figure 1c. According to eq 2, $|P_0|$ is

$$|P_0| = 1! \times 2! \times 2! \times 2! \times 2! \times 2! \times 1! \times 2! \times 1! = 64$$

By the transformation of the graph into the quasi-tree (Figure 1d,e), the number of all possible mutations $|P_b|$ of the basic quasi-tree with vertex 6 in the root can be calculated from eq 4. Since all factors $n_{jk}$, except $n_{15,2}$, equal 0, they do not contribute to $|P_b|$, so

$$|P_b| = n_{15,2}! = 2$$

Note that $n_{15,2} = 2$, since only the vertex 15 has two successors in its second permutation class. These two basic automorphisms (one is identity) result in quasi-trees A and B (Figure 3).

In the selected orbit (IV) are two vertices, 6 and 7 (Figure 1c). Therefore, $N_4 = 2$. According to eq 1 and 5, the numbers $|P|$ and $|P_n|$ are the following:

$$|P| = 2 \times 2 = 4$$

$$|P_n| = 2 - 1 + 2 = 3$$

Among the three automorphism permutations $|P_n|$, one is the root-to-root and two are the basic automorphisms. Between these two basic automorphisms, one is the identity. Compared with 64 permutations, these three permutations mean a large reduction of possible automorphisms that should be derived in order to construct the automorphism group.

**5.3. Comparison of $|P_0|$ and $|P_n|$.** The comparison of the numbers of permutations, $|P_0|$ and $|P_n|$, which should be derived to obtain the whole automorphism group, can be calculated from eqs 2, 4, and 5. The latter two equations can be combined into the eq 6. The direct comparison of eqs 2 and 6 is not

$$|P_n| = (N_i - 1) + \prod_{j=1}^{N} \prod_{k=1}^{p_j} n_{jk}! \quad (6)$$

worth the effort because the parameters $m$, $N_i$, $p_j$, and $n_{jk}$ are case dependent, but some general remarks about the values of these parameters can be made: $N_i \geq n_{jk}$ because the members of one orbit can be ordered into one (equality is valid) or more permutation classes; $n_{jk}$ = valence number of the root vertex $j$ when $p_j = 1$ (vertex $j$ has all equivalent successors), otherwise it is smaller; $n_{jk}$ = valence number −1 of the vertex $j$ which is not in the root and $p_j = 1$, otherwise it is smaller; $n_{jk} = 0$, if $j$th vertex has no equivalent successors or it is a leaf or the successors are cycle dependent. From these remarks it can be found out that the product in eq 2 consists of a smaller number of larger factors compared to the product in eq 6 which consists of a greater number of smaller factors ($n_{jk}!$).

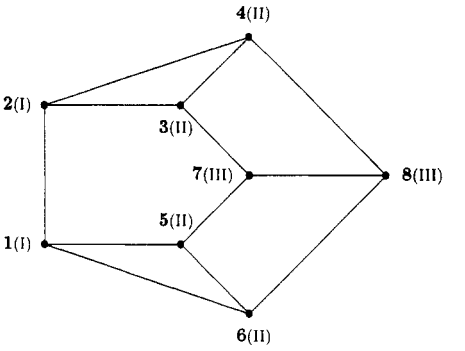


**Figure 5.** Graph of cuneane. The vertices are labeled with sequence numbers (Arabic numbers) and with orbits (Roman numbers).

Therefore, usually the relation $|P_0| > |P_n|$ but in general $|P_0| \geq |P_n|$ is valid as it will be shown on some examples later on. The exceptions are the asymmetrical graphs, where the only permutation is identity, and the symmetric graphs consisting of two vertices. In these cases an equality $|P_0| = |P_n|$ is valid.

## 6. RESULTS AND DISCUSSION

**6.1. Experimental Results.** The described algorithm has been tested on many graphs of different, mostly organic compounds. The input data for this algorithm are the graph of the compound represented by the form of a connection table[3,4,6,7] with atoms classified into orbits.

In this section, some experimental results of the algorithm will be shown on the basis of the following examples: (1) cuneane, (2) cubane, (3) 1,2,3,4-tetra(1-hydroxyethyl)cyclobutadiene, (4) 1,3,5-trihydroxybenzene, and (5) buckminsterfullerene $C_{60}$.

*6.1.1. Cuneane.* The graph of cuneane consists of eight vertices paritioned into three orbits (Figure 5). The vertices from the first orbit are placed in the root in turn. Within the basic quasi-tree of cuneane one permutation of two equivalent subtrees is possible. Therefore, there is one identity, one basic, one root-to-root automorphism (from root 1 to root 2), and one automorphisms obtained by the Cartesian product. The whole automorphism group can be represented by the matrix $\mathbf{M}_{2,2}$ containing four automorphism permutations:

1 2 3 4 5 6 7 8
1 2 4 3 6 5 8 7
2 1 5 6 3 4 7 8
2 1 6 5 4 3 8 7

*6.1.2. Cubane.* The graph of cubane (Figure 2) consists of eight equivalent vertices. As already described (see section 4), in the quasi-tree of cubane six independent permutations (permutations of vertices 2–4) are possible:

1 2 3 4 5 6 7 8
1 3 2 4 5 7 6 8
1 3 4 2 7 5 6 8
1 4 3 2 7 6 5 8
1 4 2 3 6 7 5 8
1 2 4 3 6 5 7 8

The first is identity, while the other five are basic permutations.

Since the graph of cubane has eight constitutionally equivalent vertices, seven root-to-root automorphisms must be derived in order that the automorphism group can be

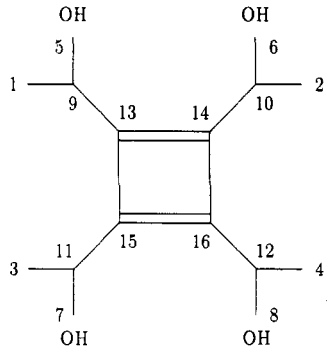


**Figure 6.** Molecular graph of 1,2,3,4-tetra(1-hydroxyethyl)cyclobutadiene.

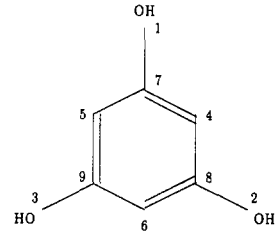


**Figure 7.** Molecular graph of 1,3,5-trihydroxybenzene.

constructed. These seven $(N_i-1)$ root-to-root automorphisms are

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 1 | 5 | 6 | 3 | 4 | 8 | 7 |
| 3 | 1 | 5 | 7 | 2 | 4 | 8 | 6 |
| 4 | 1 | 6 | 7 | 2 | 3 | 8 | 5 |
| 5 | 2 | 3 | 8 | 1 | 6 | 7 | 4 |
| 6 | 2 | 4 | 8 | 1 | 5 | 7 | 3 |
| 7 | 3 | 4 | 8 | 1 | 5 | 6 | 2 |
| 8 | 5 | 6 | 7 | 2 | 3 | 4 | 1 |

The whole group can be represented by the matrix $\mathbf{M}_{6,8}$. Therefore, it consists of 48 automorphisms: one identity, five basic, seven root-to-root automorphisms, and 35 automorphisms obtained by the Cartesian product.

*6.1.3. 1,2,3,4-Tetra(1-hydroxyethyl)cyclobutadiene.* The chemical structure of 1,2,3,4-tetra(1-hydroxyethyl)cyclobutadiene (Figure 6) has localized single and double bonds in four-membered ring due to its antiaromatic character. After its transformation into a graph, vertices 9–12 are taken as roots. The quasi-tree with vertex 9 placed in the root is the basic quasi-tree. The only automorphism of the basic quasi-tree is the identity. Therefore, the whole automorphism group represented by the matrix $\mathbf{M}_{1,4}$ consists of the identity and three root-to-root automorphisms:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 2 | 1 | 4 | 3 | 6 | 5 | 8 | 7 | 10 | 9 | 12 | 11 | 14 | 13 | 16 | 15 |
| 3 | 4 | 1 | 2 | 7 | 8 | 5 | 6 | 11 | 12 | 9 | 10 | 15 | 16 | 13 | 14 |
| 4 | 3 | 2 | 1 | 8 | 7 | 6 | 5 | 12 | 11 | 10 | 9 | 16 | 15 | 14 | 13 |

*6.1.4. 1,3,5-Trihydroxybenzene.* The atomorphism group of 1,3,5-trihydroxybenzene with delocalized single and double bonds in the benzene ring (Figure 7) consists of six automorphisms:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 2 | 1 | 3 | 4 | 6 | 5 | 8 | 7 | 9 |
| 1 | 3 | 2 | 5 | 4 | 6 | 7 | 9 | 8 |
| 3 | 1 | 2 | 5 | 6 | 4 | 9 | 7 | 8 |
| 2 | 3 | 1 | 6 | 4 | 5 | 8 | 9 | 7 |
| 3 | 2 | 1 | 6 | 5 | 4 | 9 | 8 | 7 |

**Table I.** Parameters for Different Examples

| example | $m$ | orbits | $N_i$ | $\lvert P_b\rvert$ | $\lvert P_n\rvert$ | $\lvert P_o\rvert$ | $\lvert P\rvert$ | time[a] (s) |
|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 2+4+2 | 2 | 2 | 3 | 2! × 4! × 2! | 4 | 0.05 |
| 2 | 1 | 8 | 8 | 6 | 13 | 8! | 48 | 0.11 |
| 3 | 4 | 4+4+4+4 | 4 | 1 | 4 | $(4!)^4$ | 4 | 0.05 |
| 3[b] | 4 | 4+4+4+4 | 4 | 2 | 5 | $(4!)^4$ | 8 | 0.05 |
| 4 | 3 | 3+3+3 | 3 | 1 | 3 | $(3!)^3$ | 3 | 0.05 |
| 4[b] | 3 | 3+3+3 | 3 | 2 | 4 | $(3!)^3$ | 6 | 0.06 |
| 5 | 1 | 60 | 60 | 2 | 61 | 60! | 120 | 30 |

[a] On PC 386, 33 MHz, CI = 35. [b] Conjugated double bonds are interpreted as aromatic bonds.

They form the $\mathbf{M}_{2,3}$ automorphism group that includes one identity, one basic automorphism, two root-to-root automorphisms, and two automorphisms obtained by the Cartesian product.

*6.1.5. Buckminsterfullerene $C_{60}$.* Buckminsterfullerene $C_{60}$ consists of 60 equivalent carbons that constitute 12 five-membered and 20 six-membered rings. Each five-membered ring is peri-condensed to five six-membered rings. Each six-membered ring is peri-condensed to five- and six-membered rings in turn. In the basic quasi-tree one permutation is possible. According to the 60 equivalent vertices, 59 root-to-root transformations should be made to obtain the whole automorphism group. It consists of 120 automorphisms, which can be represented by the matrix $\mathbf{M}_{2,60}$. This result is in agreement with the fact that the molecule belongs to $I_h$ symmetry group.[20]

**6.2. Comparison of Results.** Our algorithm and its results will be compared to two known algorithms for the construction of the graph automorphism groups: the algorithm described by Razinger et al.[15] and the algorithm described by Balasubramanian et al.[16]

The first algorithm generates $|P_0|$ permutations and checks if they are automorphic or not. Our algorithm generates $|P_n|$ permutations which are already automorphic, so they do not need any additional checking. In both cases the derived automorphism groups (Table I) are identical, but the execution times are different. It can be found out that for all given examples the relation $|P_0| \gg |P_n|$ is valid. With use of our algorithm, the automorphic groups for each of the examples, except the last one, are obtained in less than 0.2 s, while for the last example 30 s are needed. The algorithm described by Razinger et al. is not so fast mainly because of significantly higher number of permutations that must be made and the additional checking of the permutations.

The difference between the present approach for constructing an automorphism group of a graph, and the approach by Razinger et al.[15] is that in the latter all possible permutations of vertices within all orbits of the graph are generated, while in the former only the permutations of vertices within all permutation classes are generated. By the transformation of the graph into the quasi-tree, the vertices, which are members of the same orbit, are divided into one or more permutation classes. If the vertices of the orbit are divided into only one permutation class, the number of vertices of each permutation class is equal to the number of vertices of the orbit. Otherwise, the number of vertices of each permutation class is smaller than the number of vertices of the orbit. Therefore, in the latter case, the number of all possible permutations is reduced. The exceptions are the asymmetric graphs and some smaller symmetric graphs.

Permutation classes can be derived from the quasi-tree representation of the graph. The quasi-trees usually have lower symmetry than the graphs; therefore, the lack of the

symmetry of the quasi-tree has to be compensated for. Since the number of permutations is reduced due to the inability of the root vertex to be permuted with the other vertices, members of the same orbit as itself, the basic quasi-tree has to be transformed to another quasi-tree with another root vertex, a member of the same orbit as the former one. This transformation has to be followed by permutations of the vertices of the new quasi-tree. It can be stated that the construction of the automorphism group consists of three independent actions: (1) determination of automorphisms of the basic quasi-tree, (2) determination of all root-to-root automorphisms, and (3) multiplications of all basic automorphisms with all root-to-root automorphisms.

From the formal point of view, this approach is very similar to the generalized "wreath product" formalism by Balasubramanian described in[16,17] and the other works of the same author cited therein, although his "particle in a box" model cannot be applied directly. The automorphism group $P_G$ is defined as

$$P_G = G(H_1, H_2, \cdots, H_k)$$

where $G$, $H_1$ $H_2$, $\cdots$, $H_k$ are automorphism subgroups. The subgroup $G$ is called outer group, while the others are called inner groups. The automorphism group is constructed with multiplication of these subgroups:

$$P_G = G \times (H_1 \times H_2 \times \cdots \times H_k)$$

It is important to note that $k$, the number of inner groups, is equal to the number of permutation classes having more than two members. Each permutation class is used to generate all permutations of the successors of the vertex the permutation class has been derived for. Each thus obtained set of permutations is one of the inner groups. Therefore, the inner product $H = H_1 \times H_2 \times \cdots \times H_k$ is equivalent to the set of all basic automorphisms applied by our method. This set is also a group as mentioned before. The outer group $G$ plays a similar role as the set of root-to-root transformations, although the set is not a group in general.

Therefore, the presented approach is less demanding than the one with the tree pruning algorithm,[17] since in the former case any root-to-root transformation between any two quasi-trees, that is also a graph automorphism, is sufficient, while in the latter one has to find those automorphisms that satisfy the condition of being a group. For acyclic graphs the final result is in both cases the same, since the set of quasi-tree automorphisms is a group and therefore, all possible automorphisms are taken into account.

## 7. CONCLUSION

The construction of the automorphism group, implemented in our algorithm, is very fast mainly because of the quasi-tree representation of molecular graphs. This form allows the use of standard searching and matching procedures developed for proper trees. Furthermore, no information about the constitution of graphs is lost with quasi-trees.

The most important characteristics of the present method are speed (the construction of the automorphism groups of real chemical structures takes less than 0.2 s of CPU time (Table I)), generality (it constructs the automorphism group of any chemical structure, cyclic or acyclic) irredundancy (it generates *only* the automorphism permutations), and exhaustivity (it generates *all* automorphism permutations of a given structure). Because of these characteristics, the applicability of this method in chemistry is very wide. It is useful for different actions applied on the chemical structures: for the substructure search, for the comparison of structures, for the elimination of identical structures, for searching of identical or at least similar structures, etc.

By now, more or less modified it is incorporated in several modules of expert and information systems developed in our laboratory: CARBON[3] (substructure search), UPGEN[4] (search of identical structures), GENSTR[5-7] and GENMAS[6,7] (substructure search, elimination of identical structures), and IRSYS (substructure search). It is faster and exhaustive compared to the results obtained by the previous methods used in these systems. This is an indirect proof of the efficiency of the method.

## ACKNOWLEDGMENT

We are very grateful to Prof. Dr. J. Zupan and Dr. M. Razinger for helpful discussions and many useful criticisms. We thank the Ministry of Science and Technology of Slovenia for financial support.

## REFERENCES AND NOTES

(1) Nourse, J. G.; Carhart, R. E.; Smith, D. H.; Djerassi, C. Exhaustive Generation of Stereoisomers for Structure Elucidation. *J. Am. Chem. Soc.* **1979**, *101*, 1216–1223.

(2) Razinger, M.; Balasubramanian, K.; Munk, M. E.; Perdih, M. Stereoisomer Generation in Computer-Enhanced Structure Elucidation. Submitted for publication.

(3) Zupan, J.; Novič, M.; Bohanec, S.; Razinger, M.; Lah, L.; Tušar, M.; Košir, I. Expert System for Solving Problems in Carbon-13 Nuclear Magnetic Resonance Spectroscopy. *Anal. Chim. Acta* **1987**, *200*, 333–345.

(4) Bohanec, S.; Tušar, M.; Tušar, L.; Ljubič, T.; Zupan, J. A System for Creating Collections of Chemical Compounds Based on Structures. In Data Handling in Science and Tehnology; Karjalainen, E. J., Ed.; *Scientific Computing and Automation (Europe)*; Elsevier: Amsterdam, 1987; Vol. 6, pp 393–405.

(5) Tušar, M.; Tušar, L.; Bohanec, S.; Zupan, J. $^1$H and $^{13}$C NMR Spectra Simulation. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 299–303.

(6) Bohanec, S.; Zupan, J. Structure Generation of Constructional Isomers from Structural Fragments. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 531–540.

(7) Bohanec, S.; Zupan, J. Structure Generator GEN. *Match* **1992**, *27*, 49–85.

(8) Harary, F. *Graph Theory*, 3rd Printing Addison-Wesley: Reading, MA, Oct 1972.

(9) Trinajstić, N. *Chemical Graph Theory*; CRC Press: Boca Raton, FL, 1983; Vol. 1, Chapter 4.

(10) Morgan, H. L. Generation of Unique Machine Description for Chemical Structures a Technique Developed at Chemical Abstract Service. *J. Chem. Doc.* **1965**, *5*, 107–112.

(11) Wipke, W. T.; Dyott, T. M. Stereochemically Unique Naming Algorithm. *J. Am. Chem. Soc.* **1974**, *96*, 4834–4842.

(12) Shelley, C. A.; Munk, M. E. An Approach to the Assignment of Canonical Tables and Topological Symmetry Perception. *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 247–250.

(13) Moreau, G. A Topological Code for Molecular Structures. A Modified Morgan Algorithm. *Nouv. J. Chim.* **1980**, *4*, 17–22.

(14) Balaban, A. T.; Mekenyan, O.; Bonchev, D. Unique Description of Chemical Structures Based on Hierarchicaly Ordered Extended Connectivity (HOC Procedures). I. Algorithms for Finding Graph Orbits and Canonical Numbering of Atoms. *J. Comput. Chem.* **1985**, *6*, 538–551.

(15) Razinger, M.; Balasubramanian, K.; Munk, M. E. Graph Automorphism Perception Algorithms in Computer-Enhanced Structure Elucidation. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 197–201.

(16) Balasubramanian, K. Symmetry Groups of Chemical Graphs. *Int. J. Quant. Chem.* **1982**, *21*, 411–418.

(17) Liu, X. Y.; Balasubramanian, K. Computer Generation of Character Tables of Generalized Wreath Product Groups. *J. Comput. Chem.* **1990**, *11*, 589–602.

(18) Prelog, V.; Helmchen, G. Basic Principles of the CIP-System and Proposals for a Revision. *Angew. Chem., Int. Ed. Engl.* **1982**, *21*, 567–583.

(19) Custer, R. H. Mathematical Statements about the Revised CIP-System. *Match* **1986**, *21*, 3–31.

(20) Schmaltz, T. G.; Seitz, W. A.; Klein, D. J.; Hite, G. E. Elemental Carbon Cages. *J. Am. Chem. Soc.* **1988**, *110*, 1113–1127.