Molecular Symmetry Perception

Julian Ivanov*

27030 Cedar Road, Suite 401, Beachwood, Ohio 44122

Received August 26, 2003

An algorithm for molecular symmetry perception is presented. The method identifies the full set of molecular symmetry elements (proper and improper) and determines their coordinates. The algorithm eliminates the necessity to explore the entire graph automorphism group; as a result its computer application is extremely effective. Application to several dendrimers and fullerenes with high topological symmetry is presented.

1. INTRODUCTION

Understanding the symmetry of molecules denotes an important instrument to tackle fundamental problems in chemistry and biology. The symmetry of a molecule describes how its different parts relate to one another geometrically. Qualitative information about molecular wave function and physical properties such as dipole moments and optical activities can be obtained from the symmetry of the chemical structure. Thus, for example, existence or absence of certain symmetry elements leads to the following immediate consequences: a molecule cannot have a dipole moment perpendicular to the C_n axis; a molecule without S_n axis is chiral; a chiral molecule cannot have center of inversion; and so on.

The problem of identifying molecular symmetry has been extensively investigated in a number of studies.[1−12] Some of the algorithms[5,11] for molecular symmetry perception are based on the calculation of the moments of inertia followed by reorientation of the structure according to a specific frame of reference, and then the structure is tested for the presence of various types of symmetry elements. Although these methods are quite fast and very accurate in calculating the coordinates of the symmetry elements, they do not guarantee identification of the full set of symmetry operations for certain molecules, which results in a failure to determine the correct symmetry, point group. Thus, for instance, the algorithm for molecular symmetry recognition implemented in MOPAC[5] does not test the molecules for rotational symmetry elements of an order bigger than 8. As a result symmetry point groups such as C_n, D_n, C_{nh}, and D_{nh}, where $n > 8$, are not available.

In a series of studies[5−8] Balasubramanian exploits the simple and very effective idea of using Euclidian distances to obtain permutations that correspond to the symmetry operations. It is shown that the permutations of atoms that leave the Euclidian distance matrix of the molecule invariant comprise the permutations contained in the permutation inversion group defined by Longuet-Higgins.[2] A permutation of atoms in the molecule is an automorphic permutation, if it preserves the Euclidian distances between the atoms. Such a permutation also represents proper or improper symmetry operation of the structure under investigation. The methodology is simple and elegant; however, it requires the generation of a graph automorphism group, which is a time-consuming procedure for some types of molecules. The algorithm first divides the nuclei of the molecule into equivalence classes and generates all possible permutations $n_1!n_2!...n_m!$, where n_i are the number of atoms that belong to the ith equivalence class. For any generated permutation it checks whether the permutation belongs to the Euclidian distance group. This technique accelerates the approach for molecules that possess different equivalent classes of atoms, but it does not take any advantage in the case of structures where all nuclei are equivalent like some of the fullerenes. In this case the number of all possible permutations is $n!$, and for large n ($n > 20$) it requires additional efforts (simplifications of the structure) in order to solve the problem in reasonable time.

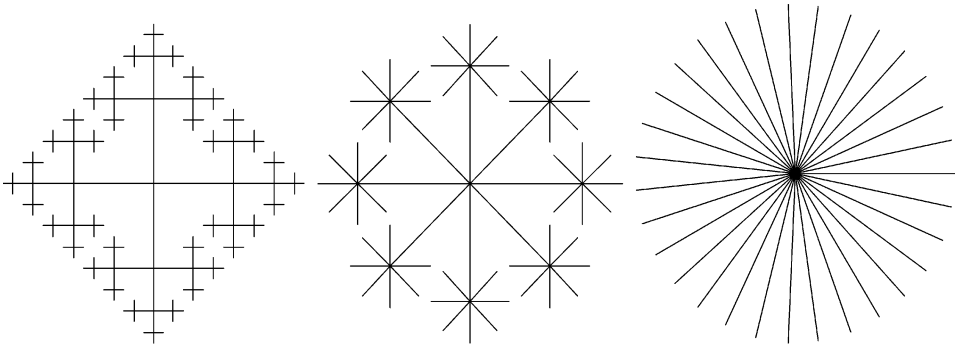
In a previous study[1] we developed simple algorithms for molecular symmetry perception that in a single procedure handle graph automorphism of the chemical structure, identify atoms permutations that correspond to molecular symmetry operations (proper and improper), calculate the coordinates and types of the symmetry elements, and finally determine the symmetry point group. The approach identifies the full set of symmetry elements for any type of chemical structure without any exception. We also showed that CPU time is quite attractive for small and moderate molecules. Although the run time for real chemical structures is acceptable, the method shows some difficulties in treatment of structures with high topological symmetry like the graphs in Figure 1. The current investigation addresses this problem and proposes a solution that results in a very fast procedure for molecular symmetry perception.

2. COMPUTATIONAL TECHNIQUES

For the sake of completeness, we first briefly recall the main steps of the symmetry identification algorithm[1] developed earlier. It starts by generating the graph automorphism group of the molecular structure. It is obtained on the basis of an oriented spanning tree of the chemical graph, constructed by the recursive depth-first search (DFS) procedure.[13,14] For any generated permutation of the graph vertices the algorithm constructs a topological name. It is formed by a sequence of the atom label, symbols for the type of the chemical bonds, and the numbers (as they appeared in the DFS tree) of the connected atoms. The purpose of the

* Corresponding author phone: (216) 896-0305; e-mail: jmi2@po.cwru.edu.

10.1021/ci0341868 CCC: $27.50
Published on Web 01/16/2004



structure **I** structure **II** structure **III**

Figure 1. Example of structures with high topological symmetry.

topological name is to control at any step of the recursive procedure if the current permutation preserves the connectivity of the structure. The name that corresponds to the first generated permutation we called the reference topological name (RTN). For the RTN the algorithm constructs the corresponding reference conformational name (RCN), which is a sequence of dihedral angles. For each permutation that belongs to the graph isomorphism group, the method generates a conformational name and compares it to the RCN. If the current conformational name is identical to the reference one, then this corresponds to a proper rotation symmetry operation. If the two names are mirror images, an improper symmetry operation has been identified. If the conformational name is different from the RCN, this means that the permutation does not preserve the geometry of the structure and, hence, does not belong to the Euclidian distance group. For the detected symmetry operations, the procedure calculates the type and the coordinates of the appropriate symmetry elements. Note the computer application of the algorithm considers two dihedral angles as identical, if the absolute value of the difference between them is less than $\pi/18$ rad (10°).

As it is well-known the graph automorphism group is related only to the connectivity of the graph, while the symmetry group of the molecule depends on the relative positions of the atoms in the 3D space. While the chemical structure can take different shapes (geometries), its connectivity remains unchanged; therefore, the symmetry group should be a subset of the graph automorphism group. Most algorithms for molecular symmetry identification (including the above one) first generate the full set of equivalent topological permutations of the nuclei (the graph automorphism group), and after that from this set they select only the permutations that preserve the Euclidian distance matrix. Unfortunately the number of the equivalent topological representations for some graphs is too large. The graph automorphism group for every structure in Figure 1 contains more than 10^{34} topological permutations. It is virtually impossible to generate all of them in real time even with the present most sophisticated computational resources. The majority of these permutations do not correspond to real symmetry operations but to simple internal molecular group rotations. The question is, how can we recognize early and avoid the generation of such permutations?

The backtrack approach we use here to handle the graph automorphism group saves some time by abandoning generations of permutations as soon as it is known that it does not preserve the connectivity of the molecular graph. The running time for this approach is of $(d_1 - 1)^{n_1}(d_2 - 1)^{n_2}...(d_m - 1)^{n_m}$ order, where n_i is the number of vertices and d_i is the vertex degree. The algorithm, however, does have a problem treating types of graphs such as the structures in Figure 1 simply because the number of the equivalent topological permutations is too large. For identifying the molecular symmetry, however, the three-dimensional geometry of the structure has to be taken into consideration, and the DFS backtrack approach can significantly decrease the CPU time of the code by using the Euclidian distances between the atoms during the recursive steps of the procedure.

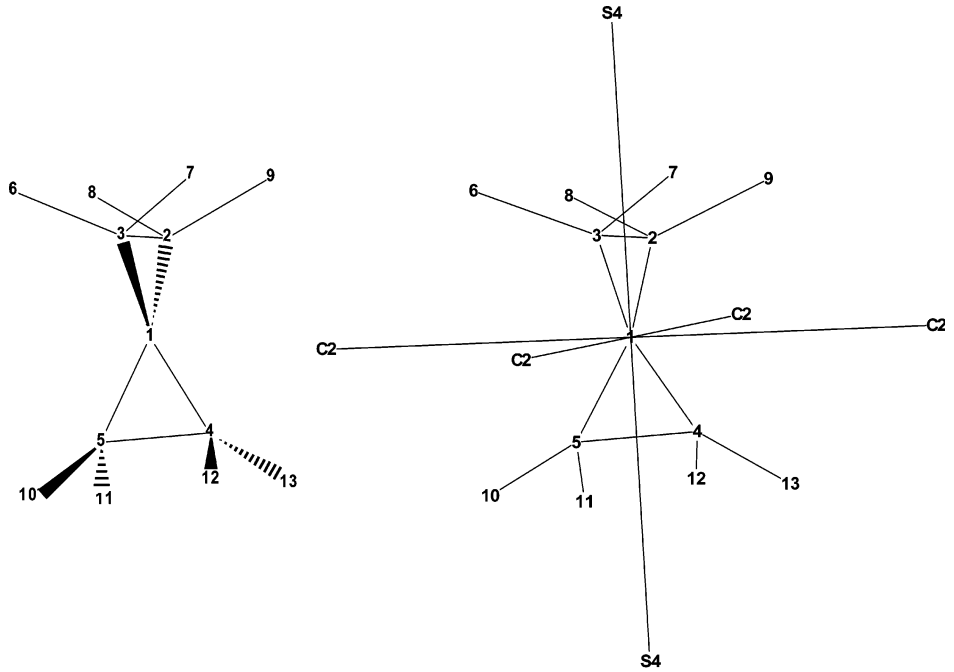
At any step the DFS procedure treats a single vertex of the chemical graph and creates a name that describe the topology of the corresponding atom. Now, to take into account the geometry of the molecule, the procedure calculates the Euclidian distances between the current *i*th atom and all $(i - 1)$ atoms already visited exactly in the order they appeared in the DFS tree. The calculated Euclidian distances construct the so-called geometrical name of the current atom, and basically this is the *i*th line of the lower triangle of the Euclidian distances matrix. Thus, the reference geometry name (RGN) of the molecule under investigation would be the lower triangle of the Euclidian distance matrix, and it contains $n*(n - 1)/2$ elements, where n is the number of atoms. The symmetry perception DFS procedure can be represented by the following pseudo C++ procedure:

```
void SymmetryDFS(CurrentAtom)
{
        i++;
        Generate topology name for the CurrentAtom
        Calculate Euclidian distances between CurrentAtom and all (i-1) visited atoms

        if (Reference name already constructed?)
                { if (ReferenceName != CurrentName) return; } else
                { ReferenceName[i] = CurrentName; }
        CurrentAtom.Visit = true;

        if (All atoms are visited)
        {
                if (Reference name already constructed?)
                        { New Symmetry Element; } else
                        { Reference name is constructed; } // Symmetry operation E
        } else
        {
                Select a new atom from the set of unvisited atoms
                DFS(NewAtom);
        }

        i--;
        CurrentAtom.Visit = false;
}
```



structure **IV**

Figure 2. Example of structure with symmetry point group D_{2d}.

After the generation of the reference name (symmetry operation identity—*E*), the code goes back to the last *i*th atom with a choice point. We define a choice point as an atom with at least two topologically equivalent adjacent atoms. This is actually the point where a single permutation occurs. At this stage the method must pick up one of the equivalent adjacent atoms, or in other words to permute them. On the next $(i + 1)$th step, it calculates the distance between the $(i + 1)$st atom and all 1, 2, ..., *i* atoms in the DFS tree and compares these distances with those in the $(i + 1)$st lane of the Euclidian distance matrix. If a single difference between the distances exists, then the permutation does not belong to the symmetry group and the algorithm runs back and continues with the next possible permutation. The accuracy threshold embedded in the computer program is 0.2 Å; i.e., the program considers two Euclidian distances as identical if the absolute value of the difference between them is less than 0.2 Å.

We illustrate the algorithm with a real example structure **IV** in Figure 2, where the planes of the two triangles 1-2-3 and 1-4-5 are perpendicular. The graph automorphism group for this structure contains 128 permutations. The following are the rules for walking through the molecular graph, embedded in the algorithm: (1) start from the vertices with the highest connectivity, and (2) continue to the adjacent vertex with the highest connectivity; the first generated permutation of the vertices (the reference) is

1 2 3 6 7 8 9 4 5 10 11 12 13

At the choice point atom 2, where the algorithm permutes vertices 8 and 9 (adjacent to atom 2), it has to be checked if this permutation preserves the Euclidian distances or in other words if distances between atom 9 and atoms 1, 2, 3, 6, and 7 are identical to the distances between 8 and 1, 2, 3, 6, and 7. The distance $6-8 < 6-9$; hence, this permutation does not satisfy the requirement for Euclidian distances matrix preservation, and the method rejects it. The most important consequence of this rejection is that the algorithm abandons all of the following permutations between vertices 4−5, 10−11, and 12−13. The eight abandoned permutations are shown in Table 1.

Table 1. Eight Permutations Abandoned by the Algorithm Due to the Permutation of Atoms 8 and 9 in Structure **IV**

1	2	3	6	7	9	8	4	5	10	11	12	13
1	2	3	6	7	9	8	4	5	10	11	13	12
1	2	3	6	7	9	8	4	5	11	10	12	13
1	2	3	6	7	9	8	4	5	11	10	13	12
1	2	3	6	7	9	8	5	4	12	13	10	11
1	2	3	6	7	9	8	5	4	12	13	11	10
1	2	3	6	7	9	8	5	4	13	12	10	11
1	2	3	6	7	9	8	5	4	13	12	11	10

Thus, the approach considers only the permutations that correspond to symmetry operations. The symmetry point group of structure **IV** in Figure 2 is D_{2d}, and the eight symmetry operations that belong to this point group are listed in Table 2 along with the corresponding atom permutations and the coordinates of the elements. The coordinates of the symmetry elements are unit vectors that coincide with the symmetry axes or are perpendicular to the symmetry planes. The right part of Figure 2 shows the molecular symmetry elements for structure **IV**. One of the C2 axes (line 4 in Table 2) coincides with the S4 axis. The two symmetry planes are not shown. The first mirror plane (line 2, Table 2) lies in the plane defined by atoms 1-2-3, while atoms 1-4-5 define the second one (line 3, Table 2).

Abandoning generations of graph automorphism permutations that do not preserve the Euclidian distances matrix dramatically speeds up the molecular symmetry perception algorithm and reduces the CPU time of the computer application, making realistic molecular symmetry identification even for structures for which the generation of graph automorphism group is virtually impossible (see Figure 1).

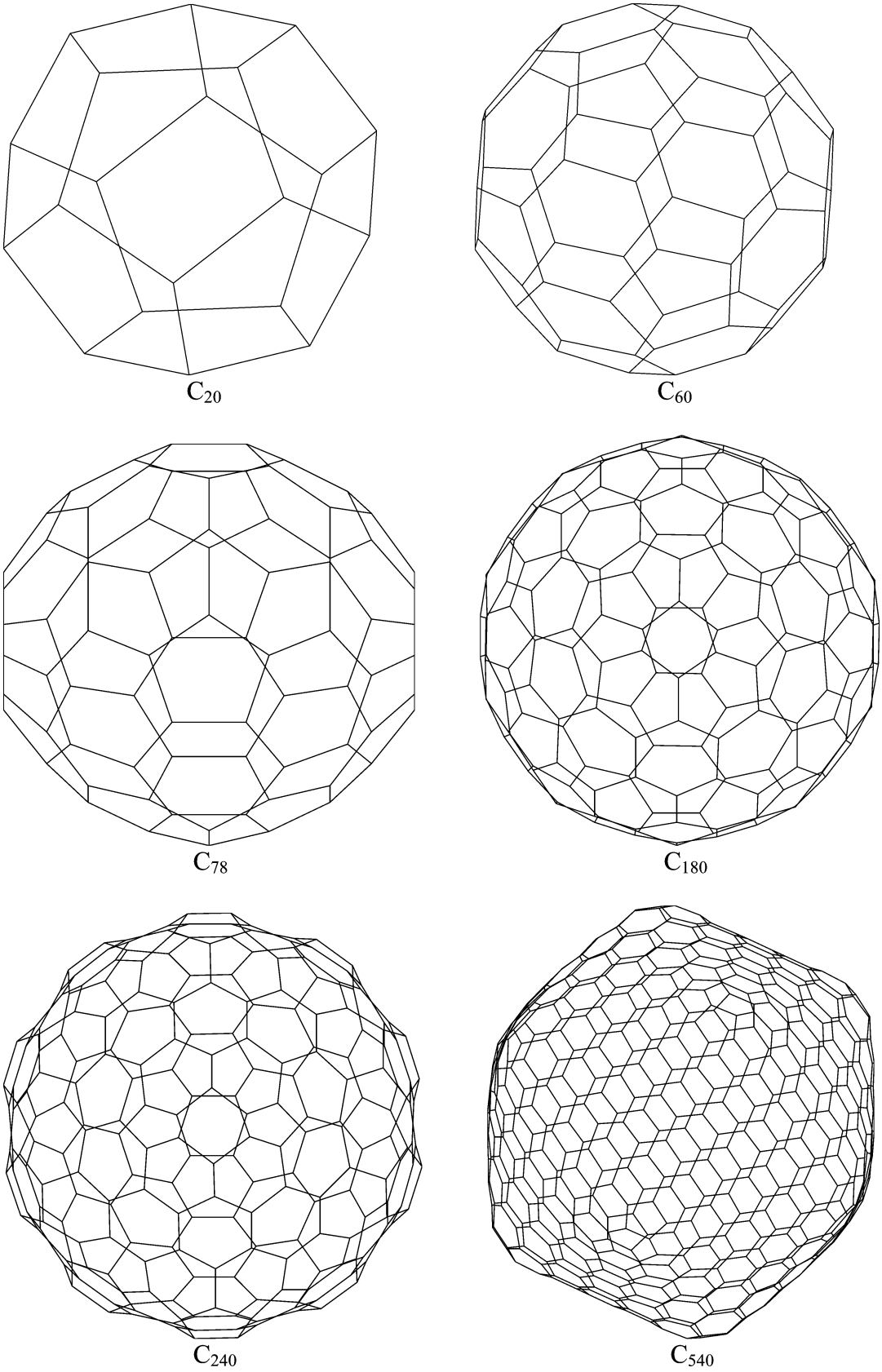


Figure 3. Structures of six fullerenes.

3. RESULTS AND DISCUSSION

The algorithm described above has been successfully implemented into a computer application using a C++ compiler. It has been tested against a variety of complex structures including fullerenes with up to 540 atoms. The results for structures in Figure 1 and Figure 3 are listed in Table 3. The CPU time test was done on a PC Pentium IV processor, 2 GHz, and 640 MB RAM memory. As seen from Table 3, even for the ultimate test of fullerene C_{540} the CPU

Table 2. Symmetry Group Permutations for Structure **IV**

permutations													symmetry operation	coordinates of the element		
1	2	3	6	7	8	9	4	5	10	11	12	13	E			
1	2	3	7	6	9	8	5	4	12	13	10	11	σ_d	0.0	0.0	1.0
1	3	2	8	9	6	7	4	5	11	10	13	12	σ_d	1.0	0.0	0.0
1	3	2	9	8	7	6	5	4	13	12	11	10	C2	0.0	1.0	0.0
1	4	5	10	11	12	13	2	3	6	7	8	9	C2	0.71	0.0	−0.71
1	4	5	11	10	13	12	3	2	8	9	6	7	S4	0.0	1.0	0.0
1	5	4	12	13	10	11	2	3	7	6	9	8	S4	0.0	1.0	0.0
1	5	4	13	12	11	10	3	2	9	8	7	6	C2	0.71	0.0	0.71

Table 3. Results and CPU Time Comparison for Some Complex Structures

struct no.	no. of atoms	graph automorphism	symmetry point group	CPU time: previous study[a]	CPU time: current study (s)
I	161	6.98×10^{41}	D_{4h}	5.95×10^{29} y[a]	0.046
II	65	1.68×10^{34}	D_{8h}	1.10×10^{22} y[a]	0.047
III	33	2.63×10^{35}	D_{32h}	1.78×10^{23} y[a]	0.359
C20	20	120	I_h	0.047 s	0.046
C60	60	120	I_h	0.578 s	0.140
C78	78	12	D_{3h}	3.891 s	0.047
C180	180	120	I_h	0.672 s	0.656
C240	240	120	I_h	1.359 s	1.110
C540	540	120	I_h	77.563 s	4.172

[a] Approximate time.

time of 4.2 s is pretty impressive. Note that the code described here does not use any simplifications for the structure of the fullerenes. All molecules are represented by the connectivity of the atoms and their three-dimensional coordinates.

From Table 3 it also can be seen the critical comparison of the CPU time between the current approach and the previous one.[1] The remarkable improvement is for the structures in Figure 1. Although the present algorithm does not solve the graph automorphism problem, which is not a subject of this study, it does solve the molecular symmetry identification problem in a quite straightforward way. It should be mentioned also that the run time listed in the table includes the time necessary for calculation of the type and the coordinates of the symmetry elements, which, depending on the complexity of the structure, could take up to 60% of the total time. Considering the issue of the molecular symmetry elements identification, there probably exists room for improvement, especially in the treatment of star structures (structure **III**, Figure 1) where the configuration name is composed of all possible ordered triads of atoms excluding the central one (see ref 1). Thus, it requires $(n - 1/3)$ elements, and for structure **III** (33 atoms) it takes more time to calculate the 4960 elements of the configuration name than the time necessary to identify 128 permutations that consist of D_{32h} symmetry point group. This issue could be a matter of future investigation.

4. CONCLUSIONS

A new algorithm and computer program for molecular symmetry identification have been developed in this study. The methodology calculates and compares the Euclidian distances between the atoms during the recursive DFS graph exploration procedure and eliminates the necessity to generate the entire graph automorphism group of the chemical structures. The result is very effective and extremely fast code where the only limitation is the physical memory of the computers. The current application can treat molecules up to 1000 atoms (145 atoms in the case of a star structure).

A demonstration version of the computer application is available upon request from the author.

REFERENCES AND NOTES

(1) Ivanov, J.; Schüürmann, G. Simple Algorithms for Determining the Molecular Symmetry. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 728−737.
(2) Longuet-Higgins, H. C. *Mol. Phys.* **1963**, *6*, 445.
(3) Masinter, L. M.; Shridharan, N. S.; Carhart, R. E.; Smith, D. H. *J. Am. Chem. Soc.* **1974**, *96*, 7714−7723.
(4) Cotton, F. A. *Chemical Applications of Group Theory*, 2nd ed.; Wiley: New York, 1981.
(5) Stewart, J. J. P. *MOPAC 2000*; Fujitsu Ltd.: Tokyo, Japan, 1999.
(6) Balasubramanian, K. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 621−626.
(7) Balasubramanian, K. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 1146−1150.
(8) Balasubramanian, K. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 761−770.
(9) Balasubramanian, K. *Chem. Phys. Lett.* **1995**, *232*, 415−423.
(10) Jiang, X. Y.; Bunke, H. *CVGIP: Graphical Models Image Process.* **1992**, *54* (1), 91−95.
(11) Yuen, K.; Chan, W. *Pattern Recognit. Lett.* **1994**, *15*, 279−286.
(12) Nedwed, K.; Gatterer, K.; Fritzer, H. P. *Comput. Chem.* **1994**, *18* (4), 271−376.
(13) Tarjan, R. E. *SIAM J. Comput.* **1972**, *1*, 46.
(14) Karabunarliev, S.; Ivanov, J.; Mekenyan, O. *Comput. Chem.* **1994**, *18*, 2, 189−193.

CI0341868