Structure Generation by the Combination of Structure Reduction and Structure Assembly[1]

Simona Bohanec

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

Received August 26, 1994[⊗]

A new method of structure generation, involving two steps, structure reduction and structure assembly, is presented. The goal of structure reduction is to reduce the number of connections between fragments and, consequently, the number of combinations that are considered in the second step. The structure assembly is a directed and controlled combinatorial process based on a generation tree. In contrast to other known generation methods, no comparisons and symmetry tests of intermediate fragments are made during this combinatorial process. Because the generation of duplicates cannot be completely avoided, the comparisons of final structures based on a quasi-tree representation of structures are performed to reject duplicates. The new method is called GEN and implemented on IBM PC and compatible computers. The effectiveness and applicability of GEN is demonstrated on several different examples and compared to some other generation methods.

INTRODUCTION

The main task of the generation process is to generate **all possible** chemical structures from the given input fragments. It is usually the central part of a larger system for structure generation that, in general, sequentially perform the following **three tasks**:

1. preparation of actual sets of fragments,
2. generation process,
3. elimination of duplicates.

These tasks are presented in Figure 1. The first task is the preparation of all possible actual sets of fragments by taking into account all different possible input data that may consist of molecular formula, structural constraints, type of final structures with exact or approximate number of rings or ring sizes, etc. The actual set is defined as a set of two or more fragments that must all be incorporated into the generated structures.

The second task is the generation process. It is a combinatorial process in which all possible combinations of connections of input fragments are tried in order to obtain all structures that satisfy the input constraints.

The third task is the elimination of identical final structures. In many generation systems, this part is an integral part of the generation process in order the avoid the generation of duplicates during the generation process. The generation is stopped whenever identical intermediate fragments are generated. However, since there are usually more intermediate fragments than final structures, we take another approach in GEN: the elimination of duplicates is performed **after** the generation process.

The main focus of this article will be the generation process, while the preparation of actual sets and the method used for the elimination of duplicates will be described in forthcoming papers.

PROBLEMS IN STRUCTURE GENERATION

The structure generation is a combinatorial process in which all fragments from the actual set are connected in all possible ways in order to obtain all structures that correspond to the input data and constraints (molecular formula, mandatory present and/or absent structural fragments, number and size of rings, presence or absence of bridges in final structures, etc.). The number of possible connections and the number of possible combinations, denoted by *CN* and *CB*, respectively, increase very rapidly with the number of free bonds *B* in the actual set

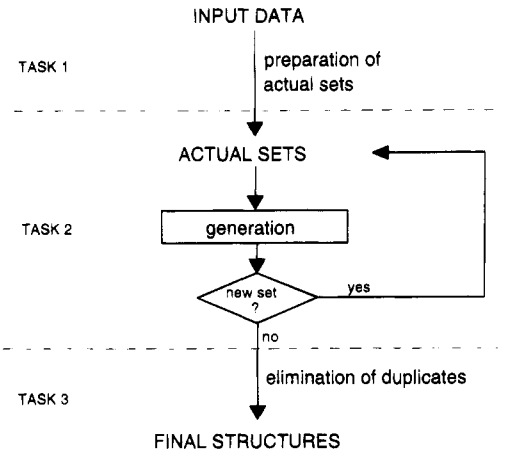

Figure 1. Basic tasks in structure generation.

$$CN = {}^1/_2 B(B-1) \quad (1)$$

and

$$CB = \prod_{i=1}^{B/2}(2i-1) = (B-1)(B-3)(B-5)... \quad (2)$$

The eq 2 consists of *B/2* factors. The increase of the number of free bonds *B* causes the increase of possible combinations of connections *CB*. When the number of combinations *CB* increase so much that these combinations cannot be examined in a reasonable amount of time, the first problem related to the generation process appears. This is the so-called **combinatorial explosion**.[2,3]

[⊗] Abstract published in *Advance ACS Abstracts*, April 15, 1995.

0095-2338/95/1635-0494$09.00/0

The second generation problem is the **generation of forbidden** structures. The forbidden structures are (1) disconnected structures, (2) inutile structures that are incompatible with input constraints, and (3) identical structures, called duplicates.[2] The detection of first two types of forbidden structures is simple and fast, while the detection of duplicates is usually complicated and time-consuming.

In order to minimize these two problems different methods were developed and explained in the literature. These methods include

- the use of standard built-in components,[4-8]
- the use of different data bases with already generated structures or substructures,[9-12]
- symmetry tests of the intermediate fragments during the combinatorial process,[4,5]
- comparisons of the intermediate fragments during the combinatorial process,[3,13-15]
- multistep generation process (after each step the intermediate structures are examined and duplicates are eliminated),[16-19] and
- the combination of several of the above approaches.

There are many open questions related to these methods. For example, (1) is the generation from built-in components exhaustive for all possible problems even if some components are not built into the generator, (2) do the built-in data bases include all possible structures or substructures and if they do, how large are they, and (3) how time-consuming are the actions applied on intermediate fragments in each step of the combinatorial process and how fast such a generation really is, etc.

It is known that the generation of duplicates cannot be completely avoided if the input fragments are of arbitrary size.[2]

METHOD

We propose a new generation method that does not include any of the possibilities for the reduction of the generation problems mentioned in the previous section. We introduce a new generation process that consists of two tasks. The first is the **elimination** of useless connections that lead to the generation of forbidden structures. The second is the **controlled combinatorial process**. Therefore, the proposed generation process consists of two sequentially performed steps: (1) **structure reduction** and (2) **structure assembly**.

During the structure reduction the generation of as many forbidden structures as possible should be avoided by the elimination of **inutile** connections. This step is similar to the structure reduction introduced by Christie and Munk, the authors of the structure generator COCOA.[7] Between COCOA and our GEN there are two evident differences: first, our GEN (during the structure reduction) takes into account **more** diverse input information than COCOA and, second, in GEN the structure assembly **follows** the structure reduction, while in COCOA it does not.

The second step of the generation process is controlled and directed combinatorial process. The control is made by testing intermediate fragments to prevent the generation of disconnected and inutile structures whose generation is not prevented in the first step. Nevertheless, by these tests the generation of duplicates can not be completely avoided. They are eliminated **after** the combinatorial process by performing a new method for structure comparisons based on a "quasi-tree" representation of structures.[20]

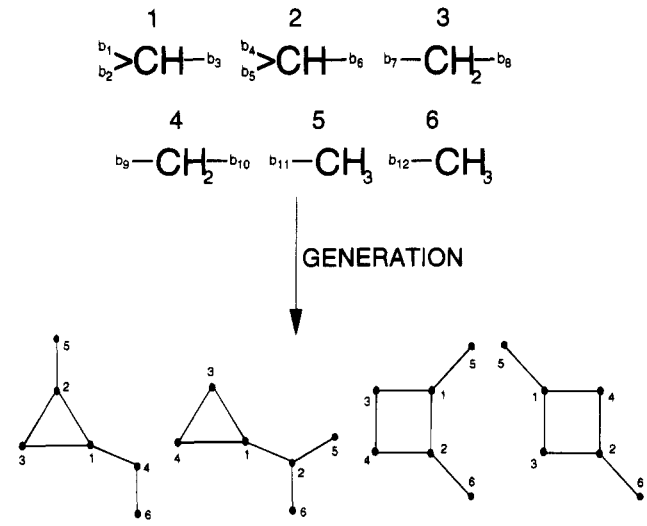

Figure 2. Generation of all structures with one ring from six input fragments.

The generation based on structure **reduction** is faster and more efficient when the structural constraints are given. On the contrary, the structure **assembly** is faster if it is performed without constraints.[7] Our idea is by using the combination of both approaches, the generation will be faster compared to the generation that includes only structure reduction **or** structure assembly.

EXAMPLE

To explain our new generation method a problem given in Figure 2 will be used as an example. This example requires a generation of all structures having **exactly one ring** from six input fragments that constitute an **actual set**. In the actual set there are three pairs of identical input fragments that have three, two, and one free bond, respectively. Free bonds of fragments are marked with b_i, where i is the identification number of the bond. The number of all free bonds is labeled as B. In our example B is equal to *12*. Therefore, to obtain the complete structure from these fragments $B/2$ (in our example *6*) connections should be made. As shown in Figure 2, there are four possible final structures.

In the following paragraphs the generation process is described and explained by the generation tree and by different connection matrices.[21,22]

STRUCTURE REDUCTION

The structure reduction is the first step of the generation process. The main task of the structure reduction is the elimination of connections that cause the generation of forbidden structures. The connections that are not eliminated are hierarchically ordered into the tree, called the **generation tree**. These ordered connections will be applied in a prescribed order during the structure assembly.

The efficiency of the structure reduction depends on the **input data**. If there are many exactly defined input data, the elimination of connections is more efficient. As a consequence, during the structure assembly fewer connections and combinations of connections are checked.

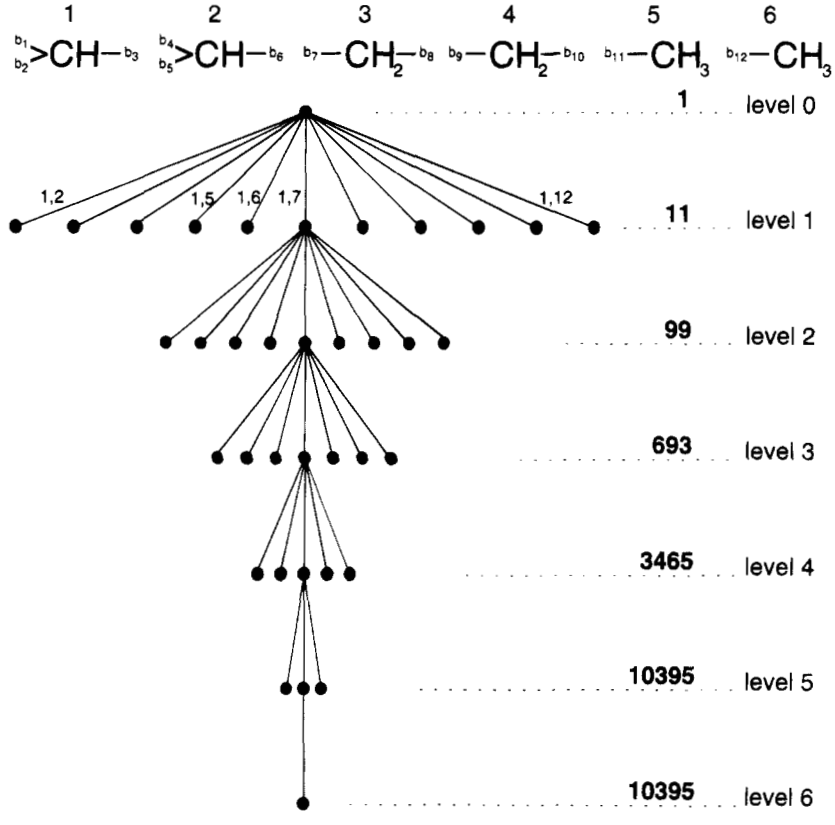

Figure 3. Complete generation tree for the set of fragments with 12 free bonds. Bold numbers at level marks represent the number of vertices in this level.

The reduction starts with the generation of the **complete generation tree** that contains **all** possible connections between fragments in the actual set. The generation tree defines the sequence of the connections that should be applied during the structure assembly. The generation tree, as any tree known in graph theory,[23,24] contains edges and vertices. The edges and the vertices represent the connections and fragments, respectively. The root vertex represents the disconnected input fragments, the vertices in the first level represent the input fragments connected with exactly **one** connection, etc. Vertices in the lowest level (called leaves) represent the final structures.

The size of the complete generation tree depends on the **number of free bonds** B of the fragments in the actual set. The generation tree has exactly $B/2$ levels. The degree of the root vertex is $(B - 1)$. The degree of all vertices in the first level is $(B - 2)$; one edge connects the observed vertex with the root vertex and $(B - 3)$ edges connect the observed vertex with vertices in the second level. From each vertex in the second level there are $(B - 5)$ edges connected with vertices in fourth level and so on. The number of vertices in the second level of the generation tree is $(B - 1)$, in the third is $(B - 1)\cdot(B - 3)$, and so on. Therefore, the number of leaves is equal to $(B - 1)\cdot(B - 3)\cdot(B - 5)\cdot \ldots \cdot (B - B + 1)$. The number of leaves is equal to the number of combinations of connection CB (eq 2). One combination of connections that leads to the generation of one final structure is represented by one branch of the generation tree. This branch goes from the root to the observed leaf of the tree.

A part of the complete generation tree for the example from Figure 2 is given in Figure 3. It has seven levels and 10 395 leaves (11·9·7·5·3·1, eq 2). Because there are 10 395 combinations of seven connections, this tree represents the generation of 10 395 final structures.

	1 b_1	1 b_2	1 b_3	2 b_4	2 b_5	2 b_6	3 b_7	3 b_8	4 b_9	4 b_{10}	5 b_{11}	6 b_{12}
1 b_1		T	T	T	T	T	T	T	T	T	T	T
1 b_2			T	T	T	T	T	T	T	T	T	T
1 b_3				T	T	T	T	T	T	T	T	T
2 b_4					T	T	T	T	T	T	T	T
2 b_5						T	T	T	T	T	T	T
2 b_6							T	T	T	T	T	T
3 b_7								T	T	T	T	T
3 b_8									T	T	T	T
4 b_9										T	T	T
4 b_{10}											T	T
5 b_{11}												T
6 b_{12}												

Figure 4. Full connection matrix with all possible connections of fragments with 12 free bonds.

In the computer the complete generation tree is represented by a **full connection matrix**[21,22] (Figure 4). The full connection matrix consists of B rows and B columns. Each row and each column represent exactly one free bond. The value of the connection matrix element e_{ij} (in the ith row and jth column) can be TRUE or FALSE. The value TRUE means that the connection between the free bonds i and j is possible, if the value is FALSE, the connection is not possible. The complete generation tree can easily be reconstructed from the full connection matrix: the elements e_{ij} in the first row of the full connection matrix represent the connections between the vertices in the zeroth and those in the first level of the complete generation tree, the elements

in the second (or in the third row if the second free bond has already been used for the first connection) represent the connections between the vertices in the first and those in the second level of the tree, etc.

The size of the complete generation tree and of the full connection matrix depends only on the number of free bonds B in the actual set. Therefore, all possible connections of fragments CN (eq 1) and all possible combinations of connections CB (eq 2) of **different** actual sets with the **same** number of free bonds can be represented by the **same** complete generation tree and with the **same** full connection matrix, respectively.

In the complete generation tree and in the full connection matrix there are many connections that cause the generation of forbidden structures. All such connections should be eliminated during the structure reduction. In the complete generation tree, the elimination can be regarded as the **pruning** of the tree. The eliminated connections get the FALSE value in the full connection matrix. These eliminations depend on the type of input fragments and on other data about the final structures.

There are two different types of pruning of the generation tree. The first is the **reduction of the tree's levels** (the tree's depth), while the second is the **reduction of the tree's breadth**.

The reduction of the tree's levels is achieved by linking some input fragments **before** the combinatorial process if some **mandatory present structural fragments** are given. One preliminary connection causes the reduction of the number of free bonds from B to $(B - 2)$. Consequently, it lowers the number of levels of the generation tree by one and reduces the connection matrix for two rows and two columns.

For example (Figure 2), if the fragment $-CH_2CH_3$ is given as the mandatory present structural fragment, the preliminary connection between one $-CH_2-$ fragment and one $-CH_3$ fragment are made. Now the generation tree has only six levels (marked with 0, 1, 2, 3, 4, and 5) instead of seven levels and 945 (= 9·7·5·3·1) leaves instead of 10395 (Figure 3).

In general, after j preliminary connections the generation tree is lower by j levels, and the connection matrix is smaller for $(j \times 2)$ rows and columns.

The reduction of the generation tree's breadth is caused by the elimination of some edges (connections) which lead to the generation of forbidden structures. There are five types of such connections:

A: the connections that link free bonds on the **same** atom;

B: the connections that lead to the generation of **identical** structures;

C: the connections that lead to the generation of **disconnected** structures;

D: the connections that lead to the generation of mandatory **absent** structural fragments;

E: the connections that link free bonds that are on the **same** mandatory present structural fragment.

Connection of type A is the first connection $e_{1,2}$ on the left side of the tree going from the root to the first level (Figure 3) because the free bonds b_1 and b_2 are on the same atom. Therefore, the connection $e_{1,2}$ should be eliminated. By its elimination the entire left branch of the generation tree is **pruned**. The same is done also with the connections $e_{1,3}$, $e_{2,3}$, etc., which link free bonds on the same atom. In the full connection matrix, all connections of type **A** are marked as shaded boxes in Figure 5a.

Connections of type B lead to the generation of identical intermediate fragments and finally to the identical structures. It is obvious that any connection between two equal atoms leads to the generation of identical fragments. For example, each of nine possible connections between the first and the second fragment lead to the same intermediate fragment >CH−CH<. Therefore, among nine connections only one ought to be used, while others should be eliminated. All connections B for the given example are marked in Figure 5b.

Connections of type C lead to the generation of disconnected structures. The generation tree defines the sequence of the connections that should be applied in the set of fragments. Sometimes, the connections that connect fragments with two or less free bonds are used **too early** during the structure assembly what causes the generation of disconnected structures. The special case of connections of type **C** is the connection of two fragments with **exactly one** free bond each. Such case represents the connection $e_{11,12}$ (Figure 5) which causes the generation of the molecule of ethane. All connections of type **C** are shown in Figure 5c.

Connections of type D cause the generation of mandatory absent structural fragments. For example, if the fragment $-CH_2-CH_2-$ is listed as a mandatory absent one, all connections between two $-CH_2-$ input fragments are eliminated. Such connections are marked in Figure 5d.

Connections of type E link free bonds on a single mandatory present structural fragment. This type of connections appears usually after the preliminary connections that cause the decrease of the generation tree's levels.

With all above mentioned eliminations, the **specific generation tree** is obtained from the complete generation tree. In the computer program it is represented by the **specific connection matrix**. For the example given in Figure 2, this process is shown schematically in Figure 6. The specific connection matrix has only 23 connections left.

The specific generation tree for the given example is shown in Figure 7. It contains only 14 leaves. Therefore, according to this tree, only 14 structures can be generated. Compared to previous 10 395 structures it means that the structure reduction reduces the number of generated structures almost 800 times.

The connections in the specific connection matrix are numbered. Each connection has the identification number ID (Figure 8a) and the equivalence number EQ (Figure 8b).

The ID numbers increase from left to right and from up to down of the matrix, while the value of the EQ numbers depends on the identification numbers of atoms that are connected by the observed connection. The identification number of the atom having the free bond b_i is marked as ID_i. The EQ number of the connection e_{ij}, which connects free bonds b_i and b_j (atoms marked with ID_i and ID_j), is marked as EQ_{ij}. All EQ_{ij} values are calculated by the equation

$$EQ_{ij} = (ID_i - ID_1)N + ID_j - 1 \qquad (3)$$

where ID_1 is the identification number of the atom on which the first free bond is located and N is the number of non-

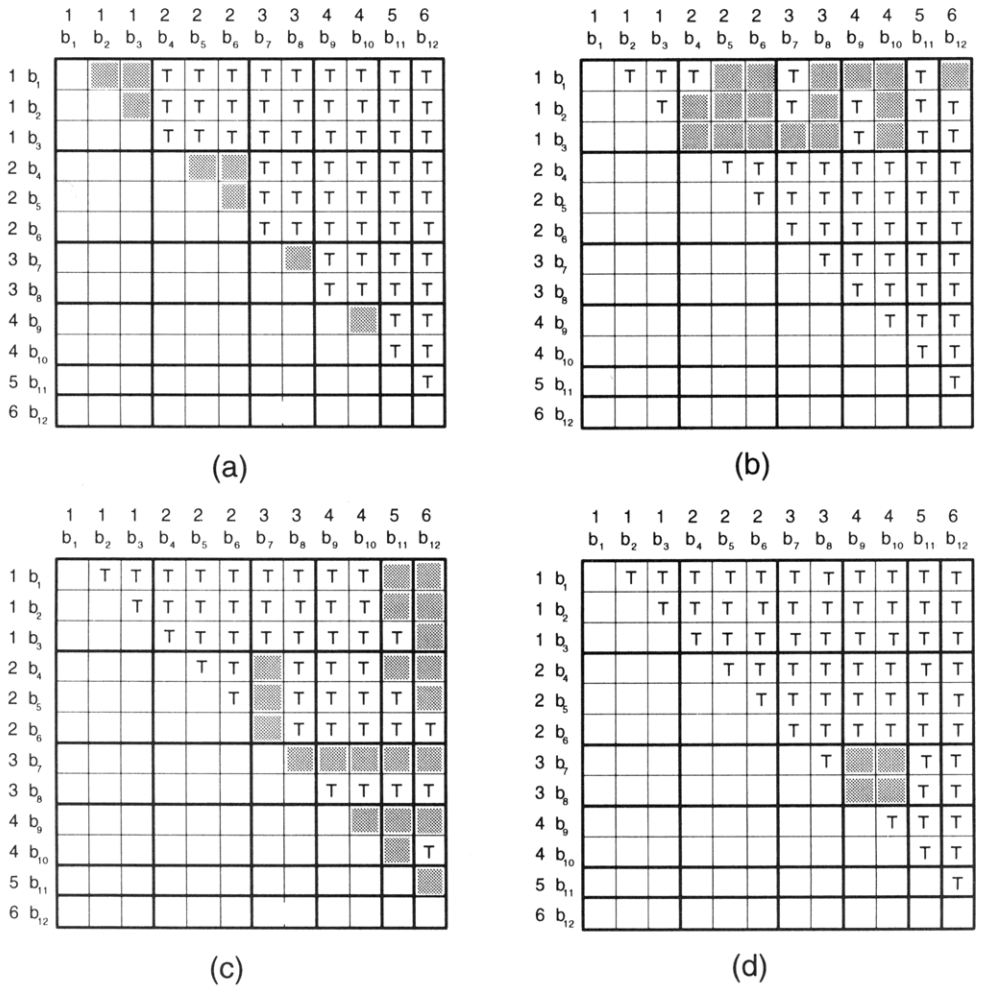

Figure 5. Full connection matrices with marked connections of type (a) **A**, (b) **B**, (c) **C**, and (d) **D.**

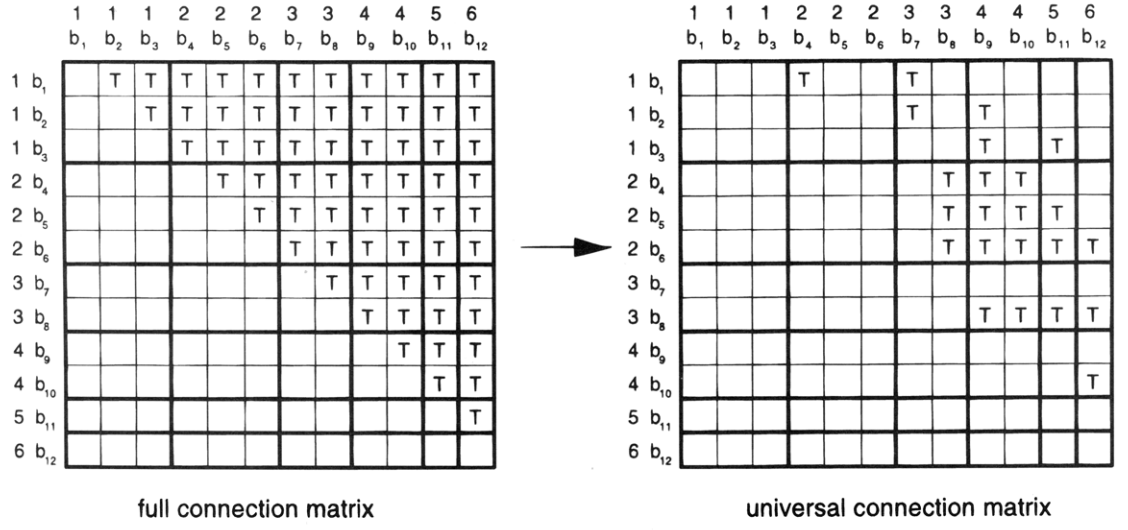

Figure 6. Full and specific (universal) connection matrix obtained after the structure reduction for the example in Figure 2. The TRUE elements are marked with T; the FALSE ones are unmarked.

hydrogen atoms in the actual set. In Figure 8 the connections with identical *EQ* numbers are grouped together and separated from the others by thick lines.

STRUCTURE ASSEMBLY

Once the specific generation tree and the specific connection matrix with numbered connections are obtained the structure assembly can begin. The structure assembly is performed by the controlled and directed combinatorial process. This combinatorial process can be described as the examination of the specific generation tree. This examination is a multistep process. One step can be viewed as a passage from one vertex to another following an edge representing a connection. Going down the tree from one to another vertex is equivalent to **adding** the passed (examined) connection into the actual set, while going up the tree is

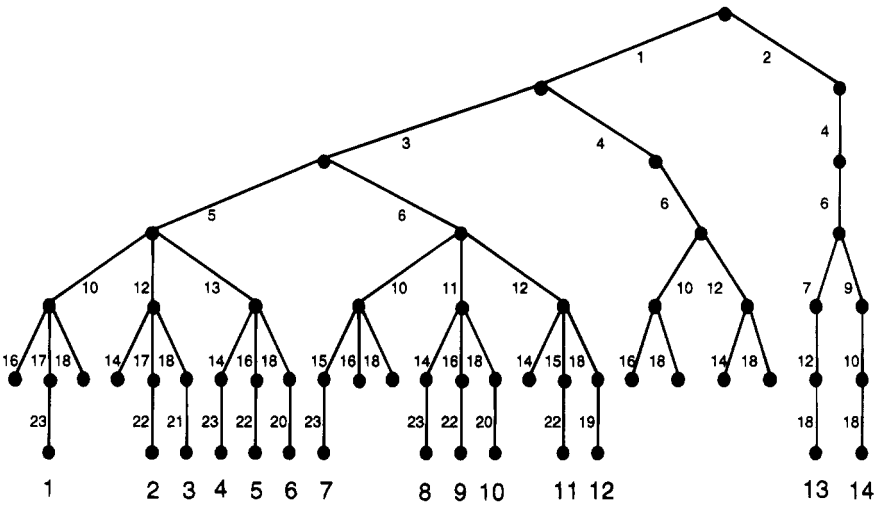

Figure 7. Specific generation tree.

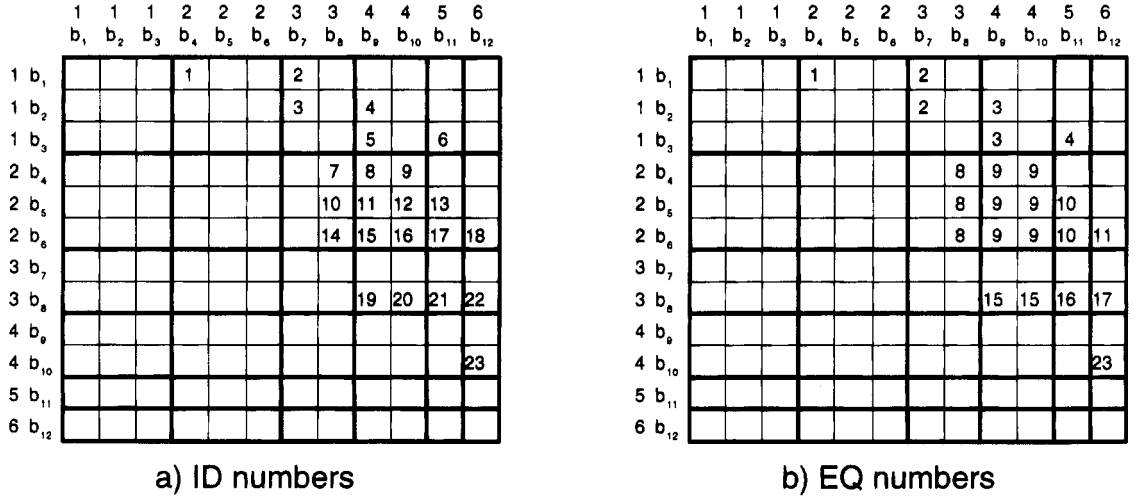

Figure 8. Numbered connections of the specific connection matrix in Figure 6.

equivalent to **elimination** (disconnection) of the connection from the set.

The examination of the specific generation tree is controlled and directed by several **rules** and **tests**. The rules are related to the edges (connections) of the specific generation tree. They make the examination of the tree more efficient by choosing the right direction (proper edge). The tests are related to the intermediate fragments represented by all vertices except the root and the leaves.

The rules are the following:

Rule 1 (R1)—the examination of the tree is determined by the depth first search algorithm: the edges which are **lower** and on the **left** compared to the other edges are examined first;

Rule 2 (R2)—the edge that is examined **must** have **larger** *ID* value than the previous one being examined;

Rule 3 (R3)—the edge that is examined **must** have **greater** (or the same) *EQ* value than the previous one. The same *EQ* value is allowed only in some special situations when the single connections can be linked together to form double or triple bonds;

Rule 4 (R4)—if there are two edges with different *EQ* values that cause the generation of the identical intermediate fragments, only the edge with the smaller *EQ* value is examined. In Figure 8 such two edges are edges 3 and 4 or connections 3 and 4 in Figure 9. They have *EQ* values 2 and 3, respectively. Because the third and the fourth input fragments are identical, these two connections are identical too.

The tests of intermediate fragments that should be performed on the fragments in **all** vertices (except the root and the leaves) must be **fast**, because they are used in **all** steps of the combinatorial process. The tests return the TRUE value:

Test 1 (T1)—if the number of connections added into the set **is smaller** than the number of intermediate fragments in the observed vertex;

Test 2 (T2)—if each intermediate fragment **has at least one** free bond;

Test 3 (T3)—if the number of intermediate fragments having exactly one free bond **is smaller** or **equal** to the number of connections that should be added into the set;

Test 4 (T4)—if the number of free bonds on one atom from the set **is smaller** or **equal** to the number of connections that should be added into the set;

Test 5 (T5)—if the number and the type of multiple bonds in intermediate fragments correspond to the input data with respect to the input data about the number and types of bonds in the final structures;

Test 6 (T6)—if the number and the size of rings in the intermediate fragments correspond to the number and the size of rings in final structures defined by the input data.

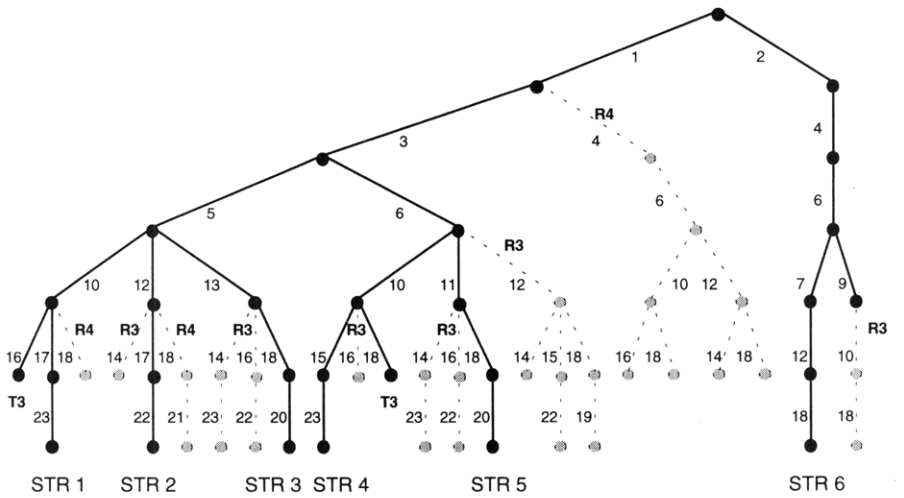

Figure 9. Examination of the specific generation tree. The links where the examination was stopped are marked with **R3**, **R4,** and **T3.**

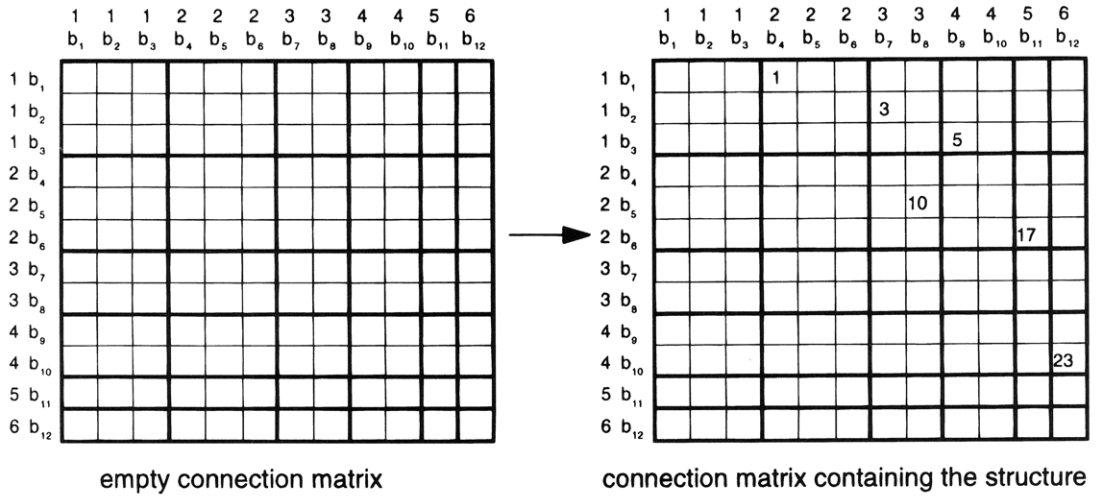

Figure 10. Filling the empty connection matrix with connections from the specific connection matrix to obtain the connection matrix containing the structure.

If all the above test returns the TRUE value, the examination of the generation tree is continued. Otherwise, the examination is stopped and redirected in a back-tracking manner.

In the specific generation tree all edges that have been examined during the structure assembly are marked with full lines (Figure 9). The places where the examination have been redirected are marked with **R3**, **R4**, and **T3**, according to the rule or test that caused the redirection in this special case.

It can be seen that the traverse of the tree has allowed the access to only six leaves. Therefore, only six structures were generated. This means that the controlled and directed combinatorial process additionally reduces the number of generated structures.

Regarding the connection matrices the structure assembly can be described as a process of **filling** an empty connection matrix (connection matrix with no connections) with the connections from the specific connection matrix by taking into account all previously mentioned rules and tests. Schematically this process is shown in Figure 10. When the empty matrix is filled with *B/2* connections the "**connection matrix containing the structure**" is obtained.

The onnection matrix containing the structure contains *B/2* connections from the specific connection matrix. In each column or in each row of this matrix there is exactly one connection. When all these *B/2* connections from the connection matrix containing the structure are used for linking fragments from the actual set, one final structure is obtained. In Figure 10 is the connection matrix that contains the first generated structure of the given example.

In Figure 9, there are six combinations of connections, represented as thick branches of the specific generation tree. The final results of the structure assembly are the following six different combinations of connections

1. 1 3 5 10 17 23

2. 1 3 5 12 17 22

3. 1 3 5 13 18 20

4. 1 3 6 10 15 23

5. 1 3 6 11 18 20

6. 2 4 6 7 12 18

These combinations determine six generated structures shown in Figure 11.

In the next step these structures are examined in order to remove the duplicates. In our generation method the

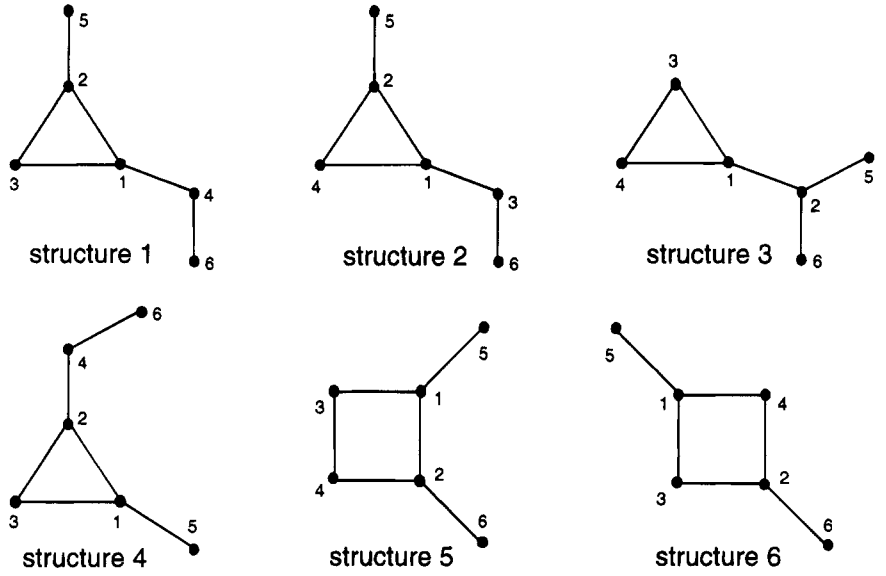

Figure 11. Six final structures before the elimination of duplicates.

Table 1. Comparison of Generation Times for Alkane Series Using Different Generators

molecular formula	no. of structures	CPU times [s]		
		AEGIS	Faulon	GEN
C_5H_{12}	3	25	10	0.05
C_6H_{14}	5	91	12	0.05
C_7H_{16}	9	409	15	0.11
C_8H_{18}	18	2153	20	0.28
C_9H_{20}	35		41	0.66
$C_{10}H_{22}$	75		94	1.81
$C_{11}H_{24}$	159		238	5.66
$C_{12}H_{26}$	355		608	19.7
$C_{14}H_{30}$	1858		3600	372
$C_{16}H_{34}$	10359		42180	18046

elimination of duplicates is done by the comparison based on the "quasi-tree" representation of structures.[20] This method will be described in detail in one of the forthcoming articles where also some comparisons of other methods for eliminations, based on canonical coding,[25-29] will be given.

For present understanding it is sufficient to enumerate only the comparisons that should be made to discard the duplicates. There are three such comparisons that include structures 1 and 2, 1 and 3, 1 and 4. These comparisons give the following result: structures 1, 2, and 4 are identical, so the last two (2 and 4) should be eliminated. The comparisons of structures 1, 5, and 6 are not needed because by their quasi-tree representation they are found as different.

Finally, the results of the given example are four different structures. These are structures 1, 3, 5, and 6.

RESULTS

Our new method for the generation of chemical structures is implemented as a computer package called GEN on personal computers IBM PC XT/AT/386/486 and compatibles. All tests are made on 486 PC (33 MHz), and results are compared to other generators.

The first test is the classical example of the generation of alkanes series. There are many generators whose efficiency was tested using this example. All generators generate the same number of alkanes. In Table 1 three generators, developed in years 1991/94, are compared: generator AEGIS,[30] Faulon's generator,[15] and our generator GEN.[21,22] Generator AEGIS works on Apple Macintosh II computer and Faulon's generator on SUN/SPARC IPC workstation which is faster compared to personal computers. Nevertheless, the results show that our generator GEN is faster compared to the other two generators.

The Faulon's generation method is based on the comparisons of intermediate fragments and stops the generation when fragment, identical to the previously generated one, is found. There are two main disadvantages related to this method: (1) many comparisons due to many intermediate fragments have to be performed and (2) the requirement of a considerable amount of computer direct access memory.

The authors of generator AEGIS argue that their generator is slow mainly because of the programming language Prolog.[30]

The second test was the generation of all possible structures from the given molecular formula applying several structural constraints described in refs 5, 21, and 22. Results of our generator are compared to the generator of the system CHEMICS[5] that is implemented on MV/2000DC computer (Table 2). Note also that in examples 1−6 and 10 both generators generate a **different** number of structures. Comparisons of results for the same molecular formulas obtained by generators MOLGEN[10,11] and AEGIS[30] show that our generator gives the correct results in all disputed cases, while the generator in system CHEMICS does not. Two significant reasons for such errors lie in the generation method implemented in the system CHEMICS:

1. the use of standard components only. If some of standard fragments are not built into the system, several structures will not be generated, and

2. the assumption that by symmetry tests of intermediate structures and the isomorphism checking by the connectivity stack method the generation of duplicates can be avoided completely. Because this is not always true, in some cases the duplicates are generated.

In our opinion, the direct comparison of the generation times is unsuitable due to the fact that the data about the generation times of system CHEMICS are too old[5]. Nevertheless, the relative time comparisons of examples, that are carried out from the same input data but using different generators and different computers, can be made. These comparisons are the following:

Table 2. The generation of Structures from the Molecular Formula and Some Mandatory Present and Mandatory Absent Structural Fragments

example	molecular formula	structural constraints		CHEMICS		GEN	
		present	absent	no. str.	time (s)	no. str.	time (s)
1	C_6H_9NO			37491	169488	35759	99668
2	C_6H_9NO	$-CO- -CH{=}CH_2$		131	1826	155	5
3	C_6H_9NO	$-CH{=}CH_2$	$-OH$	1613	129920	1987	195
4	C_5H_7NO			7075	23889	6637	2163
5	C_5H_7NO	$-CO- -CH{=}CH_2$	A−NH−A	13	380	12	1.5
6	C_4H_7NO			802	1436	764	23.2
7	C_4H_7NO	$-NHCO-$		5	34	5	0.61
8	$C_4H_5O_2Cl$			907	840	907	64.5
9	$C_4H_5O_2Cl$	$CH_3CO- -COCl$		1	6	1	0.11
10	C_3H_7NO			87	52	84	1.21
11	C_3H_7NO	$-NH_2$	Δ[a]	25	44	25	0.71

[a] "Δ" is a sign for 1,2-disubstituted 3C-ring.

•The decrease of execution times in example 1 (without constraints) with respect to the example 2 (two mandatory present structural fragments) is about **95-times** for the system CHEMICS and almost **20 000-times** for our generator GEN.

•Comparing examples 1 (without constraints) and 3 (with one present and one absent structural constraint) system CHEMICS is faster **13-times** and our GEN **500-times**.

•Comparing examples 4 (without constraints) and 5 (with two present and one absent constraint) system CHEMICS is faster **60-times** and our GEN **1400-times**, and so on.

These comparisons show how the inclusion of structural constraints affects the generation times in structure assembly concept (system CHEMICS) and in the combination of structure reduction and structure assembly concept (our new method). The concept used in GEN is evidently more efficient.

Instead of these relative time comparisons the computational complexity of measurements might give better comparison of two methods. There are two reasons why the computational complexities of generations are not given here: (1) the computational complexity depends very much on the available constraints, such as the type of generated structures (see the third example in the next paragraph) and the structural constraints, and (2) the computational complexities for other generators were not given in the literature.

The third test is the generation of different types of structures from the same molecular formula as the main input. Generator GEN offers the ability to define the type of final generated structures with different extent of accuracy: one can give the exact number of rings and their sizes or an interval restricted by the minimal and maximal number of rings and minimal and maximal size of the rings. One can also choose the output structures that contain bridges or not.

In Table 3 some examples of the generation of different **types** of final structures having the same molecular formula C_5H_8O are shown. From these results we can get a direct proof of the efficiency of our generator. For example, by the addition of the number of structures obtained by examples 2, 3, and 5 (62 + 103 + 40) we get 205 structures, that is exactly the number obtained by the first example.

As one can conclude from the literature data, other generators mainly need an exact description of the generated structures[7,16−18] or offer an alternative: the generation of **all** structures or only structures **without** rings.[10,11,30]

Table 3. Generation of Different Types of Structures with Molecular Formula C_5H_8O

example	no. of rings	ring size	bridge? (Y/N)	no. of structures	CPU times (s)
1	any	any	Y	205	5.99
2	0	*a*	*a*	62	2.14
3	1	any	*a*	103	3.02
4	1	5- or 6-membered ring	*a*	14	2.75
5	2	any	Y	40	1.48
6	2	any	N	34	1.76
7	2	one 3- and one 5-mem.	Y	3	5.88
8	2	one 3- and one 5-mem.	N	3	5.49
9	2	4- or more membered ring	Y	7	1.32
10	2	4- or more membered ring	N	1	1.37

[a] Data are not important. For example: in structures with no rings the data about the ring size is irrelevant.

CONCLUSION

A new method for the computer generation of chemical structures is described. This method is a combination of structure reduction and structure assembly concepts and includes the advantages of both concepts. The structure reduction is faster for the generation with structural constraints, while the structure assembly is faster for the generation without structural constraints. As shown in Tables 1 and 2, the new method is faster compared to methods using the structure assembly only. Until now the structure reduction concept has not been widely used, because, according to the available information, it is implemented in only one generator designed for the computer-assisted structure elucidation system CASE.[7,31,32]

Generator GEN takes into account a variety of input data given with any extent of completeness. Some of possible input data are as follows:

•exact or approximate molecular formula,

•structural constraints—mandatory present and/or mandatory absent structural fragments in the final structures, which can overlap or not,

•exact or approximate number of rings in the final structures,

•exact or approximate ring sizes in the final structures,

•number and types of free bonds if final structures should be the substructures (large fragments),

•"bridge?" option can be set either to "Y" (yes) or "N" (no) for generating all or only structures without bridges, respectively, and

•any combination of the above data.

The input data are considered in the generation process **before**, **during,** and/or **after** the combinatorial process, i.e., as soon as adequate. If they are taken into account before the combinatorial process, i.e., during the structure reduction, this affects the generation time the most.

By choosing the input data properly, the generator GEN can be used also for the generation of the following:

•structures with defined skeleton (as shown in ref 25),

•substructures with defined number and types of free bonds,

•skeletons with the exact number of rings, double, and/or triple bonds (as in refs 16−18),

•large structures with some atom groups defined as "superatoms" (as in ref 15),

•structures defined by their molecular formula, present and absent structural fragments, number of rings, size of rings, etc.

The generator GEN is written in Turbo Pascal programming language. It is incorporated into two different systems for structure generation GENMAS and GENSTR.[21,22] Both systems can be used independently, while the latter can be used also in the large package called KI CARBON SOFTWARE[33,34] developed for structure elucidation on the basis of ^{13}C NMR data. Both systems and package are implemented on personal computers IBM PC and compatibles.

ACKNOWLEDGMENT

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

REFERENCES AND NOTES

(1) This work was presented as a part of Ph.D. Thesis at the Chemistry Department of the University of Ljubljana, Slovenia.
(2) Gray, N. A. B. *Computer-Assisted Structure Elucidation,* John Wiley & Sons: New York, 1986; Chapter 1, p 325.
(3) Bangov, I. P. Computer-Assisted Structure Generation from a Gross Formula. 3. Alleviation of the Combinatorial Problem. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 277−289.
(4) Abe, H.; Okuyama, T.; Fujiwara, I.; Sasaki, S.-I. A Computer Program for Generation of Constitutionally Isomeric Structural Formulas. *J. Chem. Inf. Comput. Sci.* **1984**, *24*, 220−229.
(5) Funatsu, K.; Miyabayaski, N.; Sasaki, S.-I. Further Development of Structure Generation in the Automated Structure Elucidation System CHEMICS. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 18−28.
(6) Robien, W. Computer-Assisted Structure Elucidation of Organic Compounds III: Automatic Fragment Generation from ^{13}C-NMR Spectra. *Mikrochim. Acta* **1986**, *II*, 271−279.
(7) Christie, B. D.; Munk, M. E. Structure Generation by Reduction: A New Strategy for Computer-Assisted Structure Elucidation. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 87−93.
(8) Razinger, M.; Zupan, J.; Novi, M. Computer Generation of Chemical Structures from Known Fragments. *Mikrochim. Acta* **1986**, *II*, 411−421.
(9) Nakayama, T.; Fujiwara, Y. Structure Generation on the Basis of BCT Representation of Chemical Structures. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 218−223.
(10) Kerber, A.; Lane, R.; Moser, D. Ein Strukturgenerator für molekulare Graphen. *Anal. Chim. Acta* **1990**, short communication, *235*, 221−228.
(11) Grund, R.; Kerber, A.; Laue, R. MOLGEN, ein Computeralgebra-System für die Konstruktion molekularer Graphen. "MATCH" *Communications in Mathematical Chemistry*, Special Issue, Edited by A. Kerber, **1992**, *27*, 87−131.
(12) Carhart, R. E.; Smith, D. H.; Brown, H.; Sridharan, N. S. Applications of Artificial Intelligence for Chemical Inference. XVI. Computer Generation of Vertex-Graphs and Ring Systems. *J. Chem. Inf. Comput. Sci.* **1975**, *15*, 124−130.
(13) Bangov, I. P. Computer-Assisted Structure Generation from a Gross Formula: II. Multiple Bond Unsaturated and Cyclic Compounds. Employment of Fragments. *J. Math. Chem.* **1988**, *2*, 31−48.
(14) Bangov, I. P. Computer-Assisted Structure Generation from a Gross Formula. 4. Figurehting Against Graph-Isomorphism Disease. *Match* **1992**, *27*, 3−30.
(15) Faulon, J.-L. On Using Graph-Equivalent Classes for the Structure Elucidation of Large Molecules. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 338−348.
(16) Contreras, M. L.; Valdivia, R.; Rozas, R. Exhaustive Generation of Organic Isomers. 1. Acyclic Structures. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 323−330.
(17) Contreras, M. L.; Valdivia, R.; Rozas, R. Exhaustive Generation of Organic Isomers. 2. Cyclic Structures: New Compact Molecular Code. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 483−491.
(18) Contreras, M. L.; Rozas, R.; Valdivia, R. Exhaustive Generation of Organic Isomers. 3. Acyclic, Cyclic, and Mixed Compounds. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 610−616.
(19) Carhart, R. E.; Smith, D. H.; Gray, N. A. B.; Nourse, J. G.; Djerassi, C. GENOA: A Computer Program for Structure Elucidation Utilizing Overlapping and Alternative Substructures. *J. Org. Chem.* **1981**, *46*, 1708−1718.
(20) Bohanec, S.; Perdih, M. Symmetry of Chemical Structures: A Novel Method of Graph Automorphism Group Determination. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 719−726.
(21) Bohanec, S.; Zupan, J. Structure Generation of Constitutional Isomers from Structural Fragments. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 531−540.
(22) Bohanec, S.; Zupan, J. Structure Generator GEN. *Match* **1992**, *27*, 49−85.
(23) Harary, F. *Graph Theory*, 3rd printing; Addison-Wesley: Reading, MA, 1972.
(24) Balaban, A. T. Applications of Graph Theory in Chemistry. *J. Chem. Inf. Comput. Sci.* **1985**, 25, 334−343.
(25) Morgan, H. L. The Generation of a Unique Machine Description for Chemical Structures−A Technique Developed at Chemical Abstracts Service, *J. Chem. Doc.* **1965**, *5*, 107−112.
(26) Wipke, W. T.; Dyott, T. M. Stereochemically Unique Naming Algorithm. *J. Am. Chem. Soc.* **1974**, *96*, 4834−4842.
(27) Shelley, C. A.; Munk, M. E. An Approach to the Assignment of Canonical Connection Tables and Topological Symmetry Perception, *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 247−250.
(28) Jochum, C.; Gasteiger, J. Canonical Numbering and Constitutional Symmetry, *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 113−117.
(29) Moreau, G. A Topological Code for Molecular Structures. A Modified Morgan Algorithm *Nouv. J. Chim.*, **1980**, *4*, 17−22.
(30) Luinge, H. J. AEGIS, A Structure Generation Program in Prolog. *Match* **1992**, *27*, 175−189.
(31) Shelley, C. A.; Munk, M. E. CASE, a Computer Model of the Structure Elucidation Process. *Anal. Chim. Acta* **1981**, *133*, 507−516.
(32) Munk, M. E.; Farkas, M.; Lipkus, A. H.; Christie, B. D. Computer Assisted Chemical Structure Analysis. *Microchim. Acta* **1986**, *II*, 199−215.
(33) 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.
(34) 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 Technology'; Karjalainen, E. J., Ed.; *Scientific Computing and Automation (Europe);* Elsevier: Amsterdam, 1990; Vol. 6, pp 393−405.

CI940099R