On Using Graph-Equivalent Classes for the Structure Elucidation of Large Molecules

Jean-Loup Faulon

Fuel Science Program, The Pennsylvania State University, University Park, Pennsylvania 16802

Received January 31, 1992

This paper presents a new algorithm to enumerate the isomers of organic compounds. The method is based on the concept of equivalent classes which is now largely used in computational chemistry. The utilization of the equivalent classes enables the elucidation of molecules containing a large number of atoms in a realistic computational time. The main problem during the isomer-generation process is to avoid the presence of redundant structures. The number of identical structures can be enormous, and the process by which two structures are compared is time consuming. We show how with the equivalent classes it is possible to reduce the number of comparisons and reduce the complexity of the isomorphism check function. Exhaustivity and irredundancy of our generator are examined by using graph theory. This technique has been employed with success in the real world of structure elucidation. Results for lignin and coal structures are presented.

INTRODUCTION

Isomer enumeration in organic chemistry has been studied for more than 100 years. Many solutions have been published, but the problem is still being investigated. Our intention is not to outline a review of the literature which can be found in the publication of Balaban,[1] but rather to focus on the question: *why the diverse proposed techniques are not completely satisfactory?*

First, a mathematical formula was researched. If the problem has been resolved for specific cases, such as the alkane series[2] or the monocyclic compounds,[3] no general formulas were found. Since the 1960s, with the development of computer science, the problem ranged from a mathematical expression defining the number of isomers to a systematic method (an algorithm) which enumerated and designed the isomers. The first attempt was made with the DENDRAL project. It began from a study made by Lederberg et al.[4] for acyclic structures. The method became completely general with CONGEN[5] and more recently GENOA;[6] any types of organic structure can be treated by these computer programs. CONGEN and GENOA, which are able to enumerate the isomers of a molecular formula, are also able to generate structures when considering more restrictive constraints such as substructures (pieces of the molecular structure with known connectivity between their atoms). However, the technique employed is more a heuristic than a systematic algorithm, and a preknowledge of part of the results is necessary: GENOA uses a precompiled catalogue of 3000 elementary cyclic structures. Unfortunately, the proof of irredundancy and exhaustivity of the enumeration was never published, and differences were found between the results of these programs and other methods.[7] More systematic are the approaches of the COMBINE generator[8] and the CHEMICS program. CHEMICS is based on the concept of connectivity stack,[7] which allows an exhaustive and unique enumeration.[9] The starting point is a set of segments representing the unknown compound; a segment is a fragment containing one, two, or three atoms. To enumerate the isomers, an exhaustive permutation of all segments is processed. To avoid the redundancies, a isomorphism check is performed between the solutions. When this method is used, exhaustivity and irredundancy of the solutions can be easily proven; in fact all the permutations are considered, and the irredundancy of each solution is tested. However, the method is extremely time consuming because of the exhaustive generation of permutations. Even in the optimized version of CHEMICS[10] large molecules cannot be treated. Trying to resolve the imperfections of these techniques, new methods have been recently published.

Authors agree now that a structure generator must be exhaustive, irredundant, and effective. Generally, the exhaustivity implies the generation of redundant structures, and these structures must be removed. Considering that the isomorphism check is time consuming, the smaller the number of possible redundancies is, the more effective the method is. Bangov[11] noticed that most of the duplications during the construction process were due to permutations between equivalent elements (atoms or bonds). In particular, the bonding sites (or free bonds) of an atom are equivalent if they are in the same order (single bond, double bond, ...), and any permutation between these bonding sites generates identical structures. From this remark S. Bohanec and J. Zupan[12] have proposed an effective, exhaustive, and irredundant generator. The combinatorial process of this generator works with a matrix of bonding sites. According to the authors, the complexity of the method is independent on the size of the unknown structure and depends only on the number of bonding sites. Large molecules could be treated if all structures generated were irredundant. Unfortunately this is not the case, and the comparison between generated structures must be done considering all the atoms. Figure 1 shows an example in which two structures appear different regarding the bonding sites but are identical when the graphs are represented with all the atoms.

A similar approach has been developed by Bangov.[11] In this technique the atoms and the bonding sites of the unknown compound are partitioned into equivalent classes. The classes are more detailed than in the previous method; the main classes, called levels by Bangov, are the following ones:

h for bonding sites of heteroatoms
c for bonding sites of carbon atoms having valency greater or equal to 2
] for bonding sites which participate in a ring closure
1 for bonding sites of univalent fragment

Each of these classes are also split into sublevels, considering the number of hydrogen atoms attached, the hybridization state, and the charge density. The charge density of a given atom is calculated considering the α, β, γ, ... environments of this atom. For every level an exhaustive combination is performed, and redundant structures are eliminated. The process is applied until the High-Level, where final structures

0095-2338/92/1632-0338$03.00/0

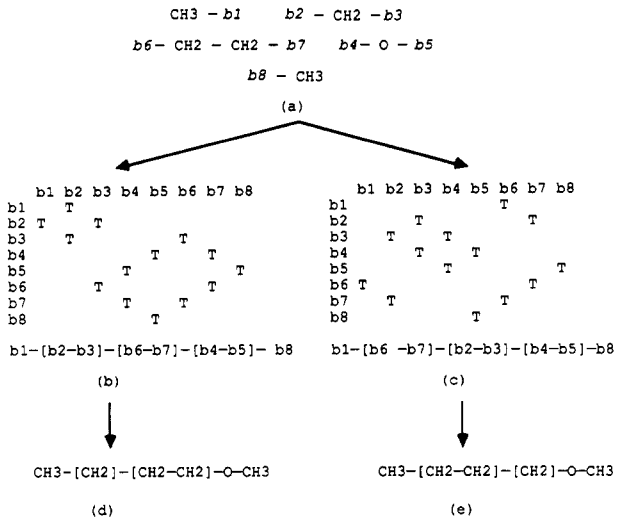

Figure 1. Five initial fragments and their respective bonding sites are represented in (a). The equivalences between the bonding sites are the following: b1 and b8, b2 and b3, b6 and b7, and b4 and b5. The figures (b) and (c) represent the adjancy matrices of two structures which can be built from the previous fragments. The structure appears to be different considering the equivalences between the bonding sites. However, the real structures containing all the atoms (d) and (e) are identical.

are generated. With this method, the number of redundant structures is considerably reduced. For acyclic compounds, all the structures generated at the High-Level are different. For cyclic structures, some of them can be identical, and the redundancies are removed. In fact, the method proposed by Bangov is in detail fairly complex, and proof that the structures generated are exhaustive and irredundant seems to be a difficult task. We will see that by using the same basic idea (the equivalent classes) defined in a formal language (graph theory) it is possible to prove the exhaustivity and the irredundancy. We will see also that in most of the cases for both acyclic and cyclic structures it is possible to avoid the final comparisons.

DEFINITIONS

In the following discussion, X represents a set of vertices, E a set of edges, and S a set of symbols. Regarding our problem, S can be the set of elements of the periodic table, or the set of elements and hybridizations, or any other convenient characterization of an atom. A molecule is represented by the graph $G = (X, E)$, where the elements of X are the atoms and the edges of E are the bonds. Each vertex is associated to an element of S, or in other words associates to an atom. Let (ϵ) be the function which associates a vertex to an element. Every element has a valence (υ) which represents the number of covalent bonds that can be formed with this element. An **atomic graph** is a graph representing a molecule which is not necessarily entirely built (i.e., part of the connectivity can be unknown). In formal language, an atomic graph $G = (X, E)$ is a nonoriented graph colored by the element of S, which verify the equation

$$\forall x \in X, \text{degree}(x) \leq \upsilon[\epsilon(x)]$$

A vertex is **saturated** if its degree is equal to the valence of the associated element. An atomic graph is saturated if all vertices are saturated. **Every molecule is a saturated atomic graph.** A saturated atomic graph is also called a molecular graph.[13]

To compute the isomers from a molecular formula, or information obtained from analytical results, is equivalent to computing all the saturated atomic graphs that can be built

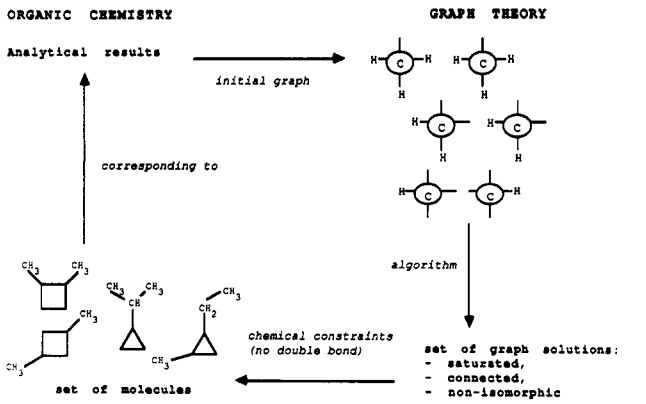

Figure 2. Isomer problem—chemistry and graph theory.

from an initial unsaturated graph (see Figure 2). In the case of a molecular formula, the connectivity between atoms is completely unknown, hence the initial graph is a set of isolated vertices. The initial graph can contain some edges when partial connectivity is known from analytical results. For example, ^{1}H NMR or ^{13}C NMR allow us to determine the partition of hydrogen. In this case the initial graph contains edges bonded to hydrogen vertices. In any case, the initial graph is unsaturated. Conversely, graphs which are solutions must be saturated, **connected**,[14] **nonisomorphic**, and they must verify the **chemical constraints**. According to Sussenguth,[15] two atomic graphs $G1 = (X, E1)$ and $G2 = (X, E2)$ are **isomorphic** if a permutation π of X can be found so that the following conditions are verified:

π is invariant for ϵ: $\epsilon o \pi = \epsilon$ (where ϵ is the function which associate to a vertex an element of S)

π matches the two sets of edges: $\pi(E1) = E2$

The **chemical constraints** are specific to each problem; however, the following constraints can be considered at least:

a vertex is not bonded to itself

the number of edges between two vertices is limited (3 in our examples, i.e., no more than triple bond)

each edge corresponds to a reasonable chemical bond (no bond O–O in our examples)

To verify if a graph is saturated, this simple method can be applied; a graph is saturated if each vertex is saturated. In the same way, the chemical constraints can be verified locally for each vertex. Both of these verifications depend linearly on the number of vertices. It is well known in graph theory that the connectivity of a graph can be checked with a linear algorithm.[16] Unfortunately, the verification of the nonisomorphism is a time-consuming process. In the worst case, the problem has an exponential [$O(n!)$] complexity, $n!$ permutations exist if X contains n vertices. In short, the major problem in terms of complexity and time consuming is the nonisomorphism verification. The purpose of this study is to reduce the number of comparisons between graphs and reduce the complexity of the isomorphism check in such a way that large molecules containing more than 100 atoms can be treated.

METHOD

The algorithm that is presented allows us to compute all the saturated atomic graphs, connected and nonisomorphic, which can be built from an unsaturated initial graph. This problem can be resolved by a simple method which determines all the ways to saturate a vertex. A similar technique is employed by the CHEMIC structure elucidation program.[9] This method treats each vertex separately and is applied recursively. The method is described in Figure 3 where the molecular formula of the unknown molecule is C_6H_{12}. The initial graphs contain

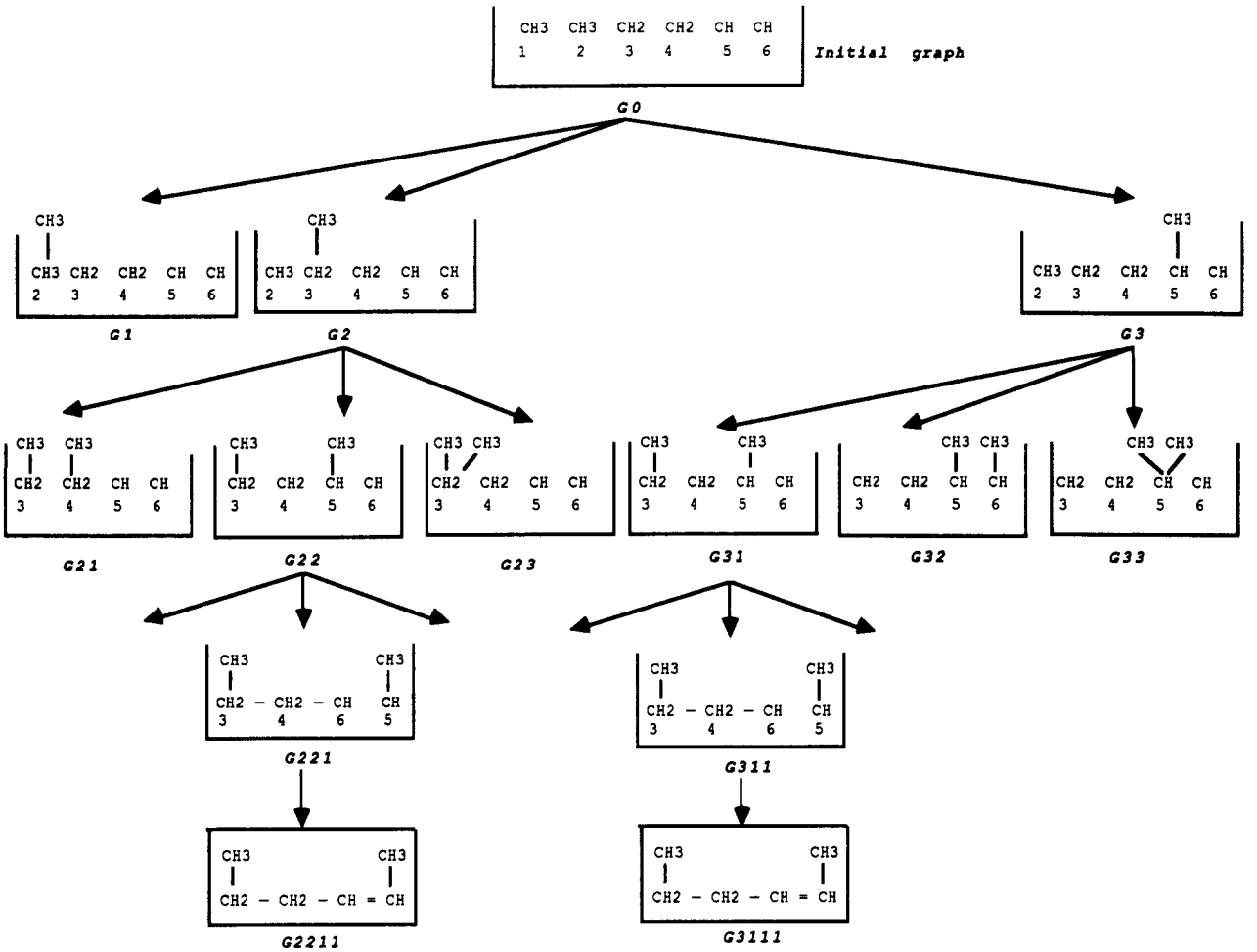

Figure 3. Vertex algorithm. This example is not detailed, and many other solutions are possible. Graphs surrounded by a rectangle are saturated.

bonds between carbon and hydrogen. This information can be determined by the combination of results from elemental analysis and ^{13}C NMR. From the first unsaturated vertex in the initial graph (number 1 in *G*0), a set of different graphs (*G*1–*G*3) is built. *G*1–*G*3 represent all the possible ways to connect the vertex 1 with the other unsaturated vertices (vertices 2–6). For each resulting graph (*G*1–*G*3), the same process is applied; a second unsaturated vertex (number 2) is chosen, and a second set of graphs is built. The graph *G*1 is rejected because the fragment CH_3–CH_3 is by itself a molecule. From *G*2, a second set of graphs is constructed: *G*21, *G*22, *G*23, and from *G*3 the set *G*31, *G*32, *G*33 is built. For each of these graphs, the process is applied again until all graphs are completely saturated. The example given in Figure 3 is detailed only for two solutions (*G*2211 and *G*3111), and we can notice that these solutions are identical. The algorithm derived from this method is outlined in Table I. The proof of exhaustivity of this method is evident because all possibilities are tested. Unfortunately, some of the saturated graph solutions can be isomorphic (Figure 3). In that case it is necessary to store every solution and eliminate the redundancies. In the worst case, the number of possible connections is exponential; therefore, the number or graphs which are stored and compared is also exponential. This method is a highly time- and space-consuming process; hence, large molecules cannot be treated.

Our algorithm tries to reduce the comparison between graphs in time and space. The principle illustrated in Figure 4 is based on a partition of the vertices into equivalent classes. According to Gutman,[13] two vertices are equivalent if a permutation exists between these two vertices which is invariant for the edges (i.e., an automorphism). In a formal language, the vertices $x1$ and $x2$ are equivalent in the atomic graph *G*

Table I. Vertex Algorithm[a]

```
vertex_algorithm (g,GS)

/* GS is the set of solutions, g is a graph */

if g is saturated, connected and the chemical constraints are respected
then
  GS = GS U* { g }  /* the graph g is added in the set
                       GS, if and only if g is non-isomorphic with the
                       graphs of GS */
else
  let x be an unsaturated vertex
  let G be the set of all different graphs obtained by saturation of x
  for every graph g of G do
    GS = vertex_algorithm (g,GS)
  done
end if
return(GS)
end
```

[a] The notations used in this algorithm and the others presented in the paper are written in algorithmic code closed from the C or Pascal languages. The operator U adds a graph in a set of graphs; the operator U* realizes the same operation with an isomorphism check.

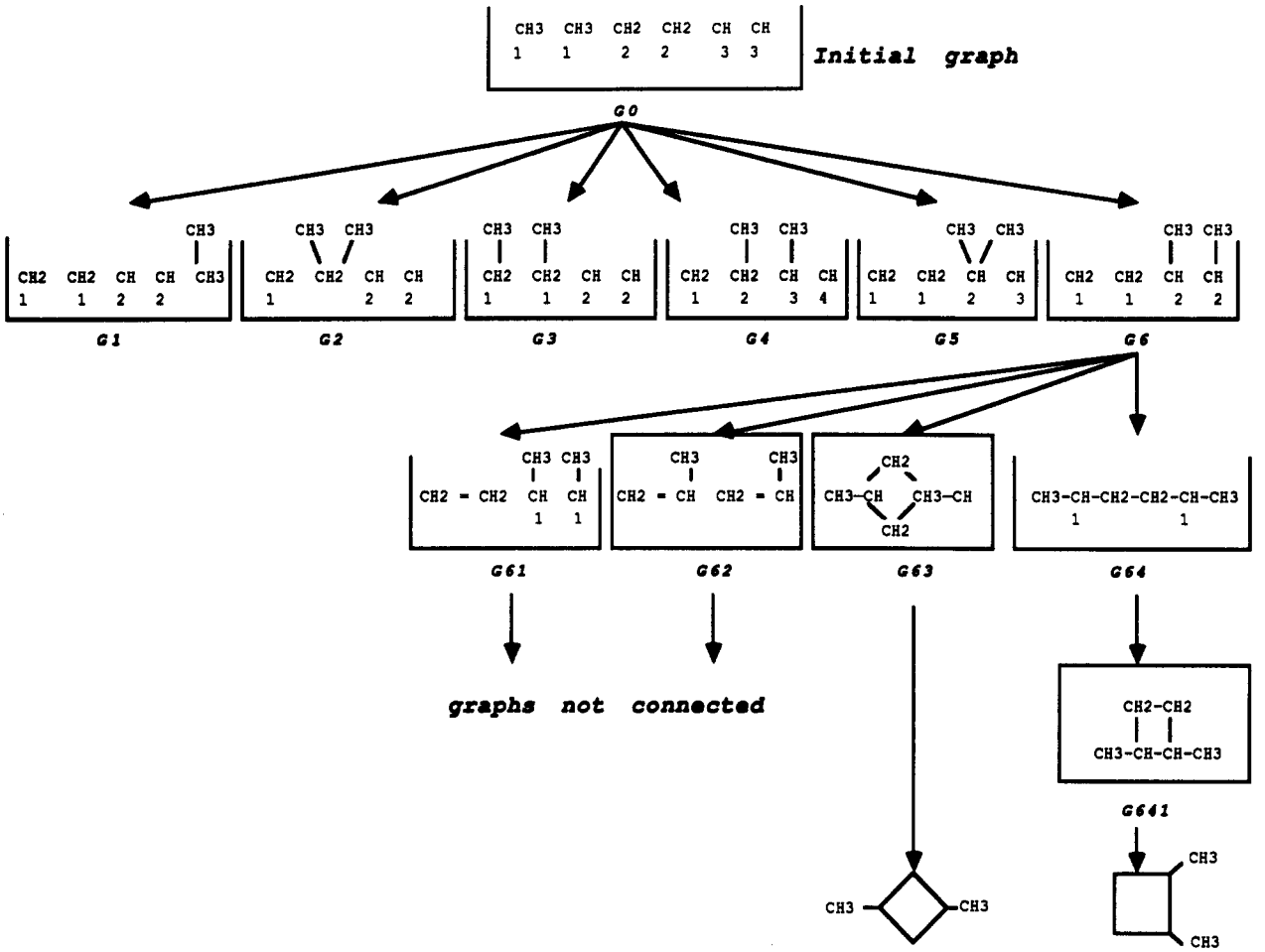

Figure 4. Class algorithm. This example is detailed only for the graph *G6*. Graphs surrounded by a rectangle are saturated.

= (X, E) colored by ϵ, if an automorphism π can be found so that

$$\epsilon o \pi = \epsilon,\ \pi(E) = E, \text{ and } \pi(x1) = x2$$

Identically to the previous method, our algorithm creates at each step a set of nonisomorphic graphs obtained after a saturation of vertices. Conversely to the previous method, our algorithm does not treat separately each vertex but treats all the vertices of one class at the same time. For a given equivalent class of the initial graph, the algorithm computes all the nonisomorphic graphs which saturate all the vertices of the class. The resulting graphs are not necessarily saturated, but the classes are now different because some edges have been added. For each resulting graph, the algorithm determines the new partition of classes. The process is recursively applied until all graphs are saturated. The algorithm is described in Table II.

The example outlined in Figure 4 is the same as the example in Figure 3. The initial graph is unsaturated, and the set of unsaturated vertices can be divided into three classes (1, 2, 3). The first step computes all the nonisomorphic graphs which saturate all the vertices of class 1. We obtain six different graphs (*G*1–*G*6). All these graphs are still unsaturated, and a new partition of classes is computed for each of them. For example, for the graph *G*6 we obtain two classes (1 and 2) of unsaturated vertices. The saturation of the vertices of class 1 in the graph *G*6 gives four different graphs (*G*61–*G*64). The two first graphs contain disconnected saturated subgraphs; therefore, they cannot be solutions because the final molecules must be connected. Thus, they are rejected. The third graph is saturated and connected; this graph is a solution. Two unsaturated vertices remain in the fourth graph; these vertices are in the same class (class 1). The saturation of the two vertices gives one graph solution *G*641. The same process detailed for *G*6 can be applied for the other graphs

Table II. Class Algorithm[a]

```
classes_algorithm (g,GS)

/* GS is the set of solutions, g is a graph */

if g is connected, saturated and the chemical constraints are
respected then
  GS = GS ∪ { g } /* g is solution, no isomorphism
                     check is necessary */
else
  compute the equivalent classes of g
  let x be an unsaturated vertex and C the class of x
  let G be the set of all different graphs obtained by saturation
  of all elements of C
  for every graph g of G do
    GS = classes_algorithm (g,GS)
  done
end if
return(GS)
end
```

[a] For notations, see footnote *a* in Table I.

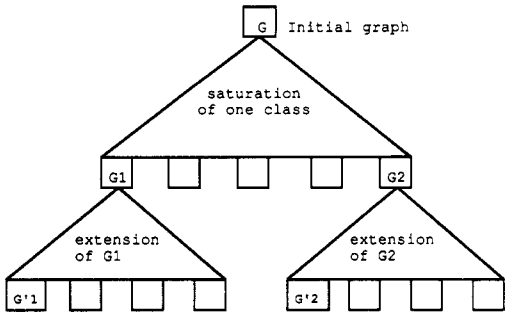

Figure 5. $G1$ and $G2$ are obtained by the saturation of all vertices of a given class. $G'1$ and $G'2$ are extensions of $G1$ and $G2$. We have to prove that if $G1 \neq G2$ then $G'1 \neq G'2$.

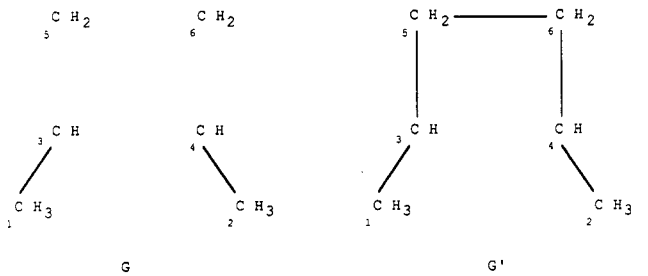

Figure 6. Extension. In this example G' is an extension of G which saturates the subset of vertices $Y = \{5,6\}$. We can remark that in G' Y is saturated; therefore, it is not possible to extend G' with edges having an extremity in Y.

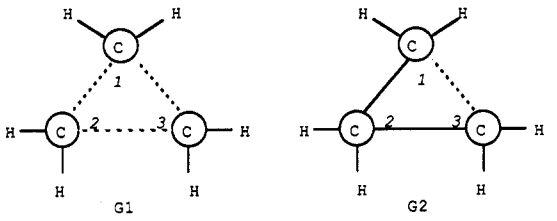

Figure 7. Ideal graph. $G1$ is an ideal graph: the missing edges (dashed lines) cannot be equivalent to the initial edges (solid line). $G2$ is not ideal because the missing edge [1,3] is equivalent to the initial edge [1,2].

$G1$–$G5$, and there are four solutions if the double bonds are forbidden (those drawn in Figure 2).

Is it possible to avoid the final comparisons of the solutions with the class algorithm? Yes, if for any two graphs, $G1$ and $G2$ obtained after the saturation of all the vertices of a class, no graph built from $G1$ is isomorphic with the graphs built from $G2$ (Figure 5). The purpose of the next section is to prove this affirmation.

IRREDUNDANCY

Some definitions are necessary before going further into the proof of the irredundancy. Let $G = (X, E)$ and $G' = (X, E')$ be two atomic graphs. Let Y be a subset of X. G' is an **extension** of G which can saturate Y if

E is included in E'

Y is saturated in E'

every edge of E'–E has at least one vertex in Y

Figure 6 gives an example of such saturation. To prove the method, we have to prove that if $G1$ and $G2$ are two extensions which saturate the same class in G, then all extensions of $G1$ are nonisomorphic with the extensions of $G2$. The next lemma and theorems prove this statement if the initial graph is ideal.[17] A graph G is ideal if in any extension of G, the edges which are created are different (nonisomorphic) than the initial edges of G (Figure 7). Here is a formal definition:

Ideal Atomic Graph. Let $G = (X, E)$ be an atomic graph. G is **ideal** if for every isomorphism π between two extensions of G, G is invariant: $\pi(E) = E$.

Lemma. Let π be an isomorphism between two extensions of $G = (X, E)$ ideal atomic graph. For every vertex x of X, x and $\pi(x)$ are equivalent in G.

Proof.

Trivial because G is ideal: $\pi(E) = E$; therefore, by definition x and $\pi(x)$ are equivalent in G.

Theorem 1. Let $G = (X, E)$ be an ideal atomic graph and x a vertex of X. Let $G1 = (X, E1)$ and $G2 = (X, E2)$ be two nonisomorphic extensions of G which saturate the class of x. No extension of $G1$ can be found to be isomorphic to the extensions of $G2$.

Proof.

Let $C_G(x)$ be the class of x in the graph G. Let $G'1$ and $G'2$ be two extensions of $G1$ and $G2$, and π is an isomorphism between $G'1$ and $G'2$. We have to prove that if $G'1$ and $G'2$ are isomorphic by π, then $G1$ and $G2$ are isomorphic by the same isomorphism π.

G is ideal and therefore $\pi(E) = E$.

Let $e = [x',x'']$ be an edge of $E1$–E.

$G1$ saturates $C_G(x)$ (the class of x): x' or x'' are in this class. Arbitrarily we will take x'.

From the previous lemma, x' and $\pi(x')$ are equivalent in G, so $\pi(x')$ is in the class of x: $C_G(x)$.

$\pi(e)$ has an extremity in $C_G(x)$, and G_2 also saturates $C_G(x)$; therefore, $\pi(e)$ belongs to E_2 and, consequently, $\pi(E1–E)$ is included in $E2$.

In the same way, it can be demonstrated that every edge of $E2$–E is an image of an edge of $E1$ by π^{-1}.

Therefore, π is also an isomorphism between $G1$ and $G2$.

Theorem 2. Let $G' = (X, E')$ be an extension of $G = (X, E)$ ideal atomic graph. If G' saturates the class of x where x is an unsaturated vertex of X, then G' is an ideal atomic graph.

Proof.

Let $C_G(x)$ be the class of x in the graph G. Let $G'1$ and $G'2$ be two extensions of G' and π be an isomorphism from $G'1$ to $G'2$.

G is ideal; therefore, $\pi(E) = E$.

Let $e = [x', x'']$ be an edge of E'–E.

G' saturates $C_G(x)$ (the class of x): x' or x'' are in this class. Arbitrarily we will take x'.

From the previous lemma, x' and $\pi(x')$ are equivalent in G, so $\pi(x')$ is in the class of x: $C_G(x)$.

$\pi(e)$ has an extremity in $C_G(x)$ so $\pi(e)$ belongs to E' and consequently $\pi(E'–E)$ is included in E'.

Therefore $\pi(E') = E'$.

Theorem 1 proves that from an ideal graph, and two nonisomorphic extensions $G1$ and $G2$, no extensions of $G1$ are isomorphic with the extensions of $G2$. Theorem 2 proves that for an ideal graph all the graphs obtained after saturation of one class are ideal. Therefore, if the initial graph is ideal, at each level the graphs produced by the algorithm are ideal. Together, theorems 1 and 2 prove the nonredundancy of the class algorithm when the initial graph is ideal. If the initial graph comes from a molecular formula, it does not contain any edge: it is an ideal graph [$\pi(\emptyset) = \emptyset$]. If the initial graph contains all the edges related to hydrogen atoms, this information can be taken from ^1H or ^{13}C NMR results. The graph is ideal because all the missing edges are connected to every atom except hydrogen. In that case a missing edge cannot be the image by an isomorphism of an inidial edge. In

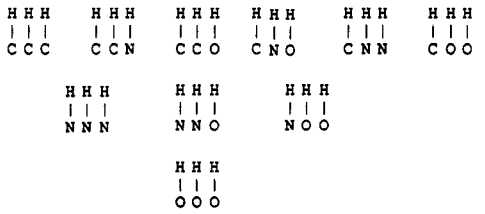

Figure 8. Initial classes are C, N, O, and H. The class to be connected is H. The figure represents the possible configurations when three hydrogens are connected.

the other cases, the ideal character has to be proven. The algorithm has been applied to geochemical organic macromolecules. From the analytical results available on these macromolecules, it is possible to define an ideal initial graph (see Results section).

EXHAUSTIVITY

This section explains how to compute all graphs which saturate an unsaturated class. In the following discussion we will consider that the graph is composed of N unsaturated classes and that the class which has to be saturated contains k elements. These elements can be connected with all other unsaturated vertices including themselves.

In first approximation, let us consider the simple case where the unsaturated vertices are not already connected together and the number of bonding sites per vertex is equal to one. In this case, the problem is equivalent to the following one: *how many figures can be formed by arranging k balls in N boxes*? The answer is

$$\sum_{q=1}^{k}\binom{q-1}{k-1}\binom{q}{N}$$

where

$$\binom{q}{N} = N!/p!\,(N-p)!$$

This number is an upper limit because we assume in the formula that each box can contain the k balls. In a real situation, the number of vertices in an unsaturated class (i.e., the number of balls per box) is limited. In an algorithmic point of view, this problem can be resolved by the computation of all unordered strings of length k that can be formed using N different letters. Let S = c1c2...ck be a string of length k; this string is unordered and can also be represented by c2c1...ck or ck...c2c1, etc. To avoid redundancies, a simple method can be employed. In every string the characters are ordered in alphabetic order, and the strings are generated in a lexicographic order.[18] For example, the different strings of three characters that can be formed with {C,N,O} are CCC, CCN, CCO, CNN, CNO, COO, NNN, NNO, NOO, and OOO. These strings represent also the 10 different ways to connect three hydrogens with carbon nitrogen and oxygen (Figure 8).

It is clear that a vertex can have more than one bonding site. Different figures can be obtained when two elements are connected on the same vertex or on different vertices, even if these vertices belong to the same class (Figure 9). Let D be the number of maximum connections that can be formed with a vertex, the maximum number of figures for N classes and k vertices is

$$\sum_{q=1}^{k}\binom{q-1}{k-1}\binom{q}{ND}$$

For example, for two classes {C,N} and two elements to be connected, if for each vertex the maximum number of

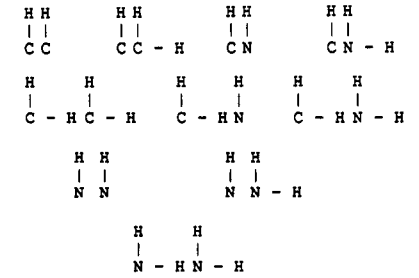

Figure 9. Initial classes are C, N, and H. The class to be connected is H. C and N can accept one or two H. The figure represents the possible configurations when two or more hydrogens are connected.

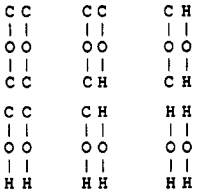

Figure 10. Initial classes are O, C, and H. The class to be connected is O. The class O contains two elements, and there are two bonding sites per element.

connections is two (D = 2), the different figures are C1C1, C1C2, C1N1, C1N2, C2C2, C2N1, C2N2, N1N1, N1N2, and N2N2. Numbers following letters represent the number of elements connected to the vertex. These strings can also be considered as the number of possibilities to connect two hydrogens with carbon and nitrogen when C and N can receive at most two hydrogens (Figure 9).

Let us consider now the case where the elements of the unsaturated class have themselves more than one bonding site. For a given element all its bonding sites (1, ...i, i+1, ...) are equivalent. To avoid redundancies, the vertex bonded to the bonding site i+1 must be greater or equal in alphabetic order than the vertex bonded to the bonding site i. For example, the class to be saturated {O} contains two elements and each element two bonding sites, if these elements are connected with the vertices of classes {C,H}, the different figures are CC CC, CC CH, CC HH, CH CH, CH HH, HH HH (Figure 10). These figures also represent the six different ways to connect an oxygen with C and H.

The previous formulas cannot be employed when the vertices of a given class are already linked together or linked with the element which belongs to the unsaturated class. However, this situation can occur and an element connected to two vertices which belong to the same class can generate two nonisomorphic graphs (Figure 11). In that particular case, all the connections between the unsaturated element and the vertices of the class must be considered. Unfortunately, some solutions generated can be identical (cf. caption of Figure 11), and a check of the nonisomorphism between the solutions must be done.

In short, for an unsaturated element and a given class, if the condition of nonisomorphism (CNI) is respected (i.e., the vertices of the class are not already linked together or linked with the element), only one vertex of this class is considered. The application of the previous formulas enables a set of solutions which are exhaustive and irredundant. When the CNI cannot be verified, all the vertices of the given class are considered and the set of solution is therefore necessarily exhaustive. However, in that case the irredundancy must be checked by applying an isomorphism check algorithm. The algorithm to connect the elements of an unsaturated class is described in Table III.

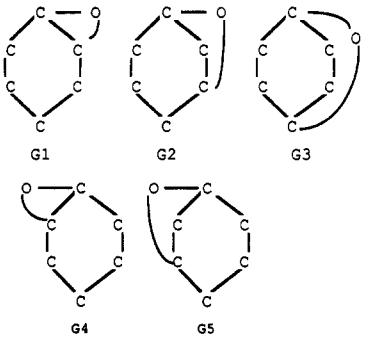

Figure 11. Initial classes are C and O. Before the connection of the unsaturated class O, all the vertices C belong to the same class. The condition of nonisomorphism (CNI) is not verified; therefore, all the vertices of class C are considered. Effectively, three nonisomorphic graphs $G1$, $G2$, and $G3$ can be found; unfortunately, by the same way two isomorphic graphs $G4 = G1$, $G5 = G2$ are generated.

The number of isomorphism checks depend on the CNI. As seen in the results, in all the tested cases this number is less than the number of atoms. However, the comparison between two graphs, even if it does not occur frequently, is time consuming. The object of the next paragraph is to show how to reduce the complexity of the isomorphism-check algorithm by using the equivalent classes.

ISOMORPHISM AND EQUIVALENT CLASSES

The isomorphism-check algorithm is based on the equivalent classes. The equivalent classes are computed each time the function **class_algorithm** is employed (Table II). The computation of the equivalent classes is a problem that has been studied in the past[19] and recently optimized.[20] Most of the solutions use the eigenvalues of the adjacency matrix. However, for our problem we propose an optimization based on the following theorem:

Theorem 3. Let $G = (X, E)$ be an ideal atomic graph and x,y two vertices of G. Let $G' = (X, E')$ be an extension of G and π' an automorphism of G'. An automorphism π of G can be found in such a way that if $\pi'(x) = y$, then $\pi(x) = y$.

Proof.

π' is an automorphism of G'; therefore, $\pi'(E') = E'$.

G' is an extension of G ideal atomic graph; therefore, $\pi'(E) = E$.

$\pi'(x) = y$ and $\pi'(E) = E$; therefore, $\pi = \pi'$ is also an automorphism of G.

In other words, the previous theorem proves that if two vertices belong to the same class in G' extension of G ideal graph, these vertices must be in the same class in G. The method to compute the equivalent classes is based on a function (**identical_class**) which compares the classes of two vertices. This function processes a matching of the two graphs beginning with the two tested vertices. By using theorem 3 to compute the equivalent classes of G', the function has to be used only between vertices which are already in the same class in G. One can notice that the number of vertices which belong to the same class reduces as the class algorithm is processed. Therefore, the complexity of the equivalent classes computation reduces as far as the class algorithm is advanced. Nevertheless, without considering the initial computation, the computation of the equivalent classes is an operation which costs less than 1 s cpu time[21] for all the results given in the next paragraph, in other words, for graphs containing 10 vertices up to 300 vertices. The isomorphism check required by the class algorithm can be resolved by using the equivalent classes. Effectively, if two graphs contain the same classes, these graphs are isomorphic. The next theorem proves this affirmation.

Theorem 4. Let $G = (X, E)$ be an ideal atomic graph. Let $G1 = (X, E1)$ and $G2 = (X, E2)$ be two extensions of G and

Table III. Saturation of a Class[a]

```
saturate_class   (c,  g,  GC)

/* c is the class, g is the graph , GC is the set of solutions */

if c is empty then
|  if CNI(g) then GC = GC U { g } /* no isomorphism check */
|  else          GC = GC U* { g }
else
|  let x be an element (unsaturated vertex) of c
|  c = c - (x)
|  let GV be the set of graphs obtained by
|  saturate_element(x,g,GV)
|  for every graph g of GV do
|  |  GC = saturate_class (c,g,GC)
|  done
end if
return(GC)
end

saturate_element  (x,g,GV)

/* x is an element of the current class (an unsaturated vertex), g is the
graph, GV is the set of solutions */

if x is saturated then
|  if CNI(g) then GV = GV U { g } /* no isomorphism check */
|  else          GV = GV U* { g }
else
|  let c0 be the class of the last vertex bonded to x
|  let d0 be the number of elements connected to the last vertex bonded
|  to x
|  for every pair (c,d) greater than (c0,d0) do
|  |  let Y be the set of vertices of class c connected to d elements
|  |  for all vertices y in Y do
|  |  |  E(g) = E(g) U [x,y]
|  |  |  GV = saturate_element (x,g,GV)
|  |  |  E(g) = E(g) - [x,y]
|  |  |  if CNI(g) then break the loop
|  |  done
|  done
end if
return(GV)
end
```

[a] For notations, see footnote *a* in Table I.

Table IV. Results for the Alkane Series

mol formula	no. of isomers	cpu time, s[a]	mol formula	no. of isomers	cpu time, s[a]
C_5H_{12}	3	10	$C_{10}H_{22}$	75	94
C_6H_{14}	5	12	$C_{11}H_{24}$	159	238
C_7H_{16}	9	15	$C_{12}H_{26}$	355	608
C_8H_{18}	18	20	$C_{14}H_{30}$	1 858	3 600
C_9H_{20}	35	41	$C_{16}H_{34}$	10 359	42 180

[a] Measured on a SUN/SPARC IPC workstation.

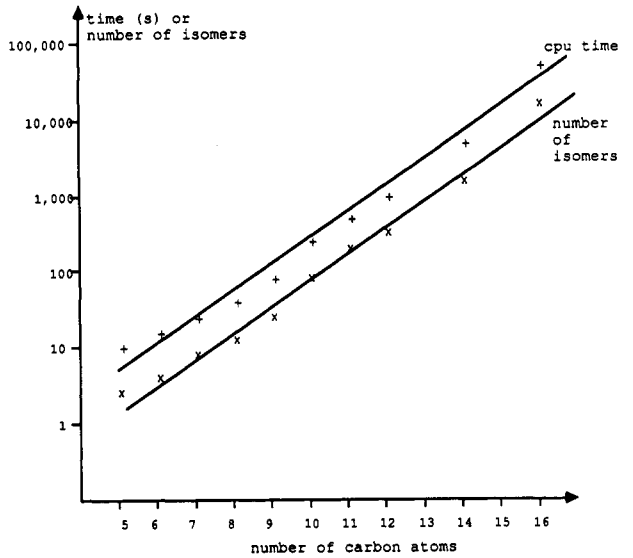

Figure 12. Results for alkanes series.

π an isomorphism between $G1$ and $G2$. If $x1$ and $x2$ are two vertices so that $\pi(x1) = x2$ then $x1$ and $x2$ belong to the same class in G.

Proof.

π is an isomorphism between $G1$ and $G2$: $\epsilon \circ \pi = \epsilon$ and $\pi(E1) = E2$.

G is an ideal atomic graph: $\pi(E) = E$.

$\pi(x1) = x2$ and $\pi(E) = E$; therefore, $x1$ and $x2$ belong to the same class in G.

From a practical point of view, the algorithm which checks the isomorphism, considers a vertex $x1$ in $G1$ and uses the function **identical_class** for every vertex of $G2$ which belongs to the class of $x1$ in G. If no vertex in $G2$ is found to be identical to $x1$, the two graphs are nonisomorphic according to theorem 4. The number of classes grows while the isomers enumeration is going on; therefore, the number of elements per class reduces in the same time, and consequently the isomorphism check is must faster at the end than at the beginning. Like the computation of the equivalent classes, the cost of the isomorphism check algorithm is minor, and the cpu time is less than 1 s[21] for all the examples given in the next section.

RESULTS

Four examples are provided in this section. The first is the classical example of the isomers of the alkanes series. For alkanes, the enumeration of the isomer is more simple because these structures are acyclic. Systematic methods have been published since the first attempt proposed by Caley.[2] The problem is still studied in terms of optimization.[22] The performance of our algorithms are optimal with the alkanes series. The initial graph does not contain any edge; therefore, the graph is ideal. The alkanes do not contain cycles, and the CNI are always verified; no isomorphism check is necessary. Table IV and Figure 12 show that for the alkanes series, the computation time progresses in the same way as the number of isomers.

The second example, already studied by Bangov,[11] can be considered as typical in structure elucidation. This example involves results of elemental analysis, NMR, and functional group analysis. The molecular formula is $C_{10}H_{20}O$. The solutions must include one OH group and one C=C group. The quantitation of C–H bonds are given by NMR and are the same as those of the 4-decen-1-ol, taken from Breitmaier and Voelter.[23] Considering this information, the initial graph

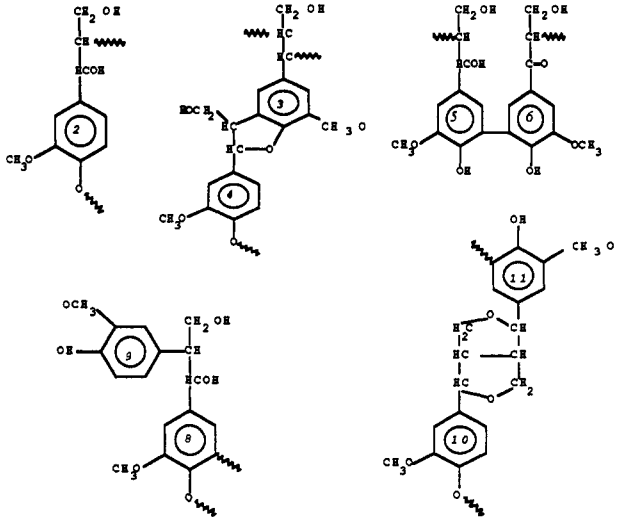

Figure 13. Figure represents the set of different fragments used by Adler to build the lignin model. The broken lines are the bonding sites. The aromatic ring numbers are those given by Adler.[25] In the original model published there are 16 monomers, but one is not completely defined and some monomers are identical. There are only 12 monomers if the boundaries of the structure are ignored (simplified model used by Hatcher[24]). According to the results of Adler, a model of 12 monomers must contain four fragments no. 2, one fragment no. 3–4, one fragment no. 5–6, one fragment no. 8–9, and one fragment no. 10–11.

is composed of the following fragments: one CH=CH fragment, one CH_2–OH fragment, six CH_2 fragments, and one CH_3 fragment. This graph is ideal because the missing edges are simple bonds between carbon atoms and the initial graph does not contain this type of bond. In all the solutions one cycle is presented (the C=C bond), and the CNI are not always verified. Our algorithm found in 50 s cpu time,[21] the following seven solutions: OH—CH_2—CH=CH—$(CH_2)_6$—CH_3, OH—$(CH_2)_2$—CH=CH—$(CH_2)_5$—CH_3, OH—$(CH_2)_3$—CH=CH—$(CH_2)_4$—CH_3, OH—$(CH_2)_4$—CH=CH—$(CH_2)_3$—CH_3, OH—$(CH_2)_5$—CH=CH—$(CH_2)_2$—CH_3, OH—$(CH_2)_6$—CH=CH—CH_2—CH_3, and OH—$(CH_2)_7$—CH=CH—CH_3.

Thirteen graph comparisons were necessary to find these molecules. Bangov found only six solutions using 57 combinations, but he probably used different constraints. With our algorithm seven graphs are solutions and 13 intermediate graphs are redundant; therefore, the number of combinations is 20.

The third example shows the performance of our algorithm for a large molecule, the lignin. Lignin is a biopolymer and is a main component of wood. The structure is intensively studied in biotechnology. Information about lignin is also important in geochemistry and coal science because the lignin structure remains during the sediment burial and skeletons of lignin have been found in coalified woods.[24] Much information is available on lignin, and one of the most advanced attempts of structure elucidation was realized by Adler.[25] Adler found in the softwood lignin the different monomers and the major possible linkages between these monomers (Figures 13 and 14). From these results Adler predicted and published a model containing 115 carbon atoms, 126 hydrogens, and 40 oxygens. However, many identical but different models can be formed from the same results, and our algorithm can determine the exact number of possibilities. For the lignin structure, the initial graph is composed of the fragments given in Figure 13. The missing edges are not all possible connections but only those given by Adler (Figure 14). The bonds D, E, F, G, and J are already present in a correct amount in the initial

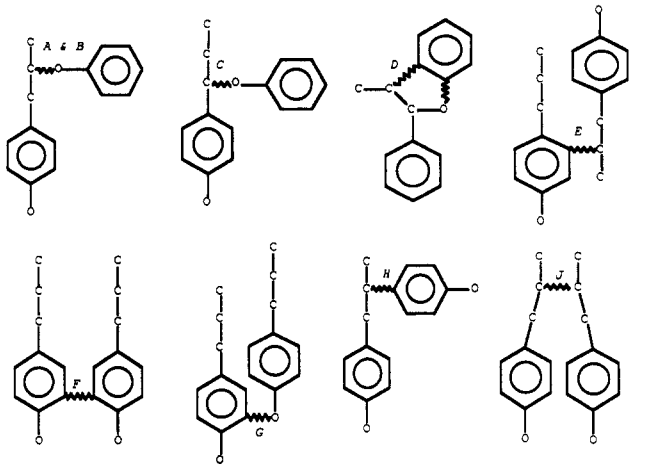

Figure 14. This figure represents the set of different interfragment bonds (the broken lines) defined by Adler. The letters (A, B, ..., J) are those given by Adler.[1] The proportions of each interfragment bonds have been measured by Adler using spectroscopic and chemical techniques. The results are the following: 50% of bonds A and B, 6–8% of bonds C, 9–12% of bonds D, 2.5–3% of bonds E, 9.5–11% of bonds F, 3.5–4% of bonds G, 7% of bonds H, and 2% of bonds J. Some other bond types were defined by Adler, but they appear only in the form of traces and their proportions are not significant.

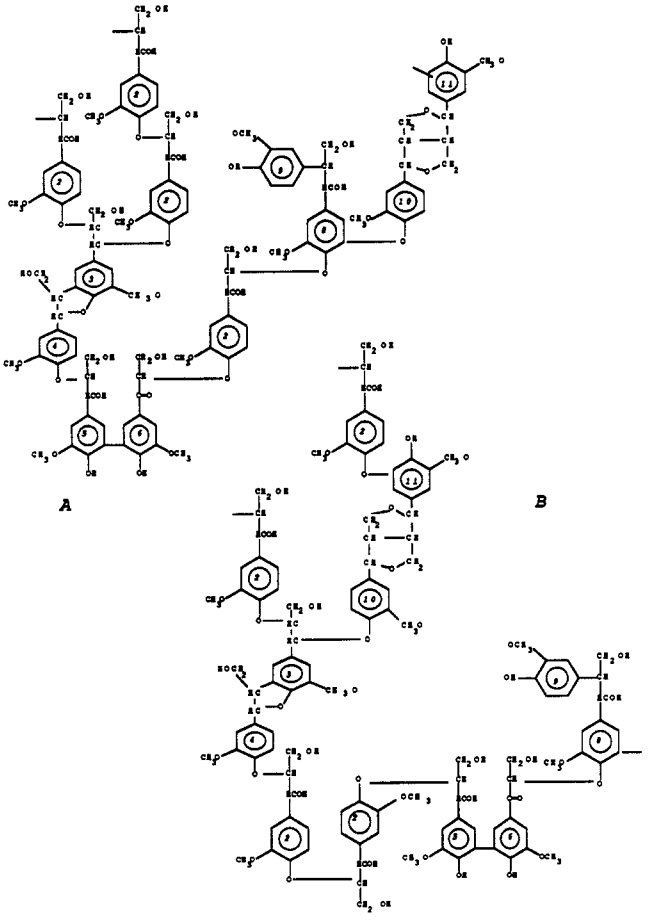

Figure 15. Two possible models for the softwood lignin structure. The structure A is the structure given by Adler.[22] The structure B is an other possibility randomly chosen among the 427 possible models.

fragment. Only the bonds A, C, and H correspond to the missing edges; these edges are different than the edges of the initial graph, which is therefore ideal. The program found 427 possible models and no graph comparisons where necessary. Two of these solutions are given in Figure 15. All the models found contain the fragments defined in Figure 13, and the interfragment bonds were added by respecting the

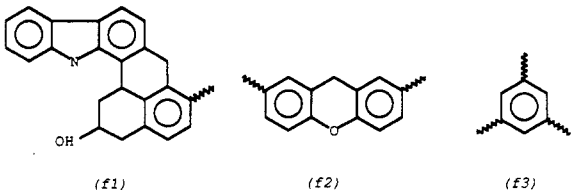

Figure 16. Some fragments of a highly volatile bituminous coal. The broken lines are the bonding sites. The final structures are composed of two fragments f1, two fragments f2, and two fragments f3.

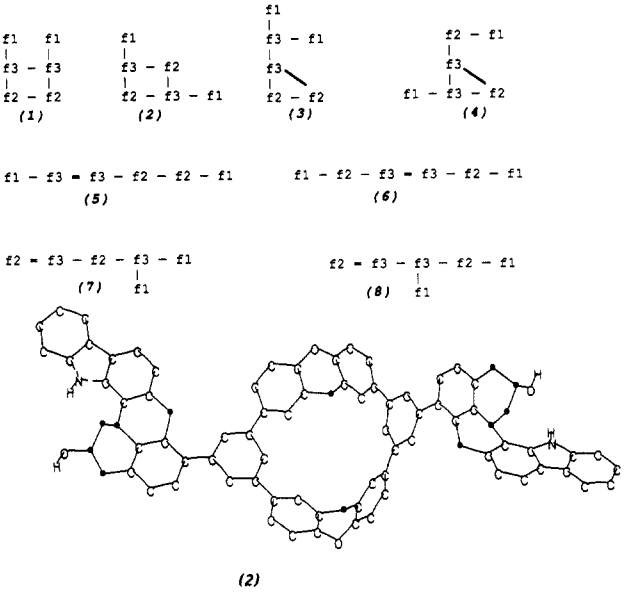

Figure 17. Fragments and their multiplicities are those defined in Figure 16. The number of possible structure is eight. After minimization, the molecular energy for structure **1** is 344.4 kcal/mol, and for structure **2**, 302.9 kcal/mol. For structures **3** and **4** the energy is greater than 350 kcal/mol, and for structures **5–8** the energy is greater than 500 kcal/mol. Structure **2** is drawn at the bottom of the figure in a 3D representation obtained after minimization of the energy. This picture has been drawn using the molecular modeling software PCMODEL (Serena software). The aliphatic carbon is represented by a dark point, and the aromatic carbon is represented by the letter C. The hydrogen atoms attached to carbon atoms are not displayed.

quantities listed in the caption of Figure 14. We have to point out that these 427 structures are only intended to define the possible models that can be built from the information given by Adler.

The final example is relative to coal structure. For this example three different fragments (in Figure 16) were chosen from the highly volatile bituminous coal model published by Shinn.[26] These fragments represent only a very small part of the whole structure, and there is no way to consider them as representative of coal. Our intention is not to define a coal model but only to examine the performance of our algorithm for this type of structure which is highly cyclic. According to Figure 16 and the associated caption, the unknown structures contain 84 carbon atoms, 58 hydrogen atoms, 4 oxygen atoms, and 2 nitrogen atoms. All the bonds are in six- or five-membered rings, and it is not possible to form these types of rings by connecting the initial fragments. Therefore, the missing bonds are not equivalent to the initial bonds and the initial graph is ideal. The construction was constrained by avoiding the possibility of bonding a fragment to itself. The program found eight different structures by using 90 graph-isomorphism checks (Figure 17). The energy of the resulting structures has been minimized by using the MMX program.[27] The minimization was employed to find a conformational correct form. In our example the structure number **2** gives

the lower energy (302.9 kcal/mol), this structure is drawn at the bottom of Figure 17.

The generator presented is part of a structure elucidation program. The program has been developed to elucidate macromolecules in organic geochemistry and fuel science. The analytical techniques which provide information on these types of macromolecules can be divided into two classes: atomic and molecular information. Briefly, the atomic information comes from analyses such as elemental analysis, NMR, and functional group analysis. These analyses are directly performed on the macromolecule, and the informations are percentages of atoms with specific environments. The second type of analyses is processed on degradation products of the macromolecule. The degradation is achieved by pyrolysis, chemical attack, or natural geological process during the sediment burial of the macromolecule. The components analyzed are not any more complex macromolecules but rather molecular fragments containing at most 100 atoms. The separation of the fragments is realized by liquid or gas chromatography, and the chemical structure of the fragments is determined by mass spectrometry. These molecular fragments form the initial graph, and the missing edges are bonds broken during the degradation. Generally these bonds, called interfragment bonds, are specific and can break more easily than those present in the molecular fragments. Consequently, in most of the cases, the initial graph is ideal. Sometimes, and this is the case with the lignin structure, the interfragment bonds are determined qualitatively and quantitatively by analytical results (Figure 14 and Adler[25]). When such information is not available, we have shown in a previous publication that it is possible to determine the quality and quantity of the interfragment bonds by doing a correlation between the atomic and molecular information.[28]

In short, in a real situation of structure elucidation, such as coal or lignin, a list of fragments and a list of interfragment bonds are first computed from the analytical results. In a second step, our generator determines all the solutions from an initial graph formed by the molecular fragments. The list of interfragment bonds is regarded during the generation as an additional chemical constraint. In a third step, the structures found by the generator are built in a 3D space by a method described previously.[17] Once built, the energy of the resulting structures can be minimized by using the molecular mechanics and dynamics programs.

CONCLUSION

A new generator to enumerate the isomers has been proposed. A first utilization can be made by using the program to generate the molecules which correspond to a molecular formula. In that case no final comparisons are necessary, and the solutions are irredundant and exhaustive.

More sophisticated is the utilization of the program from a set of information given by chemical and spectroscopic analyses. In that case the analytical results define an initial unsaturated graph. This graph contains some edges corresponding to the part of the connectivity which is known. The missing edges represent the unknown connectivity. The generator computed all the different saturated graphs which can be built from the initial graph. No final comparison is necessary if the initial graph is ideal, in other words, if the missing edges are not equivalent to the initial edges. Avoiding the final comparison is an important optimization. These comparisons are extremely time and space consuming: time consuming because the final solutions do not necessarily have the same equivalent classes, and the theorem 4 can not be employed; space consuming because each possible solution must be stored instead of displayed. Therefore, if the analytical results define an initial graph which is not ideal, rather than to compare the final solutions, it is better to modify this graph and transform it to an ideal graph. This transformation can be done by removing the initial edges that are equivalent to the missing edges. However, these edges eliminated in the initial graph must be presented in the final graphs; the solution is therefore to introduce these edges as additional chemical constraints. The detection of the possible equivalence between the initial and the missing edges can be done manually by the user of the program in simple cases. The automatic resolution of this problem by an algorithm is part of the further development of our generator.

Hopefully, in many cases, the analytical results define an ideal initial graph. This is the case for an initial graph coming from NMR results. With ^1H and/or ^{13}C NMR it is possible to determine specific connectivity such as carbon–hydrogen. Then, the initial edges correspond to bonds C–H, the missing edges are all other types of bonds and cannot be equivalent to the initial edges, therefore the initial graph is ideal. As seen in the previous section, in a real situation of structure elucidation it is possible to define an ideal initial graph, especially in geochemistry and fuel science where the studied macromolecules are degradated by diverse techniques. The degradation products form the initial graph, and the broken bonds are the missing edges. These broken bonds have a specific chemical character and generally are not present in the degradation products; hence, in most of the time, the initial graph is ideal. The compounds analyzed in organic geochemistry and fuel science are big molecules; the utilization of equivalent classes enables the elucidation of structures containing a large number of atoms: $C_{84}H_{58}N_2O_4$ for the chosen piece of coal and $C_{116}H_{126}O_6$ for the fragment of lignin structure.

ACKNOWLEDGMENT

We are pleased to acknowledge the funding provided by the U.S. Department of Energy, Sandia National Laboratories under Contract DE-AC04-76DP00789. We are grateful to Dr. P. G. Hatcher and Dr. J. M. Drappier for encouragements and constructive discussions.

REFERENCES AND NOTES

(1) Balaban, A. T. Application of Graph Theory in Chemistry. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 334–345.

(2) Cayley, A. On the Mathematical Theory of Isomers. *Philos. Mag.* **1874**, *41*, 444–446.

(3) Polya, G. Un problème combinatoire sur les groupes de permutations et le calcul du nombre des isomères des composés organiques. *C.R. Hebd. Seances Acad. Sci.* **1935**, *201*, 1167–1169.

(4) Lederberg, J.; Sutherland, G. L.; Buchanan, B. G.; Feigenbaum, E. A.; Robertson, A. V.; Duffield, A. M.; Djerassi, C. Applications of Artificial Intelligence for Chemical Inference. The number of Possible Organic Compounds. Acyclic Structures Containing C, H, O, and N. *J. Am. Chem. Soc.* **1969**, *91* (11), 2973–2976.

(5) Carhart, R. E.; Smith, D. H.; Brown, H.; Djerassi, C. Applications of Artificial Intelligence for Chemical Inference. XVII. An Approach to Computer-Assisted Elucidation of Molecular Structure. *J. Am. Chem. Soc.* **1975**, *97* (20), 5755–5762.

(6) Carhart, R. E.; Smith, D. H.; Gray, N.; Nourse, J. G.; Djerassi, C. GENOA: A Computer Program for Structure Elucidation Utilizing Overlapping and Alternative Substructures. *J. Org. Chem.* **1981**, *46*, 1708–1718.

(7) Kudo, Y.; Sasaki, S. The connectivity Stack, a New Format for Representation of Organic Chemical Structures. *J. Chem. Doc.* **1974**, *14*, 200–202.

(8) Lipkus, A. H.; Munk, M. E. Automated Classification of Candidate Structures for Computer-Assisted Structure Elucidation. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 9–18.

(9) Kudo, Y.; Sasaki, S. Principle for Exhaustive Enumeration of Unique Structure Consistent with Structural Information. *J. Chem. Inf. Comput. Sci.* **1976**, *16*, 43–49.
(10) Sasaki, S.; Kudo, Y. Structure Elucidation System Using Structural Information from Multisources: CHEMICS. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 252–257.
(11) 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.
(12) Bohanec, S.; Zupan, J. Structure Generation of Constitutional Isomers from Structural Fragments. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 531–540.
(13) Gutman, I.; Polansky, O. E. *Mathematical Concepts in Organic Chemistry*. Springer-Verlag: Berlin, 1986.
(14) G is a connected graph, if every pair of vertices in G is joined by a path.
(15) Sussenguth, E. H. A Graph-Theoric Algorithm for Matching Chemical Structures. *J. Chem. Doc.* **1969**, *5*, 36–43.
(16) Deo, N. *Graph Theory with Application to Engineering and Computer Science*. Prentice Hall: New York, 1974.
(17) Faulon, J. L. Prediction Elucidation et Modelisation Moleculaire: Algorithmes et Applications en Geochimie. Ph.D. Thesis, Ecole des Mines de Paris, Paris, 1991.
(18) $x1y1 \leq x2y2$ in lexicographic order if $x1 < x2$ or $x1 = x2$ and $y1 < y2$.
(19) 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.
(20) Rücker, G.; Rücker, C. On Using the Adjacency Matrix Power Method for Perception of Symmetry and for Isomorphism Testing of Highly Intricate Graphs. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 123–126.
(21) Measured on a SUN/SPARC workstation.
(22) Hendrickson, J. B.; Parks, C. A. Generation and Enumeration of Carbon Skeletons. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 101–107.
(23) Breitmaier, E.; Voelter, W. *^{13}C NMR Spectroscopy*; Erbel, H. F., Ed. Verlag Chemie, Weinheim and New York, 1978.
(24) Hatcher, P. G. Chemical structural models for coalified wood (vitrinite) in low rank coal. *Adv. Org. Geochem.* **1990**, *16*, 959.
(25) Adler, E. Lignin Chemistry—Past, Present, and Future. *Wood Sci. Technol.* **1977**, *11*, 169–218.
(26) Shinn, J. H. From coal to single-stage and two-stage products: a reactive model of coal structure. *Fuel* **1984**, *63*, 1187–1196.
(27) Burket, U.; Allinger, N. L. *Molecular Mechanics*. ACS Monograph 177, ACS: Washington DC, 1982.
(28) Faulon, J. L.; Vandenbroucke, M.; Drappier, J. M.; Behar, F.; Romero, M. 3D Chemical model for geological macromolecules. *Adv. Org. Geochem.* **1990**, *16*, 4–6.