A Graph-Theoretic Algorithm for Matching Chemical Structures*

EDWARD H. SUSSENGUTH, JR.**
The Computation Laboratory of Harvard University, Cambridge 38, Massachusetts
Received May 26, 1963

I. INTRODUCTION

A chemical information retrieval system consists primarily of a large file or library of chemical compounds which is to be searched in response to the request of a chemist. The chemist formulates his request as the structural diagram of a chemical. This diagram may be in either of two forms. In the first form the diagram is a complete compound, thereby indicating that the chemist wishes to locate that compound in the file which exactly matches the query structure. In the second form the diagram represents a piece or fragment of a compound, and the retrieval system is required to locate all compounds in the library which contain the query as an integral part; that is, the query item must exactly match part of the library item.

There are many chemical retrieval systems which process the first type of request efficiently. Most of these systems are also capable of handling certain fragment requests; however, the fragments which can be processed are frequently of a restricted nature. For example, in retrieval systems which are based on linear ciphers, only those fragments which are explicit in the cipher are readily detected. To allow a completely general specification of fragments it seems inevitable that a detailed atom-by-atom comparison is required of the query and library structures. A technique for making such detailed comparisons is presented in this report. This technique is novel in that it avoids the excessive backtracking and restarting required by other atom-by-atom matching procedures.

Before giving the details of the proposed algorithm, some definitions are reviewed and a brief example is presented to illustrate the over-all concepts. Then the flow diagram of the algorithm is explained in terms of additional examples. Finally, the mechanization of the algorithm for a digital computer is discussed.

This report is condensed version of the original,[1] which gives a generalization and comprehensive description of the algorithm, proofs of convergence and related topics, and applications other than chemical retrieval systems.

2. DEFINITIONS AND A PRELIMINARY EXAMPLE

The matching technique described in this report considers a chemical compound to be represented by a graph. A *graph* consists of a set of *nodes* and a set of *branches* connecting pairs of nodes. The set of nodes *connected to* node x is denoted as Γx. The operator Γ is also extended to sets, so that ΓA denotes the set of nodes connected to any node of set A. For example, in graph G of Figure 1, $\Gamma 5 = \{3,6,7\}$ and $\Gamma\{2,4\} = \{1,3\}$. Two graphs G and G* are *isomorphic* if there is a one-to-one correspondence between the nodes of G and G* which preserves a one-to-one correspondence between the branches of the graphs. In other words, for all nodes x in G there must be a unique correspondent x^* in G*, and the correspondence must be such that x is connected to y in G if and only if x^* is connected to y^* in G*. It is evident that in the graph of a chemical compound the nodes correspond to atoms and the branches to interatomic bonds, so that the nodes and branches are assigned values corresponding to the atom or bond type, respectively. For such graphs, which have values associated with their nodes and branches, the definition of isomorphism is extended in an obvious manner to ensure that the correspondents of node x and branch (x, y) have the same values as x and (x, y), respectively.

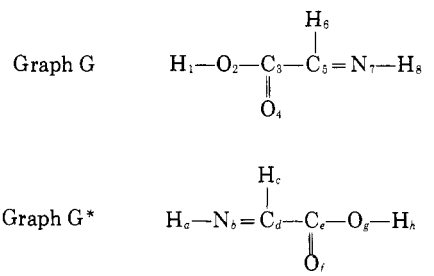


Figure 1.—The compounds for example I.

In terms of these definitions the first type request of the chemical retrieval system may be phrased as: locate that graph of the library which is isomorphic to the query graph. To rephrase the second type request in graphic terminology requires a new definition. Graph H is a *subgraph* of graph G if H can be obtained from G by removing some nodes of G and all branches adjacent to those nodes.[2] The second type request may now be restated: locate those graphs of the library that contain a subgraph which is isomorphic to the query graph.

A brief example is now presented to illustrate the over-all concepts of the algorithm. Two graphs, G and G*, representing chemical compounds, are shown in Figure 1. The problem is to determine whether or not the graphs are isomorphic, and, if they are, to delineate the explicit correspondence between nodes. It is first observed that

* Presented before the Division of Chemical Literature, 147th National Meeting of the American Chemical Society, Philadelphia, Pa., April 6, 1964. This work was supported in part by the National Science Foundation.

** International Business Machines Corp., Yorktown Heights, New York.

(1) E. H. Sussenguth, Jr., "Structure Matching in Information Processing," Doctoral Dissertation, Harvard University, 1964.

(2) Other classes of substructure are discussed in reference 1. The subgraph is the substructure of greatest interest in a chemical retrieval system.

those nodes of G* which represent carbon atoms must correspond to those nodes of G which represent carbon atoms. That is, node d must correspond to either node 3 or node 5 and no others, and node e must correspond to either 3 or 5; symbolically, $\{d,e\} = \{3,5\}$. Thus considering all node values and, in a similar manner, the bonding information, one constructs the correspondences shown on lines 1 to 6 of Figure 2. The sets on line 4 state that node f must correspond to either node 2 or 4; moreover, line 6 states that node f must correspond to either node 3, 4, 5, or 7. To satisfy both requirements f must correspond to 4. Line 4 then requires that node g correspond to 2. Using this information, lines 1 to 6 may be replaced by lines 7 to 11.

Generated by		Subset of G*	Subset of G	Line no.
Node value:	H	$\{a,c,h\}$	$\{1,6,8\}$	1
	C	$\{d,e\}$	$\{3,5\}$	2
	N	$\{b\}$	$\{7\}$	3
	O	$\{f,g\}$	$\{2,4\}$	4
Branch value:	single	$\{a,b,c,d,e,g,h\}$	$\{1,2,3,5,6,7,8\}$	5
	double	$\{b,d,e,f\}$	$\{3,4,5,7\}$	6
Partition:	lines 1–6	$\{a,c,h\}$	$\{1,6,8\}$	7
		$\{b\}$	$\{7\}$	8
		$\{d,e\}$	$\{3,5\}$	9
		$\{f\}$	$\{4\}$	10
		$\{g\}$	$\{2\}$	11
Connectivity:	line 8	$\{a,d\}$	$\{5,8\}$	12
	line 10	$\{e\}$	$\{3\}$	13
	line 11	$\{e,h\}$	$\{1,3\}$	14
Partition:	lines 7–14	$\{a\}$	$\{8\}$	15
		$\{b\}$	$\{7\}$	16
		$\{c\}$	$\{6\}$	17
		$\{d\}$	$\{5\}$	18
		$\{e\}$	$\{3\}$	19
		$\{f\}$	$\{4\}$	20
		$\{g\}$	$\{2\}$	21
		$\{h\}$	$\{1\}$	22

Figure 2.—Sets generated in example I.

Line 8 states that node b corresponds to node 7; it is now observed that those nodes connected to node b, namely nodes a and d, must correspond to those nodes connected to node 7, namely nodes 5 and 8. This correspondence and the additional correspondences obtained in a similar manner from lines 10 and 11 are shown on lines 12 to 14. Now considering lines 7 to 14 and arguing as above, the exact correspondence between the nodes of the graphs is determined, as shown on lines 15 to 22.

3. SET GENERATION TECHNIQUES

From example I and the flow diagram of Figure 3 it can be seen that the algorithm consists of two basic parts:

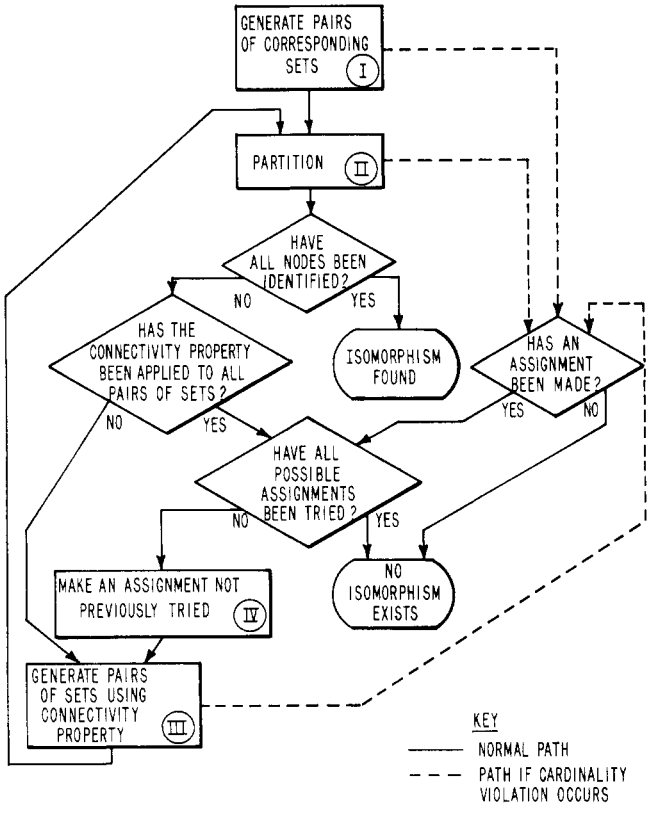


Figure 3.—Basic flow diagram of the graph matching algorithm.

(1) the generation of pairs of corresponding subsets of nodes (boxes I, III, and IV); and (2) the partitioning of these subsets to determine correspondences between individual nodes (box II). In example I, node values, branch values, and connectivity patterns of nodes were used to generate pairs of corresponding subsets. The general principle for set generation, of which these three properties are specializations, is:

> If graph G is isomorphic to graph G*, the subset of nodes of G which exhibit some property must correspond to that subset of nodes of G* which exhibit the same property.

The consequent of this principle is symbolically represented as:

$$\{x{:}x \text{ has property p}\} = \{x^*{:}x^* \text{ has property p}\}.$$

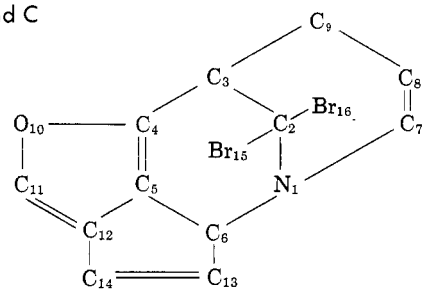
The specializations of the generating principle for the properties of node value, branch value, and connectivity are given in Figure 4. In addition to these three properties, the concepts of node degree and node order are valuable for generating sets. The *degree* of node x is the number of nodes in Γx. The *order* of node x is the least number of branches of a cycle (*e.g.*, a chemical ring) which must be traversed from x to return to itself; if x is not in a

Property	Graph	Subgraph
Node value	$\{x{:}\text{value}(x) = v\} = \{x^*{:}\text{value}(x^*) = v\}$	$\{x{:}\text{value}(x) = v\} \subseteq \{x^*{:}\text{value}(x^*) = v\}$
Branch value	$\{x{:}\text{value}((x,y)) = b\} = \{x^*{:}\text{value}((x^*,y^*)) = b\}$	$\{x{:}\text{value}((x,y)) = b\} \subseteq \{x^*{:}\text{value}((x^*,y^*)) = b\}$
Degree	$\{x{:}\text{degree}(x) = d\} = \{x^*\text{degree}(x^*) = d\}$	$\{x{:}\text{degree}(x) = d\} \subseteq \{x^*{:}\text{degree}(x^*) \geq d\}$
Order	$\{x{:}\text{order}(x) = d\} = \{x^*{:}\text{order}(x^*) = d\}$	$\{x{:}\text{order}(x) = d\} \subseteq \{x^*{:}\text{order}(x^*) \leq d\}$
Connectivity	$A = A^* \rightarrow \Gamma A = \Gamma^* A^*$	$A \subseteq A^* \rightarrow \Gamma A \subseteq \Gamma^* A^*$

Figure 4.—Summary of set generating principles.

cycle, its order is infinite. For example, in Figure 5 nodes 5 and 15 have degree 3 and 1 and order 5 and infinity, respectively.

Compound C



Fragment F

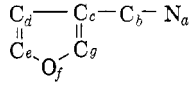


Figure 5.—The compound and fragment for example II.

A trivial, but very important, corollary of the general principle is that corresponding sets must have the same number of nodes. This, then, provides a simple test for determining that graphs are not isomorphic:

> If the number of nodes of graph G which exhibit some property is different from the number of nodes of graph G* which exhibit the same property, then G cannot be isomorphic to G*.

The principles, as described, apply when one is testing if graph G is isomorphic to graph G*. However, it is necessary to weaken the principles when the algorithm is used to test if graph G is a subgraph of graph G*, because there may be more nodes in graph G* which exhibit some particular property than there are nodes of G which exhibit the same property. For example, if node x corresponds to node x^*, the set Γx does not necessarily correspond exactly to the set $\Gamma^* x^*$ because x^* may be connected to more nodes in G* than x is in G. Therefore the principle is weakened to state only that the correspondents of the nodes of Γx must be contained in the set $\Gamma^* x^*$, realizing that some extraneous nodes may also be in $\Gamma^* x^*$; symbolically this is stated as $\Gamma x \subseteq \Gamma^* x^*$.

All the set generating properties require this modification, which allows for extraneous nodes in the subsets of G*. The degree and order properties, however, require additional weakening to provide for the fact that the correspondent of node x may have a greater degree and a lesser order than the degree and order of x itself. The weakened forms of the generating principle are given in the right-hand column of Figure 4.

Just as it could be concluded that graphs G and G* were not isomorphic if corresponding sets S and S^* did not have the same number of nodes, it may be concluded that G is not a subgraph of G* if corresponding sets S and S^* are generated and S has more nodes than S^*. Such an occurrence is called a *cardinality violation*.

As an illustration of the set generating procedures, consider example II of Figure 5. A fragment F and a compound C are shown; the test is to determine if F is contained in C, and if it is, to explicitly state the correspondences between atoms. In the first step of the algorithm, box I of the flow diagram (Figure 3), the node value, branch value, degree, and order properties are used to generate sets; the sets generated by these properties are shown in Figure 6 as lines 1 to 4, 5 to 6, 7 to 9, and 10 to 11, respectively.

4. THE PARTITIONING PROCEDURE

The second basic part of the algorithm, the partitioning procedure (box II of Figure 3), is required whenever a number of pairs of corresponding sets have been generated. The purpose of the partitioning is to reduce the number of possible correspondents of each node of G to as few nodes of G* as possible (ideally to a single node). This is done by examining the pairs of generated subsets. Let S_i and S_i^* be a pair of corresponding subsets, that is $S_i \subseteq S_i^*$. If node x is in S_i, then its correspondent x^* must be in S_i^*. Therefore, for all those i for which x is in S_i, x^* is in S_i^*; and, hence, x^* is in the intersection of those S_i^*. Symbolically

$$x^* \in \bigcap_{i \in I} S_i^* \tag{1}$$

where $i \in I$ if x is in S_i.

If there are pairs of sets S_j and S_j^* which have the same number of nodes, that is if $S_j = S_j^*$, then $\bar{S}_j \subseteq \bar{S}_j^*$ also holds. This is true because the set equality implies a one-to-one correspondence between the nodes of S_j and S_j^*, so that it is impossible for a node outside of S_j (that is, in $\bar{S}_j$) to correspond to a node of S_j^*. Therefore the correspondents nodes of $\bar{S}_j$ must be contained in $\bar{S}_j^*$, that is $\bar{S}_j \subseteq \bar{S}_j^*$. In other words if $S_j = S_j^*$ and if x is not in S_j, then x^* must be in $\bar{S}_j^*$. By using the sets which satisfy these requirements, relation 1 may be further restricted:

$$x^* \in \bigcap_{i \in I} S_i^* \cap \bigcap_{j \in J} \bar{S}_j^* \tag{2}$$

where $i \in I$ if x is in S_i and $j \in J$ if $S_j = S_j^*$ and x is not in S_j.

An example may help clarify the partitioning procedure. Consider the following pairs of corresponding sets:

{1,2,4}	{b,d,e,f}
{1,3}	{a,d}
{2,4,5}	{b,c,d,e,f}
{3,4,5}	{a,b,c,f}

Node 1 is in sets S_1 and S_2; therefore, its correspondent must be in the intersection of S_1^* and S_2^*. Node d is the only node in this intersection; therefore node 1 must correspond to node d. Similarly node 2 is in S_1 and S_3, so its correspondent must be in the intersection of S_1^* and S_3^*; moreover, sets S_2 and S_2^* have the same number of nodes, so

Generated by		Subset of F	Subset of C	Line no.
Node value:	C	{b,c,d,e,g}	{2,3,4,5,6,7,8,9,11,12,13,14}	1
	O	{f}	{10}	2
	N	{a}	{1}	3
	Br	Λ	{15,16}	4
Branch value:	single	{a,b,c,d,e,f,g}	{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}	5
	double	{c,d,e,g}	{4,5,7,8,11,12,13,14}	6
Degree:	1	{a}	{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}	7
	2	{b,d,e,f,g}	{1,2,3,4,5,6,7,8,9,10,11,12,13,14}	8
	3	{c}	{1,2,3,4,5,6,12}	9
Order	5	{c,d,e,f,g}	{4,5,6,10,11,12,13,14}	10
	∞	{a,b}	{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}	11
Partition:	lines 1–11	{a}	{1}	12
		{b}	{2,3,4,5,6,7,8,9,11,12,13,14}	13
		{c}	{4,5,12}	14
		{d,e,g}	{4,5,11,12,13,14}	15
		{f}	{10}	16
Connectivity:	line 12	{b}	{2,6,7}	17
	line 14	{b,d,g}	{3,4,5,6,10,11,12,14}	18
	line 15	{c,d,e,f}	{3,4,5,6,10,11,12,13,14}	19
	line 16	{e,g}	{4,11}	20
	line 17	{a,c}	{1,3,5,8,13,15,16}	21
	line 18	{a,c,e,f}	{1,2,3,4,5,6,9,10,11,12,13,14}	22
	line 19	{b,c,d,e,f,g}	{1,2,3,4,5,6,9,10,11,12,13,14}	23
	line 20	{c,d,f}	{3,5,10,12}	24
	line 21	{b,d,g}	{2,4,6,7,9,12,14}	25
	line 24	{b,c,d,e,g}	{2,4,5,6,9,11,12,14}	26
	line 25	{a,c,e,f}	{1,3,5,8,10,11,12,13,14,15,16}	27
	line 26	{a,b,c,d,e,f,g}	{1,3,4,5,6,8,10,11,12,13,14,15,16}	28
	line 27	{b,d,e,f,g}	{2,4,5,6,7,9,10,11,12,13,14}	29
	line 28	{a,b,c,d,e,f,g}	{1,2,3,4,5,6,7,9,10,11,12,13,14}	30
	line 29	{a,c,d,e,f,g}	{1,3,4,5,6,8,10,11,12,13,14,15,16}	31
	line 31	{b,c,d,e,f,g}	{1,2,3,4,5,6,7,9,10,11,12,13,14}	32
Partition:	lines 12–32	{a}	{1}	33
		{b}	{6}	34
		{c}	{5}	35
		{d}	{12}	36
		{e}	{11}	37
		{f}	{10}	38
		{g}	{4}	39

Figure 6.—Sets generated in example II.

the correspondent of node 2 must also be in $\bar{S}_2^*$. The intersection of S_1^*, $\bar{S}_2^*$, and S_3^* is {b,e,f}, so that, although the exact correspondent of node 2 is not explicitly determined, the number of possible correspondents is reduced to three. The correspondents of the other nodes are found in a similar manner using relation 2:

		Intersection used
{1}	{d}	$S_1 \cap S_2$
{2}	{b,e,f}	$S_1 \cap \bar{S}_2 \cap S_3$
{3}	{a}	$S_2 \cap S_4$
{4}	{b,f}	$S_1 \cap \bar{S}_2 \cap S_3 \cap S_4$
{5}	{b,c,f}	$\bar{S}_2 \cap S_3 \cap S_4$

Returning to example II of Figures 5 and 6, after completing the generation of sets in box I, the partitioning procedure (box II) is applied to the generated sets. The correspondences which are possible after the partitioning are shown on lines 12 to 16 of Figure 6. It may be noted that the sets on lines 4, 5, 7, and 11 play no part in the partition. Sets S_5^*, S_7^*, and S_{11}^* contain all the nodes of graph C and therefore cannot possibly aid in reducing the number of correspondents of any node of F. Set S_4 is empty and can never be used in relation 2. It is also obvious that it is not necessary to include duplicate pairs of sets in the families. Thus, in general, if a pair S and S^* is generated and if either (1) S is empty, or (2) if S^* contains all nodes of G*, or (3) if S and S^* are already in the families, the pair cannot be of use in the partitioning process and may be excluded from further consideration.

The algorithm step which follows the partitioning is the generation of new pairs of sets using the connectivity property. The sets of example II which are generated are shown on lines 17 to 32 of Figure 6. Those pairs in which S^* cannot be of use in the partitioning process are omitted. The pair on line 21 is generated using the pair on line 17, which in turn was generated from the pair on line 12. Thus S_{21} and S_{21}^* constitute those nodes which are reachable from node a and node 1 (that is, sets S_{12} and S_{12}^*), respectively, by traversing two branches, not just one branch. In general, the connectivity property is applied until

no new sets are generated.[3] When the connectivity property can generate no new sets, the algorithm returns to partition the new family of sets. In the example lines 12 to 32 are partitioned, and a complete correspondence between the nodes of F and a subgraph of C is found, thereby terminating the algorithm.

5. NODE-BY-NODE MATCHING

Before turning to a discussion of box IV, the only operational process in the flow diagram which has not been considered, it is of interest to digress briefly and examine how example II would be processed using the conventional technique of node-by-node matching. In the node-by-node approach, the nodes of the two structures are matched one at a time until either a valid correspondence is found or until an incompatibility arises; in the latter case, it is necessary to backtrack to a point where there is agreement and start again with different nodes.

In Figure 5, as both a and 1 are N atoms, the a–1 correspondence is a reasonable starting point. a connects to b, b is a C atom; 1 connects to 2, also a C atom; therefore b–2 is reasonable. Similarly c–3, d–4, and e–5 are reasonable. However, e connects to an O atom; 5 does not connect to an O atom; therefore, the e–5 correspondence is incompatible. Returning to d–4, one notes 4 connects to 10, but 10 is an O atom; e is not an O, so e–10 is not reasonable. Having exhausted the possibilities for d–4, one notes another incompatibility and returns to c–3 to look for another correspondent for d. d–9 and then e–8 are reasonable matches with respect to atom type but as d and e are doubly bonded and as 8 and 9 are singly bonded, neither the e–8 nor the d–9 matches are allowed. Thus one backtracks to b–2 and finds neither 15 or 16 are reasonable matches for c, thereby indicating that b–2 is also an incompatibility. Then b–6 is tried, and c–5 and d–4 follow. However d–4 is not allowed because of the double bond between 5 and 4. When d–12 is tested, the remaining matches all are compatible: e–11, f–10, g–4.[4]

The obvious distinction between the node-by-node technique and the set generating technique is the large amount of backtracking required by the former. The correspondences are arbitrarily assigned in the node-by-node technique; if the correct choice is fortuitously made at each point, the technique proceeds rapidly. It is highly unlikely, however, that the correct choice will be made at each decision point. Although arbitrary assignments may be required in the set generating technique (see below), they are made only in exceptional circumstances, and therefore there is little probability of backtracking in the latter technique. To be able to backtrack requires the saving of enough information to be able to restart from the last point of agreement, a relatively time-consuming operation. Therefore, although the basic operations required in the set generating technique are intrinsically more complex than the simple operations required by the node-by-node technique, the set generation procedure is more efficient in over-all operation.

6. THE ASSIGNMENT PROCEDURE

The majority of the calculations required by the algorithm are the generation of sets using the connectivity property (box III of Figure 3) and the partitioning of these sets into smaller sets (box II). In general, these two procedures are iterated until one of the following conditions applies: (1) the correspondents of all nodes have been identified; or (2) a cardinality violation arises; or (3) no new sets are generated. When condition 1 occurs, an isomorphism has been determined, and the algorithm terminates. When condition 2 occurs, no isomorphism is possible, and the algorithm terminates. When condition 3 occurs, however, the algorithm may not terminate because there exist some nodes for which no correspondent has yet been found and/or some nodes for which it has not yet been demonstrated that there is no possible correspondent. Ambiguities of this type can arise in two ways. First, when there is more than one isomorphism possible between graphs G and G*, there is no nonarbitrary way of selecting one of the isomorphisms. The second cause is when the properties used to generate sets are not powerful enough to separate nodes which are indeed distinguishable under some (unused) property.

Whenever condition 3 arises, no matter what its cause, there exist a pair of sets $S = \{x_1,x_2,\ldots,x_k\}$ and $S^* = \{x_a^*,x_b^*,\ldots,x_l^*\}$, where $S \subseteq S^*$.[5] Moreover, S and S^* cannot be refined by the generating properties used by the algorithm. If there is an isomorphism between G and G^*, then x_1 must correspond to either x_a^* or x_b^* or ... or x_l^*. An immediate way of resolving this ambiguity is to assign the correspondence $x_1 = x_a^*$. If no contradiction (*i.e.*, cardinality violation) arises subsequent to this assignment, the algorithm continues normally. If, however, a contradiction does arise, it may not be inferred that there is no isomorphism between G and G*, because the wrong correspondent of x_1 may have been assigned. Thus, when a contradiction does arise subsequent to the $x_1 = x_a^*$ assignment, the correspondence $x_1 = x_a^*$ is removed and the correspondence $x_1 = x_b^*$ is tried. Therefore, only after a contradiction has arisen subsequent to the assignment of x_1 to x_a^* and then to x_b^* and ... and finally to x_l^*, may it be concluded that there is no isomorphism between G and G*.

It is desirable to avoid assignments because of the problems associated with backtracking. Of course, assignments caused by multiple isomorphisms, which are called *required assignments*, are unavoidable, but because a required assignment can never result in an incorrect guess, the need for backtracking cannot arise and it is not necessary to make provision in the program for the saving of data. *Optional assignments*, which arise from a deficient property class, however, can be avoided if it is possible to discover a property powerful enough to resolve the existing ambiguity. A complete discussion of the problem of choosing a "good" class of properties is given in the reference cited in footnote 1.

(3) For simplicity in example I, the connectivity property was applied only to unit sets.

(4) This description follows that given by W. E. Cossum, G. M. Dyson, M. F. Lynch, and R. N. Wolfe, "Mechanical Searching of Chemical Substructures by Means of Atom-by-Atom Matching at CAS," mimeographed report, Chemical Abstracts Service, 1963. The node-by-node procedure was originally suggested by L. C. Ray and R. A. Kirsch, *Science*, 126, 814 (1957).

(5) $S = S^*$ in the complete isomorphism test.

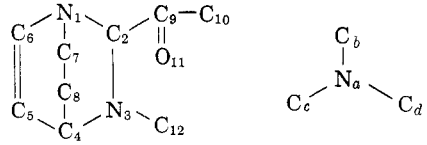


Figure 7.—The compound and fragment for example III.

Example III in Figures 7 and 8 illustrates a required assignment. Fragment F is contained in compound C in twelve different ways; the assignment on line 9 of Figure 8 essentially selects the subgraph composed of nodes 1, 2, 6, and 7 in preference to the subgraph consisting of nodes 2, 3, 4, and 12. Two additional assignments (not shown in Figure 8) are required to completely delineate the correspondence between carbon atoms.

Generated by		Subset of F	Subset of C	Line no.
Node value:	N	$\{a\}$	$\{1,3\}$	1
	C	$\{b,c,d\}$	$\{2,4,5,6,7,8,9,10,12\}$	2
Connectivity:	line 1	$\{b,c,d\}$	$\{2,4,6,7,12\}$	3
	line 2	$\{a\}$	$\{1,2,3,4,5,6,7,8,9,10,11\}$	4
	line 3	$\{a\}$	$\{1,3,5,8,9\}$	5
	line 5	$\{b,c,d\}$	$\{2,4,6,7,10,11,12\}$	6
Partition:	lines 1–6	$\{a\}$	$\{1,3\}$	7
		$\{b,c,d\}$	$\{2,4,6,7,12\}$	8
Assignment		$\{a\}$	$\{1\}$	9
Connectivity:	line 9	$\{b,c,d\}$	$\{2,6,7\}$	10
Partition:	lines 7–10	$\{a\}$	$\{1\}$	11
		$\{b,c,d\}$	$\{2,6,7\}$	12

Figure 8.—Sets generated in example III.

Example IV in Figures 9 and 10, on the other hand, illustrates an optional assignment. In this example the class of set generating properties is intentionally made deficient by omitting the order property. If sets were generated by the order property, the fact that the fragment is not contained in the compound would become apparent because of a cardinality violation in box I. However, because of the omission of the order property, it is necessary to test that all possible correspondents of node a (node 1 on line 3 and node 4 on line 8) lead to cardinality violations (lines 7 and 12) before determining that no isomorphism exists between the fragment and subgraph of the compound.

In the experience of the author, the set generating properties shown in Figure 4, namely, node value, branch value, degree, order, and connectivity, are a sufficient property class in the sense that optional assignments are never required in a complete isomorphism test. This means that backtracking never occurs, and assignments are needed only to resolve multiple isomorphisms. Although a sufficient class has not been found for subgraph tests, optional assignments are required only very infrequently.[1]

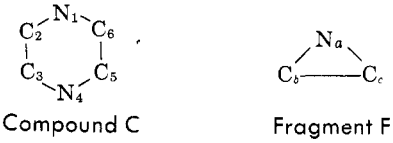


Figure 9.—The compound and fragment for example IV.

Generated by		Subset of F	Subset of C	Line no.
Node value:	N	$\{a\}$	$\{1,4\}$	1
	C	$\{b,c\}$	$\{2,3,5,6\}$	2
Assignment		$\{a\}$	$\{1\}$	3
Connectivity:	line 3	$\{b,c\}$	$\{2,6\}$	4
	line 4	$\{a,b,c\}$	$\{1,3,5\}$	5
	line 5	$\{a,b,c\}$	$\{2,4,6\}$	6
Partition:	lines 1–6	$\{a\}$	Λ	7
Assignment		$\{a\}$	$\{4\}$	8
Connectivity:	line 8	$\{b,c\}$	$\{3,5\}$	9
	line 9	$\{a,b,c\}$	$\{2,4,6\}$	10
	line 10	$\{a,b,c\}$	$\{1,3,5\}$	11
Partition:	lines 1–2,8–11	$\{a\}$	Λ	12

Figure 10.—Sets generated in example IV.

7. MECHANIZATION

The flow diagram and the examples indicate that the algorithm involves only simple concepts. The mechanization of the algorithm for a digital computer is also simple. The complexity of a computer program is often highly dependent upon the representation of data in the computer memory. The representation of sets and graphs suggested here have been found to be effective for a binary computer, such as the IBM 7090.

Sets are represented by their *characteristic vectors*, that is subset S of set $X = \{x_1,x_2,\ldots,x_n\}$ is a vector $\mathbf{s}$ such that

$$\mathbf{s}_i = \begin{cases} 1 \text{ if } x_i \text{ is in } S \\ 0 \text{ otherwise} \end{cases}$$

For example, subset $\{2,3,6\}$ of set $\{1,2,3,4,5,6\}$ is represented by vector (0,1,1,0,0,1). The characteristic vector representation is convenient, not only because of the compactness of notation, but especially because the set operations *union*, *intersection*, and *complementation* correspond to the logical vector operations *or*, *and*, and *not*, respectively. Since most digital computers have built-in logical operations, set operations are easily performed.

Graphs are represented by their *connection matrices*; that is graph G is represented by a matrix Γ which is such that

$$\Gamma_j^i = \begin{cases} 1 \text{ if node } i \text{ connects into node } j \\ 0 \text{ otherwise} \end{cases}$$

In other words the ith row of Γ is the characteristic vector of Γx_i. Figure 11 shows the connection matrix of the graph of compound C of Figure 5. (The 0 entries are omitted for legibility.) In most cases Γ is symmetric (that is, $\Gamma_j^i = \Gamma_i^j$); however, semipolar bonds can be indicated by a nonsymmetric matrix.

A convenient way to specify node values in the computer is in terms of a matrix $\mathbf{N}$ which is such that

$$\mathbf{N}_j^i = \begin{cases} 1 \text{ if node } j \text{ has value } i \\ 0 \text{ otherwise} \end{cases}$$

For graphs representing chemical compounds, the set of node values is the set of chemical elements {H,He,Li,Be, B,C,N,O,. . . ,No}. For any particular compound only a few of the chemical elements actually occur, and hence

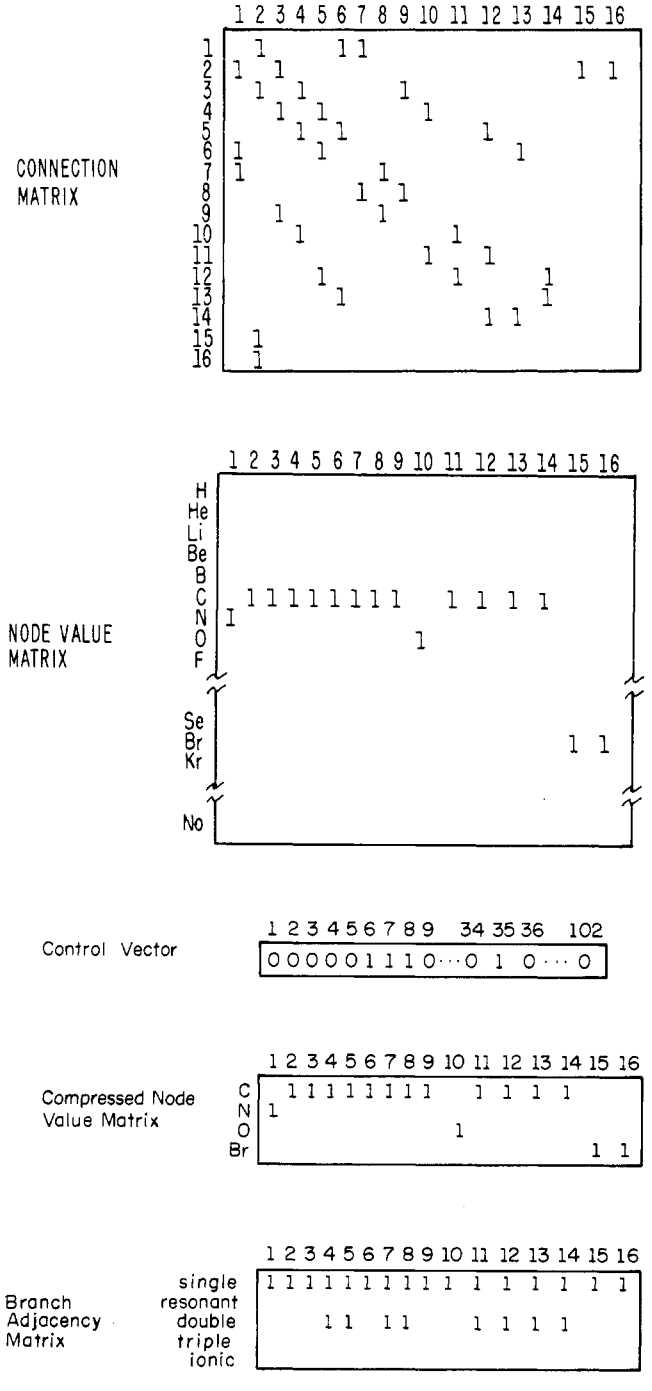


Figure 11.—The matrix representation of compound C of Figure 5.

most of the rows of **N** consist of all 0's. To conserve space only the nonzero rows are stored, and a control vector is used to indicate which elements are actually represented (see Figure 11 for an example). If it is necessary to indicate excess or deficient electrons, the charges are considered nodes of the graph, and the node value set is extended to {H,He,. . . ,No; plus charge, minus charge}.

Another matrix **B** is used to indicate the bonding of the compound. **B** is defined as

$$\mathbf{B}^i_j = \begin{cases} 1 \text{ if node } j \text{ is adjacent to a branch of value } i \\ 0 \text{ otherwise} \end{cases}$$

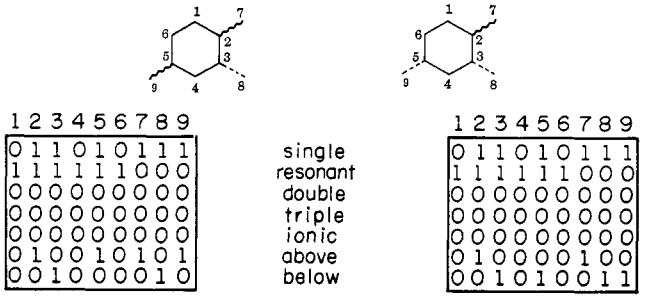


Figure 12.—Stereoisomers.

The set of branch values is {single, resonant, double, triple, ionic}. Figure 11 gives an example. Stereochemical considerations, in which some atoms are considered to be above the plane of the paper and some below, are accommodated by extending the set of branch values to {single, . . . , ionic; above, below}. See Figure 12, in which the wavy line signifies an atom above the plane, and dotted line below. Optical and *cis–trans* isomers are not distinguishable, however; for example, erythrose and threose (Figure 13) have identical representations.

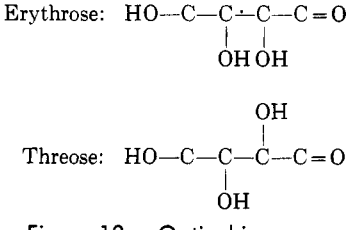


Figure 13.—Optical isomers.

It may be desirable to conduct a search with a query compound in which the specific chemical type of one or more atoms may vary over a set of prescribed types. Such a specification is equivalent to specifying several query compounds exactly. For example, in the compound of Figure 14 the value of node 1 may be either H, Cl, or F, and the value of node 2 may be either H or Br; this multiply specified query represents six uniquely specified queries. Instead of conducting individual searches for each specific query compound, it is possible to perform only one search with the multiply specified query by altering the manner in which the node value property is applied for those nodes x whose values may vary, as for example in Figure 14:

$$\{1\} \subseteq \{x^*:\text{value}(x^*) = \text{H}\} \cup \{x^*:\text{value}(x^*) = \text{Cl}\} \cup \{x^*:\text{value}(x^*) = \text{F}\}$$

and

$$\{2\} \subseteq \{x^*:\text{value}(x^*) = \text{H}\} \cup \{x^*:\text{value}(x^*) = \text{Br}\}$$

The rest of the algorithm is unchanged.

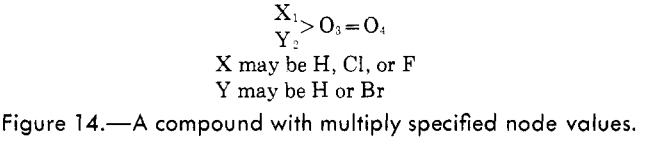
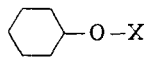


Figure 14.—A compound with multiply specified node values.

It is also possible to represent Markush compounds in which any of several different structures may be substituted at a given location. Figure 15a shows a compound

in which either a hydrogen atom, or methyl group, or phenyl group may be bonded to the oxygen atom. This multiple structure is stored in the library with all three substituents attached to the oxygen atom, as shown in Figure 15b. To find an exact match with any one of the three compounds represented, the subgraph algorithm is employed.

(a) 

X may be H, CH_3, or C_6H_5

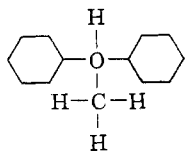
(b) 

Figure 15.—(a) A Markush compound and (b) its representation.

The algorithm has been programmed for the IBM 7090. The program itself requires about 7000 words of storage and can accomodate graphs with up to 200 nodes. In a limited test of the system, 50 compounds were chosen from Beilstein to form the library. Each compound had exactly 50 atoms and contained only the elements carbon, hydrogen, oxygen, and nitrogen; only six different molecular formulas were represented. The compounds were selected in this manner to utilize the full power of the algorithm frequently. All the compounds were reused as queries so that the algorithm was employed 1275 times. The time to determine that two compounds were not isomorphic was, in 85% of the cases, less than 0.5 msec.; the longest time was about 100 msec. The time to compute a complete isomorphism ranged from 3 to 13 sec., with an average of about 7 sec. When all hydrogen atoms were removed from the compounds (so that each compound had approximately 30 atoms), the time was reduced to the range of 0.8 to 4 sec., with an average of 2 sec. No backtracking was ever required.[6]

ACKNOWLEDGMENT

The author wishes to thank Professor Gerard Salton, Mr. Joseph Rocchio, and Dr. Robert Wall of the Computation Laboratory of Harvard University for their many valuable discussions and suggestions.

(6) A revised and more efficient program for the IBM 7090 is available from the author.

A Chemical Structure Storage and Search System Developed at Du Pont*

D. J. GLUCK
E. I. du Pont de Nemours and Co., Wilmington, Delaware
Received March 23, 1964

INTRODUCTION

As early as 1961, we in the Engineering Department of Du Pont recognized the need for a better system for recording chemical structure information for storage and subsequent retrieval. We believed that current methods and the then current development of notation systems would not completely serve our chemists' long range chemical identification needs.

Accordingly, we studied and then developed a chemical structure storage and search system. Huber[1] gave a good review of the various approaches and applications. To use his terminology, our system is topological coding.

Our initial investigation led to singling out the following needs for such a system (Figure 1).

Of primary concern is that a system provide low cost processing of large files of compounds. To serve future needs for any centralized chemical registry, a system would have to be able to process efficiently upward of 500,000 compounds.

Associated with these large files would have to be low cost input, preferably clerical, which could at sometime be converted to a mechanical operation if such were economically justified.

For long-term utility the system must permit unlimited substructure or generic searches, not restricted by pre-established groupings.

At the same time the system should be highly efficient for simple searches, particularly searches for specific compounds.

Low Cost Processing of Large Files
Low Cost (Clerical) Input
Unlimited Sustructure Searches
Simple Searches—Complete Compounds
One Form/Structure
No Ambiguities

Figure 1.—Chemical structure system needs.

* Presented before the Division of Chemical Literature, 147th National Meeting of the American Chemical Society, Philadelphia, Pa., April 6, 1964.
(1) M. L. Huber, *J. Chem. Doc.*, 5, 4 (1965).