Fragment Search in Acyclic Structures

MILAN RANDIĆ

Ames Laboratory—U.S. Department of Energy, Iowa State University, Ames, Iowa 50011

Received April 13, 1977

A graph theoretical algorithm is described which allows a search for fragments in acyclic structures. Although the topic discussed in the paper is of limited immediate use, it provides a basis for atom-by-atom matching which may be practical as a large number of unproductive matchings are eliminated from consideration. The approach is based on the concept of atom codes which register the number of neighbors one, two, and more bonds apart. Atom codes provide a means of storing information on a structure (i.e., connectivity) without an explicit use of atom labels. Atom codes also permit assignment of multiple labels to individual vertices based on some specified rules or prescribed properties.

INTRODUCTION

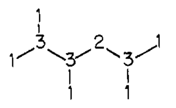
Molecular skeletons can be considered as realization of abstract graphs.[1] Such a recognition in itself is important for some chemical problems and is quite old;[2] however, in itself it does not help much in many other problems involving molecules and chemical reactions, where one is interested in *real* structures rather than in some simplified idealized abstraction. This perhaps may account for a neglect of graph theory among chemists which only recently emerged and is establishing itself as a viable branch of theoretical chemistry.[3] In part the revival of interest in graph theoretical methods followed from use of such methods, or, some concepts only, in development of chemical information science, interest in use of computers in assisting search for optimal synthetic routes, analysis of large number of combinatorial possibilities in drug molecule design, or in the work on biopolymers. There are two distinctive uses of graphs in such applications: (1) graphs provide a convenient and clear (visual) bookkeeping of multiple possibilities, and (2) graphs represent connectivity in factual structures (which may represent molecules, chemical transformations, etc). In graph theoretical schemes, briefly, one is concerned with manipulations of structures, rather than with the numerical data characterizing properties of the structure. So the terms recognition, characterization, classification, fragmentation, encoding, retrieval, decomposition, dissection, discrimination, ordering, comparison, enumeration, composition, construction and reconstruction, all have a rather specific technical connotation. However, while in the area of chemical documentation one has freedom of devising artificial codes which need not relate closely to the structural features of the system (e.g., use of trivial names for chemical compounds) for activities involving manipulations with structures, a close association of structural features supplied by the connectivity table appears essential. Codes which solely use structural information allow meaningful algebraic manipulations; an alternation of a code then corresponds to a structurally related form. Such manipulations require that we can recognize identical graphs and that we can identify fragments of interest. The problem of graph isomorphism has received considerable attention in the past and still remains of considerable interest.[4] If we have rules which uniquely prescribe labels to individual vertices (atoms in a molecule), then a comparison between two structures is trivial and reduces to relabeling of noncanonical labels. The difficulty is not only in finding such rules, which of course, should be general, so that emergence of a novel structure does not call for reconsideration, but also that the rules have to be practical. This means that one can find the canonical labeling without screening all $n!$ possibilities, or an impractically large number of possibilities. Recently such a scheme based on canonical labeling of vertices has been described[5] and demonstrated on a selection of representative examples that finding the required labels can be achieved in few steps only.[6]

A search for fragments in a larger graph is considered even more difficult.[7] While the problem of graph isomorphism is solved once coherent labels are found for the graphs considered, there is no such equivalent proposition for the problem of subgraph isomorphism (another name for fragment search). The problem is usually formulated as follows: Given two graphs G_i and G_j, is G_j isomorphic to a subgraph of G_i (we assumed that G_i has more vertices than G_j)? Here graphs have different number of vertices; if both graphs have the same number of vertices, the problem becomes that of graph isomorphism. The additional complexity of the subgraph isomorphism problem is perhaps also illustrated by the fact that in a general situation there may be numerous subgraphs which give acceptable solution. The problem, if expressed via vertex labels, becomes that of *identifying all acceptable labelings in graph G_i which preserve adjacency of a labeled graph G_j*. It would be helpful to know the number of such solutions but even this is not known.

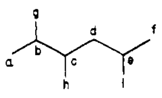
This paper outlines a strategy for attacking the problem. We use the concept of formal *atom codes* which register atomic environment by indicating the number of neighbors one bond apart, followed by the number of neighbors two bonds apart, then three bonds apart, and so on until all vertices have been exhausted. The concept of atom codes as such is not novel and has been considered by others. Thus Penny considers the connectivity around a carbon atom locating other atoms at the circumference of different radius.[8] Gordon and Kennedy[9] proposed an enumeration sequence of alkanes in terms of their skeletal graphs specially arranged so that from a graph G_n next in line, G_{n+1}, is derived by adding a vertex at the appropriate place on the "shell" of neighbors. Several applications of such or similar differentiations of the neighbors have been used for discussion of selected molecular properties.[10] Most systematically and extensively the concept of atom codes which register atomic environment has been studied and developed by Dubois in a series of publications with collaborators.[11] There is some parallelism between the scheme of Dubois and here considered codes, and if one looks for unique attributes of our approach, this is *an emphasis on mathematical formalism*, consideration of a code as a binary sequence of digits *introduced for subsequent algebraic and logistic manipulations*. From such codes one can derive others by summation, shift, erasure of an entry, and other such manipulations. Before outlining the scheme and giving illustrations, let us remark on the utility of the approach. In the present work we have limited the application to acyclic graphs. This *is* a serious limitation for practical uses, but at the same time it *is* also a very *legitimate* approach to a rather complicated problem. *If* one shows that a certain scheme works for acyclic graphs, then one may consider more complex situations systematically. This is a well-established practice in mathematical graph theory, where it is well known that a number of theorems valid for acyclic structures need yet to be demonstrated for more general graphs. For example, the famous Ulam's reconstruction conjecture *is* proven for acyclic graphs, but finding a proof for a general graph remains currently a famous unsolved problem, despite eager search for a solution.[12] The problem of extending the concept of atom codes to polycyclic system is not in the same category. The difficulty here is the presence of numerous paths in such polycyclic graphs which connects two atoms, and the question is that of selecting some rather than others. For instance, one can select the geodesic (the shortest) path, but before such suggestions can be advocated one needs to learn about the properties of such codes. This is the major reason for not rushing at this relatively early stage with extensions of the scheme to polycyclic structures.

CONCEPT OF ATOM CODES

In the following we use the word *label* to indicte a letter, number, or a name for a vertex, i.e., a purely *conventional symbol* devoid of any structural content. We similarly use the word *code* exclusively to signify a symbol that carries some *structural information* and can be interpreted in terms of graph invariants. One can assign labels to vertices of a graph in various ways, adopting a random choice or selecting special rules to be followed. Morgan considered nearest neighbors and their valencies,[13] a technique for which Wipke and Dyott[14] introduced the term "extended connectivity". A similar characterization also follows from the index eigenvector of the adjacency matrix.[15] The concept of the smallest binary code[5,6] leads to an alternative labeling which also reflects some structural features of a system. Although the above represent various characterizations of vertices, the labels do not represent structual information; their *derivation* has been a structural process. The simplest vertex codes which have a *direct* structural meaning are list of valencies, as illustrated with the 2,3,5-trimethylhexane skeleton:



Such labels are not sufficiently discriminatory and it is natural to extend and supplement such label codes with information on next nearest neighbors. One can continue the process until all neighbors for each atom have been registered. For the above example we then have:



a. 1,2,2,1,2
b. 3,2,1,2
c. 3,3,2
d. 2,4,2
e. 3,1,2,2
f. 1,2,1,2,2
g. 1,2,2,1,2
h. 1,2,3,2
i. 1,2,1,2,2

The first code means that vertex a has *one* nearest neighbor (i.e., b), *two* next nearest neighbors (c and g), *two* vertices three bonds apart (d and h), *one* vertex at the distance of four bonds (e), and finally *two* vertices at the distance of five bonds (f and i). Acyclic structures have an important property that any pair of vertices is connected by a single path. Hence, for any vertex the code is not ambiguous. The above atom codes have some properties of interest: (a) the sum of all entries in a row is constant (and equals the number of bonds in the structure); (b) equivalent vertices have necessarily identical codes (however, it appears that the opposite is not necessarily the case); i.e., (c) nonequivalent vertices may have identical codes.[16] With the codes one can perform various mathematical operations, and the entries in a code have been found to satisfy the following (d):[17]

$$\sum_i a_{1i}\cdot a_{ni} = \sum_i a_{ni} + \sum_i a_{n+1\,i}$$

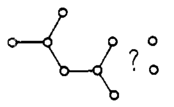
i.e., the product of the first entry and nth entry summed over all entries equals the sum of nth entry and $(n + 1)$th entry. This particular relation can be of use in verifying accuracy of the codes and in testing whether a given list corresponds to a graph or not. It represents a necessary condition, for the *legitimacy* of a code.

There are a number of important questions concerning atom codes which need to be resolved. Probably the most important question is whether a list of codes is unique for a graph; i.e., can two nonisomorphic graphs have the same list? We have examined all acyclic graphs with ten vertices and less and have not found a counterexample which would support the opposite.[16] In addition we have examined a number of larger graphs, in particular, isospectral pairs and triplets,[18] which as is known, cannot be discriminated by use of the characteristic polynomial or graph eigenvalues. So far the conjecture sustained the tests. If the conjecture is found valid, use of atom codes for isomorphism test (of acyclic graphs) may yet be the simplest procedure available. Another important problem (which also depends on the uniqueness of the code lists) is that of the graph *reconstruction* from the codes. We described an algorithm for that purpose,[16] which besides arithmetic manipulations with the codes also includes operations of shifting a code left or right, and erasing some of the entries. Other questions of interest are: does the list of codes have redundant information, and can selected codes suffice for unique characterization of a structure, or perhaps some contracted formulation. For example, Platt[19] has considered a set of integers which represents paths of different length in a molecule, as potential structural parameters for discussion of molecular thermodynamical properties. The sum of all entries in nth place in a code for all atoms gives the number of paths of length n. The derived path codes are unique (at least for all graphs with ten vertices and less), and if found general, will represent a reduced (contracted) molecular code. Another approach may be that of *truncation* of codes to a certain length. The truncated codes are implied in numerous chemical correlations and have been recently used explicitly in discussion of ^{13}C NMR chemical shifts in alkanes.[20] Obviously, the number of mentioned questions and topics here are speculative and conjectural, and future ramifications will depend on their resolutions. We have indicated them not only in order to indicate that extension of the current scheme to polycyclic systems would be premature, but also in order to stimulate further interest among chemist for these and related problems. In graph theory it is not only legitimate to indulge in conjectural work, but is also very traditional, and much of the current progress in the graph theory followed from efforts to resolve such famous conjectures as the *four-color problem*, search for conditions of *existence of Hamiltonian circuits*, and *graph reconstruction* conjecture of Ulam.[21]

There is one particular aspect of atom codes which is worth emphasizing. If one is given a list of codes without a pictorial representation of the associated graph, the labels of individual codes are redundant. They do not carry any information whatsoever and can be erased. The codes can be arranged in any order, for instance, the lexicographic order. For the example previously considered we would then have:

1.2.1.2.2
1.2.1.2.2
1.2.2.1.2
1.2.2.1.2
1.2.3.2
2.4.2
3.1.2.2
3.2.1.2
3.3.2

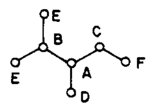
By storing this information in a computer, we would, in fact, store the structure without an explicit use of *labels*. This particular feature is of considerable importance and may find diverse applications. The list as such does not explicitly record *which atom is adjacent to which; such information has to be deduced by a reconstruction process.* The reconstruction means establishing which atom is connected to which. If we use labels (a)–(i) only as an address of individual codes, it follows from the first code:



A part of the graph can be reconstructed. The vertex adjacent to (a) can be identified, since it has on the right the same neighbors that (a) has, but *at one bond closer.* In addition at the left (part of the graph which has been assigned) it has one neighbor, so the code is: 3.1.2.2 which has label (g). In this maner, as outlined elsewhere,[16] the process of reconstruction proceeds.

SEARCHING PROCEDURE

The fact that the code represents a way of registering a structure without a reference to explicit use of atom labels is of profound use for our problem of subgraph isomorphism. As already stated the subgraph isomorphism problem cannot be reduced to that of a search for coherent labeling. But codes of two different structures nevertheless can be compared, and, as will be shortly seen, such systematic comparisons allow one to indicate the vertices in the two graphs which have a *compatible environment.* We proceed by discussing a graph with 16 vertices (Figure 1) and will search for the presence of a smaller graph:



In Table I we list atom codes for the two graphs. Since the order in which the atom codes are listed is immaterial, we adopted an arbitrary ordering. Vertices of the fragment will be designated by capital letters while vertices of the graph investigated are labeled by numbers (1–16). Otherwise the labels are arbitrary. First we will define *compatibility* of the codes of the two graphs by requiring that one code contain the other; i.e., for every entry of the two codes $a_k \geq a_k'$ must hold.

Let us start with the code A (which is 3.3) and compare this code with all the codes in Table I. The comparison shows that vertices (1), (2), (4), and (10) are the only codes with each entry being larger than the corresponding entry in the code A. Hence, if the fragment G_j is contained in the graph G_i then the vertex A of G_j can take only those sites for which the compatibility of codes has been established. In a general node-to-node matching, in principle, one has to examine *all* possibilities and follow combinations involved. Here we see that if G_j is contained in G_i we need to investigate further only *four* sites for A. Of course, some of the sites may later be found inadmissable, but the efficiency of a scheme should be measured by its ability to recognize and discard (at the earliest possible occasion) *unproductive* routes. Here we can immediately discard such matchings as placing A at site (3) and 11 other alternatives! Acceptable possibilities for vertex A are indicated in Figure 2. The process is continued with the code B, which is found compatible with seven sites in G_i, and similarly code C is found compatible (incidently) with the same seven vertices in G_i, while code D is compatible with another set of seven vertices. Finally codes E and F are found compatible with all codes of the graph G_i. The result of the comparison is illustrated in Figure 2. In a general situation some vertices in G_i may remain *unassigned*, but in this particular example all vertices have labels. In Figure 3 we have summarized the result: vertices of the graph investigated have received labels of the fragment under the search. The possibility for multiple solution is reflected by the presence of multiple labels. This ends the first stage in the search. Having found the multiple labels, finding the fragments is rather a straightforward procedure. It may appear lengthy because there are numerous fragments to be found. This should not obscure the high efficiency of the scheme.

The preparation of the complete list of all fragments can

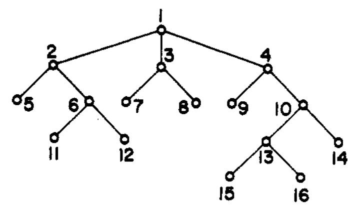


Figure 1. Acyclic graph with 16 vertices considered for the illustration of the search process.

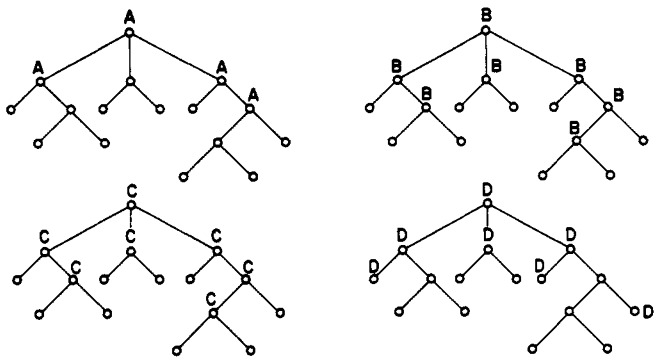


Figure 2. Assignment of labels A, B, C, and D of the fragment graph to the investigated graph.

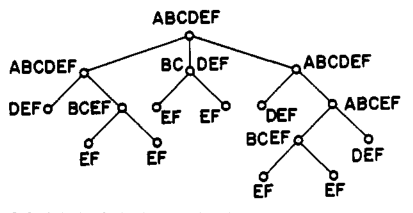


Figure 3. Multiple labels of the investigated graph.

Table I. Atom Codes for the Graphs of Figure 1[a]

				Fragment	
1	3. 6. 4. 2	9	1. 2. 4. 6. 2	A	3. 3
2	3. 4. 4. 2. 2	10	3. 4. 2. 4. 2	B	3. 2. 1
3	3. 2. 4. 4. 2	11	1. 2. 2. 2. 4. 2. 2	C	2. 2. 2
4	3. 4. 6. 2	12	1. 2. 2. 2. 4. 2. 2	D	1. 2. 3
5	1. 2. 4. 4. 2. 2	13	3. 2. 2. 2. 4. 2	E	1. 2. 2. 1
6	3. 2. 2. 4. 2. 2	14	1. 2. 4. 2. 4. 2	F	1. 1. 2. 2
7	1. 2. 2. 4. 4. 2	15	1. 2. 2. 2. 2. 4. 2		
8	1. 2. 2. 4. 4. 2	16	1. 2. 2. 2. 2. 4. 2		

[a] The entries in each line represent the number of first, second, third, etc., neighbors of the atom considered.

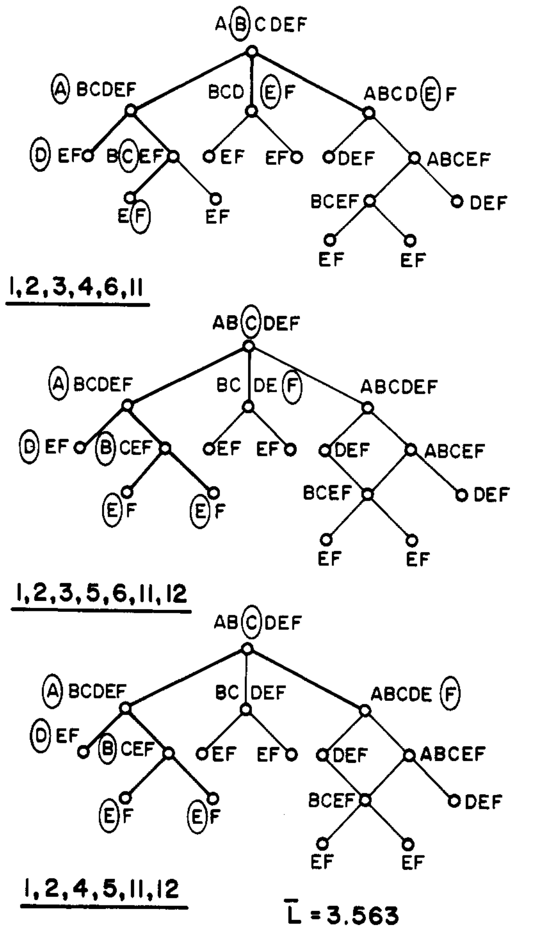


Figure 4. The searching for the fragment and listing of acceptable solutions initiated by identifying the vertex 11 as vertex E of the fragment.

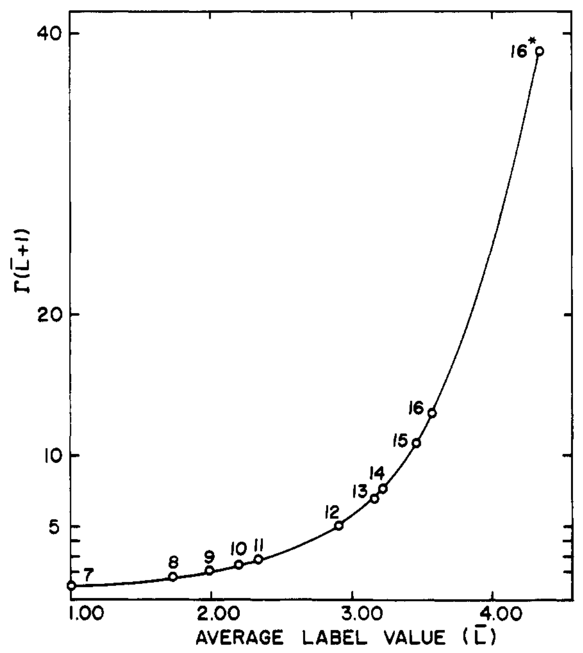
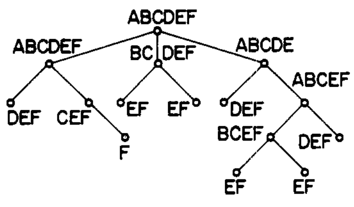


Figure 5. A plot of the factorial function Γ showing continuous reduction in the search possibilities of the fragments for graph of Figure 1 with successive erasure of exhausted vertices.

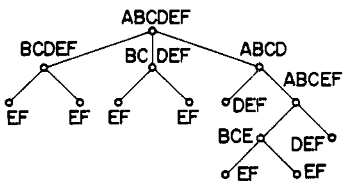
be achieved as follows: Select a vertex in the graph G_i, preferably one with as few multiple labels as possible. Such a vertex will generally be involved in fewer combinations with other vertices and *will sooner lead to examination of all possibilities in which it is involved.* When all the acceptable possibilities have been recorded, the particular vertex has been *exhausted* and as such can be *erased* from the graph G_i. In this way we arrived at a graph G_{i-1} with one fewer vertex. An erasure of a vertex will affect all previously derived codes, so in the next step one ought to redetermine atom codes for the graph with 15 vertices. The new list of atom codes is now used for a compatibility test between G_{i-1} and G_j in order to reassign the labels A–F to the truncated graph. This step is important because it may lead to a decrease in the multiplicity of the labels, and with fewer labels fewer possibilities need be examined. One should not be misled that this additional work, which requires for every step a modification of every vertex code, increases the complexity of the algorithm, giving it perhaps exponential character. The number of operations of this kind for a graph with n vertices is less than the sum of: $n + (n - 1) + \ldots + (n - j)$, which is less than $n(n + 1)/2$, i.e., gives the algorithm polynomial character.

Figure 4 illustrates the searching for the fragment and lists, one by one, the acceptable solutions. We select a single vertex to start the procedure and have taken vertex 11 which has double labels E, F. First we will examine the possibilities that may emerge from vertex 11 being compatible with code F. In the fragment G_j F is connected to C, so we search if in G_i the site 11 has in adjacency C. This condition being satisfied, we continue to search for label A adjacent to C, and we find also this condition agrees. Vertex A should have two neighbors B and D, and one such possibility is indicated in Figure 4 where we encircled the labels satisfying the adjacency conditions. Finally B should have two neighbors labeled as E, and this also is the case. Hence the fragment G_j is contained in the graph G_i with the following correspondence among the vertices: (F:11), (C:6), (A:2), (D:5), (B:1), and (E:3,4). If we order the involved vertices in a sequence of increasing numbers, the fragment found is: 1,2,3,4,5,6,11; i.e., the seven vertices have the required connectivity. In a same manner one continues the seach by examining the other possibility of the vertex 11 which is also compatible with the environment of the code E. As can be easily verified in this case, the search leads to two acceptable fragments, since once labels E, B, A, D, and C are found, there are two possible selections for a vertex labeled F. In Figure 4 these possibilities are indicated. This concludes the possibilities in which the vertex 11 could be involved; hence in continuation of the search we can eliminate, by erasure, vertex 11. So we arrive at a smaller graph with only 15 vertices, for which the new multiple labels are shown in Figure 6 (top). One should observe that several vertices have now fewer labels for their characterization. This *is* significant for the efficiency of the search. We may approximately indicate the "size" of the problem by evaluating the "average label value" ($\bar{L}$) defined as the number of all labels used in the multiple label assignment divided by the number of vertices in the graph. At the initial stage we had in all 57 labels for the 16 vertices which gives the average label value of 3.563. In a blind search for all possibilities, each vertex would have six labels; hence the average label value would be 6. If, however one discriminates valency of the vertices, then terminal vertices can have only three labels (D, E, F) while other (branching) vertices can take all six labels. Such an approach would decrease the average label value from 6 to 4.313. *Improvement from this value is possible in our approach since we take into account compatibilities imposed by all atomic neighbors,* not only the nearest neighbors given by the valency of a vertex. Each successive erasure of a vertex brings further increase in the efficiency of the search. With 15 vertices graph G_{i-1} the average label value drops to 3.467. In the next step this value is further reduced to 3.214, then to 3.154, 2.917, 2.364, 2.200, 2.000, 1.750 and finally to 1.000, which gives the last (twentieth) acceptable solution. The steps undertaken are illustrated in Figures 5 and 6 which also give after each



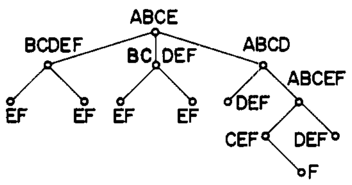
1	3.6.3.2
2	3.3.4.2.2
3	3.2.4.3.2
4	3.4.6.1
5	1.2.3.4.2.2
6	2.2.2.4.2.2
7,8	1.2.2.4.3.2
9	1.2.4.6.1
10	3.4.2.4.1
12	1.1.2.2.4.2.2
13.	3.2.2.2.4.1
14.	1.2.4.2.4.1
15,16	1.2.2.2.2.4.1

1,2,3,4,5,6,12 $\bar{L} = 3.467$



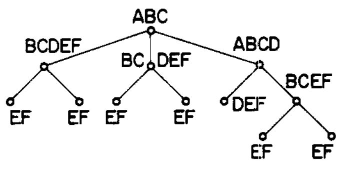
1	3.6.2.2
2,3	3.2.4.2.2
4	3.4.6
5,6 7,8	1.2.2.4.2.2
9	1.2.4.6
10	3.4.2.4
13	3.2.2.2.4
14	1.2.4.2.4
15,16	1.2.2.2.2.4

1,4,9,10,13,14,15
1,4,10,13,14,15,16 $\bar{L} = 3.214$
4,9,10,13,14,15,16



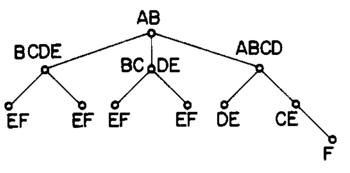
1.	3.6.2.1
2,3	3.2.4.2.1
4	3.4.5
5,6 7,8	1.2.2.4.2.1
9	1.2.4.5
10	3.3.2.4
13	2.2.2.2.4
14	1.2.3.2.4
16	1.1.2.2.2.4

1,4,10,13,14,16,9 $\bar{L} = 3.154$



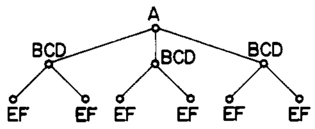
1	3.6.2
2,3	3.2.4.2
4	3.4.4
5,6 7,8	1.2.2.4.2
9	1.2.4.4
10	3.2.2.4
13,14	1.2.2.2.4

1,2,3,4,9,10,13
1,2,4,9,10,13,14 $\bar{L} = 2.917$
1,3,4,9,10,13,14



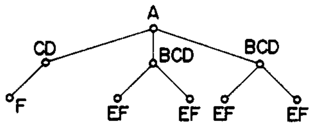
1	3.6.1
2,3	3.2.4.1
4	3.3.4
5,6 7,8	1.2.2.4.1
9	1.2.3.4
10	2.2.2.4
14	1.1.2.2.4

1,2,3,4,9,10,14 $\bar{L} = 2.364$



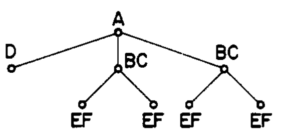
1	3.6
2,3,4	3.2.4
5,6,7 8,9,10	1.2.2.4

1,2,3,4,5,7,8 *1,2,3,4,5,9,10*
1,2,3,4,5,6,9 *1,2,3,4,5,6,10* $\bar{L} = 2.200$
1,2,3,4,5,6,7 *1,2,3,4,5,6,8*



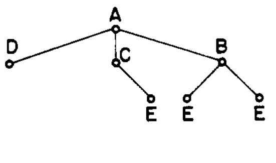
1	3.5
2.	2.2.4
3,4	3.2.3
5	1.1.2.4
7,8 9,10	1.2.2.3

1,2,3,4,5,9,10
1,2,3,4,5,7,8 $\bar{L} = 2.000$



1	3.4
2	1.2.4
3,4	3.2.2
7,8 9,10	1.2.2.2

1,2,3,4,7,9,10
1,2,3,4,7,8,9 $\bar{L} = 1.750$
1,2,3,4,7,8,10



1	3.3
2	1.2.3
3	2.2.2
4	3.2.1
8	1.1.2.2
9,10	1.2.2.1

1,2,3,4,8,9,10 $\bar{L} = 1.000$

Figure 6. The continuation of the search process: at each step list of atom codes for reduced graph are given, all acceptable solutions indicated, and the average label value shown. The exhausted vertex is then erased and the process repeated.

such step all acceptable solutions.

EFFICIENCY OF THE ALGORITHM

As the illustration showed, there was no backtracking or duplication of the search effort. The algorithm, as outlined, secures a complete list of subgraphs isomorphic to the fragment selected. The proposed algorithm is general, i.e., applicable to any acyclic structure. It is also analytical in nature; hence the actual search, the derivation of the codes, and the iteration of the process after an erasure of a vertex are suited for computer processing. The scheme could simply be incorporated as a subroutine in other computer programs involving structure manipulations.

Not all aspects of the scheme could have been introduced in a single illustration. Not infrequently in the process of assignment of the multiple labels, some vertices may not be compatible with any of the fragment codes, hence would remain unlabeled. An example is shown in Figure 7, where labels A–F are those of the fragment G_j. The unlabeled vertices cannot be a part of the sought fragment; their immediate environment does not meet the qualifications that the fragment and its connectivity impose. Such vertices should be ignored in the further analysis. This suggests that the average label value for the graph of Figure 7 should be evaluated as 56/16 = 3.500 (*ignoring* the four unlabeled vertices). But the efficiency of the search would be *increased* if those unlabeled vertices are not only *ignored*, but *deleted*, and new codes derived. As shown in the lower part of the Figure 7, fewer labels now appear, and the corresponding average label value is 52/16 = 3.250. The average label values also may serve as a (qualitative) index of the "size" of the problem; the smaller the value the fewer possibilities will in general follow. So the two examples of the graphs with 16 vertices (Figure 6 and Figure 7) are of a comparable size, though the latter is expected to involve less work. The size of the problem obviously depends on the relative size of the fragment sought. If the graph under investigation is large and vertices have many neighbors, the list of vertex codes will be lengthy and the numbers large with resultant little discrimination. Efficiency of the search in such instances can be enormously increased if the graph is broken in two (or more

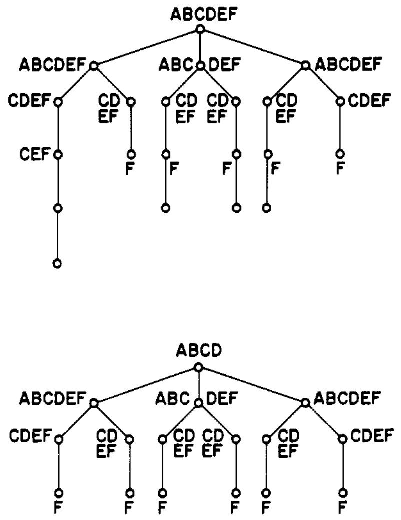
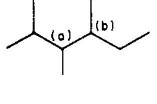


Figure 7. A graph for which initially some vertices remain unlabeled. Subsequent erasure of unlabeled vertices reduces the average label value L.

parts) and these parts are examined separately. After all fragments in the parts are found, the graph should be reconstructed (joined together by earlier erased bonds) and investigated for possibility of fragments in the area of the *ruptures*. However, to accomplish this, one only needs vertices which are as far from the rupture site as the length (the largest distance) of the fragment is; hence all more distant vertices can be erased, which will result in simplification. Thus, if a graph is broken into two parts, instead of analyzing one large graph, we deal with three smaller graphs, two giving the parts after the rupture and the third overlapping subgraph with pruned periphery. Some further simplification follows from use of apparent symmetry of graphs investigated. This is not difficult to implement, *once equivalence of vertices has been recognized*. For instance, the graph G_i, discussed previously and illustrated in Figure 4, has several equivalent vertices: the pair 7,8, the pair 11,12, and the pair 15,16. Using the vertex 11 as the start of the search we found three fragments: (1,2,3,4,5,6,11), (1,2,3,5,6,11,12), (1,2,4,5,6,11,12). If we started with the vertex 12 instead, we would find again three fragments, which will differ only from those given above by replacement of numbers 11 and 12 wherever they appear. We immediately see that this introduces only one new fragment: (1,2,3,4,5,6,12), since the latter two fragments already involve both equivalent vertices. Using symmetry we can *eliminate actually all equivalent vertices at a single step*. That is certainly a good refinement, but it calls for verification of graph symmetry, which may in some instances be quite involved.[22] The list of atom codes may suggest equivalence of vertices, since we know that equivalent vertices necessarily have identical lists of neighbors. However the opposite is not necessarily true, and two vertices which have the same code may nevertheless belong to a nonisomorphic environment. An example is given by vertices (a) and (b) in the graph shown below:



They both have the code (3.4.2). Therefore some caution is required when symmetry of the graph is to be used.

It should also be mentioned that in the special case if the two graphs, G_i and G_j, have the same number of vertices, the problem of fragment search becomes that of graph isomorphism. So the scheme also offers an alternative route to establishing if two graphs are isomorphic or not.

DISCUSSION

The topic discussed in this paper may be of importance to the chemical information field, despite the fact that in this initial stage the scheme is limited to *acyclic* structures. Substructure search systems have been described in the literature for over 10 years.[23,24] They are of interest in computer-aided examination and manipulation of chemical structural information.[25] Conceptually the simplest procedure is atom-by-atom (node-by-node) matching which one can try to improve by reducing the number of comparisons by selecting additional features, such as the least frequent atom. Some reduction in the search, as erasure of fully examined vertices, may be found also in other approaches, as in node-by-node matching. However, in our case, such an erasure will affect codes (characterization) of *all* vertices and reduce some multiple labels not only in the immediate vicinity of the erasure site. Hence, the efficiency has increased *after* such a step, not only because now we have a graph with (n − 1) nodes, but *assignment of labels has been altered.* The situation can be illustrated numerically with the concept of average label values. For the graph of Figure 4 we had initially 57 labels distributed over 16 vertices giving a mean of 3.563. *After* finding three fragments involving vertex 11 we erased this vertex and reduced the mean to 3.466. If we simply erased vertex 11 and left the labels unaltered, we would have 55 labels for 15 vertices, giving the mean 3.667. (It is not difficult to see that in such a process we would gradually impair the efficiency of the search since fewer and fewer vertices with only few labels will remain.) Instead of pursuing a closer comparison with other schemes (which is premature also in view of limited applicability of our approach), we will briefly comment on the computational complexity of the proposed algorithm.

Much of the current interest in algorithms is focused on the basic question asked in complexity theory, which concerns the time required to solve the problem and the size of a computer memory required.[26] Such analysis helps to identify classes of problems that can be solved with essentially the same algorithmic approach. Particularly important here is the concept of *reducibility* of a problem and *classification* of problems into P (polynomial) and NP (nonpolynomial) in time[27] (in fact, in complexity theory one only counts operations, and this is approximately proportional to the time necessary to perform them). Subgraph isomorphism has been classified as NP-complete,[26,27] for which there is not yet evidence that the apparent exponential growth of the combinatorial possibilities with an increase in the size of the problem (by considering larger graphs) can be reduced to the polynomial expression. By analyzing the individual steps in our approach we can see where most of the complexity resides, and where the future efforts for improvement may bring considerable benefits. Firstly, the *evaluation of atom codes* is a polynomial problem since it can be resolved by matrix manipulations. Formally, the nonzero entries in the adjacency matrix A give all the first neighbors. The square matrix A^2 similarly gives information on all paths of length 2 (again only "ones" in the matrix should be recorded). Similarly higher powers give paths of length 3, 4, etc. The *comparison of codes* of G_i and G_j is also polynomial in character; there are less than $n_i \cdot n_j$ such steps for each comparison, with i decreasing with each iteration. The repetition of finding codes and making comparison is an *additive*, not a multiplicative, feature, hence cannot change the polynomial character of the problem. The only complexity yet to be analyzed is associated with the *preparation of the list of fragments*. Here one essentially deals with a node-to-node matching; hence multiple possibilities have to be analyzed and some backtracking is possible. However, the matching of the nodes *is restricted by the assignment*, so is not left to a random process with accompanying requirement for exhaustion of all trails. Even more importantly, the amount of work does not exponentially "explode" with the increase of the size of the graph investigated. In a graph with i vertices and with a fragment having j vertices, at most one has to examine $i \cdot (j!)$ possibilities—and this assumes that the multiple labeling has not restricted assignment of labels at all. For a given fragment, $a_j\ (j!)$ is a constant (an upper limit on the number of combinatorial possibilities involved), so increase of the size of the investigated graph by an additional vertex increases the number of additional work by a *constant* (even if it may be large). This would suggest again polynomial character for the scheme. With an increase of the fragment, of course the complexity increases with $(j + 1)!$, seemingly exponential growth. However, one should also observe that the larger the fragment the fewer multiple labels will appear (this is also illustrated in Figure 4 where one can look at the successive steps as problems in which the *relative* size of the investigated graph and the fragment gradually increase, until at the end the two graphs have the same number of vertices and the test becomes that of graph isomorphism). *So apparent increase in* $(j)!$ *is more than compensated by reduction in multiple labels*, as demonstrated by convergence of the average label values in Figure 4. So although overall it may be somewhat ambiguous to clearly characterize the algorithm as polynomial or not, in practice the number of possibilities to be examined is dramatically reduced in a comparison with a node-to-node search in which information on more distant neighbors is not explicitly introduced. Hence, we are optimistic that in realistic applications the scheme will prove efficient and will remain within practical limits.

ACKNOWLEDGMENT

The author wishes to thank Dr. A. Zamora for valuable comments and Professor R. C. Read (Waterloo) for some unpublished material. The work has been supported through Ames Laboratory by the U.S. Department of Energy, Division of Basic Energy Sciences.

REFERENCES AND NOTES

(1) N. L. Biggs, E. K. Lloyd, and R. J. Wilson, "Graph Theory 1736–1936", Clarendon Press, Oxford, 1976, Chapter 4, p 55.
(2) A. Cayley, *Phil. Mag.*, **47**, 444 (1874). J. J. Sylvester, *Nature* (*London*), **17**, 284 (1877).
(3) A. T. Balaban, D. Farcasiu, and R. Bănică, *Rev. Roum. Chim.*, **11**, 1205 (1966); A. T. Balaban, Ed., "Chemical Applications of Graph Theory", Academic Press, London, 1976; I. Gutman and N. Trinajstič, *Fortschr. Chem. Forsch.*, **42**, 49 (1973); A. Graovac, I. Gutman, and N. Trinajstič, "Topological Approach to the Chemistry of Conjugated Molecules" (Lecture Notes in Chemistry No. 4), Springer-Verlag, Berlin, 1977; D. H. Rouvray, *J. Chem. Soc. Rev.*, **3**, 355 (1974).
(4) R. C. Read, *J. Graph Theory*, in press.
(5) M. Randič, *J. Chem. Phys.*, **60**, 3920 (1974); **62**, 308 (1975).
(6) M. Randič, *J. Chem. Inf. Comput. Sci.*, **17**, 171 (1977).
(7) A. V. Aho, *Acta Crystallogr.*, **33**, 5 (1977).
(8) R. H. Penny, *J. Chem. Doc.*, **5**, 113 (1965)
(9) M. Gordon and J. W. Kennedy, *J. Chem. Soc., Faraday Trans. 2*, **69**, 484 (1973).
(10) E.g., J. E. Dubois and J. Chretien, *J. Chromatogr. Sci.*, **12**, 811 (1974).
(11) J. E. Dubois in "Computer Representation and Manipulation of Chemical Information", W. T. Wipke et al., Ed., Wiley, New York, N.Y., 1974, p 239. J. E. Dubois in "The Chemical Application of Graph Theory", ref 3.
(12) For a survey, see J. A. Bondy and R. L. Hemminger, *J. Graph Theory*, in press.
(13) H. L. Morgan, *J. Chem. Doc.*, **5**, 107 (1965).
(14) W. T. Wipke and T. M. Dyott, *J. Am. Chem. Soc.*, **96**, 4834 (1974).
(15) M. Randič, *J. Chem. Inf. Comput. Sci.*, **15**, 105 (1975).
(16) M. Randič, to be published.
(17) F. E. Harris (Salt Lake City), private communication, 1977.
(18) The number of isospectral graphs representing skeletons of interest in chemistry is growing. See T. Živkovič, N. Trinajstič, and M. Randič, *Mol. Phys.*, **30**, 517 (1975); W. C. Herndon and M. L. Ellzey, Jr., *Tetrahedron*, **31**, 99 (1975), and other publications of these authors.
(19) J. R. Platt, *J. Phys. Chem.*, **56**, 328 (1952).
(20) M. Randič, submitted for publication in *J. Chem. Phys.*
(21) There is extensive literature on these famous problems. The first two (four color conjecture and Hamiltonian circuits problem) are to be found outlined in most elementary introductory books, like: R. J. Wilson, "Introduction to Graph Theory", Academic Press, New York, N.Y., 1972. Such expositions should be supplemented with more recent developments, e.g., *Science*, **193**, 564 (1976). For Ulam's conjecture see ref 12.
(22) For an outline to discern the symmetry properties of graphs, see M. Randič, *Chem. Phys. Lett.*, **42**, 283 (1976); *Croat. Chem. Acta*, **49**, 643 (1977).
(23) W. E. Cossum et al., "Advances in Automatic Chemical Substructure Searching Techniques", *J. Chem. Doc.*, **5**, 33 (1965); J. E. Armitage and M. F. Lynch, *J. Chem. Soc.*, *C*, 521 (1967); R. J. Feldmann, S. R. Heller, K. P. Shapiro, and R. S. Heller, *J. Chem. Doc.*, **12**, 41 (1972); R. J. Feldman and S. R. Heller, *ibid.*, **12**, 48 (1972); M. M. Cone, R. Venkataraghavan, and F. W. McLafferty, *J. Am. Chem. Soc.*, **99**, 7668 (1977).
(24) For less applicative aspect of the problem (i.e., mathematical underlaying basis), see G. Levi, *Calcolo*, **9**, 341 (1972); E. H. Sussenguth, Jr., *J. Chem. Doc.*, **5**, 36 (1965); G. Salton and E. H. Sussenguth, Jr., *Proc. AFIPS 1964*, Spartan Books, New York, N.Y., 1964, p 587.
(25) We give a list of references originating from different laboratories more to indicate various sources where interested reader should search for additional material: E. J. Corey, *Q. Rev. Chem. Soc.*, **25**, 455 (1971); J. Gasteiger, P. D. Gillespie, D. Marquarding, and I. Ugi, *Fortschr. Chem. Forsch.*, **48**, 1 (1974); J. Lederberg, *Proc. Natl. Acad. Sci. U.S.A.*, **53**, 134 (1965); H. Gelernter, N. Sridharan, A. J. Hart, S.-C. Yen, F. W. Fowler, and H.-J. Shue, *Top. Curr. Chem.* (*Fortschr. Chem. Forsch.*), **41**, 114 (1973); C. K. Johnson and C. J. Collins, *J. Am. Chem. Soc.*, **96**, 2514 (1974); R. Fugman, U. Dölling, and H. Nickelsen, *Angew. Chem. Int. Ed. Engl.* **6**, 723 (1967); J. B. Hendrickson, *J. Am. Chem. Soc.*, **93**, 6847 (1971); W. T. Wipke and T. M. Dyott, *ibid.*, **96**, 4825 (1974); M. Bersohn and A. Esack, *Chem. Rev.*, **76**, 269 (1976); A. J. Thakkar, *Top. Curr. Chem.*, **39**, 3 (1973); T. M. Gund, P. v. R. Schleyer, P. H. Gund, and W. T. Wipke, *J. Am. Chem. Soc.*, **97**, 743 (1975); A. Zamora, *J. Chem. Inf. Comput. Sci.*, **16**, 40 (1976); D. H. Smith, L. M. Masinter, and N. S. Sridharan in ref 11, p 287; V. V. Serov, M. E. Elyashberg, and L. A. Gribov, *J. Mol. Struct.*, **31**, 381 (1976); J. C. J. Bart and E. Garagnani, *Z. Naturforsch.*, **31b**, 1646 (1976).
(26) A. V. Aho, J. E. Hopcroft, and J. D. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley, Reading, Mass., 1974.
(27) S. A. Cook, "Proceedings of the Third Annual Association of Machinists, Symposium of Theory of Computing", 1971, p 151. R. M. Karp, "Complexity of Computer Computations", R. Miller and J. Thatcher, Ed., Plenum Press, New York, N.Y., 1972, p 85.
(28) L. C. Ray and R. A. Kirsch, *Science*, **126**, 814 (1957).