(2) (a) Fujita, S. *J. Chem. Inf. Comput. Sci.* **1986**, *26*, 205. (b) Fujita, S. *J. Chem. Inf. Comput. Sci.* **1986**, *26*, 212. (c) Fujita, S. *J. Chem. Inf. Comput. Sci.* **1986**, *26*, 224. (d) Fujita, S. *J. Chem. Inf. Comput. Sci.* **1986**, *26*, 231. (e) Fujita, S. *J. Chem. Inf. Comput. Sci.* **1986**, *26*, 238. (f) Fujita, S. *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 99. (g) Fujita, S. *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 104. (h) Fujita, S. *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 111. (i) Fujita, S. *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 115. (j) Fujita, S. *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 120. (k) Fujita, S. *J. Chem. Soc., Perkin Trans. 2* (in press). (l) Fujita, S. *Yuki Gosei Kagaku Kyokaishi* **1986**, *44*, 354.
(3) (a) Frèrejacque, M. *Bull. Soc. Chim. Fr.* **1939**, *6*, 1008. (b) Plotkin, M. *J. Chem. Doc.* **1971**, *11*, 60. (c) Zamora, A. *J. Chem. Inf. Comput. Sci.* **1976**, *16*, 40. (d) Bersohn, M. *J. Chem. Soc., Perkin Trans. 1* **1973**, 1239. (e) Esak, A. *J. Chem. Soc., Perkin Trans. 1* **1975**, 1120. (f) Schmidt, B.; Fleischhauer, J. *J. Chem. Inf. Comput. Sci.* **1978**, *18*, 204. (g) Gasteiger, J.; Jochum, C. *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 43. (h) Roos-Kozel, B. L.; Jorgensen, W. L. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 101. (i) Hendrickson, J. B.; Grier, C. L.; Toczko, A. G. *J. Chem. Inf. Comput. Sci.* **1984**, *24*, 195.
(4) (a) Corey, E. J.; Wipke, W. T. *Science (Washington, D.C.)* **1969**, *166*, 178. (b) Corey, E. J.; Petersson, G. A. *J. Am. Chem. Soc.* **1972**, *94*, 460.
(5) Fujamann, R.; Dölling, U.; Nickelsen, H. *Angew. Chem., Int. Ed. Engl.* **1967**, *6*, 723.
(6) Wipke, W. T.; Dyott, T. M. *J. Chem. Inf. Comput. Sci.* **1975**, *15*, 140.
(7) Hückel, W.; Danneel, R.; Schwartz, A.; Gerke, A. *Liebigs Ann. Chem.* **1929**, *474*, 121.
(8) For the construction of an ITS, see ref 1 and 2a.
(9) The ITS is stored in a computer in terms of an ITS connection table in which connectivity between each pair of nodes is represented by a complex bond number (*a b*) or a color-bond number (*o*, *i*, *p*). See Appendix I.
(10) The starting molecules are generated by the PS operation (projection to starting stage), which is deletion of in-bonds in a graphical expression. The products are reproduced by the PP operation (projection to product stage), which deletes out-bonds in a graphical expression. See Appendix I.
(11) Multi-tied rings are not adopted here because of the need to save various arrays that are concerned with rings detected. However, another algorithm that considers both tied and multi-tied rings for covering conditions would be adopted for some purposes.
(12) Hirano, S.; Hara, H.; Hiyama, T.; Fujita, S.; Nozaki, H. *Tetrahedron* **1975**, *31*, 2219.
(13) An algorithm for detecting all rings has been reported. See: Kudo, Y. in *Organic Synthesis and Computers* CMC Press: Tokyo, 1983; p 49. See also ref 6.
(14) Meiser, W. *Ber.* **1899**, *32*, 2055.
(15) Felix, D.; Muller, R.; Horn, U.; Schreiber, R.; Eschenmoser, A. *Helv. Chim. Acta* **1972**, *55*, 1276.
(16) Tarbell, D. S. *Org. React.* **1944**, *2*, 1.
(17) Donaruma, L. G.; Heldt, W. Z. *Org. React.* **1960**, *11*, 1.
(18) Criegee, R.; Kaspar, R. *Liebigs Ann. Chem.* **1948**, *560*, 127.
(19) von Auwers, K.; Ziegler, K. *Liebigs Ann. Chem.* **1921**, *425*, 217.
(20) (a) Fujita, S.; Nozaki, H. *Bull. Chem. Soc. Jpn.* **1971**, *44*, 2827. (b) Fujita, S.; Nozaki, H. *Yuki Gosei Kagaku Kyokaishi* **1972**, *30*, 679.
(21) (a) Ryan, A. W.; Stobaugh, R. E. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 22. (b) Freeland, R. G.; Funk, S. A.; O'Korn, L. J.; Wilson, G. A. *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 94.

Automated Classification of Candidate Structures for Computer-Assisted Structure Elucidation

ALAN H. LIPKUS[1] and MORTON E. MUNK*

Department of Chemistry, Arizona State University, Tempe, Arizona 85287

Received March 12, 1987

In computer-assisted structure elucidation, a large number of candidate structures for the unknown compound can be generated. A computer program designed to aid in the recognition of significant differences between these structures is described. The goal of the program is to group the candidates into meaningful classes with a minimum of input from the user; each class is distinguished by a substructure its members have in common. A computational approach is developed based on the combinatorial problem of set covering. The program evaluates its results using an information-theoretical criterion. An application of the program to a real-world structure problem is presented.

INTRODUCTION

The field of computer-assisted structure elucidation now comprises many different computer programs designed to facilitate the process by which a chemist deduces the structure of an unknown compound from spectroscopic and chemical data.[2] Of central importance to this field are programs capable of generating all the molecular structures satisfying certain given constraints that reflect the current state of knowledge about the unknown compound. These so-called structure generators allow the chemist to examine all plausible candidates for the unknown at any stage of the elucidation process. Several of these programs, such as ASSEMBLE,[3] CONGEN,[4] GENOA,[5] and COCOA[6] are highly sophisticated and offer considerable breadth in the range of constraining information they can accept from the chemist. The recently developed program COCOA is designed to receive most of its structural input directly from spectrum-interpreting programs. Some other more specialized structure generators[7,8] developed earlier also function in this manner.

When the available information is insufficient to narrow the set of generated structures to one, this set must be examined for the purpose of finding additional information with which to constrain structure generation more severely. The result of this examination is often the planning of new experiments. This step in the computer-assisted structure elucidation process requires the chemist to perceive chemically or spectroscopically meaningful differences between the candidate structures. This may be impractical, however, if the number of candidates generated is very large. For this reason, structure-editing programs have been devised to assist the chemist in examining large sets of candidate structures.[9,10] These programs allow the creation of subsets of candidates possessing certain substructural features, or combinations of features, selected by the user. Organizing the candidates in this manner can facilitate the perception of experimentally meaningful structural differences.

In the present work, a different approach is explored to the problem of examining the potentially large number of candidate structures generated for an unknown compound. A computer program is described that, like a structure editor, places candidates into different groups on the basis of selected substructural features. Unlike an editor, however, this program aims to select these features largely by itself, with a minimum of input from the user.

CLASSIFICATION SCHEME FOR CANDIDATE STRUCTURES

A method of classifying candidate structures is proposed that is both useful to the chemist and amenable to computational treatment. In this classification scheme all the candidate

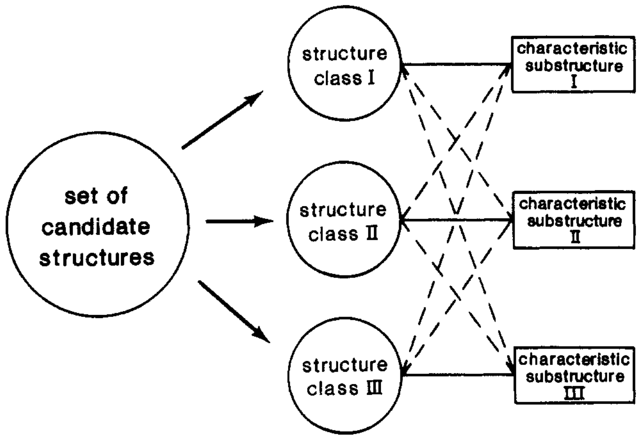

set of candidate structures
structure class I
structure class II
structure class III
characteristic substructure I
characteristic substructure II
characteristic substructure III

Figure 1. Schematic representation of the classification problem. The goal is to classify the candidate structures for an unknown compound into several groups—in this case, three. The structure classes must be related to a set of characteristic substructures as indicated by the solid and dashed lines. (Solid line) Every structure in the class contains the substructure. (Dashed line) No structure in the class contains the substructure.

structures are grouped into appropriate "structure classes", each of which satisfies two conditions. First, each class is characterized by some substructure its members have in common; this substructure will be called the "characteristic substructure" of that class. Second, no structure in one class can contain the characteristic substructure of any other class. The problem of finding such a classification scheme is represented diagrammatically in Figure 1. The object of the program to be described is to solve this "classification problem".

The definition of this problem makes it impossible for two different characteristic substructures to be present in any one of the candidate structures; i.e., the substructures are mutually exclusive. A consequence of this constraint is that the assignment of structures to their respective classes, on the basis of a given set of characteristic substructures, is unique: each structure is assigned to the one class whose characteristic substructure it contains. Thus, any solution to the classification problem is completely described by its characteristic substructures. The effort to solve this problem is essentially a search for these substructures.

It is important to understand how the chemist might use this classification scheme in practice. The ultimate goal in analyzing a set of characteristic substructures would be to determine, perhaps through new experiments, which of the mutually exclusive substructures is actually present in the unknown. If this goal can be met, all of the structure classes except one can be eliminated from further consideration. When this cannot be done, it may be possible to demonstrate that one or more of the substructures is not present in the unknown. If so, the entire structure class associated with each such substructure can be eliminated.

Information-Theoretical Criterion. It must be assumed that there may be more than one solution to the classification problem. Therefore, some criterion is needed for ranking the different solutions in terms of their potential usefulness to the chemist.

This can be treated quantitatively by using information theory. For m possible events that occur with probabilities $p_1, p_2, ..., p_m$, information theory defines the "entropy" as[11]

$$H(p_1, p_2, ..., p_m) = -K\sum_{i=1}^{m} p_i \ln p_i \qquad (1)$$

where K is some positive constant. H is a measure of the uncertainty regarding which event will occur. If $p_1 = 1$ and $p_2 = ... = p_m = 0$, there is complete certainty, and, accordingly, the entropy is zero. The entropy attains its maximum when all the events are equally likely:

$$H(1/m, 1/m, ..., 1/m) = K \ln m \qquad (2)$$

Indeed, this is intuitively the most uncertain situation. Another view of H is that it is a measure of information, i.e., the information that must be supplied before we know with certainty which event will occur.

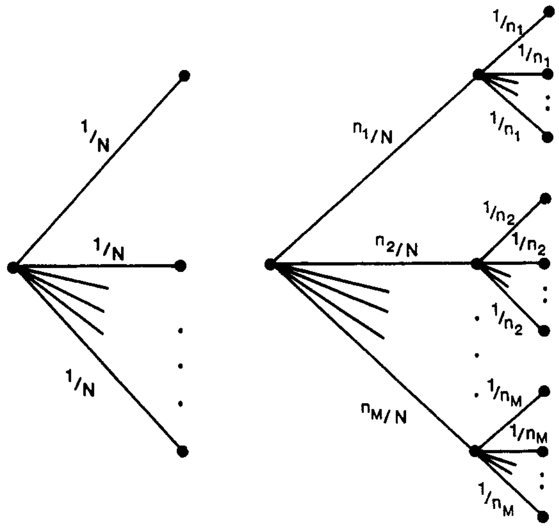


Figure 2. Two diagrams (of the type used by Shannon[11]) representing the problem of choosing from among N candidate structures having equal probabilities. The problem can be seen as (left) a choice from among N structures each with probability $1/N$ or (right) a two-step process in which the first step is a choice from among M structure classes having sizes $n_1, n_2, ..., n_M$.

Consider a structure elucidation problem in which there are N candidate structures, each of which is equally likely to be the unknown. The problem is to choose from among N possibilities each with probability $1/N$. This is diagrammed on the left in Figure 2. Suppose these N structures have been grouped into M classes containing $n_1, n_2, ..., n_M$ structures. The problem can now be decomposed into two consecutive choices: first, choose from among the M classes, which have probabilities $n_1/N, n_2/N, ..., n_M/N$; second, if the ith class was selected, choose from among n_i candidates each with probability $1/n_i$. This two-step process is diagrammed on the right in Figure 2.

The decomposition of the original problem does not change the entropy H (this is one of the conditions under which H is derived[11]). Each diagram in Figure 2 gives rise to a different expression for the total entropy, and these can be equated:

$$H(1/N, 1/N, ..., 1/N) = H(n_1/N, n_2/N, ..., n_M/N) + \sum_{i=1}^{M}(n_i/N)H(1/n_i, ..., 1/n_i) \qquad (3)$$

The two terms on the right-hand side correspond to the two-step process; the second term, corresponding to the second step, is a summation of entropies weighted by the probability of the first step. The sum of these two terms equals $K \ln N$, which is the total uncertainty associated with the structure problem. As the first term is made larger, by changing $n_1, n_2, ..., n_M$, the second term becomes smaller. The second term is the expectation value of the uncertainty remaining after the first step. Thus, the larger the first term is, the less uncertainty we expect there to be after choosing from among the M classes. We therefore take the function $U(n_1, ..., n_M) = H(n_1/N, ..., n_M/N)$ as a measure of the potential usefulness a set of structure classes has for the chemist (in calculating H from eq 1, K is set equal to 1).

This function U can be used to rank different solutions to the classification problem. The highest ranked of several solutions will be the one for which this function is greatest. The maximum of U is attained when

$$n_1 = n_2 = \ldots = n_M = N/M \qquad (4)$$

or when this condition is met as closely as possible if N/M is not an integer. The "best" solution for given M will therefore be the one whose structure classes are most nearly equal in size.

This information-theoretical criterion depends only upon the sizes of the structure classes; it does not take into account the nature of the characteristic substructures. Clearly, a more powerful criterion would be one based on the chemical and/or spectroscopic properties of the substructures themselves. However, the development of a computer program capable of such an analysis would constitute a formidable exercise in artificial intelligence, requiring a vast knowledge base in chemistry and spectroscopy. The program we describe can be regarded as a measure of the extent to which useful results can be produced without a knowledge base.

BASIS FOR A COMBINATORIAL APPROACH

This problem of classifying candidate structures is closely related to a well-known combinatorial problem. Thus, the computational approach used is based on methods from the field of combinatorics.

Set-Covering Problem. Suppose we have a set of objects, A, and a family of subsets of A. A cover of set A is any collection of these subsets whose union is A itself. The set-covering problem[12] is generally concerned with the enumeration of such covers subject to various restrictions. A common restriction is that each object in A be contained in only one of the subsets forming the cover, in which case the cover is said to be disjoint. (Henceforth, the term "cover" will imply "disjoint cover".)

The algorithm for finding set covers can be illustrated with the following simple problem. The binary table in Figure 3a describes four subsets of the set {1, 2, 3, 4}; the table denotes the presence of an object in a subset by a one. We wish to find all covers of this set among subsets A, B, C, and D.

The enumeration of set covers, like many combinatorial problems, can be conveniently represented by a "search tree".[13] Such a tree is arranged so that a thorough search of it will generate all solutions. A search tree for the problem just posed is shown in Figure 3b. Every node except the root corresponds to a subset. The immediate descendants of each node correspond to all of the subsets that are (alphabetically) greater than it; this will avoid the generation of covers that are simply permutations of one another.

The search tree is traversed by visiting its nodes in depth-first order.[13] Every time a previously unvisited node is reached, a trial solution to the covering problem is constructed and tested. It consists of all the subsets lying along the shortest path from the root down to and including the current node. In the tree of Figure 3b, only the trial solutions at the starred nodes—A, C and B, D—are found to be covers. If a trial solution contains two subsets that have an object in common, further searching deeper into the tree can yield only nondisjoint covers. This unproductive searching is avoided by backtracking from the current node. Backtracking also takes place when a cover has been found since the addition of another subset must lead to a nondisjoint cover. Those portions of the search tree avoided by backtracking are shown in dashed lines.

Classification Problem and Set Covering. It is relatively simple to formulate the classification problem so that it is equivalent to a set-covering problem. Consider a set S consisting of N candidate structures. This is the set to be covered. A family of subsets of S is needed from which covers can be

a

	1	2	3	4
A	1	0	0	1
B	0	0	1	1
C	0	1	1	0
D	1	1	0	0

b

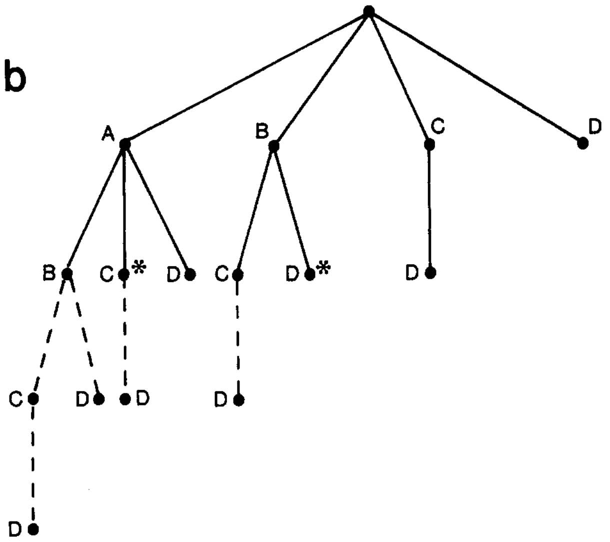


Figure 3. Simple problem in enumerating disjoint covers of the set {1, 2, 3, 4}: (a) Binary table describing the four subsets to be used. Subset A, for example, is {1, 4}. (b) Search tree for the problem. Stars indicate the nodes at which the two disjoint covers, A, C and B, D, are found. Dashed lines indicate branches that would be avoided by the search algorithm, as described in the text.

formed. Let {s} be the subset of S composed of all the structures in S that contain substructure s. Suppose $s_1, s_2, \ldots, s_R$ is the irredundant list of all substructures contained by any of the structures in S (obviously, this list is extremely long; i.e., $R \gg N$). The corresponding family of subsets is $\{s_1\}, \{s_2\}, \ldots, \{s_R\}$.

In principle, the classification problem can be solved exhaustively by finding all covers of S from this family of subsets. For example, suppose that the first M of these subsets form a cover. This means they satisfy two conditions

$$\{s_1\} \cup \{s_2\} \cup \ldots \cup \{s_M\} = S \qquad (5)$$

and

$$\{s_i\} \cap \{s_j\} = \{\} \qquad i \neq j \qquad (6)$$

where {} is the empty set. Equation 5 is the basic condition for set covering; eq 6 is the specific condition for disjointness. This cover clearly constitutes a solution to the classification problem: $\{s_1\}, \{s_2\}, \ldots, \{s_M\}$ are the structure classes, and $s_1, s_2, \ldots, s_M$ are their respective characteristic substructures, whose mutual exclusivity is ensured by eq 6. Finding all covers will yield all solutions to the problem.

Unfortunately, the computing time needed to search for all of these covers will be prohibitive in practice. Inspection of Figure 3b shows that the degree of branching in a search tree for set covering is governed by the number of subsets used. Thus, with a family of R subsets, viz., $\{s_1\}, \ldots, \{s_R\}$, a tree is created that is so highly branched that the outcome of any attempt to search it would almost certainly be a "combinatorial explosion".

The computational load can, however, be reduced to a practical level by introducing constraints that restrict the allowed characteristic substructures. The result, as described in the next section, is a way of inexhaustively solving the classification problem as a series of covering problems, each of which is relatively small in scope and easily solved. This is representative of a strategy often used in combinatorics: dividing an intractable problem into simpler subproblems.

A COMPUTER PROGRAM TO CLASSIFY CANDIDATE STRUCTURES

The goal of the program is to find as many characteristic substructures for a given set of candidates as the chemist requests; this amounts to assigning the candidates to a user-specified number of structure classes. The program divides this task into a series of covering problems that progressively increase the number of characteristic substructures. The program begins with an initial substructure common to all candidate structures in set S and continues until the desired number of substructures is reached. Whenever the number of substructures is increased, it is done by the simple mechanism of replacing one of them with two or more new substructures.

The problem of finding these new substructures can be posed as a problem in disjoint covering. Assume that $s_1, ..., s_m$ is a valid set of characteristic substructures; i.e., $\{s_1\}, ..., \{s_m\}$ form a cover of S. Suppose we wish to find several substructures with which to replace s_m. We can do so by finding substructures $s_{1'}, ..., s_{k'}$ satisfying the condition that $\{s_{1'}\}, ..., \{s_{k'}\}$ form a cover of $\{s_m\}$. This condition is sufficient to ensure that $\{s_1\}, ..., \{s_{m-1}\}, \{s_{1'}\}, ..., \{s_{k'}\}$ form a cover of S. As a result, $s_1, ..., s_{m-1}, s_{1'}, ..., s_{k'}$ is also a valid set of characteristic substructures. The number of substructures is thereby increased from m to $m + k - 1$, and the structure class associated with s_m is, in effect, divided into k smaller classes.

A covering problem like this one must be solved by the program whenever the number of characteristic substructures is to be increased. But such problems can be easily solved only if they are constrained in some way. A simple but effective constraint is to require that the substructures chosen as replacements are "expansions" of the substructure to be replaced. We define an expansion of a substructure as any new substructure formed from it by the addition of one bond. Because of this constraint, the replacement substructures will seem to have "grown" from the substructure they replace. In reference to this, the substructure selected from the current characteristic substructures as the one to be replaced will be called the seed and will be denoted as s_{seed}; the set $\{s_{seed}\}$ consists of all candidate structures containing the current seed.

In general terms, the goal of this constrained covering problem is to find covers of $\{s_{seed}\}$ among subsets $\{s_1\}, ..., \{s_r\}$, where substructures $s_1, ..., s_r$ represent all the expansions of the seed that are found in $\{s_{seed}\}$. These substructures can be generated by examining each candidate in $\{s_{seed}\}$, locating the seed, and adding to it, in every possible way, one adjoining bond, i.e., a bond that is not in the seed but does involve at least one seed atom. If the seed occurs more than once in a candidate, all its different locations are taken into account. Figure 4 illustrates the generation of one-bond expansions of a substructure within α-pinene. This example shows that the number of unique expansions may be less than the number of adjoining bonds. It also shows that adding a bond to a substructure does not necessarily add an atom; the added bond may instead close a ring.

When a cover of $\{s_{seed}\}$ is found, the corresponding substructures are used to replace the seed; e.g., if the cover is $\{s_{1'}\}, ..., \{s_{k'}\}$, then s_{seed} is replaced by substructures $s_{1'}, ..., s_{k'}$. This type of covering problem will be called simply a "cover search". The main effort of the program is to solve a series of cover searches, each time replacing the current seed with two or more expansions of itself.

Input and Output. The program must receive three items as input: a set of candidate structures for some unknown, the

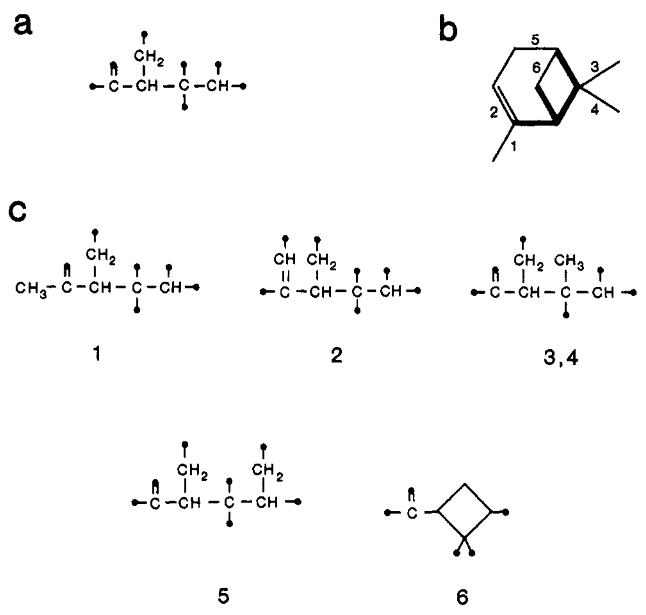


Figure 4. Expansions of a substructure: (a) Substructure whose expansions within α-pinene are to be obtained. (b) Structure of α-pinene. The location of the substructure is indicated in boldface, and its adjoining bonds are numbered 1–6. (c) The five possible expansions. The number below each substructure indicates which bond was added to the original substructure.

number of characteristic substructures (structure classes) desired, and a starting characteristic substructure, the initial seed. All subsequent characteristic substructures will contain the initial seed. It can be any substructure common to all of the candidates—thus, at the beginning of the program, $\{s_{seed}\}$ = S. Since one or more substructures deduced from the spectral and/or chemical behavior of the unknown compound generally serve as input to the structure generator producing the set of candidates, a common substructure can almost always be found.

The program returns as output one solution to the classification problem. It consists of the requested number of characteristic substructures as well as identification numbers of the candidates in the respective structure classes. The solution returned is the best—in an information-theoretical sense—the program can find, viz., the one for which the function U is greatest.

Bit-Level Operations. The method used to solve the cover searches is the algorithm for disjoint covering, which has already been discussed. This algorithm can be implemented so as to take advantage of the speed of certain bit-level operations. This requires that the problem be given a particular representation in the computer. Each subset of candidates is represented in binary fashion (as in Figure 3a) by bit strings of length N, where N is the number of candidates. If a subset contains candidate i, then bit i of the corresponding string is a 1.

Given the problem of covering $\{s_{seed}\}$, bit-string representations must be created for $\{s_1\}, ..., \{s_r\}$, where $s_1, ..., s_r$ are the expansions of the seed. To do this, the candidates in $\{s_{seed}\}$ are searched, and all expansions of the seed are obtained. As each expansion is found, it is put into a canonical connection-table representation[14] and compared to an irredundant list of the previously found expansions. If it does not match any of these, it is added to the list and is associated with a new bit string in which bit j is a 1 (assuming the expansion was found in candidate j) and the other $N - 1$ bits are zeros. If the expansion does match one on the list, the bit string associated with it is updated by changing bit j to a 1. In this manner, each expansion of the seed becomes associated with a bit string representing the corresponding subset of S.

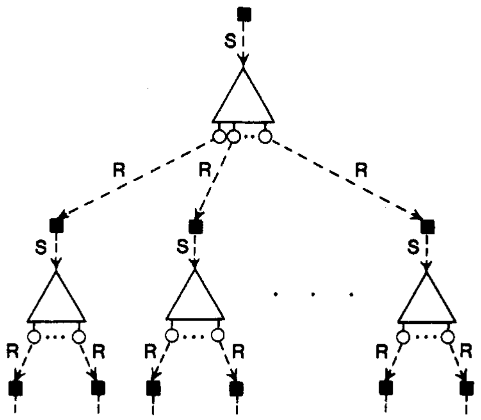


Figure 5. Tree illustrating the program. Each square represents a set of characteristic substructures. Each triangle represents a cover search. Arrows marked "S" denote the selection of a new seed from the current set of substructures. Arrows marked "R" denote the replacement of the current seed to yield a new, larger set of characteristic substructures. See text for further discussion.

The union and intersection of subsets from the family $\{s_1\}$, ..., $\{s_r\}$ must be determined repeatedly in performing a cover search. A rapid means of doing so is provided by logical operations on the associated bit strings. The bit-level logical operations used are OR and AND. A collection of subsets satisfies the condition for covering $\{s_{seed}\}$ (cf. eq 5) if ORing together all their bit strings yields the bit-string representation of $\{s_{seed}\}$. It also satisfies the condition for disjointness (cf. eq 6) if ANDing any two of the bit strings yields a string of N zeros. By use of these operations, the algorithm rapidly finds all the covers of $\{s_{seed}\}$ that can be formed from the above family of subsets.

Tree Representation for the Program. The operation of the program can be illustrated by the tree shown in Figure 5. Each square represents a set of characteristic substructures; the topmost square represents the set consisting only of the initial seed. As one proceeds deeper into the tree, the squares encountered represent progressively larger sets of substructures. The arrow from each square marked S symbolizes the process by which a new cover search is initiated. This involves selecting a new seed from the current set of substructures and finding and storing, as described earlier, all the expansions of the seed and their corresponding subsets of candidates.

The program incorporates a simple heuristic for selecting a new seed from among the current characteristic substructures: the substructure chosen is the one associated with the largest structure class. Dividing the largest class will on average result in a size distribution closer to overall equality than dividing any of the other, smaller classes and will therefore produce a "better" solution.

Each triangle in Figure 5 represents the search tree (e.g., Figure 3b) for a cover search. Each attached circle represents a node at which a cover of $\{s_{seed}\}$ is found; more than one cover may exist. No criterion has been established for selecting one cover in preference to the others; therefore, the consequences of using each are explored. The arrow from each circle marked R symbolizes the process of creating a larger set of substructures from the previous set. This process is simply the replacement of the current seed as determined by the cover of $\{s_{seed}\}$ that has been found.

The operation of the program is essentially a search of this tree. The rules guiding the search depend upon whether or not the user-requested number of characteristic substructures has yet been attained. Every time a new set of substructures is created, the number of substructures, m, is compared with the requested number, M. The outcome of this comparison determines the next steps taken by the program. There are three possibilities to consider:

(i) If $m < M$, there are too few substructures, so a deeper level of the tree must be reached. The program initiates a new cover search, searches until the first (leftmost) cover is found, and uses this cover to create a larger set of substructures. If the search ends without finding any cover, the program must backtrack, i.e., go to a higher level of the tree and explore a new branch. To backtrack, the program returns to the most recently visited node of the previous cover search. Starting from this node, which represents a used cover, the search is resumed in order to find the next unused cover. If this search ends without finding another cover, the program continues to backtrack. The program ends when the topmost cover search finds no more covers.

(ii) If $m > M$, there are too many substructures. The program backtracks as described above.

(iii) If $m = M$, the program stores the current set of characteristic substructures if no previous set has been stored. Otherwise, if the value of $U(n_1, ..., n_M)$ for the current set is greater than that for the stored set, the program replaces the stored set with the current set. The identification numbers of the candidates in the structure classes are also stored. The program then backtracks. The set of M substructures stored at the end of the program is returned to the user as the best solution since it has the largest value of U encountered.

It is not necessary to test this final set of substructures for mutual exclusivity. This property, which is obeyed by any set of substructures produced by the topmost cover search (due to the disjointness condition), is automatically preserved at all lower levels of the tree. The reason it is preserved is that any structure class not containing a particular substructure can never contain any expansion of it, and any substructure not present in a particular class can never be present in any smaller class resulting from its division. Thus, the characteristic substructures in each set encountered in the tree are always mutually exclusive.

Use of the Branch-and-Bound Technique. The above rules for searching the tree in Figure 5 can be made more efficient by incorporating a test to recognize unproductive branches of the tree. The basis for this test is the fact that the size distribution of a set of structure classes constrains the size distribution of any set of classes derived from it by dividing some or all of its classes. For example, from three classes whose size distribution is (75, 15, 10), one could derive (25, 25, 25, 15, 10) but could not derive the more equal distribution (25, 20, 20, 20, 15). With a method of determining the most equal size distribution of a set of M classes that can be derived from a given set of m ($m < M$) classes, the program can, while searching the tree, put an upper bound on the value of U that can be attained among the solutions derived from the current set of m substructures. If this bound does not exceed the value associated with the currently stored set of M substructures, there is no reason to search for these solutions, and the program can immediately backtrack. This strategy is that of the so-called branch-and-bound technique.[13]

Using this technique in the present problem requires determining how a given set of m structure classes of sizes n_1, n_2, ..., n_m can be divided into a total of M classes so that U is maximized. To describe this division of classes, let d_i be the number of classes into which the ith class is divided. It is assumed that each of the m classes is divided equally; i.e., the ith class is divided into d_i classes each containing n_i/d_i candidates. The fact that such an equal division cannot actually be achieved when n_i/d_i is noninteger is irrelevant to this analysis; acceptance of possible noninteger values, for reasons of simplicity, means only that the upper bound on U may be somewhat larger than necessary. The value of U for these M classes is given as

$$U = -\sum_{i=1}^{m} \frac{n_i}{N} \ln \frac{n_i}{Nd_i} \qquad (7)$$

To obtain a bound on U, integer values for $d_1, d_2, ..., d_m$ must be found that maximize eq 7.

A simple algorithm for this purpose begins by arbitrarily choosing values for $d_1, d_2, ..., d_m$ such that $n_i \geq d_i \geq 1$ and $d_1 + ... + d_m = M$. Each pair of these numbers is examined to determine whether U can be increased by either of two types of incremental change. The pair d_i, d_j is changed to $d_i + 1$, $d_j - 1$ if

$$n_i \ln (1 + 1/d_i) + n_j \ln (1 - 1/d_j) > 0 \qquad (8)$$

It is easily shown from eq 7 that this change results in an increase in U if eq 8 holds. On the other hand, d_i, d_j is changed to $d_i - 1$, $d_j + 1$ if this change increases U, i.e., if

$$n_i \ln (1 - 1/d_i) + n_j \ln (1 + 1/d_j) > 0 \qquad (9)$$

If neither of these conditions is satisfied, no change is made. (Note that eq 8 and 9 cannot both be true; if they were, adding them would give

$$n_i \ln (1 - 1/d_i^2) + n_j \ln (1 - 1/d_j^2) > 0 \qquad (10)$$

which is false since the logarithm of a number less than one is negative.) All pairs are examined again if at least one change has been made. This continues until one complete pass through all the pairs results in no further change, at which time the algorithm stops. The final values for $d_1, ..., d_m$ substituted into eq 7 yield the upper bound, B.

As an example, consider a problem in which 100 candidates are to be put into six structure classes. Assume that the current partial solution in the tree search consists of three classes with sizes 65, 30, and 5. The values for d_1, d_2, and d_3 found by the algorithm are 3, 2, and 1, respectively, and thus $B = 1.713$. If the currently stored solution is (25, 20, 20, 15, 15, 5), for which $U = 1.709$, this partial solution will not be rejected because $B > U$. However, if the stored solution is (20, 20, 20, 20, 15, 5), for which $U = 1.722$, it will be rejected because $U > B$. It is clear that the usefulness of this technique for avoiding branches of the tree depends upon how early in the search a solution with relatively large U can be found.

Maximizing the Substructures. As defined, a characteristic substructure is common to all the members of its structure class, but this does not mean it is a "maximal" common substructure. It is maximal only if it cannot be expanded further and still qualify as a common substructure. If it is nonmaximal, on the other hand, it can be expanded into a larger substructure that is also common to all members of the class. The treatment of nonmaximal characteristic substructures is an important feature of the program.

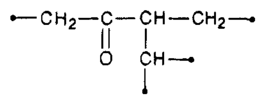
Whenever a substructure is selected as the seed, it is expanded until it is maximal if it was not already. Only then does the program initiate the appropriate cover search. The reason this is advantageous is that there are always bonds adjoining a nonmaximal substructure that are common to all the candidates in its class. These bonds obviously cannot be used to differentiate the candidates. But in the process of expanding to a maximal substructure, these bonds are superseded by new adjoining bonds that can be used to differentiate the candidates. It is therefore reasonable to expect the search for covers of $\{s_{seed}\}$ to be more successful if the seed is maximal.

Another aspect of maximizing substructures concerns those characteristic substructures the program is to return as output. It is preferable for these substructures to be maximal with respect to their structure classes. Otherwise, the chemist will receive less structural information than is actually present. The program has thus been designed to process further all the substructures in storage at the finish. Each is expanded until it is maximal with respect to its class of candidates and is returned as output in that form.

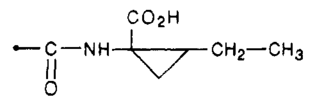
EXAMPLE AND DISCUSSION

Applying the program just described to a real-world structure elucidation problem illustrates many aspects of the program's behavior and demonstrates its potential as well as its limitations.

Candidate Structures for Coronatine. The structure of coronatine ($C_{18}H_{25}O_4N$), a phytotoxic microbial product, was originally determined from spectroscopic, chemical, and X-ray analysis of it and its derivatives;[15] computer-assisted methods were not employed. A set of 57 plausible candidates (Figure 6) for coronatine was generated by ASSEMBLE when the following published structural inferences[15] were submitted to it: (1) the presence of the ketone-containing fragment



in which the carbonyl group is part of a five-membered ring but not a three- or four-membered ring; (2) the presence of the amide-containing fragment



in which the amide carbonyl group is conjugated to a carbon–carbon double bond that is not part of a three-, four-, or five-membered ring; (3) the presence of a total of two ethyl groups and no methyl groups other than those of the ethyl groups; (4) the presence of only one vinylic hydrogen; and (5) the absence of an allenic linkage. Among these candidates is the actual structure reported[15] for coronatine (Figure 6, structure 4).

Classification Results and Discussion. Using the ketone fragment as the initial seed and requesting two structure classes, the program returns the output shown in Figure 7. Inspection of characteristic substructures I and II reveals that they have been derived by adding to the initial seed, respectively, a methine carbon (>CH–) and a methylene carbon (–CH_2–). Substructure II has been expanded further, into a ring, to make it maximal with respect to its structure class. The disparity between the sizes of the two classes is great, but this is evidently the best distribution the program could find given the inherent constraints of the problem. Although less probable, the actual structure of coronatine belongs to the smaller class. Because of this, the disparity in class sizes is potentially advantageous: if the chemist can successfully identify either the presence of substructure II in the unknown or the absence of substructure I, then 50 of the 57 candidates can be eliminated. This example does not, however, refute the argument in favor of an equal distribution of class sizes. That argument is essentially statistical and must be evaluated over many examples.

When the program is requested to group the candidates into three, four, or five structure classes—still using the ketone fragment as the initial seed—it is unable to do so. It is not surprising to find that the program cannot always solve the classification problem. The only solutions may be in regions of "solution space" not explored by the program because at this stage of development it tries to divide only the largest of the current classes. On the other hand, there may be no solutions at all because the constraints of the problem are impossible to satisfy. In this regard, it is instructive to consider why no larger solution can be derived from the one in Figure

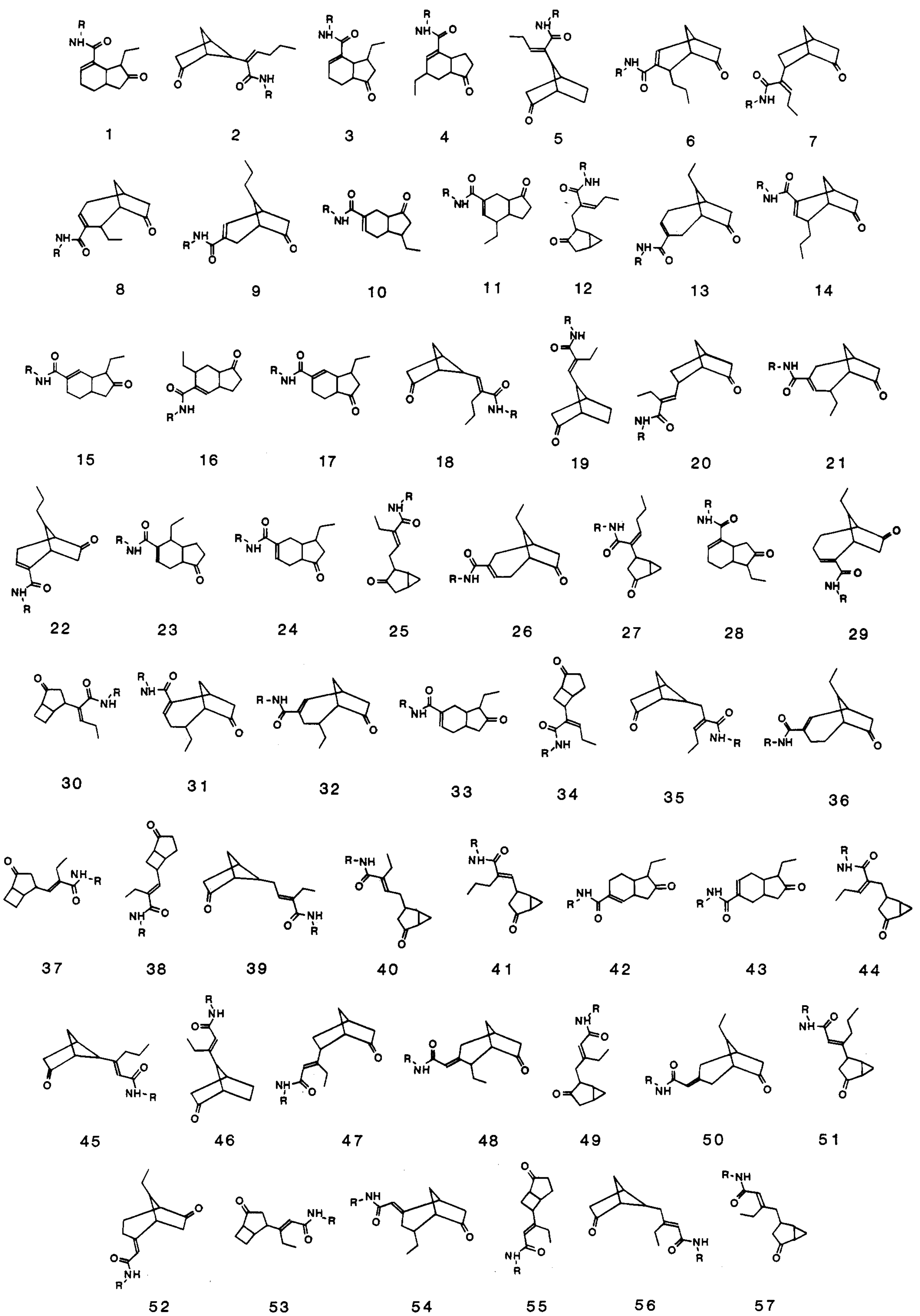
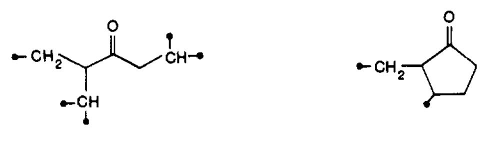


Figure 6. Candidate structures for coronatine generated by ASSEMBLE using the input data described in the text. Candidate identification numbers were assigned by ASSEMBLE. The actual structure of coronatine is number 4. R group: 1-(1-carboxyl-2-ethyl)cyclopropyl residue.

CHARACTERISTIC SUBSTRUCTURES:



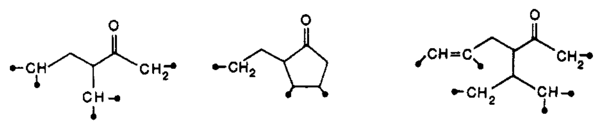
I II

STRUCTURE CLASSES:

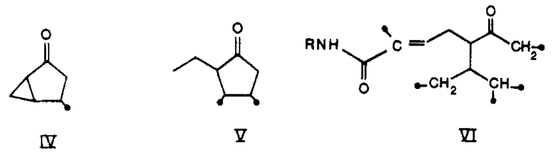
	candidate numbers
I (50 candidates):	1, 2, 3, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 35, 36, 37, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57
II (7 candidates):	4, 11, 16, 23, 34, 38, 55

Figure 7. Classification of the coronatine candidates into two structure classes using the ketone fragment.

CHARACTERISTIC SUBSTRUCTURES:



I II III



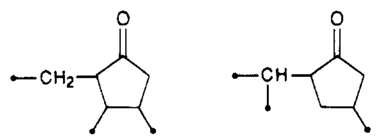
IV V VI

STRUCTURE CLASSES:

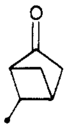
	candidate numbers
I (22 candidates):	2, 4, 6, 7, 8, 14, 16, 18, 20, 21, 31, 32, 34, 35, 38, 39, 45, 47, 48, 54, 55, 56
II (11 candidates):	3, 5, 17, 19, 29, 30, 36, 37, 46, 52, 53
III (7 candidates):	9, 10, 11, 12, 13, 49, 50
IV (6 candidates):	27, 40, 41, 44, 51, 57
V (6 candidates):	1, 15, 28, 33, 42, 43
VI (5 candidates):	22, 23, 24, 25, 26

Figure 8. Classification of the coronatine candidates into six structure classes using the ketone fragment.

7. A five-membered ring, which is required, can be formed from substructure I in either of two ways:

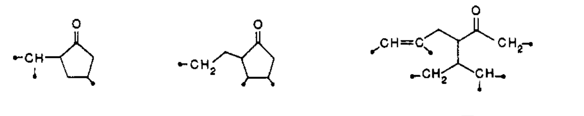


It may appear that replacing substructure I with these two substructures should yield a solution for three classes. However, the above substructures are not mutually exclusive; they are both present in each of the six candidates (numbers 2, 18, 35, 39, 45, and 56 in Figure 6) containing the bicyclic moiety

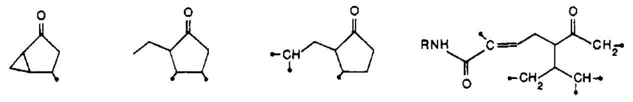


These six candidates should really be put into their own class, but the program has no way of doing this at this time.

CHARACTERISTIC SUBSTRUCTURES:



I II III



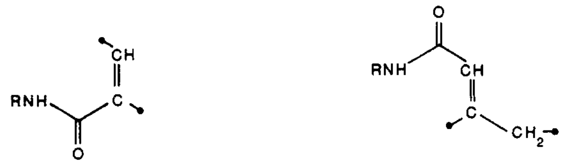
IV V VI VII

STRUCTURE CLASSES:

	candidate numbers
I (17 candidates):	2, 6, 7, 8, 14, 18, 20, 21, 31, 32, 35, 39, 45, 47, 48, 54, 56
II (11 candidates):	3, 5, 17, 19, 29, 30, 36, 37, 46, 52, 53
III (7 candidates):	9, 10, 11, 12, 13, 49, 50
IV (6 candidates):	27, 40, 41, 44, 51, 57
V (6 candidates):	1, 15, 28, 33, 42, 43
VI (5 candidates):	4, 16, 34, 38, 55
VII (5 candidates):	22, 23, 24, 25, 26

Figure 9. Classification of the coronatine candidates into seven structure classes using the ketone fragment.

CHARACTERISTIC SUBSTRUCTURES:



I II

STRUCTURE CLASSES:

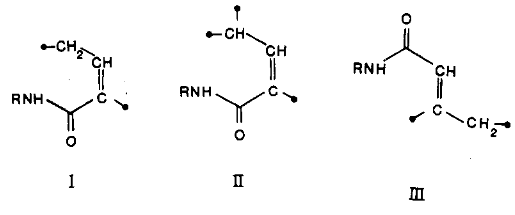
	candidate numbers
I (44 candidates):	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44
II (13 candidates):	45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57

Figure 10. Classification of the coronatine candidates into two structure classes using the amide fragment.

The search for solutions is again successful when six classes are requested. The solution found, shown in Figure 8, is unrelated to the solution for two classes. Exactly how substructures I–VI originate with respect to the ketone fragment is not easy to perceive because of their expansion into maximal substructures. Close inspection shows that all of them have been derived from the initial seed by adding a bond to the methylene carbon β to the carbonyl. This is most easily seen in substructure I.

The output of the program for seven structure classes is shown in Figure 9. This result is clearly an extension of the previous one for six classes. This means that the path by which it was found led first to the solution for six classes, which was then extended by dividing the largest class (I in Figure 8) into two smaller classes (I and VI in Figure 9). Of the seven characteristic substructures, five contain rings. The development of larger, more specific substructural features, such as rings, is one reason a large number of classes may be more helpful than a small number. Substructures I, II, and IV–VI

CHARACTERISTIC SUBSTRUCTURES:



STRUCTURE CLASSES:

	candidate numbers
I (26 candidates):	1, 2, 3, 5, 7, 8, 10, 12, 13, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 34, 35, 39, 40, 43, 44
II (18 candidates):	4, 6, 9, 11, 14, 15, 16, 17, 18, 19, 20, 21, 32, 36, 37, 38, 41, 42
III (13 candidates):	45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57

Figure 11. Classification of the coronatine candidates into three structure classes using the amide fragment.

can be viewed as a sampling of the ring environments of the ketone carbonyl. This sampling is incomplete since substructures III and VII do not contain rings. Nevertheless, to get similar information with a structure-editing program would be a tedious job, requiring a lengthy series of queries.

The solutions obtained for two, three, and four structure classes using the amide fragment as the initial seed are presented in Figures 10–12. It can be seen that each solution is simply an extension of the previous one. These results are especially interesting in that the vinylic proton appears in every characteristic substructure. The NMR properties of this proton could provide a spectroscopic basis for determining which one of the substructures is present in the unknown. Consider the four substructures in Figure 12. If the vinylic proton is found to be α to the carbonyl, substructure III is the only compatible choice. If it is at the β position and vicinally coupled to exactly one proton, substructure I must be chosen. The two remaining substructures differ in the number of protons to which the vinylic proton is allylically coupled: one (substructure II) or two (substructure IV). If it can be correctly determined that substructure I is present in the unknown, the number of valid candidates will be narrowed to those in class I; this is less than one-third the original number even though class I is the largest class. This is an example of a fairly equal distribution of class sizes.

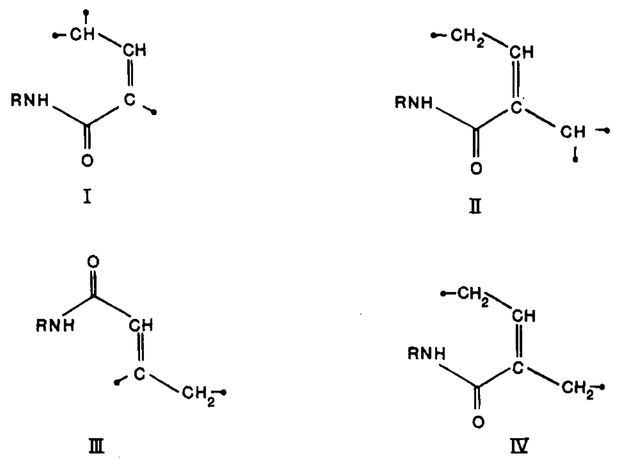
The fact that the characteristic substructures in Figure 12 have obvious experimental implications cannot be credited to the program since it operates without any chemical knowledge base. This example does suggest that the creation of structurally meaningful classes can lead to classes that are spectroscopically meaningful as well. Even when the substructures obtained do not clearly suggest new experiments, as in Figure 9, they still provide potentially useful information.

CONCLUSIONS

The program described in this paper addresses a previously neglected problem in computer-assisted structure elucidation: automating the analysis of a large number of candidate structures. The present work represents only a "first-generation" effort in this direction. Many of the choices made in developing the program were intended to keep it as simple as possible. At the expense of making the program somewhat more complex, there are a number of ways its current level of performance could be improved.

One way of giving the chemist more control over the nature of the output is for the program to preprocess the candidate

CHARACTERISTIC SUBSTRUCTURES:



STRUCTURE CLASSES:

	candidate numbers
I (18 candidates):	4, 6, 9, 11, 14, 15, 16, 17, 18, 19, 20, 21, 32, 36, 37, 38, 41, 42
II (14 candidates):	1, 2, 3, 5, 7, 8, 22, 23, 27, 28, 29, 30, 31, 34
III (13 candidates):	45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57
IV (12 candidates):	10, 12, 13, 24, 25, 26, 33, 35, 39, 40, 43, 44

Figure 12. Classification of the coronatine candidates into four structure classes using the amide fragment.

structures according to a user-created definition of similarity. This working definition would specify the extent to which certain distinguishing features of atoms, such as element type, hybridization, or ring membership, are to be recognized. For example, changing the element type of every heteroatom to X would prevent the program from distinguishing between heteroatoms on the basis of their element types while still allowing it to distinguish heteroatoms from carbon atoms. This preprocessing approach has already been used successfully in a program for finding common substructures among a given set of structures.[16] The goal of the preprocessing would be to emphasize features in the candidates that are experimentally important and suppress those that are unimportant.

As demonstrated in the coronatine example, the program is sometimes unable to find a solution for the given input. A particularly simple example of such an impasse is the following. Substructure s_0, which has two free valences, is the initial seed and appears in each candidate in one of three environments: C–s_0–N, C–s_0–O, or N–s_0–O. There are thus three expansions of s_0, viz., s_0–C, s_0–N, and s_0–O. Any two of these expansions are both present in some of the candidates, so an impasse is reached because no choice of mutually exclusive expansions is possible. This impasse arises because the program expands substructures by only one bond at a time. If the program would be modified to consider expansions formed by adding two bonds at a time, most impasses would likely be avoided, although at the cost of increased computational time.

Judging from the examples presented, the program in its present form already shows an impressive ability to organize a large set of candidate structures. It could be easily integrated into a system for computer-assisted structure elucidation, like CASE,[6] where its role would be to aid in the recognition of significant differences between the candidate structures and possibly facilitate the planning of new experiments. The program operates by solving a strictly defined combinatorial problem and leaves to the chemist the more difficult task of knowledgeably interpreting the results. Thus, the design of this program follows the general aim of computer-assisted structure elucidation: to use certain narrow, but powerful, capabilities of the computer so that the expert knowledge of the chemist can be applied more effectively.

ACKNOWLEDGMENT

The financial support of this work by the National Institutes of Health and The Upjohn Co. is gratefully acknowledged.

REFERENCES AND NOTES

(1) Present address: National Center for Biomedical Infrared Spectroscopy, Battelle—Columbus Laboratories, 505 King Avenue, Columbus, OH 43201.
(2) Munk, M. E.; Shelley, C. A.; Woodruff, H. B.; Trulson, M. O. "Computer-Assisted Structure Elucidation". *Fresenius' Z. Anal. Chem.* **1982**, *313*, 473–479.
(3) Shelley, C. A.; Hays, T. R.; Roman, R. V.; Munk, M. E. "An Approach to Automated Partial Structure Expansion". *Anal. Chim. Acta* **1978**, *103*, 121–132.
(4) Carhart, R. E.; Smith, D. H.; Brown, H.; Djerassi, C. "Applications of Artificial Intelligence for Chemical Inference. 17. An Approach to Computer-Assisted Elucidation of Molecular Structure". *J. Am. Chem. Soc.* **1975**, *97*, 5755–5762.
(5) Carhart, R. E.; Smith, D. H.; Gray, N. A. B.; Nourse, J. G.; Djerassi, C. "GENOA: A Computer Program for Structure Elucidation Utilizing Overlapping and Alternative Substructures". *J. Org. Chem.* **1981**, *46*, 1708–1718.
(6) Munk, M. E.; Farkas, M.; Lipkus, A. H.; Christie, B. D. "Computer-Assisted Chemical Structure Analysis". *Mikrochim. Acta* **1986** II, 199–215.
(7) Kudo, Y.; Sasaki, S. "Principle for Exhaustive Enumeration of Unique Structures Consistent with Structural Information". *J. Chem. Inf. Comput. Sci.* **1976**, *16*, 43–49.
(8) Lipkus, A. H.; Munk, M. E. "Combinatorial Problems in Computer-Assisted Structural Interpretation of Carbon-13 NMR Spectra". *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 38–45.
(9) Shelley, C. A.; Munk, M. E. "CASE, A Computer Model of the Structure Elucidation Process". *Anal. Chim. Acta* **1981**, *133*, 507–516.
(10) Lindsay, R. K.; Buchanan, B. G.; Feigenbaum, E. A.; Lederberg, J. *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project*; McGraw-Hill: New York, 1980; pp 65–66.
(11) Shannon, C. E.; Weaver, W. *The Mathematical Theory of Communication*; University of Illinois: Urbana, 1949; pp 48–52.
(12) Wells, M. B. *Elements of Combinatorial Computing*; Pergamon: New York, 1971; Chapter 6.
(13) Ibid., Chapter 4.
(14) 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.
(15) Ichihara, A.; Shiraishi, K.; Sato, H.; Sakamura, S.; Nishiyama, K.; Sakai, R.; Furusaki, A.; Matsumoto, T. "The Structure of Coronatine". *J. Am. Chem. Soc.* **1977**, *99*, 636–637.
(16) Varkony, T. H.; Shiloach, Y.; Smith, D. H. "Computer-Assisted Examination of Chemical Compounds for Structural Similarities". *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 104–111.

Further Development of Structure Generation in the Automated Structure Elucidation System CHEMICS

KIMITO FUNATSU, NOBUYOSHI MIYABAYASHI, and SHIN-ICHI SASAKI*

Laboratory for Chemical Information Science, Toyohashi University of Technology, Tempaku, Toyohashi 440, Japan

Received April 20, 1987

The automated structure elucidation system CHEMICS has been expanded to allow the compound with all or any of the elements C, H, O, N, S, and the halogens to be analyzed. To realize the improvement, a new set of primary, secondary, and tertiary components to be used commonly for both spectral analyses and structure construction was established. Twelve attributes of the components were prepared for the determination of the bonding priorities between the components. This makes it possible to construct a tertiary component by giving an attribute of bonding partner to a secondary component. A new structure generator for the improved system has been developed on the basis of the established components set, using the authors' connectivity stack method. The generator was endowed with a function that takes information about the macrocomponent (substructure to be present or absent designated by a user) into the system; candidate structures with or without macrocomponent are generated in accordance with the information.

INTRODUCTION

We have already reported in a series of publications that CHEMICS, an automated structure elucidation system, can serve for analyzing the structures of organic compounds consisting of various combinations of C, H, and O atoms.[1] One of the most important problems to be solved in the study of automated structure elucidation is how to generate appropriate structure on the basis of given information. To solve the problem, CONGEN introduced a peculiar method based on the graph theory.[2] CASE coped with it through a modified canonicalization method of the connection table.[3] In the authors' CHEMICS, the connectivity stack method in which the connectivity matrix is differently expressed is used.[4] In the present study, we make an attempt to expand the system to allow nine atomic species, including C, H, O, N, S, and the halogens, to be analyzed. To realize this, it is necessary to modify the components by which a full structure is generated, previously stored in a computer, and to improve the generator, the core of CHEMICS.

The practical efficiency of structure elucidation work would be greatly increased if a user can use, in addition to spectral data, substructural data (referred to as "macrocomponent" in terms of CHEMICS) obtained from other sources.[5] In the system proposed in this paper, processing of macrocomponents can be incorporated into the structure generation algorithm by coordinating them closely with each other. Furthermore, macrocomponents desired to be incorporated into the candidate structures of an unknown compound and those desired to be excluded can be separately processed or treated.

ESTABLISHMENT OF COMPONENTS

We have previously reported that CHEMICS requires three hierarchical classes of components, i.e., primary, secondary, and tertiary. The same classification is also adopted in this