capable of detecting predicted solutions for errors created by the discrete time model and correcting them.

ACKNOWLEDGMENT

We thank Dr. J. C. Marshall and W. A. Schlieper for their valuable comments and criticisms. We thank Lenny Holmes for supplying the enzyme and NAD. This research was supported by the National Science Foundation under Grant CHE-8415295.

LITERATURE CITED

(1) French, S. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*; Wiley: New York, 1982.
(2) Cooper, R. B. *Introduction to Queuing Theory*; North-Holland: New York, 1981.
(3) Phillips, D. T. *Fundamentals of Network Analysis*; Prentice-Hall: Englewood Cliffs, NJ, 1981.
(4) Harrington, P. B.; Isenhour, T. L. "An Expert System for Temporal Optimization of Robotic Procedures". Presented at the 1986 Pittsburgh Conference, Atlantic City, March, 1986.
(5) Harrington, P. B.; Isenhour, T. L. "Expert Strategies for the Temporal Optimization of Robotic Procedures". Presented at the 1987 Pittsburgh Conference, Atlantic City, March, 1987.
(6) Isenhour, T. L. "Robotics in the Laboratory". *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 292.
(7) Schlieper, W. A.; Isenhour, T. L.; Marshall, J. C. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 159.
(8) Horowitz, E.; Sahni, S. *Fundamentals of Data Structures*; Computer Science: Taft, CT, 1976.
(9) Nilsson, N. J. *Principles of Artifical Intelligence*; Tioga: Palo Alto, CA, 1980.
(10) Pearl, J. *Heuristics*; Addison-Wesley: Reading, MA, 1984.
(11) Colowick, S. P., Kaplan, N. O., Eds. *Methods in Enzymology*; Academic: New York, 1955; p 495.

An Efficient Graph Approach to Matching Chemical Structures

O. OWOLABI

Department of Computer Science, University of Strathclyde, 26 Richmond Street, Glasgow G1 1XH, U.K.

Received February 19, 1988

A matrix minimization method for the generation of canonical orderings for the nodes of a graph is outlined, with particular emphasis on its application in matching chemical structures. By use of both the connectedness and the attributes of each node in the representative graph, the modified adjacency matrix is reduced to a canonical form. This canonical matrix is unique for each distinct compound represented. For storage and comparison, a canonical linear string is generated for each compound. All known structures are compiled only once in this efficient manner, and the string representation also permits best-match retrieval. Examples are given to illustrate the procedure.

INTRODUCTION

One of the most widespread uses of computers in chemical information systems is in the matching of chemical structures. Looking for the name and other information associated with a particular structure involves the matching of the structure against a library of known structures.

Due to the nature of chemical compounds, it is natural to map them onto the graph structure for convenience of manipulation. One of the main advantages of representing a complex structure such as a chemical compound by the graph is that the description can be in terms of the constituting primitives of the compound. In this case the primitives are the atoms present in the compounds of interest.

The matching of two graphs, however, is not a trivial problem. The most straightforward way to solve this problem is to conduct a node-to-node matching. In this scheme, all the possible mappings between the two graphs are generated. Each is then evaluated to determine whether it represents a legal mapping between the two compounds or not. This process will yield $n!$ possible mappings between two graphs with n nodes each. This is a computationally expensive procedure.

To overcome the limitation imposed by this naive approach, various alternatives have been proposed. The most obvious one is to combine the straightforward exhaustive approach with backtracking. With backtracking, a check is made every time a new node is added to a possible solution. Once a failure is encountered, that particular mapping is abandoned, and a new one is generated. In this way, all the possibilities are not exhaustively searched. This can be further improved by mapping only between nodes of the same type.

This is the basis of the node partitioning strategy employed by Unger[1] and Sussenguth.[2] To compare two graphs, the node set is partitioned into equivalence classes according to various node properties. These properties include node type, node degree, branch type, etc. Assignments are made between equivalent classes in the two compounds. New partitions are then generated. These two steps are iterated until either all the nodes have been matched or it is established that the compounds do not match.

There are $n!$ ways of ordering the nodes of a single n-node graph. Thus, the major problem in graph matching is to determine whether any of these orderings of a candidate graph are equivalent to the stored model graph. This consideration has given rise to the idea of seeking what amounts to the canonical ordering of the graphs to be matched. In this approach, only the canonical forms of the graphs being considered are compared. To check a database for an unknown structure, the search structure is reduced to its canonical form. This is then compared against the canonical forms of the stored structures to see whether there is a matching compound in storage.

Some of the earlier works in the generation of canonical orderings for graphs were carried out by Scoins[3] and Nagle.[4] The approach has been further explored by Randić.[5] The basic method is to represent an n-node graph by an $n \times n$ connectivity matrix. Each row is taken as a decimal number. The rows and columns of this matrix are then rearranged until the rows are sorted in ascending order. This is done by starting

0095-2338/88/1628-0221$01.50/0

from the top of the matrix and exchanging each row with the one below it if it has a greater numerical value. Where two rows are exchanged, the corresponding columns are also exchanged. This is done until no more exchanges take place. Hendrickson and Toczko[6] employed a similar method but sought rather to maximize the connectivity matrix.

Using the simple connectivity matrix gives no indication of the node and connection types that are present in the graph. It is also not easy to constrain the process of row and column exchanges.[5,7] According to Mackay[7] this method will sometimes fail to yield an absolute minimum for a given graph configuration. It is also possible for the process to enter a cycle. These two conditions are due to the fact that an exchange of columns following an exchange of rows might require that rows already arranged be rearranged; i.e., exchanging two columns can change the index number of the rows involved.

The method employed by Wipke and Dyott,[8] which is based on the method of Morgan,[9] involves the use of extended connectivity. In addition to using the extended connectivity of the atoms, which is the number of its non-hydrogen connectivity, it also uses the stereochemical properties of the structure.

Another approach to the generation of canonical names for chemical structures is that of Jochum and Gastaiger,[10] which employs an invariant—called the number of the outermost occupied neighbor (NOON)—based on the paths and distances between the atoms. Properties used to discriminate between atoms include atomic number, number of free electrons, and connectivity.

The technique presented here is based on the graph minimization approach to canonical ordering and attempts to overcome the problems associated with the basic method. Ordering the rows of the adjacency matrix by the number of entries rather than by the size of the index number will result in a method that terminates because column exchanges following row exchanges cannot affect the ordering of the rows that are already arranged; i.e., the number of entries in every row will always remain the same.

The node properties are also employed to select which row should precede the other in the rearrangement process. It is thus possible to more easily discriminate between any two nodes. Finally, while previous methods do not reflect the link types in the adjacency matrix, this method employs a unique identifier for every link type in the canonical string that is generated.

DEFINITIONS

A graph $G = (V, E)$ is a structure that consists of a set of vertices (also known as nodes) $V = \{v_1, v_2, ...\}$ and a set of edges (also known as arcs) $E = \{e_1, e_2, ...\}$. Each edge e is incident to the elements of a pair of vertices $\{u,v\}$, which are not necessarily distinct; i.e., each edge connects two vertices.

A graph is usually represented by a connectivity or adjacency matrix. For a graph with n nodes, the connectivity matrix is an $n \times n$ matrix such that

$$M_{ij} = 1 \quad \text{if node } i \text{ is incident node } j$$
$$= 0 \quad \text{otherwise}$$

A relational graph is a graph in which attached to each node is a node label drawn from a set of node labels. These node labels represent the primitives of the structure being represented—in this case, the names of atoms present in the compounds of interest. Attached to each edge is also an edge label that is drawn from the set of edge labels—in this case the type of link between the two atoms connected by the edge.

A modified connectivity or adjacency matrix, as used in this paper, for a relational graph is the $(n + t) \times (n + t)$ matrix, where n is the number of variable nodes and t the number of terminal nodes, such that

$$M_{ij} = k \quad \text{if node } i \text{ is incident node } j$$
$$= 0 \quad \text{otherwise}$$

where k is the integer code for the type of connection between node i and node j. The terminal nodes constitute the set of node labels.

GRAPH MINIMIZATION METHOD

(A) Concepts. The method presented here is based on the canonical ordering of the nodes of the graph representing a chemical structure. This node ordering process operates on the graph's modified adjacency matrix to yield the canonical constant-order reduced matrix. This canonical matrix is unique for a given structure. The nodes are ordered by connectedness and then by node attributes. The structure is then reduced to a string by "walking" the graph in this canonical order. The similarity between structures can also be measured in terms of similarity between their canonical strings.

Apart from using an efficient graph minimization procedure in the matching process, the method allows all the known structures to be compiled into the canonical form only once. Subsequently, only the unknown input description need be compiled into its canonical string. This string is then matched against stored canonical strings.

In human–computer interfaces, a problem that is often encountered is that of inaccurate or incomplete data and query entry. Thus, it is possible for a structure to be specified with a missing node or arc. Graph matching methods will generally not find a match in such situations. With the canonical string representations, however, the most similar structure can be retrieved in cases where there is no exact match.

(B) General Description. The summary of the procedure is first given, followed by a detailed explanation of each stage:

(i) Representation. The chemical structure is turned into a modified adjacency matrix representation.

(ii) Matrix Minimization. The adjacency matrix is minimized into the canonical matrix form.

(iii) String Representation. The canonical matrix representation is then reduced into a string by doing a "raster scan" of the matrix.

(iv) Best-Match Retrieval. The resulting canonical string is matched against stored strings. In the event of failure to find an exact match, the most similar structures can be retrieved.

(C) Representation. In our representation, we have used two types of nodes—variable nodes and terminal nodes. The number of variable nodes in any particular structure is equal to the number of atoms in the compound, while the number of variable nodes is equal to the number of possible labels that could be attached to the nodes.

The method requires that all the possible labels—all the atoms—that could be present be identified, along with the types of links that could connect any two atoms, e.g., single or double. Each compound is then represented by a modified connectivity matrix in which a unique integer code represents the type of link between any two atoms. The top half of this matrix—those rows representing the terminal nodes—is the same for all the compounds in the domain of interest, because no arcs originate from a terminal node. An arc labeled "is.a" is employed to connect each variable node to the appropriate terminal node.

A chemical structure is described in terms of its primitive parts (which are the atoms) and the relationships (i.e., the bonds) between them. For example, the structure of Figure 1 can be described as follows:

a is.a H
b is.a O
c is.a C

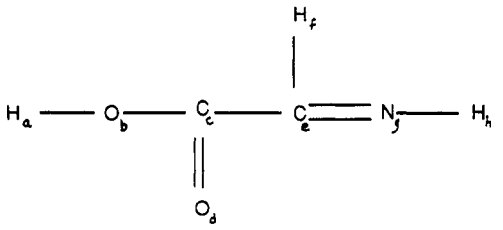

Figure 1.

d is.a O
e is.a C
f is.a H
g is.a N
h is.a H
a single b
b single a
b single c
c single b
c double d
c single e
d double c
e single c
e single f
e double g
f single e
g double e
g single h
h single g

The nodes of the graph formed by the structure are a, b, c, d, e, f, g, h, C, H, N, and O, where the last four are the constant nodes. The arc labels are "is.a", "single", and "double".

The description of a chemical structure is input in the form shown above, in which each input line represents an arc in the descriptive graph. The description as given is then reduced into the connectivity matrix. To ensure that the connectivity matrix is in some initial order, the description is sorted, first by the relation field and then by the third field within each relation.

We have called the particular adjacency matrix as used in this method the modified adjacency matrix. In this matrix each relation type is represented by a unique integer value. We can have, for example, single 1, double 2, and is.a 3. Thus, the above description will be represented by the matrix

		1	2	3	4	5	6	7	8	9	10	11	12
C	1	0	0	0	0	0	0	0	0	0	0	0	0
H	2	0	0	0	0	0	0	0	0	0	0	0	0
N	3	0	0	0	0	0	0	0	0	0	0	0	0
O	4	0	0	0	0	0	0	0	0	0	0	0	0
a	5	0	3	0	0	0	1	0	0	0	0	0	0
b	6	0	0	0	3	1	0	1	0	0	0	0	0
c	7	3	0	0	0	0	1	0	2	1	0	0	0
d	8	0	0	0	3	0	0	2	0	0	0	0	0
e	9	3	0	0	0	0	0	1	0	0	1	2	0
f	10	0	3	0	0	0	0	0	0	1	0	0	0
g	11	0	0	3	0	0	0	0	0	2	0	0	1
h	12	0	3	0	0	0	0	0	0	0	0	1	0

with the initial node ordering 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.

(D) Matrix Minimization. The problem we are tackling is that in the case of rows representing features of the structure there is no absolute naming scheme. We wish in these cases to allocate the same names to each comparable component in different graphs being compared. Not all features need to be treated in this way, as some properties are known literally, e.g., C, H, N, and O, which will be invariant across descriptions.

In principle, the rows of the connectivity matrix are considered as a series of numbers. The aim is to reorder rows and columns until the matrix from top to bottom represents the rows in ascending order if each row is considered as being a number. To ensure that the process will always terminate, we consider the row with the greatest number of nonzero entries to be the largest.

The matrix minimization is performed by attempting to "move down" rows with the largest number of nonzero entries while "moving up" those with the least number of entries. The process is carried out in stages until no more movements are possible. If at any stage there is more than one row with the largest number of entries, the one with the more significant entry is moved. When two rows are exchanged, the corresponding columns are also exchanged.

The rows representing the constant nodes are not included in the minimization process. The constant nodes, being the attribute nodes, are used to constrain the reordering process. Where two rows contain the same number of nonzero elements, selecting the one with the more significant entry is, in effect, using the node properties to discriminate between the two. This ensures that the resulting matrix is in a canonical order.

This process can be illustrated with the following starting matrix:

	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	3	0	0	0	0	2	1	0	0
6	0	0	0	3	2	0	0	0	0
7	3	0	0	0	1	0	0	1	2
8	0	3	0	0	0	0	1	0	0
9	0	0	3	0	0	0	2	0	0

In this initial matrix, the row with the largest number of entries is row 7. Thus, rows (and columns) 7 and 9 are exchanged, and the new matrix is

	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	3	0	0	0	0	2	0	0	1
6	0	0	0	3	2	0	0	0	0
7	0	0	3	0	0	0	0	0	2
8	0	3	0	0	0	0	0	0	1
9	3	0	0	0	1	0	2	1	0

At the next iteration, only the first eight rows are examined. Of these, row 5 contains the largest number of entries. It is exchanged with row 8. The new matrix is

	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	3	0	0	0	0	0	0	1
6	0	0	0	3	0	0	0	2	0
7	0	0	3	0	0	0	0	0	2
8	3	0	0	0	0	2	0	0	1
9	3	0	0	0	1	0	2	1	0

In this matrix, the first seven rows are now examined. Rows 5–7 all contain three entries. Row 5 is selected for exchange with row 7 as it contains the leftmost entry. The new matrix is

	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	3	0	0	0	0	0	2
6	0	0	0	3	0	0	0	2	0
7	0	3	0	0	0	0	0	0	1
8	3	0	0	0	0	2	0	0	1
9	3	0	0	0	2	0	1	1	0

Rows 5 and 6 are now exchanged based on the above criterion.

	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	3	0	0	0	2	0
6	0	0	3	0	0	0	0	0	2
7	0	3	0	0	0	0	0	0	1
8	3	0	0	0	2	0	0	0	1
9	3	0	0	0	0	2	1	1	0

At this stage no exchanges are possible. This is the canonical matrix that is reduced into the representative string. Therefore, the final ordering of the nodes is as follows: 1, 2, 3, 4, 6, 9, 8, 5, 7.

It should be pointed out at this stage that this matrix minimization process will always terminate and that it will always generate a unique canonical matrix for every relational graph. This is obvious for the case where every node has a different number of connections from all the other nodes.

The more common situation, however, is that there is more than one node with the same number of connections. The first constraint that is applied in such a case is that of the node properties. This is so because the more significant positions that are used to discriminate between nodes with the same number of connections actually represent the terminal nodes in the matrix. These attribute nodes remain invariant throughout the minimization process as they are not reordered.

Where two nodes both have the same number of connections and the same attributes, then the current largest row is selected. In this case, it does not really matter which one is chosen since the nodes are identical. It could also be possible to use the types of connections to distinguish between nodes, since each connection type is represented by a different integer in the matrix.

In this manner, a row is selected for exchange during each pass of the matrix until there is no longer any need for exchange. Each time a row is selected and put in its appropriate place in the reordered matrix, that particular row of the reordered matrix is no longer exchanged. Hence, the process will always terminate. This method is therefore different from those that attempt the minimization without the constraints that we have applied.

(E) Canonical String Representation. The minimized matrix is then scanned to turn the description into an ordered string. This is done by assigning each node a token and then doing a raster scan on the matrix. The terminal nodes are given distinct identifying node tokens, while the other nodes are given a common unspecified node token. The algorithm for doing this is given below in outline form.

In the algorithm, n and t are the number of variable and terminal nodes, respectively, mat is the adjacency matrix, and node is a vector storing the order of the nodes.

```
Scan(string):
1. a := 'A'; (* Assign node tokens. *)
   for i := 1 to t do begin
      token[i] := a;
      a := succ(a)
   end;
   for i := t+1 to n+t do
      token[i] := 'N';
2. index := 0; i := t;
3. if row i contains nonzero entries the begin
      string[index] := token[i];
      index := index+1;
      j := 1
   end
   else goto step 6;
4. if mat[node[i],node[j]]>0 then begin
      string[index] := mat[node[i],node[j]];
      index := index+1;
      string[index] := token[j];
      index := index+1
   end;
5. if j<n+t then j := j+1; goto step 4;
6. if i<n+t then i := i+1; goto step 3
7. end.
```

For the example of Figure 1 given previously, the new node ordering is 1, 2, 3, 4, 10, 7, 5, 12, 11, 6, 8, 9, yielding the reduced matrix

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	3	0	0	0	0	0	0	0	2
6	0	3	0	0	0	0	0	0	0	0	1	0
7	0	3	0	0	0	0	0	0	0	1	0	0
8	0	3	0	0	0	0	0	0	1	0	0	0
9	0	0	0	3	0	0	0	1	0	0	0	1
10	0	3	0	0	0	0	1	0	0	0	2	0
11	3	0	0	0	0	1	0	0	0	2	0	1
12	3	0	0	0	2	0	0	0	1	0	1	0

Nodes 1–4 are given tokens A, B, C and D, and C and D, respectively. All other nodes are assigned the unspecific token N. The matrix is scanned starting from row 3 down. Where there is a nonzero entry in position (i,j), the three symbols token[i], matrix[i,j], and token[j] are written into the string in that order, except that for every row i token[i] is written only once. For this example the resulting string will be

N3D2NN3B1NN3B1NN3B1NN3D1N1NN3C1N2NN3
A1N2N1NN3A2N1N1N

We can estimate the size of the string generated for a given structure in the following manner. Along with an entry in the adjacency matrix to represent each node connection, an entry is also employed to indicate the node type. In addition to writing the token corresponding to every nonterminal row in the string, two symbols are also written for every entry in the adjacency matrix. Thus, the total number of symbols in the string is given by

$$\sum_{i=1}^{n}(2(n_i + 1) + 1) = 2\sum_{i=1}^{n}n_i + 3n = 4n_t + 3n$$

where n is the number of atoms in the structure, n_i is the number of bonds for each atom, and n_t is the total number of bonds. For a structure with, say, 30 atoms and 30 bonds, the number of symbols in the string is about 210 bytes. Use of a binary connection table, on the other hand, results in 900 entries. Representing this by bitlists will require about 113 bytes. This is in addition to storage space for other information like the type of atoms and bonds, etc. As a result, it is likely that the method described in this paper will utilize an amount

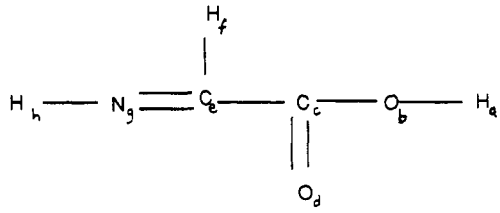

Figure 2.

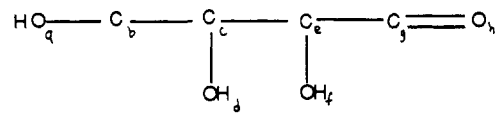

Figure 3.

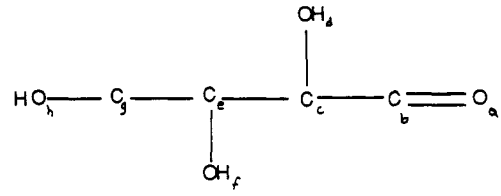

Figure 4.

of space similar to what other methods require.

(F) Best-Match Retrieval. The final stage in comparing any two given descriptions is the computation of the similarity value for the strings generated in the manner described above. The dictionary of known structure is set up to facilitate quick exact retrieval. Where an exact match cannot be found, however, some string similarity measure can be used to retrieve the structures most similar to the given one. Several algorithms are available for doing this.[11,12]

A useful way to organize the dictionary of strings to accommodate efficient similarity matching is to partition the file according to the length of the strings or the number of atoms and bonds in the corresponding structure. If the file were partitioned by string length, then only a few partitions with string length close to the search string need be examined. For exact matching, hashing can be used with the partition address used to offset the hash value.

The *n*-gram method[12] in which the strings are represented in the form of a table detailing the fixed-length substrings present in each string is another useful way of handling best matches. The similarity of any two strings is based on the number of *n*-grams (substrings of length 3, for instance) they possess in common. The *n*-gram table is usually maintained as a set of inverted files in bit array format. Considered a matrix, there is a row for each *n*-gram and a column for each string. For matching, the table is consulted to determine the number of *n*-grams common to the strings. The stored string with the highest similarity value, based on the number of common *n*-grams, is selected. An efficient implementation of this technique is described in reference 13.

The method outlined in this paper is capable of retrieving stored compounds, both exactly and approximately, in an efficient manner. Figures 1–4 show two different chemical structures that can be identified by this method. In the earlier example shown in the paper, only four different node types (C, H, N, O) are present in the system. As many node types as are present can be accommodated, however.

By including OH among the primitive node types that can be present, the structures of Figures 1 and 2 reduce to the string

N3D2NN3B1NN3B1NN3B1NN3D1N1NN3C1N2NN3
A1N2N1NN3A2N1N1N

while those of Figures 3 and 4 both yield

N3E1NN3E1NN3E1NN3D2NN3A2N1NN3A1N1NN3
A1N1N1NN3A1N1N1N

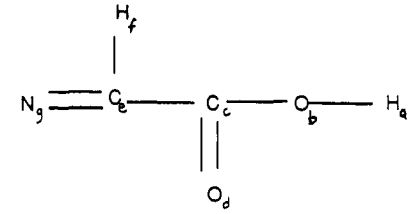

Figure 5.

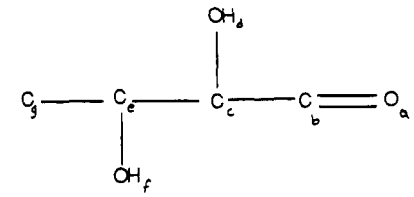

Figure 6.

Table I. Similarities for Figures 1–6

	figure					
figure	1	2	3	4	5	6
1	1.00	1.00	0.76	0.76	0.79	0.64
2	1.00	1.00	0.76	0.76	0.79	0.64
3	0.76	0.76	1.00	1.00	0.67	0.83
4	0.76	0.76	1.00	1.00	0.67	0.83
5	0.79	0.79	0.67	0.67	1.00	0.75
6	0.64	0.64	0.83	0.83	0.75	1.00

To illustrate how this method will perform in a situation where a structure is specified, say, with a missing atom or bond, we removed the atom labeled "h", along with the corresponding bonds, from both Figures 2 and 4. The resulting structures are shown in Figures 5 and 6. The canonical strings for the structures were generated, and each of the six are then compared with all the structures in turn.

The strings were compared by using the string similarity distance.[11] To put the computed distances in perspective, each is transformed to a fraction by computing the function[13]

$$(ab - 2*d)/ab$$

where *ab* is the sum of the lengths of the two strings compared and *d* is the computed distance. The resulting similarity values are given in Table I.

It can be seen from Table I that the algorithm reckons the structure of Figure 5 to be closer to those of Figures 1 and 2, while Figure 6 is closer to Figures 3 and 4. Thus, correct similarity matching is performed.

CONCLUSIONS

An efficient graph minimization method for generating unique canonical names for chemical structures has been presented. The graph minimization process always stops because it is constrained by both node connectedness and attributes. Further efficiency is gained by the possibility of compiling all known compounds into the canonical form only once. Best-match retrieval can also be performed because of the canonical string representation that is employed.

REFERENCES

(1) Unger, S. H. "GIT—A Heuristic Program for Testing Pairs of Directed Line Graphs for Isomorphism". *Commun. ACM* **1964**, *7*, 26–34.
(2) Sussenguth, E. H. "A Graph-Theoretic Algorithm for Matching Chemical Structures". *J. Chem. Doc.* **1965**, *5*, 36–43.
(3) Scoins, H. I. "Placing Trees in Lexicographical Order". *Mach. Intelligence* **1968**, *3*, 43–60.
(4) Nagle, J. F. "On Ordering and Identifying Undirected Linear Graphs". *J. Math. Phys. (N.Y.)* **1966**, *7*, 1588–1592.
(5) Randić, M. "On Canonical Numbering of Atoms in a Molecule and Graph Isomorphism". *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 171–180.

(6) Hendrickson, J. B.; Toczko, A. G. "Unique Numbering and Cataloguing of Molecular Structures". *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 171-177.
(7) Mackay, A. L. "A Metaphor for Molecular Evolution". *J. Theor. Biol.* **1975**, *54*, 399-401.
(8) Wipke, W. T.; Dyott, T. M. "Stereochemically Unique Naming Algorithm". *J. Am. Chem. Soc.* **1974**, *96*, 4834-4842.
(9) Morgan, H. L. "The Generation of a Unique Machine Description for Chemical Structures". *J. Chem. Doc.* **1965**, *5*, 107-113.
(10) Jochum, C.; Gasteiger, J. "Canonical Numbering and Constitutional Symmetry". *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 113-117.
(11) Wagner, R. A.; Fischer, M. J. "The String-to-String Correction Problem". *J. Assoc. Comput. Mach.* **1974**, *21*, 168-178.
(12) Angell, R. C.; Freund, G. E.; Willett, P. "Automatic Spelling Correction Using a Trigram Similarity Measure". *Inf. Process Manage.* **1983**, *19*, 255-261.
(13) Owolabi, O.; McGregor, D. R. "Fast Approximate String Matching". *Software Pract. Exper.* **1988**, *18*(4), 387-393.

^{13}C NMR Assignments of the Bases in Oligodeoxynucleotides: An Automated Procedure Using Bayesian Statistics†

TIMOTHY J. HYMAN, EILIS A. BOUDREAU, GILLES G. MARTIN,‡ BÉAT M. JUCKER, PHILIP N. BORER, and GEORGE C. LEVY*

NIH Research Resource for NMR and Data Processing and CASE Center, Syracuse University, Syracuse, New York 13244-1200

Received May 20, 1988

A statistical method, Bayes Maximum Likelihood, has been applied to the classification of base ^{13}C NMR resonances in DNA oligomers. An accuracy of 100% for carbon class discrimination was achieved for a preliminary training set of four oligomers using the following four parameters: (1) the chemical shift; (2) the temperature at which the spectrum was obtained; (3) the difference in chemical shift from the C5 resonances; and (4) a sequence factor representing the neighboring nucleotides. Classification of a fifth oligomer, previously assigned and not contained in the original training set, gave reasonable carbon class assignments.

INTRODUCTION

^{13}C NMR is a valuable tool for studying the conformation of nucleic acids and is particularly useful for studying internal dynamics in DNA oligomers. The most significant feature of ^{13}C NMR in contrast to ^{1}H NMR is that the carbon signals appear over a chemical shift range almost 20 times larger than for proton resonances. Since structural effects on the chemical shifts are on the order of 1-2 ppm for both nuclei, ^{13}C assignment is much simpler. The preferred methods for assigning the carbon signals are (1) selective isotopic substitution or (2) coherence transfer from the protons, where the proton spectrum can be unequivocally assigned by COSY and NOESY techniques.[1] Recently, the coherence-transfer method has been refined to be very sensitive by use of the two-dimensional proton-detected double quantum ^{1}H-^{13}C correlation experiment.[2-8] Both assignment methods require a large investment in time and effort by experts in chemical synthesis and/or NMR spectroscopy and further require facilities that are not routinely available.

A third method involves the comparison of monomer and oligomer reference spectra. This method has several drawbacks: (1) Peak assignment is time consuming. (2) A large amount of documentation is needed for comparison. (3) NMR experts are required to perform peak assignments. (4) Errors can occur when only comparative assignment is performed. An automated statistical approach to these assignments would be attractive.

An ideal computer program to perform peak assignment would possess a database of spectra for which all the peaks are known. Using expertise built into the software, the computer would compare unknown peaks to its database and automatically assign these unknown peaks to the correct carbon class. This could be performed in seconds by a laboratory technician vs hours for an NMR expert using current comparative assignment methods. Even if isotopic substitutions or 2-D methods are available, the classification scheme outlined here would provide independent confirmation of the assignments and help to resolve ambiguities that are inevitable when current methods are used.

There are several statistical methods to assign an unknown sample to one of several populations represented in a training set. The field of pattern recognition provides abundant literature on the optimal method depending on the training set used. Among these commonly used methods are clustering, Fisher discriminant analysis, principal component analysis, and nearest-neighbor classification. Bayes Maximum Likelihood (BML), which will be implemented in this study, is one of the most powerful and most general multivariate pattern recognition methods.

METHODS

(A) Collection and Assignment of ^{13}C Data. ^{13}C NMR data obtained on the molecules (1) $[d(TAGCGCTA)]_2$, (2) $[d(GGTATACC)]_2$, (3) $[d(CGCGCG)]_2$, and (4) $[d(GGCCT)]_2$ constitute the training set in this study. The synthesis and purification of the oligonucleotides and collection of the one-dimensional ^{13}C NMR spectra have been described previously.[9] The conditions for these experiments have been reviewed elsewhere.[7,8,10] A fifth molecule, $[d(TCGCG)]_2$, was implemented in testing the program's ability to assign carbon classes to the chemical shifts of an unassigned spectrum. Therefore, this molecule was not included in the training set. The test oligomer was classified by the program for spectra taken at 10, 30, and 75 °C.

The term "carbon class" refers to all resonances that correspond to a specific carbon type in a molecule. For example, there are two resonances belonging to a guanine carbon 2 (GC2) in $[d(TAGCGCTA)]_2$, shown in Figure 1. Individual carbon positions are specified as, for example, G3,2, which refers to guanine carbon 2 in chain position 3 from the 5'-end of the oligomer. This paper addresses only the base carbon

† This work was supported by grants from the National Institutes of Health (RR01317, GM32691, and GM35069).
* To whom correspondence should be addressed.
‡ Present address: Ecole Centrale de Paris, Department de Mathematiques Appliquees, 92230 Chatenay Malabry, France.

0095-2338/88/1628-0226$01.50/0