

## ARTICLES

## Algorithms for the Identification of Three-Dimensional Maximal Common Substructures

ANDREW T. BRINT and PETER WILLETT\*

Department of Information Studies, University of Sheffield, Western Bank, Sheffield S10 2TN, U.K.

Received November 20, 1986

Two algorithms are described for the identification of the maximal substructures common to two (or more) three-dimensional chemical structures, where a substructure consists of a set of atoms and the associated interatomic distances. The algorithm of Crandell and Smith involves a breadth-first tree search procedure in which substructures are expanded as they are shown to be common to all of the molecules under consideration. The clique-detection algorithm involves the identification of cliques in the correspondence graph linking matching atoms and interatomic distances in pairs of structures that are being compared. This second algorithm is shown to be substantially faster in operation than the Crandell and Smith algorithm when applied to structures taken from the Cambridge Crystallographic Data Bank, and an extension to the algorithm is described that allows it to be used for the identification of the maximal substructure common to arbitrary numbers of molecules.

## THREE-DIMENSIONAL MAXIMAL COMMON SUBSTRUCTURES

Many areas of chemical research depend upon the identification of the structural features common to molecules. With improvements in computer hardware and software technology, there is increasing interest in procedures that would allow this identification to be carried out by automatic means. Typical application areas for this sort of research are in mass spectral search systems, quantitative structure-activity relationships, automatic chemical reaction indexing, and similarity searching in chemical structure files.<sup>1-7</sup> To date, research work into the automatic identification of common substructures has generally involved the comparison of two-dimensional (2-D) chemical structure diagrams, but there is no reason why the techniques should not be extended to chemical structures for which full three-dimensional (3-D) coordinate data are available. A particularly important application here is the ability to identify the largest substructure common to a structurally disparate set of 3-D molecules of known activity, since this common substructure is likely to represent, or at least contain, the *pharmacophoric pattern*, i.e., the geometric pattern of atoms responsible for the observed activity.<sup>8,9</sup>

We are currently involved in the development of computer systems for pharmacophoric pattern searching using techniques analogous to those developed for 2-D substructure searching in chemical information systems. To date, we have established a methodology for the selection of interatomic distances that form the basis for a first-level screening search<sup>10</sup> and subsequently demonstrated that these screens allow highly efficient pharmacophoric pattern searches to be carried out.<sup>11</sup> Most recently, we have evaluated a range of algorithms that can be used for *geometric searching*, the 3-D equivalent of atom-by-atom searching which takes place only for those few molecules matching at the screen search level.<sup>12</sup> The present paper compares two algorithms that allow the identification of the maximal common substructure(s) (MCS) in 3-D chemical compounds. The substructures considered here consist of a set of atoms, together with the associated inter-

atomic distances; thus the MCS will not, in general, correspond to sets of atoms linked together by chemical bonds. The first of the two algorithms is that of Crandell and Smith,<sup>13</sup> which works by finding all common substructures of size  $N$  atoms and then extends these so as to identify common substructures of size  $N + 1$  atoms. The second algorithm is based on the graph-matching algorithm of Barrow et al.,<sup>14,15</sup> which has been used for the identification of maximal common subgraphs in computer vision research.

## CRANDELL-SMITH ALGORITHM

The algorithm described by Crandell and Smith<sup>13</sup> for finding the 3-D substructures in common between a set of molecules involves taking all the common substructures of size  $N$  atoms associated with each molecule and adding an extra atom to each of them. These enlarged substructures are then canonically named, as detailed below, so as to allow them to be compared rapidly with the enlarged substructures associated with other molecules. If a substructure is not found in all of the other molecules' lists, it is deleted from consideration. The surviving substructures form the common substructures of size  $N + 1$ .

The selection of an atom to add to a substructure in the growing step is done by consulting a distance matrix associated with each molecule. This matrix contains the distances between all atoms in the molecule. Negative distances in the matrix indicate those distances that are not included in the current set of common substructures, while atoms associated with positive distances are available for addition to the common substructures in the next growth phase of the algorithm. Thus, the algorithm consists of the following basic steps: (1) Create the initial distance tables; set  $N = 1$ . (2) Grow substructures of size  $N + 1$  atoms from the common substructures of size  $N$  atoms identified in the previous iteration. (3) Name the substructures. (4) Compare the substructures to identify those that are still in common; if there are none, then stop. (5) Amend the distance tables and return to step 2. Crandell and Smith<sup>13</sup> describe modifications to this basic procedure to allow for a common starting substructure to be specified, which must be contained in any substructures produced by the algorithm,

\* Author to whom all correspondence should be addressed.

and to cater for stereochemistry; these modifications are not considered here.

**Creation of the Distance Tables.** The comparison of the substructures in step 4 is implemented efficiently by representing each of the interatomic distances in each molecule by a single integer code, this being done by the following clustering procedure. A list is formed of the interatomic distances present in the molecules for each pair of atomic types present. These lists are then sorted into ascending order and the distances in them clustered together so that a distance belongs to the same cluster as its predecessor if the difference in their values is less than the tolerance value, which is taken to be 0.09 Å as in the original paper;<sup>13</sup> otherwise, a new cluster is formed. The clusters are then numbered starting from one, and groups that do not contain atom pairs from every molecule have their numbers negated. A distance matrix,  $D$ , is associated with each molecule, and the element  $D[I, J]$ ,  $I < J$ , contains the group number for the interatomic distance between this molecule's  $I$ th and  $J$ th atoms; these group numbers are referred to subsequently as distances.

**Growing.** Each substructure associated with a molecule is grown by enlarging its atom set by one. This is effected by numbering each atom in a structure from one up to the number of (non-hydrogen) atoms present and then adding an atom with a number greater than any of the atoms in the current set of atoms comprising a common substructure and whose distances in the distance table to these atoms are positive. Where it is possible to add several different atoms, a new substructure is produced for each of them. In the first iteration, the grown substructures are the individual atoms in each molecule. Thus, the common structures are grown by a breadth-first tree search procedure.

**Naming.** Each node set, i.e., a set of atoms comprising a common substructure, produced in the growth stage is given a canonical name by taking each of the  $N(N-1)/2$  atom pairs in the substructure (where  $N$  is the size of the substructure) and forming a triple consisting of the two atom types (with the larger coming first) and the relevant distance entry in the distance table. Once these  $N(N-1)/2$  triples have been sorted, each triple need only be represented by its distance element, since this implicitly contains the atom type information.

**Comparing.** The named substructures for each of the molecules are compared with those of the other molecules. If another molecule is found that does not have this substructure among its substructures, then the substructure is deleted; hence, the  $N$ -atom substructures surviving this step are those in common.

**Amending the Distance Tables.** After the comparison stage, each nonnegative entry in the distance tables has its atom pair checked to see whether it still occurs in the relevant molecule's list of node sets. If it does not, the distance entry is negated so as to avoid growing substructures that contain this atom pair at some later point in the procedure.

### CLIQUE-DETECTION ALGORITHM

Any 3-D molecule can be regarded as a graph in which both the nodes and the edges have labels, these corresponding to the atomic types and the interatomic distances, respectively. This being so, it is possible to use established graph-theoretical procedures for the identification of pharmacophoric patterns, and we have adopted the maximal common subgraph algorithm of Cone et al.,<sup>1</sup> Barrow et al.,<sup>14,15</sup> and Levi.<sup>16</sup> Given a pair of graphs  $A$  and  $B$ , a *correspondence graph*,  $C$ , can be formed by the following process: (1) Create the set of all pairs of nodes, one from each of the two graphs, such that the nodes of each pair are of the same type. (2) Form the graph  $C$  whose nodes are the pairs from step 1. Two nodes  $(A_I, B_X)$ ,  $(A_J, B_Y)$

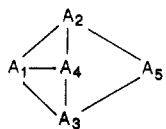
are marked as being connected in  $C$  if the values of the arcs from  $A_I$  to  $A_J$  and  $B_X$  to  $B_Y$  are the same. Maximal common subgraphs then correspond to the *cliques* of the correspondence graph, where a clique is a subgraph in which every node is connected to every other node and which is not contained in any larger subgraph with this property. Examples of the use of this technique for the handling of 3-D chemical structure information have been reported by Brint and Willett,<sup>12</sup> Golender and Rozenblit,<sup>17</sup> and Kuhl et al.<sup>18</sup>

An alternative graph-theoretical approach to MCS identification has been described by McGregor,<sup>19</sup> his algorithm involving a depth-first tree search in which each node in the tree represents the pairing of a node from the first graph with one from the second. The algorithm involves the definition of a maximal common subgraph as being the subgraph contained in both graphs that has the largest number of edges, a characterization which is of importance when the algorithm is applied to the identification of the bond changes that have occurred in chemical reactions.<sup>4</sup> In the present context, however, the algorithm is less applicable since it allows the identification only of the largest common subgraph, whereas the method described here finds all such subgraphs that are not contained in a larger subgraph; in addition, the data structures required by McGregor's algorithm would be very demanding of internal storage in the present context.<sup>20</sup>

We have used an MCS algorithm for clique detection previously, in a comparison of geometric searching algorithms for pharmacophoric pattern matching.<sup>12</sup> Specifically, given a query pattern  $P$  and a structure  $S$ , an MCS algorithm can be used to determine whether  $P$  is a subgraph of  $S$  simply by determining whether the MCS between the two is identical with  $P$ . These geometric searching experiments used the widely cited algorithm of Bron and Kerbosch<sup>21</sup> for the detection of the cliques; however, the identification of common substructures is computationally demanding, and thus several different clique-finding algorithms were coded and compared to see which was the most efficient for this application. The algorithms chosen were those described by Bron and Kerbosch,<sup>21</sup> Golender and Rozenblit,<sup>17</sup> Gerhards and Lindenberg<sup>22</sup> (specifically, version 1 of their algorithm 1), Loukakis and Tsouros,<sup>23</sup> and Loukakis;<sup>24</sup> these algorithms are described in detail in the original papers and by Brint.<sup>20</sup> However, as is demonstrated under Experimental Results below, the Bron-Kerbosch algorithm is very much faster in operation than the others, and thus only this algorithm will be presented here.

The algorithm operates by means of a backtracking tree search. At each level,  $D$ , of the tree search, there are two sets,  $N_D$  and  $C_D$ , of nodes of the graph that are connected to every node in the set  $M_D$ , which consists of the  $D$  nodes under consideration for inclusion in the next clique.  $N_D$  contains the nodes that have already been tried in the attempt to enlarge  $M_D$  and  $C_D$  those candidate nodes that have yet to be tried. The algorithm moves to the next level of the tree search by moving a candidate node from  $C_D$  to the trial set  $M_D$ , which then becomes  $M_{D+1}$ . The sets  $N_{D+1}$  and  $C_{D+1}$  are then calculated by removing from  $C_D$  and  $N_D$  those nodes not connected to the candidate node. When backtracking occurs, the node most recently added to  $M_{D+1}$  is added to  $N_D$  and removed from  $C_D$ , and the level of the search becomes  $D$  from its previous value of  $D+1$ . A clique is found when both  $C_D$  and  $N_D$  are empty; if only  $C_D$  is empty, then  $M_D$  is a subset of a clique that has already been output.

The selection of a candidate node from  $C_D$  is done so as to increase the likelihood of a point in  $N_D$  being connected to all points in  $C_D$ . (When this happens, further extensions to  $M_D$  from  $C_D$  cannot remove this point from  $N_D$ ; therefore,  $N_D$  can never become empty by extending  $M_D$ , so backtracking needs to occur.) This can be done by choosing that point in  $N_D$  which

**Table I.** Operation of the Bron-Kerbosch Clique-Detection Algorithm<sup>a</sup>


	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>	A <sub>4</sub>	A <sub>5</sub>
A <sub>1</sub>	1	1	1	1	0
A <sub>2</sub>	1	1	0	1	1
A <sub>3</sub>	1	0	1	1	1
A <sub>4</sub>	1	1	1	1	0
A <sub>5</sub>	0	1	1	0	1

<i>D</i>	<i>M<sub>D</sub></i>	<i>N<sub>D</sub></i>	<i>C<sub>D</sub></i>	action
1	1	{}	2 3 4	
2	1 2	{}	4	
3	1 2 4	{}	{}	output clique
2	1 2	4	{}	
1	1	2	3 4	
2	1 3	{}	4	
3	1 3 4	{}	{}	output clique
2	1 3	4	{}	
1	1	2 3	4	
1	2	1	4 5	
2	2 5	{}	{}	output clique
1	2	1 5	4	
1	3	1	4 5	
2	3 5	{}	{}	output clique
1	3	1 5	4	
1	4	1 2 3	{}	
1	5	2 3	{}	

<sup>a</sup>*M<sub>D</sub>* is the set forming the basis of the next clique, *C<sub>D</sub>* the candidates for addition to *M<sub>D</sub>*, and *N<sub>D</sub>* the previously rejected elements of *M<sub>D</sub>* that are connected to every element of *M<sub>D</sub>*. When a choice is possible over which element from *C<sub>D</sub>* should be used to extend *M<sub>D</sub>*, that element is chosen which is connected to the least number of elements of *N<sub>D</sub>*.

**Table II.** Number of Substructures Grown After Step 3 of the Crandell-Smith Algorithm for a Pair of Size-16 Molecules

iteration no.	no. of growths	
	A	B
1	16	16
2	111	113
3	450	472
4	1180	1195
5	2098	2212
6	2711	3023
7	2730	3081
8	2103	2246
9	1213	1266
10	506	517
11	144	145
12	25	25
13	2	2

is connected to the most elements of *C<sub>D</sub>*, and then every time a candidate is taken, selecting a point in *C<sub>D</sub>* that is not connected with the chosen point. The workings of the algorithm are illustrated in Table I.

#### ALGORITHMIC ENHANCEMENTS

The two methods that have been used for finding common substructures have major drawbacks. Crandell and Smith's algorithm can compare several molecules with one another, but our experiments, and those reported by Crandell and Smith, show that it is very expensive both in CPU time and in the storage space required if there is a large common substructure. On the other hand, the clique-finding method can deal very well with comparing two molecules even when there is a large common substructure, but extending the comparison to more than two molecules greatly increases the size of the correspondence graph. For *M* molecules containing SIZE<sub>1</sub>, SIZE<sub>2</sub>, ..., SIZE<sub>*M*</sub> atoms, respectively, the number of elements in the correspondence graph can be as high as SIZE<sub>1</sub><sup>2</sup> × SIZE<sub>2</sub><sup>2</sup> × ... × SIZE<sub>*M*</sub><sup>2</sup> in the worst case. This makes the basic clique-detection procedure quite impracticable if more than

**Table III.** Times (in CPU Seconds on a Prime 9950) Taken by Various Stages of the Crandell-Smith Algorithm

step	time taken
initialize	0.3
grow	5
name	90
compare	88
amend	31
total	214

**Table IV.** Number of Substructures Grown After Step 3 of the Crandell-Smith Algorithm When the Updating of the Distance Tables Is Omitted

iteration no.	no. of growths	
	A	B
1	16	16
2	111	113
3	450	472
4	1180	1195
5	2098	2212
6	2733	3032
7	2737	3082
8	2104	2330
9	1213	1302
10	506	526
11	144	146
12	25	25
13	2	2

**Table V.** Times (in CPU Seconds on a Prime 9950) Taken by Various Stages of the Crandell-Smith Algorithm When the Updating of the Distance Tables Is Omitted

step	time taken
initialize	0.3
grow	5
name	91
compare	88
total	184

two molecules of realistic size are to be processed.

**Reducing the Time Taken by Crandell and Smith's Algorithm.** Table II shows the number of common substructures grown after step 3 of each generation of the processing of a pair of molecules A and B, each containing 16 non-hydrogen atoms. This pair of molecules was generated by selecting a 16-atom structure from the Cambridge Crystallographic Data Bank (CCDB)<sup>25</sup> and then reordering the atoms and distorting some of the interatomic distances so as to obtain two different, but structurally related, molecules. Table III shows the run times taken by each step of the Crandell-Smith algorithm when comparing A and B. These run times, like all of the others in this paper, are in CPU seconds and were obtained from FORTRAN 77 programs run on a Prime 9950 computer.

In considering these times, the reader should note that the comparison stage includes 68 s of CPU time which was used for sorting the name list associated with each molecule into ascending order. This was done to allow a rapid, one-pass comparison of the two lists. If this sort is omitted, the comparison stage can swamp the computation; e.g., the use of unsorted lists for this pair of molecules resulted in a comparison stage taking no less than 916 CPU s.

It is possible to remove the amending step, which is not absolutely necessary and which is included only as a way of increasing the efficiency of the algorithm. The effect of removing the amendment step is shown in Tables IV and V, which refer to the same size-16 molecule as in Tables II and III. Table IV again lists the numbers of substructures grown in step 3 of the algorithm, while Table V gives the timings for the various stages of the processing when the amendment stage

is excluded. A comparison between Tables II and IV shows that the removal of the amendment stage results in only a slight increase in the number of common substructures; however, this is more than compensated for by the greater overall speed of execution, as evidenced by the running times in Tables III and V. Brint<sup>20</sup> presents extensive results obtained with other pairs of structurally related molecules, all of which confirm the behavior illustrated here. Accordingly, all of the subsequent results reported here relate to experiments in which the distance tables were not updated after the comparison stage.

**Extending the Clique-Finding Algorithm to More than Two Molecules.** Previous applications of the clique-detection approach to chemical structure problems have been restricted to the comparison of pairs of molecules only.<sup>1,12,17,18</sup> This is a severe limitation of the algorithm, and we have extended the method to allow the comparison of  $M$  ( $M > 2$ ) molecules. The approach taken is to select one of the molecules and find its common substructures with each of the other  $M - 1$  molecules in turn. If the selected molecule is of size SIZE, then each common substructure can be represented as a set whose elements are taken from the atom numbers 1, 2, ..., SIZE. Hence, the problem of finding substructures common to the  $M$  molecules becomes one of intersecting  $M - 1$  sets, one coming from each of the  $M - 1$  groups of common substructures. Unfortunately, it is not possible to simply generate all of the possible intersections because if there are  $K$  common substructures in each group, then there are  $K^{M-1}$  ways of intersecting the sets. One way to tackle this problem is to grow the subsets in common, in much the same way as the Crandell-Smith algorithm grows the common substructures. More specifically, for each group  $I$  the union of all sets is taken so as to form a set UNION( $I$ ). A set INTERSECT is formed by intersecting all the UNION( $I$ )'s, and the common substructures of size one are all the elements of this set. The algorithm then proceeds as follows: (1) Grow each substructure by adding an element of INTERSECT that is greater than every element of the substructure. If there are several possible elements from INTERSECT that can be added, then an enlarged substructure is produced for each of these. (2) Test every enlarged substructure to see whether it is contained in at least one set from each group. If it does not, then it is eliminated. (3) Check every element of INTERSECT to see if it is still contained in a substructure. If it is not, then it is eliminated from INTERSECT and from any of the sets containing it. Any set whose size is less than or equal to the current substructure size is also eliminated, and the algorithm goes back to step 1.

## EXPERIMENTAL RESULTS

In this section, we describe the experiments that were carried out to test the algorithms described in the previous sections using structures taken from the CCDB.

**Comparison of Clique-Detection Algorithms.** These experiments compared the five clique-detection algorithms discussed under Clique-Detection Algorithm. For each pair of molecules, the first stage of Crandell and Smith's algorithm was applied to cluster the distances and the correspondence graph produced from the resulting distance tables: the clique-finding algorithms were then run on this correspondence graph.

Several pairs of molecules were generated, containing between 16 and 25 non-hydrogen atoms. In each case, a range of distortions was imposed so as to create MCSs of different sizes. A typical set of results is shown in Table VI, which gives the run times required to find all the cliques in the correspondence graph generated from the 25-atom molecules ( $C_{15}NO_9$ ); the runs for the Gerhards-Lindenberg algorithm were all aborted after 2400 CPU s. The results in this table, and others presented by Brint,<sup>20</sup> indicate that the Bron-Ker-

**Table VI.** Times (in CPU Seconds on a Prime 9950) for Finding All Cliques in the Correspondence Graph Resulting from the Distortion of a 25-Atom Structure<sup>a</sup>

algorithm	largest clique size; no. of cliques >size 4						
	8;	9;	10;	13;	17;	21;	25;
BK	934	1293	1106	471	340	152	106
GR	15.3	17.7	14.8	7.7	6.5	5.0	3.8
GL	80.3	102.4	78.4	25.5	19.3	11.5	5.5
LT	all runs >2400						
L	154.1	187.8	153.1	79.8	66.1	46.9	32.3
	156.5	187.5	152.1	77.2	62.4	46.2	30.5

<sup>a</sup>The algorithms are as follows: BK = Bron and Kerbosch, GR = Golender and Rozenblit, GL = Gerhards and Lindenberg, LT = Loukakis and Tsouros, and L = Loukakis.

**Table VII.** Times (in CPU Seconds on a Prime 9950) To Identify the MCS for Distorted Pairs of Molecules

(A) 14 Atoms					
algorithm	size of the largest common substructure				
	7	8	10	12	14
CS	4.5	5.3	12.0	47.9	266.3
clique	2.1	2.3	2.4	2.3	2.4
(B) 15 Atoms					
algorithm	size of the largest common substructure				
	6	7	9	11	15
CS	2.6	3.2	4.7	19.1	70.6
clique	1.5	1.5	1.5	1.5	1.4
(C) 20 Atoms					
algorithm	size of the largest common substructure				
	6	7	9	11	14
CS	8.6	14.6	19.0	59.8	*
clique	4.1	4.0	3.6	3.8	3.8

bosch algorithm is substantially faster in operation than the other algorithms tested here. These use a variety of data structures and heuristics to prune branches of the depth-first tree search as quickly as possible; however, it would seem that these strategies actually increase the computation rather than reduce it. This may be due to the fact that, with the exception of the Golender-Rozenblit algorithm that has been designed for chemical applications and which performs quite well in our tests, these heuristics were designed and tested for clique detection in randomly generated graphs; the chemical graphs used here, conversely, are far from random in that there are strong dependencies between the various interatomic distances. In view of these results, the Bron-Kerbosch algorithm was adopted as the basis of the clique-detection algorithm used for the experiments described in the remainder of this paper.

**Comparing Two Molecules.** Molecules were chosen from the CCDB, the atoms reordered, and both the original molecule and its copy distorted slightly as described above. The Crandell-Smith algorithm and the Bron-Kerbosch clique-finding algorithm were run on pairs of molecules of sizes 14 ( $C_{12}NO$ ), 15 ( $C_{10}N_3OCl$ ), and 20 ( $C_{13}N_3O_3S$ ). The results of these runs are listed in Table VII. Asterisked results correspond to a run where the Crandell-Smith algorithm exceeded the available virtual storage; this was often the case when the MCS was of size 15 atoms or greater. It will be seen from the figures in the table that the clique-detection method is more efficient in operation than the Crandell-Smith algorithm. The difference is substantial even when the MCS is quite small and becomes very large indeed when there is a high degree of structural similarity between the two molecules being compared.

A further set of experiments involved molecules of sizes 9 ( $C_8O$ ), 15 ( $C_{13}O_2$ ), 24 ( $C_{24}$ ), and 28 ( $C_{27}O$ ); these were searched for in a set of 250 molecules taken from the CCDB to identify molecules with MCSs of size 7 atoms or greater

**Table VIII.** Times (in CPU Seconds on a Prime 9950) for Finding the MCS between a Query Molecule and Its Collection of Molecules

(A) 9 Atoms					
molecular size					
	17	22	23	29	36
no cluster	1.2	2.2	2.1	2.4	2.6
cluster	1.5	3.5	3.4	3.8	3.9
CS	3.2	9.8	10.1	5.1	4.9

(B) 15 Atoms					
molecular size					
	20	21	22	23	36
no cluster	3.2	2.2	3.2	3.8	13.0
cluster	5.7	3.9	5.6	6.4	36.5
CS	22.2	7.5	8.0	20.2	*

(C) 24 Atoms				
molecular size				
	13	14	24	34
no cluster	6.7	7.1	22.4	36.5
cluster	28.4	29.4	109.9	>2400
CS	*	*	*	*

(D) 28 Atoms					
molecular size					
	22	25	26	29	33
no cluster	19.5	22.5	9.5	34.9	28.0
cluster	>600	97.2	82.3	218.9	>600
CS	*	*	*	*	*

**Table IX.** Numbers of Cliques Produced with and without Distance Clustering in the Collection of Molecules Matching a Query Molecule of 24 Atoms

clique size	molecule size			
	13	14	24	34
No Clustering				
5	27	70	109	181
6	28	64	132	118
7	16	56	86	31
8	2	10	28	7
9	0	0	0	1
10	0	0	0	6
11	0	0	0	2
12	0	4	8	0
With Clustering				
5	2952	3368	15298	
6	380	516	2670	
7	168	176	442	
8	56	48	136	
9	16	24	32	
10	4	4	8	
11	0	0	0	
12	4	4	8	

(size 5 for the 15-atom query molecule). In the case of the clique-detection approach, two sets of experiments were carried out. The first set used the actual distances with an error tolerance of 0.15 Å in the construction of the correspondence graph; the second set adopted the Crandell-Smith approach with the distances clustered into groups with an error limit of 0.09 Å. The results in Table VIII again show that the clique-detection method is substantially faster in operation than the Crandell-Smith method, even when the clustered distances are used; when unclustered distances are used, the difference between the two methods becomes much more pronounced. The clique-detection method processes clustered distances much more slowly because of the greater number of cliques present in the correspondence graph. An example of this behavior is shown in Table IX, which gives the numbers of cliques identified when the distances were or were not clustered when the 24-atom molecule was compared with the four

**Table X.** Times (in CPU Seconds on a Prime 9950) for Identifying the MCS Using (A) Three, (B) Four, (C) Six, and (D) Nine Structures of Size-20 Atoms<sup>a</sup>

(A) Three Structures				
largest common substructure size				
	6	9	11	12
cluster	6.1 (1.1)	7.3 (2.0)	9.2 (4.1)	9.0 (3.9)
CS	16.5	36.1	72.8	103.4

(B) Four Structures				
largest common substructure size				
	6	8	9	12
cluster	23.0 (12.4)	17.6 (8.7)	16.0 (7.4)	17.7 (9.0)
CS	25.9	50.6	60.6	198.5

(C) Six Structures			
largest common substructure size			
	8	12	14
cluster	34.0 (15.6)	22.5 (7.3)	28.2 (16.4)
CS	98.7	245.1	*

(D) Nine Structures			
largest common substructure size			
	7	9	11
cluster	43.6 (14.0)	45.4 (15.0)	34.3 (11.0)
CS	209.0	206.5	*

<sup>a</sup>Times in parentheses are the times required by the INTER-SECT/UNION stage of the clique-detection procedure.

molecules that had cliques of size 7 or greater in common; the numbers of cliques are not listed for the 34-atom molecule with clustered distances since this run had to be aborted after 2400 CPU s.

The large number of cliques identified when the clustering method is used causes two major problems, especially with large molecules. First, the Crandell-Smith algorithm frequently exceeds the available storage, while the clique-finding method takes much longer to finish. Second, even if the algorithms do terminate, in a practical situation the common substructures that they output have to be evaluated for their significance, and this may be difficult if very different distances have been grouped together by the clustering procedure. Unfortunately, despite the shortcomings of the clustering approach, some way has to be used with Crandell and Smith's algorithm for converting the interatomic distances into integers so that the grown substructures can be compared with each other. It would, of course, be possible to use the actual distances, together with some error range, but then the computational requirements of the comparison stage would become totally impracticable. The Crandell-Smith algorithm should be better than the clique-finding method at dealing with large molecules, e.g., of size 40 non-hydrogen atoms or greater, because the correspondence graph for the clique-finding approach then becomes very large. However, even with such molecules, the Crandell-Smith approach could not be used if the MCS was at all large, owing to the storage problems that have been mentioned previously.

**Comparing More than Two Molecules.** The distortion procedure described above was used to generate sets of structurally related compounds to test the ability of the clique-finding approach to deal with more than two molecules; experiments were also carried out using the Crandell-Smith algorithm. An example of the results obtained are shown in parts A-D of Table X, which give the observed run times for the comparison of three, four, six, and nine molecules, respectively, of size 20 non-hydrogen atoms; further results are presented by Brint.<sup>20</sup> The order in which the molecules are processed affects the clique-finding approach because it compares the first molecule with each of the others in turn, and thus different orderings can result in different run times.

However, experiments showed that the variation in run time was very small, and thus just a single time is listed in the tables.

The times in parentheses in these tables correspond to the time taken by the INTERSECT/UNION stage of the clique-finding approach, and it will be seen that the  $M - 1$  comparisons of pairs of molecules generally take more time than the INTERSECT/UNION stage of the clique-finding method, except when the MCS is very large. Therefore, the efficiency of the clique-finding approach is likely to be roughly linearly related to the number of molecules under consideration.

## CONCLUSIONS

There is increasing interest in the development of techniques for the processing of 3-D chemical structure data bases,<sup>8-13,18,26,27</sup> in this paper, we have studied one such technique, this being the identification of the 3-D substructures common to sets of molecules. The clique-finding approach has been found to be considerably faster than the Crandell-Smith algorithm. Various different clique-detecting algorithms were compared in this context, and the widely used algorithm of Bron and Kerbosch was found to be considerably superior to the others; in addition, the approach has been extended to allow the processing of more than two molecules. However, finding cliques in large graphs is very demanding in terms of storage and time, and the algorithm of Crandell and Smith is thus more appropriate when large molecules need to be processed and when the MCS is quite small. A major limitation of this latter approach is that the distance-clustering stage results in distances being considered as equivalent even though their difference is quite large; the clique-finding algorithm, conversely, can use the original distances if required.

In their original paper,<sup>13</sup> Crandell and Smith suggested several enhancements to their basic algorithm, and it is of interest to see how these could be included in the clique-detection algorithm. Their first suggested enhancement was to allow the user of the system to specify a *seed pattern*, i.e., a set of atom types and associated distances that could be used as the starting point for the growth of the common substructures, rather than the individual atoms. A simple way of providing this facility would be to check for the presence of this seed pattern in each clique (of the same size as the pattern) as it is generated by using a geometric searching algorithm;<sup>12</sup> the further growth of the clique could then be terminated if the seed pattern proved to be absent. Crandell and Smith also suggested that it would be useful for the system to be run interactively, with the user having control over which common substructures are allowed to act as the basis for subsequent iterations of the algorithm. Such an approach is easy to implement in their algorithm since it is a breadth-first search, where all of the common substructures of size  $N$  atoms can be displayed prior to the generation of the common substructures of size  $N + 1$  atoms. The clique-detection algorithm, conversely, operates in a depth-first manner; thus, if the program discovered  $x$  cliques of size  $N$  atoms, the user would need to be polled  $x$  times, with a substantial increase in the running time of the program. One way around this would be for the user to specify certain constraints prior to program execution, e.g., to request the generation of all common substructures that contained at least two heteroatoms or to prune all common substructures that contained the pattern of atoms and distances which characterize a benzenoid ring. Finally, Crandell and Smith describe a mechanism for including stereochemical information in the naming stage. This is achieved by selecting a pattern of atoms that describe a plane through a molecule and then characterizing individual atoms and substructures by their orientations with respect to this plane. Although we have not explored this idea in detail, it could be

included in the clique-detection algorithm by using such stereochemical atomic identifiers in the construction of the correspondence table that is used for the generation of the cliques. Ghose and Crippen<sup>28</sup> discuss the use of stereochemical information in ligand binding studies with the algorithm of Kuhl et al.,<sup>18</sup> which is similar to the clique-detection algorithm reported here.

## ACKNOWLEDGMENT

We thank Michael Lynch, Eleanor Mitchell, and the referees for comments on an earlier draft of this paper and the Science and Engineering Research Council for the award of a Research Studentship to A.T.B.

## REFERENCES AND NOTES

- (1) Cone, M. M.; Venkataraghavan, R.; McLafferty, F. W. "Molecular Structure Comparison Program for the Identification of Maximal Common Substructures". *J. Am. Chem. Soc.* **1977**, *99*, 7668-7671.
- (2) Cammarata, A.; Menon, G. K. "Pattern Recognition. Classification of Therapeutic Agents According to Pharmacophores". *J. Med. Chem.* **1976**, *19*, 739-748.
- (3) Streich, W. J.; Franke, R. "Topological Pharmacophores. New Methods and Their Application to a Set of Antimalarials. Part 1: The Methods LOGANA and LOCON". *Quant. Struct.-Act. Relat.* **1985**, *4*, 13-18.
- (4) McGregor, J. J.; Willett, P. "Use of a Maximal Common Subgraph Algorithm in the Automatic Identification of the Ostensible Bond Changes Occurring in Chemical Reactions". *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 137-140.
- (5) Willett, P., Ed. *Modern Approaches to Chemical Reaction Searching*; Gower: Aldershot, U.K., 1986.
- (6) Bersohn, M.; Fujiwara, S.; Fujiwara, Y. "A Method for the Machine Detection of Near Equivalence of Major Substructures in a Molecule". *J. Comput. Chem.* **1986**, *7*, 129-139.
- (7) Willett, P. "Similarity and Clustering Methods in Chemical Information Systems"; Research Studies: Letchworth, U.K., 1987.
- (8) Gund, P. "Three-Dimensional Pharmacophoric Pattern Searching". *Prog. Mol. Subcell. Biol.* **1977**, *5*, 117-143.
- (9) Gund, P. "Pharmacophoric Pattern Searching and Receptor Mapping". *Annu. Rep. Med. Chem.* **1979**, *14*, 299-308.
- (10) Jakes, S. E.; Willett, P. "Pharmacophoric Pattern Matching in Files of 3-D Chemical Structures: Selection of Interatomic Distance Screens". *J. Mol. Graphics* **1986**, *4*, 12-20.
- (11) Jakes, S. E.; Watts, N.; Willett, P.; Bawden, D.; Fisher, J. D. "Pharmacophoric Pattern Matching in Files of 3-D Chemical Structures: Evaluation of Search Performance". *J. Mol. Graphics* **1987**, *5*, 41-48.
- (12) Brint, A. T.; Willett, P. "Pharmacophoric Pattern Matching in Files of 3-D Chemical Structures: Comparison of Geometric Searching Algorithms". *J. Mol. Graphics* **1987**, *5*, 49-56.
- (13) Crandell, C. W.; Smith, D. H. "Computer-Assisted Examination of Compounds for Common Three-Dimensional Substructures". *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 186-197.
- (14) Barrow, H. G.; Burstall, R. M. "Subgraph Isomorphism, Matching Relational Structures and Maximal Cliques". *Inf. Proc. Lett.* **1976**, *4*, 83-84.
- (15) Barrow, H. G.; Tenenbaum, J. M. "Computational Vision". *Proc. IEEE* **1981**, *69*, 572-595.
- (16) Levi, G. "A Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs". *Calcolo* **1972**, *9*, 341-352.
- (17) Golender, V.; Rozenblit, A. "Logical and Combinatorial Algorithms for Drug Design"; Research Studies: Letchworth U.K., 1983.
- (18) Kuhl, F. S.; Crippen, G. M.; Friesen, D. K. "A Combinatorial Algorithm for Calculating Ligand Binding". *J. Comput. Chem.* **1984**, *5*, 24-34.
- (19) McGregor, J. J. "Backtrack Search Algorithms and the Maximal Common Subgraph Problem". *Software Pract. Exper.* **1982**, *12*, 23-34.
- (20) Brint, A. T. "Matching Algorithms for Handling Three-Dimensional Molecular Coordinate Data". Ph.D. Thesis: University of Sheffield, 1987.
- (21) Bron, C.; Kerbosch, J. "Algorithm 457. Finding All Cliques of an Undirected Graph". *Commun. ACM* **1973**, *16*, 575-577.
- (22) Gerhards, L.; Lindenberg, W. "Clique Detection for Nondirected Graphs: Two New Algorithms". *Computing* **1979**, *21*, 295-322.
- (23) Loukakis, E.; Tsouros, C. "A Depth First Search Algorithm to Generate the Family of Maximal Independent Sets of a Graph Lexicographically". *Computing* **1981**, *27*, 349-366.
- (24) Loukakis, E. "A New Backtracking Algorithm for Generating the Family of Maximal Independent Sets of a Graph". *Comput. Math. Appl.* **1983**, *9*, 83-89.
- (25) Allen, F. H.; Bellard, S.; Brice, M. D.; Cartwright, B. A.; Doubleday, A.; Higgs, H.; Hummelink, T.; Hummelink-Peters, B. G.; Kennard, O.; Motherwell, W. D. S.; Rogers, J. R.; Watson, D. G. "The Cambridge Crystallographic Data Centre: Computer-Based Search, Retrieval,

- Analysis and Display of Information". *Acta Crystallogr., Sect. B: Struct. Crystallogr. Cryst. Chem.* **1979**, B35, 2331-2339.
- (26) Lesk, A. M. "Detection of 3-D Patterns of Atoms in Chemical Structures". *Commun. ACM* **1979**, 22, 219-224.
- (27) Esaki, T. "Quantitative Drug Design Studies. V. Approach to Lead Generation by Pharmacophoric Pattern Searching". *Chem. Pharm. Bull.* **1982**, 30, 3657-3661.
- (28) Ghose, A. K.; Crippen, G. M. "Geometrically Feasible Binding Modes of a Flexible Ligand Molecule at the Receptor Site". *J. Comput. Chem.* **1986**, 6, 350-359.

## An Expert System for Machine-Aided Indexing<sup>†</sup>

CLARA MARTINEZ,\* JOHN LUCEY, and ELLIOTT LINDER

American Petroleum Institute, 156 William St., New York, New York 10038

Received December 5, 1986

The Central Abstracting & Indexing Service of the American Petroleum Institute (API-CAIS) has successfully applied expert system techniques to the job of selecting index terms from abstracts of articles appearing in the technical literature. Using the API Thesaurus as a base, a rule-based system has been created that has been in productive use since February 1985. The index terms selected by computer are reviewed by a human index editor, as are the terms selected by CAIS's human indexers. After editing, the terms are used for printed indexes and for online computer searching.

### INTRODUCTION

The Central Abstracting & Indexing Service (CAIS) of the American Petroleum Institute (API) has produced indexes of patents and technical literature of interest to the petroleum and petrochemical industries since 1964. The documents to be indexed are selected from more than 150 journals in seven different languages and from about 50 recurring meetings and conferences. The abstractors on the CAIS staff select the documents according to specific rules and write an informative abstract of about 200 words. After being edited and proofread, the abstracts are used to prepare the alerting bulletins distributed to subscribers. At the same time, a copy of each abstract goes to the technical indexing staff for indexing.

Indexing is done by use of a controlled vocabulary. An indexer reads the abstract and tries to index all concepts using the valid terms in the API Thesaurus. The index is then edited to ensure accuracy and completeness (Figure 1).

Producing a high quality database using human indexers is expensive. For a long time CAIS had considered automated processing techniques to reduce the cost of indexing. Automated indexing as well as machine-aided indexing<sup>1-12</sup> was investigated.

In 1979, the API Special Task Force Comparing Automatic and Manual Indexing conducted a comparative study of retrieval of information from a system with automatic indexing versus that from the manually indexed API files. Later, the API Automatic Indexing Task Force was created to look into advanced techniques for information processing. These studies concluded that no existing system was suitable for our purposes. We therefore started our machine-aided indexing (MAI) project in 1982. It was felt that MAI would provide the flexibility necessary to handle the special features, i.e., the controlled vocabulary, roles, and links, of the API technical databases. The application of expert systems to a project of this type was also noted.<sup>13</sup>

The purpose of this paper is to present what we are doing at CAIS. It has been considered the first attempt to develop an automated indexing system that models human indexing behavior.<sup>14</sup>

### KNOWLEDGE BASE

The MAI system is a rule-based expert system; that is to say, we are using the knowledge of our experts to create additional indexing rules and to modify or delete existing ones.

The machine scans the text of the abstract of a document and compares it with text fragments stored in the Knowledge Base. When a match occurs and certain specified conditions are met, the Knowledge Base supplies the appropriate index terms from the API Thesaurus. We refer to the package of a specific text fragment with any special conditions and the associated index terms as a "rule". The original Knowledge Base consisted of the index terms and cross-references in the 1982 edition of the API Thesaurus.

An example of a rule in the Knowledge Base is

TEXT: NAPHTHA

TERM: NAPHTHA

The text found in the abstract is followed by the term to be indexed. In this case the text and the term are the same. In another example

TEXT: LNG

TERM: LIQUEFIED NATURAL GAS

the rule is derived from a cross-reference in the API Thesaurus. In this case the term to be used is not the same as the text.

In order to create additional rules, a batch of about 1000 abstracts of documents indexed in 1982 was run against the rules in the API Thesaurus. The MAI indexing for these abstracts was then compared to the human indexing. This first round of the MAI was analyzed for good terms selected (HITS), good terms not selected (MISSED), and bad terms selected (NOISE). Statistical data were produced for the NOISE and MISSED terms, and lists were produced for the terms in descending order of frequency (Figure 2).

The results of the first round were HITS 40% and NOISE 38%.

It was found at that point that punctuation and plurals were causing problems. Handling the irregular plurals and eliminating most of the punctuation raised the percentage of HITS to 44% and lowered the NOISE to 21%. By reviewing samples of the abstracts in which the high-frequency terms were errors, we could determine what changes were needed in the

<sup>†</sup> Presented, in part, before the Division of Chemical Information, Symposium on Applications of Artificial Intelligence in Chemical Information, 191st National Meeting of the American Chemical Society, New York, NY, April 15, 1986.