AND Thiocyanide_Add

## APPENDIX 3. DIFFERENT FORMATS FOR EXPERTISE

**Expertise in Text Format.**

1. After adding the masking reagents, if the difference between two metal ions distribution coefficient is larger than 4.5, then these two metal ions can be separated.
2. If these two metal ions can be separated, then check their distribution coefficients. The one needed to mask its distribution coefficient must be smaller than 0.01.
3. If the masked metal ion's extraction efficiency is smaller than 0.004, then this masking reagent can be used.
4. If this masking reagent can be used, then create the rules for the masking reagent and relevant metal ions.

**Expertise in Rules Format.**

1. IF Dis_Coe_Diff ≥ 4.5
   THEN Separable
2. IF Separable
   THEN Check_D AND Not_Extract_D ≤ 0.01
3. IF Not_Extract_E_Value ≤ 0.004
   THEN Masking_Reagent_Ok
4. IF Masking_Reagent_Ok
   THEN Create_Rule

## APPENDIX 4. QUERY PROCESS TO FIND CONDITIONS FOR $HG^{2+}$ AND $CU^{2+}$ SEPARATION

⟨**user query**⟩SELECT (Ke- ∝ + 2 Log_Chelate_Conc[a] + 2[b])/2
FROM BROMIDES,CHELATES
WHERE CHELATES.METAL = BROMIDES.METAL
AND CHELATES.METAL = 'Hg2+'
⟨**system response**⟩pH = 0.7

Footnotes for the query above. (*a*) Log_Chelate_Conc stands for the log value of the concentration of the dithizone in carbon tetrachloride. (*b*) 2 is the minus log value of 0.01; the value of 0.01 is from rule 2 from Appendix 3.

## APPENDIX 5. RULE CREATED FOR $HG^{2+}$ - $CU^{2+}$ SEPARATION BY USING BROMIDE AS MASKING REAGENT

RULE CREATED:
IF Metal_Ions = Cu2+ AND Condition = In_Hg2+
AND Extractant = Dithizone
THEN Bromide_Add AND pH ≤ 0.7

## REFERENCES AND NOTES


(1) Sawyer, C. N.; McCarty, P. L. *Chemistry for Environmental Engineering*, 3rd ed.; McGraw Hill: New York, 1980; p 514.
(2) Irving, H. *The Solvent Extraction of Metal Chelates*; The Macmillan Company: New York, 1964.
(3) Chalmers, R. A. *Solvent Extraction of Metals*; Van Nostrand Reinhold: London, 1970.
(4) Keyes, J. *AI Expert* **1989**, *4* (2), 60.
(5) Kitayama, S. *DEC Professional* **1988**, *July*, 52.
(6) Naecker, P. *DEC Professional* **1989**, *May*, 58.
(7) Schlieper, W. A.; Marshall, J. C.; Isenhour, T. L. Using Analytical Data to Build Expert Systems. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 159.
(8) Hu, D. *C/C++ for Expert Systems*; MIS: Portland, 1989; p 497.
(9) Barr, A.; Feigenbaum, E. A.; Cohen, P. R. *The Handbook of Artificial Intelligence*; William Kaufmann: Los Altos, CA, 1981; Vol. 1, p 23.
(10) Barr, A.; Feigenbaum, E. A.; Cohen, P. R. *The Handbook of Artificial Intelligence*; William Kaufmann: Los Altos, CA, 1981; Vol. 1, pp 192, 197.
(11) Inczédy, J. Tyson, J., Translation Ed.; *Analytical Applications of Complex Equilibria*; John Wiley & Sons Inc.: New York, 1976.
(12) Dehne, T. *Anal. Chem.* **1990**, *62*, 565A.


# Automorphism and Equivalence Classes

JOHN FIGUERAS
65 Steele Road, Victor, New York 14564

Received November 19, 1991

Previously described determinations of equivalence classes of atoms in molecules have been based upon the use of graph invariants. Because invariants do not adequately convey all of the information present in a connection table, more or less complicated methods of refinement must be applied to compensate. A method for finding equivalence classes is presented that is theoretically complete because it is based on all possible mappings of a graph onto itself. This method, usually considered to be computationally impractical, is in fact a useful approach to the determination of topological symmetry. The procedure uses preliminary processing based on a few graph ivariants and the Morgan algorithm to induce an initial partition, which markedly accelerates subsequent mappings of a graph onto itself. As new symmetries are discovered, they are used to induce new partitions, which accelerates the process even more. In recalcitrant cases, where the usual graph invariants of atom identity and bond distribution are not sufficient to induce partitioning, the determination of ring membership may be useful.

## INTRODUCTION

The determination of equivalence classes for atoms in a structure is equivalent to finding mappings of a structure onto itself (automorphisms), a special case of determining graph isomorphism. This can be seen from the following argument. Two atoms $x$ and $y$ are defined as equivalent if and only if their interchange leaves the structure unaltered. Operationally, this may be implemented by showing that the original structure is superimposable on an exact copy when atom $x$ in the original is overlaid on atom $y$ in the copy (automorphism). Methods for determining isomorphism (automorphism) are discussed in a paper by Read and Corneil.[1] Their work suggests that the only theoretically sound determination of isomorphism is that based upon atom-by-atom matching using a backtracking search algorithm, generally considered a brute-force solution. To avoid this approach, previous workers have tried to define different calculable atom parameters (graph invariants) that rigorously distinguish between nonequivalent atoms. If such a graph invariant could be found, then atoms having common values of such an invariant could rigorously be assigned to the same equivalence class. Read and Corneil showed that definition of such an invariant is equivalent to determining iso-

0095-2338/92/1632-0153$03.00/0 

| ORIG, i | ATOMSET[i] |
|---|---|
| 1 | a c d e f g |
| 2 | b |
| 3 | a c d e f g |
| 4 | a c d e f g |
| 5 | a c d e f g |
| 6 | a c d e f g |
| 7 | a c d e f g |
| 8 | h |

**Figure 1.** ATOMSETs containing correspondents with respect to atom type and bond distribution.

morphism. They also concluded that graph invariants alone are unreliable as a basis for determining isomorphism (and equivalence classes) because no invariant adequately embodies the whole environment of a given vertex, which is the graph itself. Recent papers illustrate the problems that arise when graph invariants are used for determination of topological symmetry.[2-4] In this paper I will show that the use of theoretically rigorous atom-by-atom matching in fact leads to a practical method for finding topological symmetry when applied to chemical structures. The final algorithm uses preliminary classification (partitioning) of atoms based on atom and bond properties (or ring membership) and takes advantage of discovered symmetries to reduce the computational burden.

## THE ALGORITHM

The connection table used to represent the chemical structure is an adjacency list, an $n \times 6$ array; $n$ is the number of atoms in the structure, and each may have up to 6 connections. Two atom properties, *atomic number* and *composite bond code* (see Implementation section), describe the nature of each atom and the distribution of bonds around each atom. In some cases, discussed later, *ring membership* is used in place of composite bond code. Two copies of the connection table and atom properties, called ORIG and COPY in this paper, are required. For each atom $i$ in ORIG, the program evaluates a set, ATOMSET[i], that contains those atoms in COPY having the same atom properties as atom $i$. Each ATOMSET[i] may be viewed as a first approximation to the equivalence class containing atom $i$. Figure 1 displays a simplified illustration of ATOMSETs that uses atomic number and composite bond codes as a basis for classification (in the final application, Morgan extended connectivities rather than composite bond codes are used). ATOMSETs serve two purposes: they express the partitioning of atoms based on atom properties, which accelerates the determination of equivalence classes, and they supply candidate atoms for carrying out the atom-by-atom search. Final equivalence classes are contained in an array of sets, EQUIV[i]; i.e., EQUIV[i] contains those atoms equivalent to $i$. Initially, EQUIV[i] is initialized to a singleton set [i]; at a minimum, a given atom must below to its own equivalence class.

As argued above, if atoms $k$ and $m$ are equivalent, then COPY and ORIG will be superimposable if atom $m$ in COPY overlays atom $k$ in ORIG. We may regard the original structure ORIG as a *template* structure to be overlaid on COPY. Each ATOMSET[i] may contain several atoms from COPY that are possible correspondents to atom[i]. The constraint that atom $k$ in ORIG overlays on atom $m$ in COPY is invoked by replacing ATOMSET[k] with the singleton [m]. The process of overlaying ORIG onto COPY is carried out by means of atom-by-atom search, which in effect superimposes the two by producing a list of one-to-one correspondences between the atoms in COPY and the atoms in ORIG. Since the atom-by-atom search draws its candidate atoms from ATOMSETs, the assignment ATOMSET[k] := [m] imposes the required constraint. The pair of atoms $k$ and $m$ will be referred to as a *constrained pair*. A successful mapping of COPY onto ORIG for a particular constrained pair implies that the atoms in the constrained pair must, by definition, be equivalent. Theoretically, this process should be carried out for all possible pairs of atoms $(k,m)$ in ORIG and COPY. Partitioning reduces the number of pairs drastically, and further reduction occurs because a successful mapping also produces $n - 1$ correspondences between the remaining atoms in ORIG and COPY, which likewise must be considered equivalent. All of these equivalences are added to the equivalence sets EQUIV. Once an atom has been assigned to any EQUIV[i], it is no longer eligible for use in a constrained pair with atom $i$. If an assignment fails for a constrained pair $(k,m)$, the nonmatching atom $m$ may be removed from ATOMSET[k]. As equivalences are added to the equivalence sets EQUIV and nonmatching atoms are removed from ATOMSETs, the sets EQUIV and ATOMSET become identical, which terminates the algorithm. In pseudo code, the algorithm is stated as follows:

Let $N$ be the number of atoms in the structure and initialize each ATOMSET[i] to the set of atoms in COPY that correspond to atom $i$ in ORIG with respect to atom properties (atomic number and Morgan extended connectivities based on composite bond codes). Initialize each set EQUIV[i] to the singleton set [i]:

```
for i := 1 to N do
  while ATOMSET [i] <> EQUIV[i] do
    1. Extract an atom k from the difference set
       ATOMSET[i] - EQUIV[i].
    2. Backup ATOMSET[i]. Replace ATOMSET[i] with
       singleton set [k].
    3. Carry out an atom-by-atom search that maps ORIG
       onto COPY and record the one-to-one correspond-
       ences produced by the search in an array, MATCH,
       where each MATCH[j] contains the atom in COPY
       corresponding to atom j in ORIG.
    4. If the search is successful do the following:
         For each atom m = i to N do if MATCH[m] is not
       in EQUIV[m] then
          begin
         Add MATCH[m] to EQUIV[m].
         For each atom j in EQUIV[m] add EQUIV[m] tp
           EQUIV[j].
          end
         else if the search is not successful then do following:
         Remove atom k from the backup set (see step 2).
         For each atom m = i to N do if atom m is in
           EQUIV[i] then remove k from ATOMSET[m].
    5. Restore ATOMSET[i] from backup and return to
       the top of the WHILE loop.
```

Steps 1 and 2 select an unused constrained pair. In the event of a successful mapping, step 4 adds newly uncovered correspondences in COPY to the equivalence sets EQUIV. Step 4 also updates the equivalence classes corresponding to each atom in the updated classes. The argument behind this second step is as follows: if atom $i$ is equivalent to atom $k$, then atom $k$ must be equivalent to atom $i$: less trivially, if the set EQUIV[i] contains atom $k$ then the set EQUIV[k] must contain atom $i$. In the event that the mapping of COPY onto ORIG fails, the ELSE block in step 4 removes the nonmatching atom $k$ from ATOMSET[i], and also removes $k$ from all atom sets attached to atoms equivalent to $i$. The reasoning behind this latter step is that if atom $k$ is a nonmatch to atom $i$, it must also be a nonmatch for all atoms equivalent to $i$. The atom-by-atom search is described in the Appendix.

## IMPLEMENTATION

The computer program was written in Turbo Pascal, v 5.5, for an IBM-type personal computer (80286 CPU operating at 10 mz, with zero wait states). The program provides a graphical user interface for convenient structure input and generation of connection tables. An atomic number and composite bond code is assigned by the program to each atom of the input structure. The composite bond code is computed as the product of individual bond codes for each bond (to non-hydrogen substituents) around a given atom. The individual bond codes are prime numbers (2 for a single bond, 3 for a double bond, 5 for a triple bond, and 7 for an aromatic bond); the product of bond codes for a given atom produces a unique number characterizing the bond distribution around the atom. The composite bond codes are used as the basis for computation of "extended connectivities", by the Morgan algorithm.[5] The original Morgan algorithm used only the number of non-hydrogen bonds at each atom: the extended form in this work employs the composite bond code at each atom, and therefore includes the bond-type distribution around each atom.

Computational results are recorded for both ORIG and COPY connection tables. The Morgan algorithm is used as described in the original paper; i.e., it is applied iteratively until it fails to produce a finer partition.

When all atoms are of the same type and have the same composite bond codes, Morgan partitioning fails because each cycle of the algorithm gives a uniform increase in the Morgan sum. Even in such recalcitrant cases, atom-by-atom searches will succeed in discovering equivalence classes, but processing may be lengthy. In these cases, it may be useful to determine ring membership as a basis for partitioning (Appendix). It should be noted that contrary to the role played by partitioning in other topological symmetry algorithms, partitioning here is not central to finding equivalence classes, but is used to reduce the computational burden.

## RESULTS AND DISCUSSION

Computer trials are reported in Table I for a set of structures (see Chart I for the structures) extracted from two recent publications by Günther and Günther,[3,4] who developed an algorithm for the determination of topological symmetry based upon analysis of successive powers of the adjacency matrix. These structures include several produced as "challenges" by previous authors. On the whole, processing times are satisfactorily short, and the numbers of classes extracted by the program agree in every case with those previously reported. Not shown in the table is a trial with a structure having 118 atoms and 45 fused six-membered rings, which required 5.11 s of processing time. This was the largest structure tried.

Structures **21–23** show appreciable decreases in processing time as a result of ring partitioning, particularly for structure **22**. However, ring partitioning cannot always be counted on to improve efficiency; with structure **20**, ring partitioning actually increased processing time by about 4 s (results not reported in Table I). The computation shows some dependence upon the sequence of numbering of the atoms in the structure: processing time for structure **22** without ring partitioning was reduced from 8 s to 2 s simply by renumbering (compare **22** and **23** in Table I with ring partitioning disabled). Structures **25** and **26**, examples of highly-connected structures, show no anomalous behavior. Structure **28** is included to illustrate application of the algorithm to a typical complex molecule, morphine.

**Table I.** Equivalence Class Determinations

| structure | *n* | classes | CPU time, s |
|---|---|---|---|
| **1** | 8 | 8 | 0.33 |
| **2** | 8 | 2 | 0.11 |
| **3** | 8 | 3 | 0.38 |
| **4** | 8 | 4 | 0.41 |
| **5** | 12 | 7 | 1.32 |
| **6** | 16 | 3 | 2.47 |
| **7** | 18 | 6 | 2.14 |
| **8** | 17 | 17 | 2.77 |
| **9** | 20 | 1 | 2.72 |
| **10** | 25 | 3 | 0.67 |
| **11** | 25 | 5 | 1.10 |
| **12** | 8 | 6 | 0.22 |
| **13** | 9 | 9 | 0.04 |
| **14** | 10 | 3 | 0.16 |
| **15** | 11 | 6 | 0.79 |
| **16** | 14 | 14 | 0.11 |
| **17** | 18 | 8 | 0.38 |
| **18** | 18 | 8 | 1.17 |
| **19** | 30 | 16 | 0.44 |
| **20** | 42 | 14 | 25.10 |
| **21** | 8 | 3 | 0.44 |
| **21** | 8 | 3 | 1.54[a] |
| **22** | 10 | 3 | 0.93 |
| **22** | 10 | 3 | 8.02[a] |
| **23** | 10 | 3 | 1.11[b] |
| **23** | 10 | 3 | 2.30[a,b] |
| **24** | 19 | 19 | 0.34 |
| **25** | 18 | 1 | 1.89 |
| **26** | 8 | 2 | 0.68 |
| **27** | 24 | 2 | 0.84 |
| **28** | 21 | 21 | 0.22 |

[a] Ring partitioning turned off. [b] Structure **23** with renumbered atoms. Note: Structures **1–11** appear in ref 3. Structures **12–21** appear in ref 4.

## APPENDIX: RING FINDING AND SUBSTRUCTURE SEARCH

A modification of Zamora's path-tracing algorithm[6] was used to find the number and sizes of rings to which each atom belongs. An arbitrary limit of seven was placed on the size of the largest ring to be discovered to keep search times under control. For purposes of partitioning, not all rings have to be found, but rings within the limited set must not be overlooked or incorrect partitioning will result. The ring-detecting algorithm uses an exhaustive, backtracking depth-first search to find all linearly independent smallest rings with fewer than eight members, starting from each ring closure bond in the structure. The following simple routine, which does not appear to have been previously described, was used to find ring closure bonds directly from the connection table:

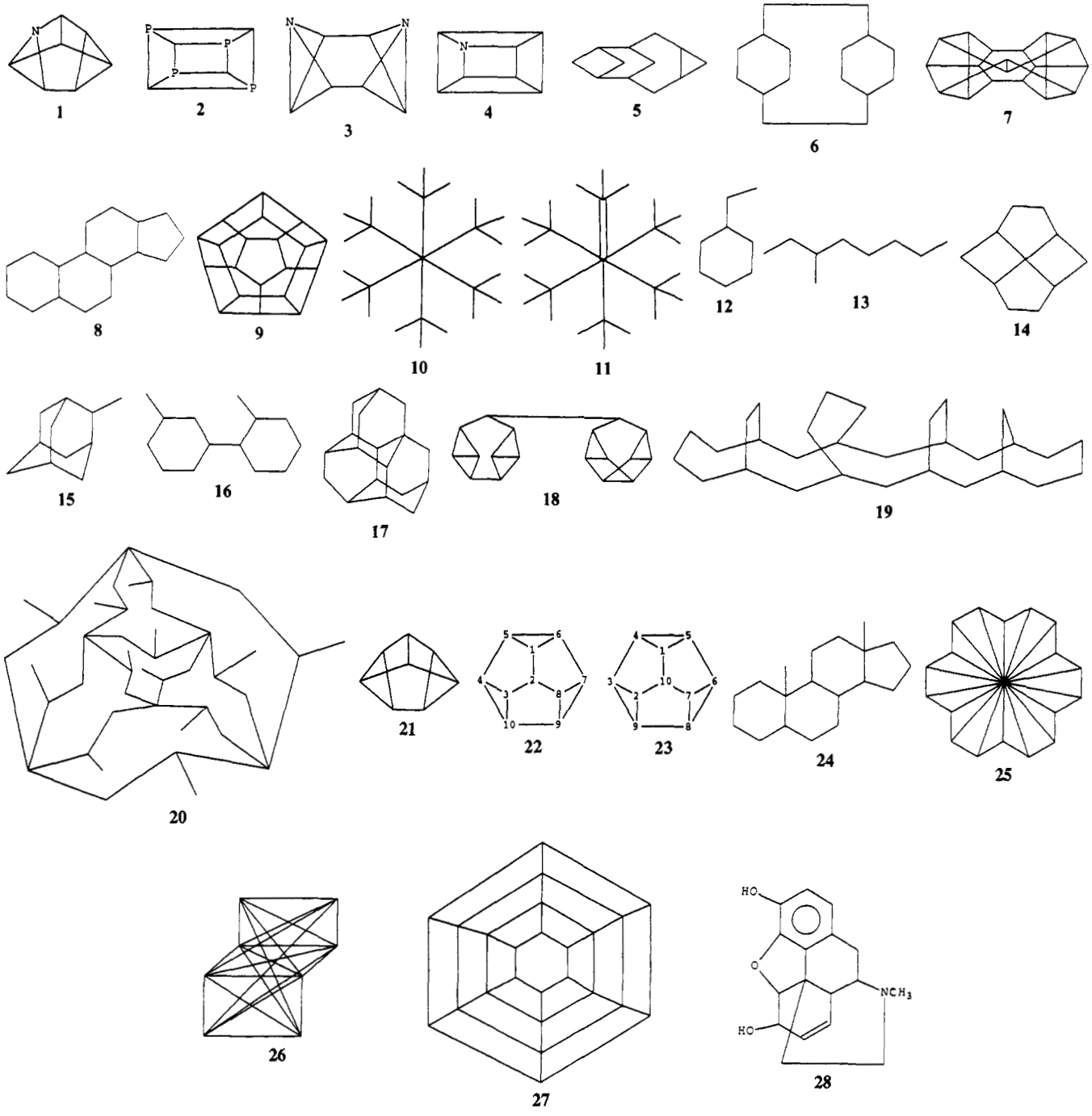
For every atom $i$ (row $i$) in the connection table:

(1) for each atom $k$ attached to atom $i$, if atom $i$ and atom $k$ have been marked, and $k > i$, then add them to the list of ring closure bonds.

(2) mark atoms $k$ and $i$.

The atom-by-atom search is a standard backtracking algorithm as described in works on graph theory.[7] Two features were implemented to improve programmability and efficiency. The ORIG connection table is renumbered by means of a depth-first traversal of the structure and made nonredundant (connections between two atoms appear only once in the table). The depth-first renumbering produces a new connection table containing no forward references; as will appear later, this is important for simplifying implementation of the backtracking search algorithm. The second feature is the establishment of an array of sets, ATOMSET[i], to contain those atoms in COPY that are possible correspondents to atom $i$ in ORIG. The ATOMSETs, conveniently implemented as Pascal data structures of type *set*, serve as repositories for candidate atoms used by the backtracking algorithm. Initial sets are formed by applying two steps of the Sussenguth set reduction algorithm.[8,9] In the first step, for each atom $i$ in ORIG, those atoms in COPY that have the same atomic number and extended Morgan connectivity are assigned to ATOMSET[i]. The

Chart I




second step refines atom sets by applying a connectivity requirement[9] closely related to the Ullmann algorithm:[10]

(1) Form the set O of atoms *a*, *b*, *c*, ... attached to atom *i* in ORIG by bonds of a given bond type T.
(2) Record in set C those atoms in COPY attached to each atom in ATOMSET[*i*] by a bond of type T. (Set C contains possible correspondents for each atom in set O.)
(3) For each atom *a*, *b*, *c*, ... in set O, compute the intersection of ATOMSET[a], ATOMSET[b], ... with set C.

The last step will often result in the reduction of ATOMSETs; the resulting atom sets now contain atoms that correspond to atom *i* with respect to atom type, Morgan connectivities (based on composite bond codes), and nearest neighbors bonded by a particular bond type.

The backtracking algorithm[7] is stated succinctly as follows: Assume that one-to-one correspondences have been found between atoms in COPY and the first *k* atoms in ORIG; the list of correspondences will be maintained in a vector *F*. Proceed as follows:

(1) Select atom $k + 1$ from ORIG.
(2) Seek a candidate atom in COPY that corresponds to the selected atom.
(3) If the seek in step 2 succeeds, increment *k*, store the index of the candidate in F[k].
If $k = N$ (the number of atoms in ORIG) then declare a match and quit, otherwise return to step 1.
(4) If the seek in step 2 fails, then decrement *k*.
If $k = 0$, then declare failure and quit, otherwise return to step 1.

Selection of a candidate atom, step 2, is based on the following considerations. Let us assume that correspondences, recorded in the array *F*, are known for the first *k* atoms of ORIG, these atoms occurring in the first *k* rows of the ORIG connection table. We will assume also that there are no forward refer-

ences in the connection table; i.e., all of the entries in the ORIG connection table through the $k$th row refer to atoms for which correspondences have been found, and therefore have entries in the array $F$. The depth-first renumbering of the connection table referred to previously is applied exactly for this purpose: to produce a nonredundant connection table that contains no forward references. On this basis, selection of a candidate node corresponding to the $k + l$th atom proceeds as follows:

(1) For each atom $m$ attached to atom $k + 1$ in ORIG do:
  (a) Find the corresponding atom, $c$ = F[m], in COPY. Because there are no forward references, $c$ is guaranteed to exist.
  (b) Form a set $P_m$ containing all atoms $p$ for which $p$ is attached to $c$ and $p$ is in ATOMSET[k+1].
(2) The set of candidates corresponding to atom $k + 1$ is the intersection of all of the sets obtained in step b, above. Return one of these nodes as a candidate node, holding the rest in reserve for backtracking.

The argument behind these steps is straightforward: atom $m$ is attached to $k + 1$; atom $c$ corresponds to atom $m$; atom $p$ is attached to atom $c$, and if it is contained in the set ATOMSET[k+1], then atom $p$ is a possible correspondent to atom $k + 1$. Each connection $m$ produces its own set $P_m$ of possible correspondents $p$ to $k$, hence step 2 is invoked, requiring that all of these sets $P_m$ must jointly contain correspondents to atom $k + 1$.

The performance of the atom-by-atom search alone was timed for 100 iterations of the mapping of a given structure ORIG onto its COPY, using 22 structures spanning a size range of 10-120 atoms, on a microcomputer with an 80386D microprocessor (33 mHz clock). The structures were mostly ring assemblies, with a few large, branched alicyclic structures. Processing times per 100 iterations ranged from 2.09 s for the smallest structure to 63.61 s for the largest. A plot of log $t$ vs log $n$ was quite linear with some scatter. The approximate equation of the line was $t = n^{1.45/20}$. No great precision is claimed for these results, but they do give an order-of-magnitude estimate of performance.

## REFERENCES AND NOTES


(1) Read, R. C.; Corneil, D. G. The Graph Isomorphism Disease. *J. Graph Theory* **1977**, *1*, 339. See also Carhart, R. E. Erroneous Claims Concerning the Perception of Topological Symmetry. *J. Chem. Inf. Comput. Sci.* **1978**, *18*, 108-110.
(2) Liu, X.; Balasubramanian, K.; Munk, M. E. Computational Techniques for Vertex Partitioning of Graphs. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 263-269.
(3) Rücker, G.; Rücker, C. Computer Perception of Constitutional (Topological) Symmetry: TOPSYM, a Fast Algorithm for Partitioning Atoms and Pairwise Relations among Atoms into Equivalence Classes. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 187-191.
(4) Rücker, G.; Rücker, C. Isocodal and Isospectral Points, Edges, and Pairs in Graphs and How To Cope with Them in Computerized Symmetry Recognition. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 422-427.
(5) Morgan, H. L. The Generation of a Unique Machine Description for Chemical Structures—A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, *5*, 107.
(6) Zamora, A. An Algorithm for Finding the Smallest Set of Smallest Rings. *J. Chem. Inf. Comput. Sci.* **1976**, *16*, 40.
(7) Nijenhuis, A.; Wilf, H. S. *Combinatorial Algorithms*; Academic Press: New York, 1978.
(8) Sussenguth, E. H., Jr. A Graph-Theoretic Algorithm for Matching Chemical Structures. *J. Chem. Doc.* **1965**, *5*, 36.
(9) Figueras, J. Substructure Search by Set Reduction. *J. Chem. Doc.* **1972**, *12*, 237.
(10) Willett, P.; Wilson, T.; Reddaway, S. F. Atom-by-Atom Searching Using Massive Parallelism. Implementation of the Ullmann Subgraph Isomorphism Algorithm on the Distributed Array Processor. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 225-233.


# Limits of Classification. 2. Comment on Lawson and Jurs


LOUIS HODES

National Cancer Institute, Bethesda, Maryland 20892


Received February 20, 1991


Lawson and Jurs[1,2] have two recent papers on clustering using data on a diverse set of 143 published acrylates. From our work[3] on clustering large diverse sets of compounds, we have found that a diverse set of compounds will generally have a dual nature—some clustered and some scattered compounds. Are the five published[2] clusters a good classification of that set of compounds? A further analysis throws more insight into the interrelations among the acrylates.


## INTRODUCTION

Lawson and Jurs[1,2] have two recent papers on clustering using data on a diverse set of 143 published acrylates. Their first paper[1] explores clustering tendency with a modified Hopkins method. Their second paper[2] performs the clustering by the Kmeans and Isodata methods.

Their purpose in clustering was to guide the sampling of compounds for toxicity testing. One compound from each cluster can be chosen on the assumption that the other compounds in the same cluster should have similar toxicity.

From our work[3] on clustering large diverse sets of compounds, we have found that a diverse set of compounds will generally have a dual nature—some clustered and some scattered compounds, the distribution dependent on the criteria for clustering. It is our contention that algorithms which insist on a complete clustering of a diverse set will often force disparate compounds into unsuitable clusters.

There is a question as to how far this phenomenon, which we called a limit to classification, shows up in a relatively small set such as the 143 diverse acrylates. Are the five published[2] clusters a good classification of that set of compounds?

Here we fill in some gaps in the Lawson and Jurs work. A more precise analysis results in more flexibility in achieving their goal of sampling acrylates for toxicity testing.

Our relevant paper,[3] the third in a series on clustering a large number of compounds, was subtitled "the limits of classification". This paper can be considered number 4 in the series on clustering, but it is more fitting as the second paper dealing with the problem of classification. Also, this paper deals intimately with the data and methods of Lawson and Jurs[1,2] to the extent that it requires the reader to be somewhat