Table I. Statistics of CAS Registration

FILE	# COMPOUNDS INPUT	# NAMES MATCHED	% NAME MATCH	# STRUCTURES MATCHED	# STRUCTURES REGISTERED	# NOT REGISTRABLE
MASS SPEC.	49160	17097	34.7	31418	7279	645
XRAY	11148	2527	22.7	8621	1096	0
CNMR	2824	1887	66.8	909	63	28
REGISTERED PESTICIDES	2608	2028	77.7	221	147	359
OIL & HAZARDOUS MATERIALS	915	758	82.8	116	98	41
PESTICIDE ANALYTICAL STDS.	389	344	88.4	45	2	0
DRINKING WATER	309	213	68.9	85	10	1

When the entry is found in this file, the REGN, name, and molecular formula for the compound are retrieved, but if the structure is not in the file, a new REGN is assigned, and the chemical is then registered by CAS and a Ninth Collective Index name is generated. If it is not possible to generate a structure from the information initially provided, consultation with the contracting agency ensues, and if a structure still cannot be defined, the entry is put aside and does not receive a CAS REGN. At present CAS is devising a method whereby these "unregistrable" chemicals (e.g., asphalt) can be assigned a REGN; however, implementation is still a number of months away.

At present, the compounds in seven data bases have been completely registered by CAS, and those in several other files are in the process of registration. The CIS files that have been completed are the MSDC–EPA–NIH Mass Spectral Data Base,[1] the EPA–NIH Carbon-13 Nuclear Magnetic Resonance Data Base,[2] and the Cambridge Crystal Structure file.[3] The EPA files that have been registered are the Pesticide file, the Oil and Hazardous Materials file, the file of Toxic Substances in Water, and the Pesticide Reporting file. Table I gives details of these files, including the number of compounds in each one, the number of successes in each step of the matching process and the number of "materials" that were not assigned CAS REGN, either because no structure could be drawn or because an undefined mixture of isomers was involved.

Once a complete file has been passed through the registration process, the next steps in the duplication removal are fairly clear. The file must first be sorted on the REGN to identify duplicates. Then the quality of each data set for a given REGN must be established. In the case of the mass spectral data base, this is done by means of a program developed by McLafferty and co-workers, which calculates a "quality index" for each spectrum. Finally, all but the best of a series of spectra associated with the same REGN are discarded. Work is now in progress on this phase of the process and is continuing without any notable difficulty.

ECONOMICS

The cost of registration is variable because the amount of work in each case is to some extent unpredictable. With the files discussed above, the cost of obtaining a REGN and the connection table for each compound has averaged about $3.00. This is a high figure, but one which, for the reasons enumerated below, is countenanced. By careful planning of the workload, it has been found that a stream of compound names moving at the rate of about 2000 per month can handled comfortably by CAS.

CONCLUSIONS

The benefits that accrue from the CAS registration process as it is described here are varied, and depend to some extent upon one's perspectives. From the point of view of the CIS system managers, there are some immediate gains, viz., the REGN and a reliable connection table which can be used in the CIS substructure programs. Perhaps the most important benefits are in the longer term. The reliability of the entire CIS, or at least of its data, is enhanced by the addition of the REGN's, and the value of the interfile link that is provided by the REGN's is basic to the structure of the CIS.

From the point of view of the user, the most obvious change is that duplicates are removed from the files. The more important advantages are that compounds with ill-defined or undefinable structures are no longer proffered to him as "answers" to his searches and that the answers which are suggested can be linked to the standard chemical literature through the REGN.

ACKNOWLEDGMENT

One of us (S.R.H.) wishes to thank M. Yaguda, M. Springer, and W. Greenstreet for their support of this project.

REFERENCES AND NOTES

(1) R. S. Heller, G. W. A. Milne, R. J. Feldmann, and S. R. Heller, *J. Chem. Inf. Comp. Sci.*, **16**, 176 (1976).
(2) B. A. Jezl and D. Dalrymple, *Anal. Chem.*, **47**, 203, (1975).
(3) O. Kennard, D. G. Watson, and W. G. Town, *J. Chem. Doc.*, **12**, 14 (1972).

A Flexible Interactive Graphics System for Searching Atom Connectivity Matrices

BO E. H. SAXBERG, DANIEL S. BLOM, and B. R. KOWALSKI*
Department of Chemistry, University of Washington, Seattle, Washington 98195

Received April 14, 1976

An interactive screen-generating structure search system with graphic input and display capabilities is described. The system allows great flexibility in searching atom connectivity matrices for interactively defined substructures or complete molecular structures. Host computer programs are written in UCI Lisp and Fortran, and are accessed via an intelligent graphics terminal. Results on a small collection of molecules are presented and potential applications are discussed.

Large molecular data bases have become necessary in many areas of chemistry. Computerized spectral analysis, for example, requires that a large file of molecules and their spectra be kept for reference. Pharmaceutical companies and other industries[1,2] dealing with molecular design need to keep files of molecules that have been studied in the past, as well as those undergoing current research. Large molecular data bases are also necessary for pattern recognition applications,[3] e.g., using

features of many molecules to establish correlations to biological activities.

Often one does not want to study all of the molecules in a large data base at one time, but rather to select a subset of the data base. As this selection is frequently made on the basis of structural features, substructure searching programs which search through a library of molecules for the occurrences of certain substructural features are of current interest.[4,5] One difficulty in handling large data files is to make this selection process as fast as possible, and at the same time to have all of the necessary information present in the data. Search routines can generate a file containing condensed information based on the structural features of the molecules which are structurally described in the data base.

There have been several approaches to the representations of molecular structures and several approaches to the search routines. One of the most condensed representations is the simple screen, usually a bit screen with a bit turned on or off depending on whether a particular substructure is present in a molecule (though it can take other forms, e.g., a list of integers). The searching of screens is limited in their description of the structure of the molecule. In addition, if the screens are encoded by hand for each molecule, there is an error factor that must be taken into consideration.

Fragment codes contain more information than screens, depending on the complexity of the code used.[6,7] For example, they can give information on interconnections between the various substructures used to make up the code. Structure searches can be performed on the code, looking for various symbols or groups of symbols, but this will naturally be slower than matching numbers in a screen. Fragment codes still do not necessarily provide complete information on the structure of a molecule, since they are limited by the set of fragments chosen for the code.

Linear notations provide complete and unique information on the structure of a molecule in a line of code. The most common linear notation is Wiswesser Line Notation[8] (WLN), though there are others (IUPAC, for example). Searches can be performed on WLN (and other notations) to find substructures, though there are some difficulties.[8,9] In addition, if the search routine is to be interactive, either the user must learn the code, or encoding and decoding programs must be written.

Connection tables[8] specify completely the actual interconnections between the various atoms in the molecule. The structure is thus totally and uniquely represented in tables (except for stereoisomers, which may be handled by designating certain atoms as being above and below an imaginary plane passing through the molecule) which are readily used in searching for any substructure. A connection table is necessary for an atom-by-atom iterative search routine, such as Meyer's routine,[10] or for a set reduction search routine, such as Sussenguth's routine.[11] Because of the detail of description (and in some cases redundancy in information), connection tables tend to take up a lot of storage space, more than most other representations.

We have developed a system that interactively allows the user either to select molecules from a file of connection tables on the basis of substructural features through an atom-by-atom recursive search, or to create a file of screens based on substructures entered by the user. The file of screens can then be used for preliminary screening before performing an atom-by-atom search. The molecules in the connection table file are drawn with a light pen on a graphics terminal, so entries are fast and errors are easily detected and corrected (Figure 1). Another approach is to enter the substructures through some sort of scanning system, such as Woodward and Isenhour have developed, which would reduce on-line time to

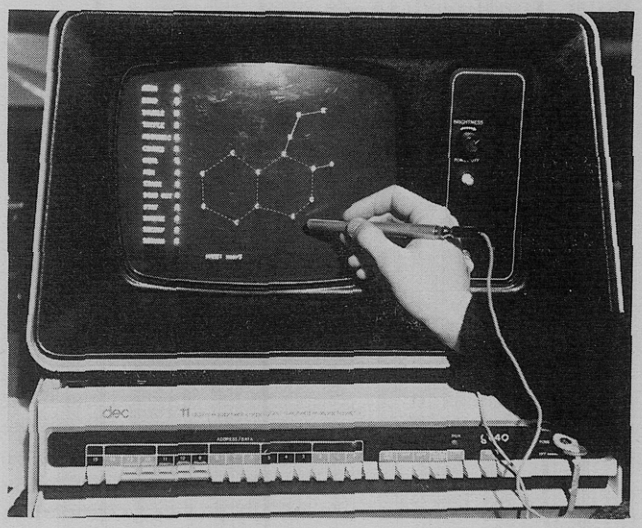

Figure 1.

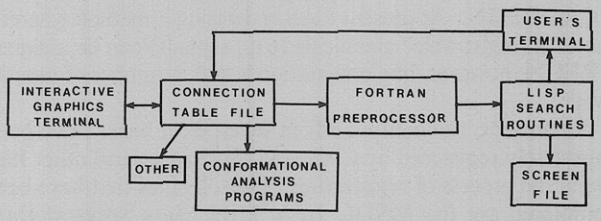

Figure 2.

a minimum.[12] However, as we are interested in creating an interactive system, it makes sense to give the user the ability to fully use the system through a single terminal, and a CRT terminal is ideal for this approach.

The substructures used in the search can be entered in two ways. The first method is used during the operation of the search program, when the substructures are entered in essentially a connection table format. This is fast and convenient for small substructures, and the code is transparent to the user. As this method requires only the simplest of computer terminal keyboards, it allows the user to search a file even if he may not have a graphics terminal available (for example, searches may be conducted from a portable terminal in the laboratory, etc.)

The second method is used when searching for large substructures or when identifying a whole molecule; it allows the user to create a file of connection tables for the substructures (or molecules) to be searched for, by drawing the substructures on the graphics terminal (just as the molecules were drawn). Input to the search routine then consists of the file containing the connection tables of the substructures to be used in the search (which lets the user keep files of substructures which can be selected and combined to make up the file to be used in the search). In any case, the user need not learn any complex coding method.

SYSTEM DESCRIPTION

The system runs on a Digital Equipment Corp. PDP-10 in Fortran and UCI Lisp (from University of California at Irvine). Lisp was chosen in order to take advantage of Lisp's recursive list processing features in dealing with molecules and structures represented as lists. Input and output are from the user's terminal and/or disk files. An outline of the system appears in Figure 2.

When the molecules in the data base are being drawn on the graphics terminal, an assembly language program in the terminal's minicomputer converts the structures to a connection

table format, which contains a name tag for the molecule, the connection table of the molecule, and three-dimensional coordinates for each atom. The three-dimensional coordinates are included so that the file can be used as input to conformational analysis programs[13] which allow the user to calculate low-energy conformation in vacuo or some solvent, and to rotate the resultant structure in real time and view it in three dimensions. The file can also be used as input to other user programs since it is in Fortran format.

Once the molecules have been entered, a preprocessing program creates another file with the same connection tables, but in Lisp readable format and without the three-dimensional coordinates and other information irrelevant to the search routine. If the user has drawn in the substructures to be used in the search, the Fortran preprocessing program must also be used to convert the substructure connection tables. After all input intended for the Lisp search routine is in Lisp readable files, the search routine is used to search through the connection tables of molecules.

The Lisp program asks the user for: (1) the substructures to search for, and (2) how to combine the number of occurrences of each substructure in each molecule to generate the screen. The latter allows the user to combine the number of occurrences of several different substructures into one number in the screen. This is useful when the user wishes to treat two or more substructures as equivalent, even though their actual structural representations may be unique. The search routine finds not only an occurrence of a substructure, but *all* occurrences of a substructure in a molecule. (The user can, of course, use the program to find a molecule in the data file by searching for the structure of the molecule.)

The output of the search routine is a screen in Fortran readable format, consisting of a name tag and a list of integers for each molecule. Each integer occupies one location in the screen; the number actually written in each location is the total number of occurrences in the molecule of those substructures which were assigned to that particular location in the screen. If the user directs the output of the search to the user's terminal, the program tells the user directly which molecules have or do not have certain structural features (and how many). Using the name tags, he can then go back and retrieve the molecules of interest to him and display them on the graphics terminal. If the output is directed to a file, this file may be used as a screen file to preselect molecules for future applications of the search routine, or for any other purpose defined by the user. For example, the screen file could be used in the interactive chemical information retrieval system CRYSRC, a generalized modular system.[14]

SYSTEM OPERATION

When drawing the molecules which constitute the data base, or the substructures to be searched for, the user can enter the following atom types: H, C, N, O, P, S, Cl, Br, I, F, or X (X representing any atom type). He may also specify the bond types as single, double, triple, and aromatic. There is also an option for the program to insert hydrogens to fill valence requirements (when drawing molecules this is useful, but when drawing substructures it is not used). The Fortran readable connection tables generated by the graphics terminal program are put in the user specified file. The Fortran preprocessor asks for the name of the connection table file to be used, and for the name of the file where the Lisp readable connection tables will be placed.

The Lisp program asks for the name of the Lisp readable file to be used, the name of the output device (the user's terminal and/or a file), and the substructures to be searched for. If the user has drawn the substructures on the graphics terminal, the user gives the name of the Lisp readable file containing the substructures. The Lisp program will read the substructures from this file and search for their occurrences.

If instead the user wants to enter the substructures interactively during the operation of the search routine, he must give the program a list of substructures to be used in the search. Each substructure is a list of ⟨bondpair⟩'s. Each ⟨bondpair⟩ is of the format: (⟨atype⟩⟨asub⟩⟨btype⟩⟨atype⟩⟨asub⟩). which describes a bond of type ⟨btype⟩ between two atoms, each atom having an atom type ⟨atype⟩ and a subscript number ⟨asub⟩. A carbon atom with a single bond to another carbon is represented as: (C 1 1 C 2); a carbon with a double bond to an oxygen is written: (C 1 2 O 2). The subscript numbers must be uniquely assigned; i.e., only one atom may have any given subscript number, and there is only one subscript number for any atom. However, the same subscript numbers can be used in different substructures, since the subscript numbers are relevant only within any one substructure. How the subscript numbers are assigned to the atoms is not important, but it is easier for the user to check and correct errors if they are assigned in an orderly, linear fashion.

The possible ⟨atype⟩'s and ⟨btype⟩'s that are currently included in the program are listed in Table I. An ⟨atype⟩ or ⟨btype⟩ of X represents any atom type and any bond type, respectively. A ⟨btype⟩ of R represents an aromatic bond. As the substructure is represented as a list of ⟨bondpair⟩'s, the user is, in effect, writing a nonredundant connection table for each substructure. Figure 3 shows the substructures whose ⟨bondpair⟩ representations are entered to the program in Figure 4.

Table I

Possible ⟨atype⟩'s to be used in a ⟨bondpair⟩		
Letter	No.	Atom type meaning
H	1	Hydrogen
C	2	Carbon
N	4	Nitrogen
O	6	Oxygen
P	8	Phosphorus
CL	9	Chlorine
F	10	Fluorine
S	11	Sulfur
Br	14	Bromine
I	15	Iodine
X		Any atom type

Possible ⟨btype⟩'s to be used in a ⟨bondpair⟩	
Code	Meaning
1	Single bond
2	Double bond
3	Triple bond
R	Aromatic
X	Any bond

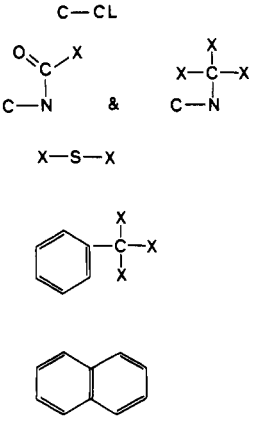

Figure 3.

```
WHAT FILE DO YOU WANT TO SCREEN?*(TSTJUN.DAT)


WHAT FILENAME DO YOU WANT FOR THE SCREEN?*(SCRN1.DAL)


STRUCTURES TO BE SCREENED FOR?*( ((C 1 1 CL 2))
*  ((C 1 1 N 2) (N 2 1 C 3) (C 3 2 O 4) (C 3 1 X 5))
*  ((C 1 1 N 2) (N 2 1 C 3) (C 3 1 X 4) (C 3 1 X 5) (C 3 1 X 6))
*  ((X 1 1 S 2) (S 2 1 X 3))
*  ((C 1 R C 2) (C 2 R C 3) (C 3 R C 4) (C 4 R C 5) (C 5 R C 6)
*   (C 6 R C 1) (C 1 1 C 7) (C 7 1 X 8) (C 7 1 X 9) (C 7 1 X 10))
*  ((C 1 R C 2) (C 2 R C 3) (C 3 R C 4) (C 4 R C 5) (C 5 R C 6)
*   (C 6 R C 1) (C 1 R C 7) (C 7 R C 8) (C 8 R C 9) (C 9 R C 10)
*   (C 10 R C 6))  )


 CORRESPONDENCE BETWEEN STRUCTURES AND SCREEN ELEMENTS?*( (1) (2 3)
* (4) (5) (6) )


STRUCTURES USED TO GENERATE SCREEN ARE:
((C 1 1 CL 2))

((C 1 1 N 2) (N 2 1 C 3) (C 3 2 O 4) (C 3 1 X 5))

((C 1 1 N 2) (N 2 1 C 3) (C 3 1 X 4) (C 3 1 X 5) (C 3 1 X 6))

((X 1 1 S 2) (S 2 1 X 3))

((C 1 R C 2) (C 2 R C 3) (C 3 R C 4) (C 4 R C 5) (C 5 R C 6) (C 6 R C
 1) (C 1 1 C 7) (C 7 1 X 10) (C 7 1 X 11) (C 7 1 X 12))

((C 1 R C 2) (C 2 R C 3) (C 3 R C 4) (C 4 R C 5) (C 5 R C 6) (C 6 R C
 1) (C 1 R C 7) (C 7 R C 10) (C 10 R C 11) (C 11 R C 12) (C 12 R C 6)
)


CONTINUE?*Y


NIL
*
```

Figure 4.

```
1AWXPARENT    0  0  0  0  0
2AW49819      1  0  1  0  0
3AW96792      0  0  1  0  0
4AW43460      2  4  0  1  0
5AX57996      0  0  1  0  0
6AX57987      0  0  0  0  0
7AW41733      0  0  3  0  1
8AW49800      1  0  1  1  0
9AV95585      2  4  0  1  0
10AV95629     0  0  1  0  0
11AX26704     0  4  0  2  0
12AX26722     2  4  0  2  0
13AX26688     2  4  0  1  0
14AX26259     1  1  0  1  1
15AX26277     2  4  0  1  0
16AX25225     2  4  0  1  0
17AX25207     0  4  0  1  1
18AX20971     0  1  1  0  0
19AX20962     0  0  1  0  0
20AX20122     0  4  1  0  0
21BC06514     1  4  0  1  0
 2BCERR       0  0  1  0  1
 0XXXXX       0
```

Figure 5.

Once the substructures to be used in the search have been entered, the program asks for the manner in which the number of occurrences of each substructure in a molecule will be combined in each location of the screen with those of the other substructures. This is done by entering a list of structure numbers (a structure number of a substructure represents the order in which it was entered; e.g., the third structure entered has a structure number of 3) for each location in the screen. As an example, if the user has entered three substructures and wants the occurrences of the first and third substructures to be combined in the first location, and the occurrence of the first element alone to be placed in the third location, the list of structure numbers for the first location would be (1 3); for the second location, (2 3); and for the third location, (1). To establish correspondence to the order of locations in the screen, the lists of structure numbers are arranged in a list according to the order of their corresponding locations, to wit: ((1 3) (2 3) (1)). This is the correspondence list the user enters to tell the program how to combine the number of occurrences of the various substructures. Note that the number of occurrences of any one substructure can influence several locations in the screen. The user need not enter a correspondence list if the locations in the screen are to correspond directly to the substructures as they were entered; he never needs to enter a correspondence list such as: ((1) (2) (3) ...). Figure 4 shows a correspondence list as it is entered to the program.

There are error checks which allow the user to correct mistakes made in entering the ⟨bondpair⟩ structure representation and the correspondence list without starting the program over. Once the substructures and the correspondence list have been verified, the search routine proceeds to search through each molecule in the specified file for the occurrences of the substructures by performing an atom-by-atom recursive search. The first atom in the substructure is matched with one of the same ⟨atype⟩ in the molecule, and the routine then recursively searches for matches to the other atoms in the substructure, with the proper bonds. As it does this, it builds up a structural correspondence list containing information as to which atoms in the molecule match atoms in the substructure. If the match is completely successful, this list is added to another list containing structural correspondence lists from previous matches to the same substructure, unless the latest structural correspondence list is a duplication (reflection or rotation) of one already found (searching for all possible occurrences of a benzene ring in a benzene ring will give 12 matches, of which 11 are duplications which must be detected and eliminated). Whenever the search routine has searched through a particular part of the molecule for a structure and stops due to a successful complete match or a failure in that branch, the atom at which the routine stopped is marked so that the routine will not duplicate a search through an area that has already been searched for a given correspondence match context; i.e., there is no redundancy in the search as the routine attempts to complete any one structural correspondence list.

Since subroutines are used to match the atoms and bonds, it is easy for the user to create new ⟨atype⟩'s and/or ⟨btype⟩'s which match several different atom types or bond types in a molecule. For example, an ⟨atype⟩ OS could easily be created to match either an oxygen *or* a sulfur in a molecule. The ability of the user to easily create his own search terms gives him great flexibility in determining how specific a search will be.

Once all the occurrences of a substructure in a molecule have been found, the program appends the total number of occurrences of the substructure to a list containing the name tag and the results of previous searches for other substructures in that molecule. When all the substructures have been searched for, the program adds together the number of occurrences of the substructures and orders the numbers thus obtained according to the correspondence list the user entered earlier. The screens consisting of the name tag and search results for each molecule are then returned either to the user's terminal or to a user specified file. Figure 5 contains a sample of the screens produced by the substructures and correspondence list in Figure 4.

RESULTS AND DISCUSSION

We have tested the screen generator on a small data base of about 65 molecules (w/structures relevant to mediterranean fruit fly sex attractants). The search routine was very fast

when the substructure of interest contained some unique feature, e.g., a chlorine atom, a triple bond, a benzene ring, etc. (for example, searching for an ether linkage through the file of 70 molecules of average size 19–20 atoms took roughly 7 s, of cpu time). However, as we were interested in the effect of carbon chain branching on biological activity, we entered some highly symmetrical substructures, such as a neopentane with symmetrical specifications on the ways it may be attached to the rest of the molecule, which resulted in very large run times. When handling molecules with many potential matches for such a symmetrical substructure, we found the run time to increase to an unacceptable level.

One problem resides in the necessity for finding every possible occurrence of a substructure for pattern recognition applications. Obviously, if one is interested in simply a binary result (yes, the structure does occur, or no, it does not), this problem is not encountered. However, in potential applications with pattern recognition that we envisioned, the system must be able to quantify its search results, so that there is some hope of pattern recognition revealing discriminatory features relative to biological activity, which may be dependent not just on the existence of a substructural feature, but also how many times it appears. Since we must find every possible occurrence of a substructural feature, there must of necessity be some redundancy in the results of a linear, recursive search routine (the duplication mentioned earlier) which will need some bookkeeping to eliminate. Take the possible ways to match a neopentane onto a neopentane, including the hydrogens: $4*6^5$ possible ways! This is obviously an extreme case, since the neopentane is going to be attached in some way to the rest of a molecule reducing the possible number of hydrogen matches, but it does point out the problem a linear recursive search has with very symmetrical situations, when asked to find *every* occurrence of a substructural feature and eliminate all duplications.

It would appear, at first, that some combination of a fragment notation with a connection table matrix would be optimal, in that searching for certain very symmetrical substructures as neopentane would be trivial if the molecule were encoded for them. A problem comes in attempting to find the occurrence of a substructure which is implicitly extant in the molecule, but is not obviously visible in the code, due to the manner in which the molecule was encoded. For example, in the case of four neopentanes around a single central carbon, fragment coding for neopentanes would miss either the fact that there were four neopentanes in addition to the central one, or a central one in addition to the four in the outer part. In addition, interfacing the fragment code to the linear, recursive search algorithm does not appear to be a trivial problem.

Our observation of the performance of the system indicates it to be a very useful interactive tool with respect to a structure–activity study dealing with appearances of substructures containing significant features: carbonyl groups, epoxy groups, ether linkages, etc. However, response time increases above acceptable levels (say, 1 min) when highly symmetrical carbon skeletons are of interest, due to the tremendous number of possibilities encountered by the search routine.

Potential applications of the system are numerous. When a large number of molecules must be entered in a file, we have found in our experience it is faster to draw the molecules on the graphics terminal than to type their structures. Because the run time is expected to be considerable, in relation to interactive use, if a large data base (several thousand molecules) is used, it is evident that the creation of a file of screens by batch processing (which can easily be done by this system) is a first priority. The file of screens will be more compact

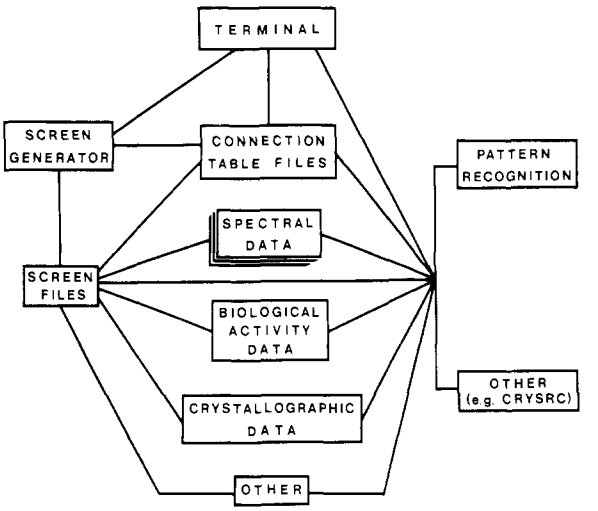

Figure 6.

and faster to search than the file of connection tables. If the user needs to identify a molecule, programs can first match the screen generated from the molecule with the screens in the screen file, and then do an atom-by-atom search on these molecules pulled from the screen files to conclusively establish identity.

The program can also be used to select molecules from a connection table file on the basis of substructural features of interest to the scientist. This feature would be particularly useful if spectral data or other data are also stored on the file. For example, if a particular substructural feature is thought to be present in an unknown molecule, spectra of molecules that contain the substructure could be retrieved by a search through the structures of molecules with spectra on file. These spectra could then be compared with the spectrum of the unknown molecule for verification. This is the reverse procedure for spectral search systems. However, searching the spectral files alone will often miss related molecules because a part of the spectrum may be obscured by the contribution of the entire structure.

Another important application of this system comes from recent studies applying pattern recognition to molecular structure–biological activity correlations.[3,15] In these studies, a list of substructural features thought to be related to biological activity could be compiled. The system would then search each structure in a collection of previously tested molecules and tabulate the occurrences of each substructural feature in each molecule. These features would then be correlated to the tested biological activity of the molecules by pattern recognition methods in hopes of establishing a useful structure–activity relationship.

CONCLUSIONS

The advantages of the system lie in the ease and accuracy of the graphic input, the fact that the user need not learn any complex code to interact with the system, and the fact that the results of the search can be returned in a form that is readily interpreted by the user at the terminal, as well as being readily utilized by other programs. In addition, errors are easily seen as the molecules are being drawn using the graphic input capability; it is possible to reduce error even more by using duplicate entries whose screens can be cross-checked for any discrepancies.

The simplicity of the system allows many potential applications in combination with other research tools (Figure 6). As one of the principal research goals in chemistry is to correlate molecular structure with other properties, it is evident that along with the creation of large molecular data files there

must be some way to select molecules on the basis of their structural features. Hence, search routines will continue to be an important area of computer applications to chemistry as long as structural features of molecules are of interest.

ACKNOWLEDGMENT

The authors wish to thank J. R. Koskinen and C. J. Appellof for the development of the PDP-11/05 program used for graphical input of molecular structures. The authors wish also to thank the National Science Foundation for partial financial support during the summers for Mr. Blom and Mr. Saxberg under the NSF Undergraduate Research Participation program, Grants EPP 75-04525 and SMI 76-03095.

LITERATURE CITED

(1) S. J. Frycki, F. F. Giarrusso, P. A. Roskos, D. E. Dancsecz, S. J. Lucania, and D. M. O'Brien, "Computerized Monitoring of the Inventory and Distribution of Research Chemicals", *J. Chem. Doc.*, **13**, 136–145 (1973).
(2) E. H. Eckermann, J. F. Waters, R. O. Pick, and J. A. Schafer, "Processing Data from a Large Drug Development Program", *J. Chem. Doc.*, **12**, 38–40 (1972).
(3) G. Redl, R. D. Cramer IV, and C. E. Berkoff, "Quantitative Drug Design", *Chem. Soc. Rev.*, 273 (1974).
(4) F. G. Stockton and R. L. Merritt, "The Shell Chemical Structure File System", *J. Chem. Doc.*, **14**, 166–170 (1974).
(5) J. L. Schultz, "Handling Chemical Information in the DuPont Central Report Index", *J. Chem. Doc.*, **14**, 171–179 (1974).
(6) H. Skolnik, "A Notation Symbol Index for Chemical Compounds", *J. Chem. Doc.*, **11**, 120–124 (1971).
(7) H. Skolnik, "A Chemical Fragment Notation Index", *J. Chem. Doc.*, **11**, 142–147 (1971).
(8) M. F. Lynch, J. M. Harrison, W. G. Town, and J. E. Ash, "Computer Handling of Chemical Structure Information", American Elsevier, New York, N.Y., 1971, pp 68–69.
(9) J. E. Crowe, P. Leggate, B. N. Rossiter, and J. F. B. Rowland, "The Searching of Wiswesser Line Notations by Means of a Character-Matching Serial Search", *J. Chem. Doc.*, **13**, 85–92 (1973).
(10) E. Meyer in "Computer Representation and Manipulation of Chemical Information", W. T. Wipke, S. R. Heller, R. J. Feldmann, and E. Hyde, Ed., Wiley, New York, N.Y., 1974, pp 108–111.
(11) E. H. Sussenguth, "A Graph-Theoretic Algorithm for Matching Chemical Structures", *J. Chem. Doc.*, **5**, 36–43 (1965).
(12) W. S. Woodward and T. L. Isenhour "Computer Controlled Television Scan System for Direct Encoding of Chemical Structure Models", *Anal. Chem.*, **46**, 422–426 (1974).
(13) A. J. Hopfinger, "Conformational Properties of Macromolecules", Academic Press, New York, N.Y. 1973.
(14) J. Villarreal, Jr., E. F. Meyer, Jr., R. W. Elliot, and C. Morimoto, "CRYSRC: A Generalized Chemical Information System Applied to a Structural Data File", *J. Chem. Inf. Comput. Sci.*, **15**, 220–225 (1975).
(15) K. C. Chu, R. J. Feldmann, M. B. Shapiro, G. F. Hazard Jr., and R. I. Geran, "Pattern Recognition and Structure-Activity Relationship Studies. Computer-Assisted Prediction of Anti-tumor Activity in Structurally Diverse Drugs in an Experimental Mouse Brain Tumor System", *J. Med. Chem.*, **18**, 539–545 (1975).

Reliability of Nonparametric Linear Classifiers

A. J. STUPER and P. C. JURS*

Department of Chemistry, The Pennsylvania State University, University Park, Pennsylvania 16802

Received June 18, 1976

The consequences of developing nonparametric linear discriminant functions using data sets which have a small ratio of samples to variables per sample are investigated. The lower limit to the ratio of samples to measurements per sample is dependent on the number of variables used but is approximately 3:1. Studies have been done with fully characterized computer generated data sets. Results are reported which indicate that classifiers developed with data sets having a ratio of less than 3:1 are likely to form relationships where none exist, and they are unable to distinguish relationships which do exist. Also, below this limit, feedback feature selection processes are shown to lose their effectiveness.

In the past five years a number of articles concerning applications of pattern recognition have appeared in the chemical literature. Many of these have reported the utility of nonparametric discriminant functions in providing insight into relationships contained within sets of chemical measurements. An assumption inherent in the use of such functions is that the ability to correctly dichotomize the data into classes is meaningful. Alternatively, successful classification is thought to imply that the classification parameters are related to the observed properties through some indirect or complex function. These assumptions can be tested only if certain criteria are met. These criteria deal with the minimum ratio of samples to measurements per sample required to demonstrate a relation within the data. This paper will investigate these criteria and discuss parameters which indicate the reliability of such relations.

Before the limitations of nonparametric discriminant function development can be established, the framework in which such techniques are employed must be understood. The basic procedure is to take n measurements of each object which is to undergo analysis, treating each measurement as an independent axis. This results in a data set for which each object is represented by a point in the n-dimensional space formed from the measurements made upon it. It is assumed that members of a set most similar to each other will tend to cluster in limited regions in this n-dimensional space. The ability to develop a function capable of separating the groups is thought to imply that the measurements must somehow correlate to these clustering properties.

The ability of a function to separate clusters within a set of data is dependent upon the dichotomization ability of the discriminant function. Dichotomization ability is the total number of two class groupings which can be made by the discriminant function. Different functional forms will have different dichotomization abilities. The dichotomization ability for a linear function is given by the following equation[1]

$$D(N,n) = 2\sum_{k=0}^{n} C_k^{N-1}$$

where $C_k^{N-1} = (N-1)!/(N-1-k)!k!$, N is the number of samples, n is the number of measurements or variables per sample, and k is an index describing how the groupings are taken.

The total number of dichotomies possible for a set of samples, regardless of its n space distribution is 2^N. [The only assumption made is that the data are well distributed. A data set is well distributed if no subset of $n+1$ points lies on a $n-1$ dimensional hyperplane.] Any classifier which could effect all of the 2^N possible dichotomies would always indicate that the desired clusters were present. This behavior is independent