

- (6) Meyer, E. F., "Towards an Automatic, Three Dimensional Display of Structural Data," *J. Chem. Doc.* **10**, 85 (1971).
- (7) Meyer, E. F., "Three Dimensional Graphical Models of Molecules and a Time-Slicing Computer," *J. Appl. Cryst.* **3**, 392 (1970).
- (8) Meyer, E. F., "Interactive Computer Display for the Three Dimensional Study of Macromolecular Structures," *Nature* **232**, 255 (1971).
- (9) *Chem. Eng. News*, "TV Displays 3-D Structures," p. 25, May 17, 1971.
- (10) Tometsko, A. M., "Computer Approaches to Protein Structure. I. Analysis of Atomic Distances," *Comput. Biomed. Res.* **3**, 229 (1970).
- (11) Tometsko, A. M., "Computer Approaches to Protein Structure II. Model Building by Computer," *Ibid.*, **3**, 690 (1971).
- (12) Tometsko, A. M., "Computer Approaches to Protein Structure. III. Transformation of Atomic Coordinates," *Ibid.*, **4**, 407 (1971).
- (13) Portigal, L. D., and Minicozzi, W. P., "Computer Generated Display and Manipulation of a General Molecule," *J. Chem. Ed.* **43**, 790 (1971).
- (14) Kennard, O., and Watson, D. G., "Molecular Structures and Dimensions," Vols. 1 and 2, Crystallographic Data Center, Cambridge, England, 1970.
- (15) Quantum Chemistry Program Exchange, Chemistry Department, University of Indiana, Bloomington, Ind., Program No. 178.
- (16) Quantum Chemistry Program Exchange, Chemistry Department, University of Illinois, Urbana, Ill., Program No. 141.
- (17) Quantum Chemistry Program Exchange, Chemistry Department, University of Illinois, Urbana, Ill., Program No. 137.
- (18) Sproull, R. J., Stanford University "Topics in Computer Graphics," unpublished manuscript, p. 92-4, 1972.
- (19) Kiefer, J., DCRT, NIH unpublished results, 1971.
- (20) Johnson, C. K., "ORTEP," Oak Ridge Thermal Ellipsoid Program, 1964.
- (21) Heller, S. R., and Feldmann, R. J., "DCRT/CIS X-Ray Modeling System Users' Manual," Division of Computer Research and Technology, Bethesda, Md., August 1972.

## Substructure Search by Set Reduction\*

JOHN FIGUERAS

Research Laboratories, Eastman Kodak Co., Rochester, N. Y. 14650

Received October 17, 1972

**A PL/1 implementation of a substructure search system based on set reduction is described. The set reduction algorithm is based on set theory and Boolean algebra rather than the graph-theoretic approach described by Sussenguth [*J. Chem. Doc.* **5**, 36 (1965)]. The use of ordered numerical codes for atom properties permits rapid list processing for fast rejection of non-matches, and the formulation of an efficient method for set generation. Time trials with a small file of organic chemical structures indicate that the algorithm can be economically used for substructure (or complete structure) sequential searches on a file containing 30,000-50,000 computer-coded structures.**

The connection table is a widely used device for the compact storage of complete chemical structure information. It has the disadvantage, however, that structural relationships which are explicit in the original chemical structure are merely implied in the connection table, so that it is not possible to conduct straight-forward searches for fragments of structure. This paper describes an algorithm based on the use of sets which provides a route to structure matching when the structures are coded as connection tables. The algorithm is closely related to a graph-theoretic algorithm published by Sussenguth<sup>1</sup> in 1965, but is quite different in approach and organization and contains some novel features.

### THE CONNECTION TABLE

A specially formulated connection table is used in order to expedite the substructure search program. The coding conventions employed follow closely the practice of *Chemical Abstracts*<sup>2</sup> with some modifications. The N<sup>th</sup> row of the table refers to the atom with index N, and the entries

in the row give atom indices and bond type for each atom connected to atom N. A typical table is shown in Table I for ethyl acetate. Entries in the body of the table (listed under "Connections and Bonds") are composite numbers, the last digit of each entry giving the nature of the bond joining two atoms; the remaining frontal digits give the index of the attached atom. For the connection table, the bonds are coded as follows:

- 1—single
- 2—double
- 3—triple
- 4—benzenoid
- 5—tautomer
- 6—charge delocalized

The definitions for "tautomer" and "charge delocalized" bonds can be found in *Chemical Abstracts* reports.<sup>2</sup> The use of composite numbers for joint representation of atom indices and associated bond types is a convenience that reduces the amount of information storage and the amount of subscripting required in the computer program. The composite numbers are easily resolved into their components by simple arithmetic. The table shows, for example, that atom 3 is connected to atom 4 by a double

\*Presented at Division of Chemical Literature, 163rd Meeting, ACS, Boston, Mass., April 12, 1972.

Table I. Connection Table and Atom Codes for Ethyl Acetate

$  \begin{array}{c}  \text{O} \\  \parallel \\  \text{CH}_3 - \text{C} - \text{OC}_2\text{H}_5 \\  \begin{array}{ccc}  5 & 4 & 2 \quad 1  \end{array}  \end{array}  $			
Atom No.	Atom Code	Code Count	Connections and Bonds
1	206202	1	21
2	108202	1	41 11
3	108102	1	42
4	106304	1	21 32 51
5	106201	1	41

bond, as indicated by the entry 32 in the row of the table for atom 4. The table is symmetrized—i.e., connections are shown going both ways—unlike CA practice;<sup>2</sup> this is necessary for maximizing search speed.

Stored along with the connection table is a vector containing *atom codes*. The atom code is a string of numbers of the form MMAABBB where MM is a number between 1 and 99 giving the multiplicity<sup>a</sup> of the atom, AA is the atomic number of the atom,<sup>b</sup> and BBB is a bond pattern code, described below. The code for atom 1 in Table I shows that it is a carbon atom (atomic number 6) with multiplicity 2—i.e., it is a C<sub>2</sub> group (see footnote this page)—and with bond pattern code of 202.

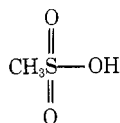
The bond pattern code is derived from the distribution of bonds around an atom. The code is a composite number, BBB = 1000\*N + S, where N is the number of bonds around the atom and S is the *sum of bond numbers*. The bond numbers (different from those used in the connection table, see above) have the following values:

- 0 benzenoid bond; bonds in fully conjugated cyclic systems
- 1 single bond
- 2 double bond
- 4 tautomer bond
- 8 charge-delocalized bond
- 16 triple bond

For example, three single bonds give a bond pattern code 303. The bond pattern code is a unique specification of bond distribution for any arrangement of bonds up to a total valence of six.

The atom code is rich in structural information. It is analogous to a fragment code, since, for example, carbonyl oxygen, ether oxygen, and hydroxyl oxygen each have characteristic and different atom codes. As shown in Table I, the atom codes occur in decreasing order of magnitude. This sequence of atom codes determines the

<sup>a</sup>The computer program from which the connection table is obtained contracts polymethylene chains and multiply-occurring single terminal atoms to the form X<sub>n</sub>, where n is the *multiplicity*. A chain of twenty methylene groups is contracted to C<sub>20</sub> and chloroform is contracted to CHCl<sub>3</sub>. Structure input to the program may be in either contracted or uncontracted form. Contraction offers several advantages: (1) it reduces the amount of storage and search time needed for molecules with long chains; (2) it removes the ambiguity attached to writing alkyl groups (e.g., C<sub>2</sub>H<sub>5</sub>- vs. CH<sub>3</sub>CH<sub>2</sub>-) and multiply-occurring terminal single atoms (e.g.,



vs. CH<sub>3</sub>SO<sub>3</sub>H).

<sup>b</sup>Certain structural features are included in the connection table by representing them as pseudo-atoms, assigning them normally unused atomic numbers; for example, a positive charge is considered as an atom with atomic number 93. The use of pseudo-atoms as descriptors can be extended to other structural features such as stereochemical configuration.

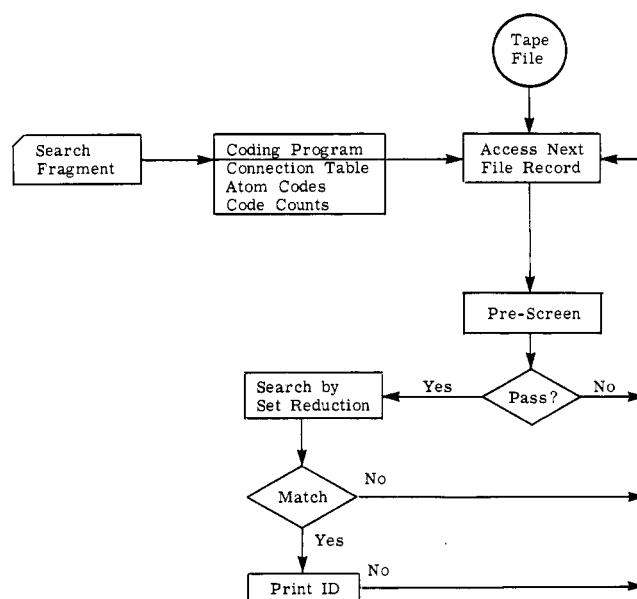


Figure 1. Over-all search logic

numbering sequence of atoms in the molecule; the connection table is based on this numbering. A count of the number of occurrences of each atom code is also stored as part of the record for the molecule (Table I). The atom codes and their counts are used in the prescreen portion of the search program as described below.

## ORGANIZATION OF THE SEARCH PROGRAM

The flow chart in Figure 1 outlines the organization of the search routine. The search fragment is entered from punched cards, using line-by-line punching of the structure.<sup>c</sup> The fragment is coded to obtain a connection table, atom code list, and code count list. These coded data are used for the search of the tape file, which itself is a collection of connection tables, atom codes, and code counts.

The prescreen is a portion of the program in which a very rapid structure survey is made. This prescreen was programmed to be as efficient as possible because search times are determined as strongly by the rapidity with which structures are rejected as by the rapidity with which they are matched. If the prescreen is passed, search by set reduction is carried out, and matches through set reduction are identified in the printout from the program.

## PREScreen

The first prescreen step is a check to make sure that the current file structure has more atoms than the search (query) fragment. If this is not so, the file structure is rejected and the next one obtained from the tape file. Otherwise, query and file structures are compared for atom code compatibility; each atom code in the query must not occur more frequently than it does in the file structure. For example, if the query structure contains two carbon-

<sup>c</sup>One punched card is used to represent one line of structure, as suggested by P. Horowitz and E. M. Crane in their privately distributed report, HECSAGON (1961). Special, simple conventions are required to permit structure representation within the limitations of the 029 keypunch keyboard. Punched-card structure input is impractical for file creation, but a great convenience for fragment input by chemists who rarely have access to chemical typewriters, light pens, RAND tablets, etc.

yl oxygens—i.e., atom code 108102 occurs with a code count of 2—any file record which contains fewer than two carbonyl oxygens can be rejected as a possible match. The use of ordered lists of atom codes and prior determination of code counts greatly facilitates this comparison by eliminating the item-by-item comparison and counting that would be required if random lists were used. If the prescreen is passed, the program moves into the more detailed, time-consuming search by set reduction.

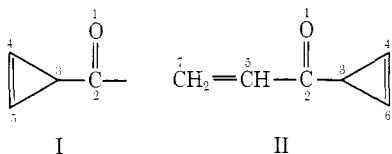
### PRINCIPLES OF SET GENERATION AND SET REDUCTION

The way in which various sets can be generated and represented is illustrated in Table II. The set of all carbon atoms can be represented by a *characteristic vector* with a zero in position 1 indicating that atom 1 is *not* a carbon atom and unit bits in positions 2 through 7 indicating that atoms 2 through 7 *are* carbon atoms. Similarly, the second characteristic vector gives the distribution of atoms having one double bond, and the third characteristic vector represents the set of all atoms having at least one single bond.

A new vector is formed from the arithmetic product of the elements in corresponding positions of each of the three vectors to give the Boolean product of the three sets. The product vector gives the distribution of carbon atoms (property 1) having one double bond (property 2) and at least one single bond; (property 3) i.e., it corresponds to a joint property set. The formation of Boolean products in this way comprises *set reduction* in Sussenguth's sense, since the number of unit bits in the Boolean product vector cannot be greater than the number in each of the elementary vectors, and is frequently less.

### SEARCHING BY SET REDUCTION: THE FIRST-ORDER SEARCH

We shall consider the problem of showing that the query structure I is present in the file structure II, which



represents one of a number of structures stored in coded form in the tape file. The atom code for each query atom (second column of Table III) is used as a basis for establishing a characteristic vector that gives the distribution of that atom code in the file structure (third column of Table III). Atom code lists and code counts for both file and query structures are used for this construction; the

Table II. Representation and Reduction of Sets

Set		Characteristic Vector
All carbon atoms		0111111
Atoms with one double bond		1101111
Atoms with at least one single bond		0111110
Boolean product		0101110

Table III. Set Generation Based on Atom Codes

Query Atom	Atom Code	Characteristic Vector for File Structure
1	108102	1000000
2	106304	0100000
3	106303	0010000
4	106203	0001110
5	106203	0001110

ordered nature of these lists permits formation of the characteristic vectors with minimum list searching and counting.

The characteristic vectors in Table III represent sets of file atoms that are possible correspondents to each query atom with respect to the joint properties contained in the atom code. For example, the last characteristic vector shows that file atoms 4, 5, and 6 are possible correspondents to query atom 5.

Connectivity and bonding information are introduced into the algorithm using the scheme in Figure 2. According to this scheme, a given query atom A is selected with a given bond type at that atom. The query connection table is used to find the set B of atoms attached to A by the given bond type. Using the correspondences based on atom codes (as in Table III), the set of the file atoms  $C_1, C_2 \dots$  is obtained that are possible correspondents to atom A. The connection table for the file structure is used to obtain the set D of file atoms attached to the file atoms  $C_1, C_2 \dots$  by the selected bond type. The sets B and D contain atoms that correspond to each other with respect to connectivities and bonds; that is, each atom in each set is attached by the same kind of bond to the same kind of atom. This process is carried out for all query atoms and all query bond types.

The numerical example in Figure 3 illustrates this scheme. Query atom 3 is attached by single bonds to query atoms 2, 4, and 5. Table III shows that file atom 3 is the only possible correspondent to query atom 3, based on atom code. File atom 3 is attached by single bonds to

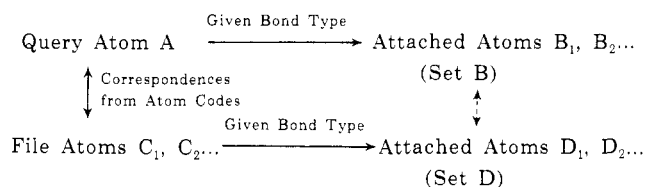


Figure 2. Set generation from connectivities and bonds

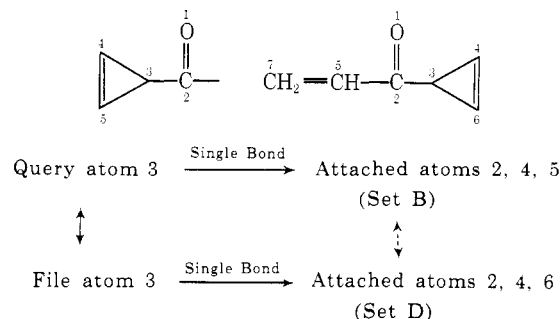


Figure 3. Numerical example of set generation based on connectivities and bonds

file atoms 2, 4, and 6. Thus, file atoms 2, 4, and 6 are possible correspondents to each of query atoms 2, 4, and 5 with respect to single-bonded connectivities. We can therefore associate with each of the query atoms, 2, 4, and 5, a file characteristic vector 0101010 giving possible correspondences with respect to connectivities and bonds only. Query atoms 2, 4, and 5 now have *two* associated file characteristic vectors (sets): one based on connectivities and bonds (0101010, above) and one based on atom codes (Table III). These pairs of vectors (Table IV) are merged by Boolean multiplication to give new correspondences (last column, Table IV) that hold with respect to both atom codes and bond connectivities.

The procedure described in Figure 2 differs from Sussenguth's original work,<sup>1</sup> which did not take explicit account of the nature of the bond *between* atoms. For this reason, Sussenguth's algorithm may not differentiate between certain isomers that differ only in which atom pair controls a multiple bond. For example, the fragments  $\text{=CH-CH=}$  and  $\text{-CH=CH-}$  would yield identical sets of matching structures in Sussenguth's algorithm. Ming and Tauber<sup>3</sup> remarked on the occurrence of false drops arising from this source. They proposed the introduction of extra nodes between atoms to carry bonding information and verified the validity of this approach to give greater search discrimination. However, as they also realized, the use of extra nodes gives a significant increase in required core storage and introduces the added difficulty that connected atoms are no longer represented by adjacent nodes. The use of extra nodes is averted by the scheme in Figure 2, which incorporates the required bonding information directly from the connection table with no increase in storage requirements.

The scheme given in Figure 2 is carried out for each query atom and its associated bond types. The final result for query structure I and file structure II is given in Table V. One-to-one correspondences are obtained for query atoms 1, 2, and 3, but multiple isomorphisms are obtained for query atoms 4 and 5. The multiple isomorphisms result from molecular symmetry that makes file atoms 4 and 6 equivalent. Sussenguth<sup>1</sup> used an assignment procedure to resolve multiple isomorphisms. This procedure required back-tracking with concomitant increases in core requirements to store intermediate arrays, and with significant increases in time requirements. Sussenguth's principal reason for using his assignment procedure

was the observation that multiple isomorphisms arise either from molecular symmetry (as in our current example) or from failure of the algorithm because of an insufficient property set. He therefore invoked assignment to guard against the latter possibility. We have chosen to omit the assignment procedure, accepting isomorphisms as a concomitant of molecular symmetry and depending upon experiment to determine the adequacy of the algorithm.

## USE OF HIGHER-ORDER CONNECTIVITIES

The search considered so far is a first-order search in that nearest neighbors—as given in connection tables—are used for the search. If the first-order search gives complete 1-1 correspondences, a match is indicated, and the algorithm moves to the next file structure. If, however, multiple isomorphisms are obtained, the algorithm moves to a higher-order search, using more remote connectivities. The scheme for obtaining correspondences based on higher-order connectivities is shown in Figure 4. The program contains a simple algorithm for obtaining higher-order connection tables for the query and file structures. These connection tables are used with query atom A to generate the set B of  $n^{\text{th}}$  nearest neighbors, and with the corresponding file atoms  $C_1, C_2 \dots$  to generate the set D of  $n^{\text{th}}$  nearest neighbors. Sets B and D now contain possible correspondents based on higher-order connectivities alone, and these can be expressed as characteristic vectors and merged with existing vectors by Boolean multiplication. The algorithm terminates and a match is indicated when two successive higher-order searches produce no change in the characteristic vectors. The higher-order search algorithm in the present version uses neighbors of  $2^k$  order ( $k = 1, 2, 3 \dots$ ). Such a search is particularly simple to generate by replacing the current connection tables with their second-order tables before each cycle of higher-order search. A higher-order search using 2, 3, 4  $\dots n^{\text{th}}$  orders was also written; it did not give different search results except for possibly longer match times with large queries.

The higher-order search is time-consuming, each one requiring about as much time as the first-order search alone. The higher-order search does eliminate false drops in some cases, but more experience is required to determine whether the additional discrimination that it provides justifies the increase in search time.

## DETERMINATION OF NONMATCHES

Nonmatches detected during set reduction appear in various guises although all of them involve some form of cardinality violation—that is, the number of atoms with a given combination of properties is larger in the query structure than in the file structure. During set reduction, two nonmatch conditions may arise:

1. A characteristic vector becomes null. A null vector may appear after a Boolean multiplication and signals that

Table IV. Sets Corresponding to Query Atoms 2, 4, and 5 and Set Merging

Query Atom	Connectivity-Based Sets	Atom-Code-Based Sets	Boolean Products
2	0101010	0100000	0100000
4	0101010	0001110	0001010
5	0101010	0001110	0001010

Table V. Final Sets After First-Order Search

Query Atom	Characteristic Vector
1	1000000
2	0100000
3	0010000
4	0001010
5	0001010

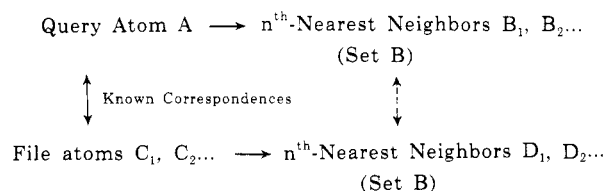


Figure 4. Higher-order searching:  $n^{\text{th}}$  nearest neighbors

the query atom has no correspondent in the file structure. The algorithm is terminated whenever a null vector appears.

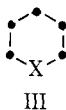
2. A multiple assignment occurs. *Multiple assignment* arises when the number of query atoms having a given set of properties is larger than the number of corresponding file atoms. A null characteristic vector is a special case of this. In the general case of a multiple assignment, one file atom is assigned as a possible correspondent to more than one query atom. A multiple assignment manifests itself when  $n$  identical characteristic vectors contain fewer than  $n$  unit bits each. A check for multiple assignments is made after each order of search and the algorithm terminates if one is found.

### SET CONTRACTION

The file structure is usually much larger than the query structure and will usually contain a number of atoms that have no correspondents in the query structure. As a result of set reduction, the number of possible correspondents to query atoms either decreases or remains the same; it can never increase. Consequently, noncorrespondents that appear after the first-order search can never become correspondents in the higher-order search and may be eliminated. An example of such a noncorrespondent appears in Table V: atom 7 of the file structure has no correspondents in the query structure (as shown by the column of zeroes for file atom 7). Elimination of noncorrespondents may result in a substantial reduction in the number of atoms in the file structure that have to be considered and a consequent decrease in search time, provided that the saving in search time outweighs the additional processing time required to effect contraction. In practice, this appears to be the case.

### SPECIAL TERMINATORS

Special terminators are required to facilitate searches for heterocycles, and to specify substitution patterns on rings. A request for all six-membered, heterocyclic compounds would be represented, for example, by structure



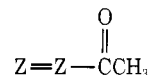
III, in which X represents a noncarbon atom. Three special terminators are incorporated into the program:

X-noncarbon, tautomeric  
Y-noncarbon, nontautomeric  
Z-carbon

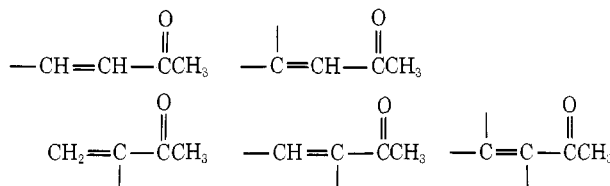
The terminators are handled as follows. In the coding program, they are treated as atoms with special atomic numbers. They are included in the connection table for the query structure. In the set reduction program, special vectors are associated with the terminators. For example, if X or Y (defined above) appears in the query structure, the characteristic vector which is initially established for X or Y has unit bits for all noncarbon atoms in the file structure, and zero bits otherwise. These vectors associated with special terminators are carried through set reduction in the same way as the rest of the vectors, and eventually correspondences are obtained between the X or Y atoms and certain atoms of the file structure. Special precautions are required when these general terminators are introduced because they may cause the emergence of

extraneous unit bits that upset the detection of multiple assignments. Accordingly, a modification of the multiple assignment check is used when special terminators appear.

Special terminators are not limited to specification of heterocycles but are of use in formulating generic searches. For example,



represents all methyl ketones having  $\alpha,\beta$ -unsaturation. All structures of the following types would be accessed by a search based on this structure:

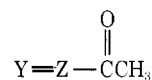


This points up the useful role of Z in providing a method for allowing hydrogen to be among the possible substituents on carbon. This is the only convenient way in which hydrogen can be specified as a substituent.

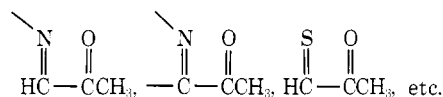
Analogously, structure IV represents all aromatic heterocycles containing five carbon atoms and one nitrogen atom, and a request based on this structure would give all pyridines, quinolines, isoquinolines, acridines, etc.



X- and Y-terminators also have a variety of uses. For example, a search for



would yield all compounds in which an acetyl group was joined to carbon (substituted or unsubstituted) doubly bonded to a hetero atom; e.g.,



X- and Y-terminators function alike unless there is a possibility for tautomerism. That is, for example, a search based on



would give all tautomeric, acetyl-containing structures (acetic acid, thioacetic acid, etc.), while a search based on



would give all nontautomeric, acetyl-containing structures (acetyl halides, *N,N*-dimethylacetamide, acetanilide, etc.).

## IMPLEMENTATION

The coding program and search algorithm were written in PL/I, which has excellent routines for character manipulation and bit handling. The present version of these programs can accommodate up to 96 atoms per molecule, with a total valence up to six. The coding program consists of 791 statements and requires about 1 sec of computer time to code the search query. The search algorithm is available in two versions: the complete version (428 statements) and a short version (300 statements), which omits the higher-order search. Both versions required about 200K bytes of storage.

A small file of 314 structures (selected from Company files) was created to test the program and obtain timing data. Figure 5 is a computer-generated histogram showing the distribution of molecular sizes in the file. The average molecular size in the file was 14.8 atoms, and the size range was 3 to 74 atoms. The average underestimates true molecular size because of the contraction feature in the coding program.

1	0
2	0
3**	2
4*****	5
5*****	8
6*****	8
7*****	8
8*****	15
9*****	22
10*****	27
11*****	33
12*****	35
13*****	27
14*****	5
15***	3
16*****	9
17*****	8
18*****	18
19*****	15
20*****	9
21*****	11
22*****	11
23*****	7
24****	4
25**	2
26*****	7
27*	1
28*	1
29*	1
30	0
31	0
32**	2
33*	1
34*	1
35	0
36	0
37**	2
38*	1
39	0
40*	1
41	0
42	0
43	0
44	0
45	0
46	0
47	0
48	0
49	0
50*	1
51	0
52	0
53**	2
54	0
55	0
56	0
57	0
58	0
59	0

Figure 5. Computer-generated histogram

## SEARCH RESULTS

Table VI contains timing data for a number of fragment searches on the small file, using the complete, basic routine. Asterisked bonds represent benzenoid bonds; thus, the second fragment represents azomethine compounds in which the nitrogen atom is connected to an aromatic ring. *Total match time* is the total amount of time required for matches by set reduction during the whole file scan. *Total search time* is the total amount of time required to do the search. The *average match time* is obtained by dividing total match time by the number of hits. *Scan rate* is obtained by subtracting total match time from total search time to obtain the amount of time required to scan the file of 314 structures exclusive of matches, and extrapolating the difference to a rate expressed as compounds per minute. The scan rate and average match times give some indication of how the search could be expected to behave with larger files. Average match times are on the order of 0.1 sec for the fragments in Table VI; The average scan rate for twelve searches was 37,700 compounds per min. By extrapolation, the search routine ought to give searches on files of 30,000 compounds with CPU times on the order of a few minutes.<sup>d</sup>

The scan rate appears to be independent of molecular size, although it varies greatly with structure. The variation depends upon the extent to which set reduction and higher-order searching are required in order to determine a mismatch; in the case of the first fragment (Table VI), the ubiquitous carbonyl group and disubstituted methylene carbon atom require frequent use of set reduction and higher-order search, resulting in a low scan rate. Match times increase with the size of the search fragment, of course.

The second last example in the table, illustrating the use of special terminators, caused retrieval of all of the substituted pyridines in the experimental file.

In all cases reported in Table VI, there were no false drops and recall was one hundred per cent.

Table VII illustrates a feature first pointed out by Ming and Tauber.<sup>3</sup> Searches for the joint occurrence of two fragments can be carried out simply by listing the fragments. Space must separate the fragments so that they are properly interpreted by the coding program as unbonded. The first search, for example, is for all structures which contain both a nitrile and an amine group. Match times increase, of course, with the number of fragments specified.

Table VIII reports timing data using the short version of the search program for some of the fragments appearing in Tables VI and VII, against which they should be compared. The most striking feature of the data obtained with the short version—in which the higher-order search is deleted—is the marked reduction in match times; the short version gives roughly a fivefold reduction in match time. On the whole, scan rates are not affected by using the short version except in the first, sixth, and ninth examples in Table VIII, for which the short version gives a marked improvement in scan rate (compare with Table VI). However, the short version has less discrimination

<sup>d</sup>Scan rates will obviously depend upon the distribution of structural types in the file and the nature of the search fragment. The scan rates obtained with the small file used in this work are considered *indicative* only. It is considered significant that scan rates have a magnitude on the order of 10,000–60,000 structures per min; if rates on the order of 100–1,000 structures per min had been obtained, the use of the algorithm with large files would probably not be considered for reasons of cost. Note Added in Proof: More recent work with a file of 7000 structures confirms the timing data obtained with the small file reported here.

# SUBSTRUCTURE SEARCH BY SET REDUCTION

than the complete version; the first example in Table VII gave six false drops, and the seventh example gave one false drop. The short version may be especially useful for searches involving large, complex fragments, where the higher search speed would be desirable to avoid excessively long match times, and where false drops may be a minor problem.

Although the search algorithm was designed primarily for doing substructure searches, it is equally effective for complete structure retrieval. In complete structure retrieval, it can be assumed that the structure only occurs once in the file so that the search can be terminated once a hit is made—this feature is built into the search algorithm. Table IX reports timing data for complete structure searches using both the complete and short versions of the program. In no case, with either program, were false drops obtained. The timing data obtained for complete structure retrieval are comparable with those obtained in fragment searches. The largest structure retrieved in a complete structure search contained 37

Table VI. Fragment Searches (Complete Version)

	No. Hits	Total match Time, Sec	Total search Time, Sec	Avg. Match Time, Sec	Scan Rate Compds/Min
	7	0.77	2.13	0.11	13,800
	11	0.50	1.11	0.05	31,000
	18	0.16	0.61	0.01	42,000
	6	0.44	0.79	0.07	54,000
	6	0.72	1.90	0.12	16,000
	12	0.80	1.23	0.07	44,000
	1	0.80	1.09	0.80	65,000
	7	0.95	2.45	0.14	12,500
	4	1.73	2.02	0.43	65,000
	8	1.74	2.77	0.22	18,300
	0	0	0.35	0	54,000


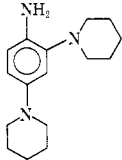
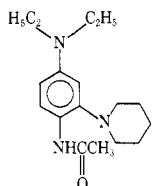
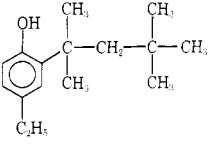
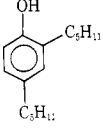
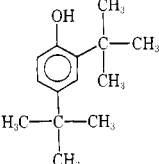
Table VII. Fragment Searches (Complete Version)

	No. Hits	Av Match Time, Sec	Scan Rate Compds/Min
	1	0.00	57,200
	3	0.01	90,000
	3	0.08	57,300
	19	0.09	18,300
	5	0.27	42,000
	7	0.08	29,000

Table VIII. Fragment Searches (Short Version)

	No. Hits	Av Match Time, Sec	Scan Rate Compds/Min
	13	0.025	31,400
	11	0.01	29,400
	18	0.004	46,000
	6	0.045	49,300
	12	0.015	43,000
	1	0.17	64,000
	8	0.035	19,500
	4	0.14	46,000
	8	0.04	43,000
	1	0.00	54,000
	3	0.01	61,000
	3	0.02	55,200

Table IX. Complete Structure Searches

	Full Version			Short Version	
	No. Atoms	Match Time, Sec	Scan Rate Comps/Min	Match Time, Sec	Scan Rate Comps/Min
	10	0.04	55,000	0.04	55,000
	19	0.80	95,900	0.16	55,200
	18	0.49	46,000	0.10	48,100
	16	0.45	39,600	0.09	42,500
	9	0.14	42,800	0.04	51,000
	15	0.19	51,100	0.09	63,100

atoms; the match time was 1.74 sec in the complete search algorithm, and 0.41 sec in the short version; the scan rate was 78,000 structures per min.

## MODIFICATIONS

Two modifications of the search program were written, but were only cursorily explored. First, a higher-order search routine was written incorporating 2, 3, 4 . . . order

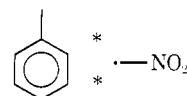
Col 1	
A	Fragment A
-	Fragment B
C	Fragment C

Figure 6. A search with extended logic

connectivities (in place of  $2^k$  connectivities in the first version). Investigation using the trial file showed no substantial differences in performance except for appreciably longer match times for large structures using the new routine—e.g., a 37-atom structure gave a match in 1.74 sec with the  $2^k$  order search, and required 2.59 sec with the  $k$ -order search.

In a second modification, the logic capabilities of the algorithm were extended to permit negative searches. Figure 6 is an example of input for a search with extended logic; character choice for column 1 determines the nature of the search, a minus (-) sign yielding a negative search and any other character a positive search. Figure 6 causes a positive search for A, a negative search for B, and a positive search for C. The test for fragment C is made only on the hits which pass A and not B. That is, the searches are cascaded on the same structure until a definite nonmatch is achieved.

The extended logic produces an AND search which is somewhat different from that obtained by joint listing of fragments (as in Table VII) in that fragment B may be contained in fragment A in the extended logic search. This is not necessarily true for searches using joint listing. For example, a search for aromatic nitro compounds having singly substituted phenyl rings



cannot give nitrobenzene as a match on joint listing, whereas nitrobenzene is a possible match using the extended logic search. Simple searches with the extended logic search indicate that it is as fast as the basic routine alone.

## ACKNOWLEDGMENT

I thank David H. Willis, of our Computer Center, for his assistance in PL/I programming. I also wish to acknowledge the advice of Tom Grout, who suggested the use of ordered lists for rapid list searching.

## LITERATURE CITED

- (1) Sussenguth, E. H., Jr., "A Graph-Theoretic Algorithm for Matching Chemical Structures," *J. Chem. Doc.* **5**, 36 (1965).
- (2) CAS Registry, System and Program Documentation Manual, Released by Systems Development, Chemical Abstracts Service, Columbus, Ohio, Copyright 1968.
- (3) Ming, T. K., and Tauber, S. J., "Chemical Structure and Substructure Search by Set Reduction," *J. Chem. Doc.*, **11**, 47 (1971).