# BCT Representation of Chemical Structures

TAKASHI NAKAYAMA and YUZURU FUJIWARA*

Institute of Information Sciences and Electronics, University of Tsukuba, Sakura-mura, Niihara-gun, Ibaraki, 305 Japan

Received June 11, 1979

A method is described for representing chemical structures hierarchically by means of a BCT (block-cutpoint tree), which is a structural unit having chemical significance. Memory requirements and the flexibility of data manipulation are improved compared with direct processing of adjacency matrices.

## I. INTRODUCTION

In processing structural information about chemical compounds by computer, memory efficiency and processing time depend largely on the method of representation. Graph theory has been a powerful tool in representing a structure in terms of graphs by assigning atoms and bonds to vertices and edges. One of the typical methods is a structure representation by a connection table.[1] The adjacency matrix can represent the connectivity relation of a chemical structure, and most of the structural information is contained in a connection table. Practically, the compact connection table, such as those devised by Gluck,[2] is effective for saving storage requirement. However, it is not easy to use a connection table directly and efficiently in various application systems, such as a substructure search system. These problems are due to the concept of chemical structure as simply a connectivity relationship among atoms. Actually, when an observer recognizes a chemical structure, the direct connections among atoms are not always basic to the recognition. It is believed that the various intermediate concepts between chemical compounds and atoms are organized hierarchically, and that they are properly selected during the recognizing procedure.

We have tried to describe a chemical structure hierarchically by means of not only the structural unit "atom", but also the intermediate concept "block". The block used here is defined graph theoretically, and it is believed that the block has some kind of chemical meaning. And that is one of the reasons why a "block" is used as an intermediate structural unit. By introducing a block, a chemical structure is described in the intermediate form of tree structure, called a block-cutpoint tree (BCT). The canonical form of a tree is obtained simply. It is sufficient for the description of blocks to prepare a "block dictionary". This leads to major savings in memory, as explained later.

Methods to represent chemical structures may be divided into two systems. One is called linear notation systems and the other is called topological systems. Many systems have been devised and implemented for various purposes.[4-11] The BCT representation of chemical structures belongs to the latter category.

## II. BCT REPRESENTATION

**1. Definitions.** A "graph" $G$ is formally denoted by a doublet $(X, U)$, where $X = \{x_1, \ldots, x_n\}$ is a set of "vertices" and $U = \{u_1, \ldots, u_m\}$ is a set of "edges" which are the elements of the Cartesian product of $X$: $X \times X = \{(x,y) | x \in X, y \in Y\}$.[12,13] If the edges in $U$ have no direction, the graph is "nondirected". A "path" in $G$ is any sequence of edges in which the final vertex of an edge is the initial vertex of the next one. A "loop" is an edge whose initial and final vertices are the same. A "cycle" is a path $x_1 \ldots x_\rho$ where the initial vertex $x_1$ coincides with the final vertex $x_\rho$. A graph is "connected" if, for any two distinct vertices of $X$, there is at least one path between the two. The number of edges joined to vertex $x$ is called the "degree" of vertex $x$ and is denoted by deg $x$. The "length" of a path $x_0x_1x_2 \ldots x_n$ is the number of edges, $n$, in the path. The "distance" $d(x_1,x_2)$ is the length of the path between the two vertices $x_1$ and $x_2$. A graph is "simple" if the following conditions are satisfied: (1) it has no loops and (2) the number of edges between any two distinct vertices is at most one. Given a graph $G = (X, U)$, a "subgraph" is the graph $G' = (Y, (Y \times Y) \cap U)$ with $Y \subset X$. A "tree" is a connected graph without a cycle. A graph $G = (X, U)$ is a "bigraph" or a "bipartite" graph if the set $X$ can be partitioned into two subsets ($X = X_1 \cup X_2$, $X_1 \cap X_2 = \phi$) so that all edges in $U$ have one terminal vertex in $X_1$ and the other in $X_2$. A vertex $x$ is a "central" vertex if $x$ is a vertex of a connected graph such that the maximal distance from $x$ is minimal. A chemical structure can be regarded as a nondirected connected simple graph. The concepts of a cutpoint and a block are used to define BCT.[3] A vertex $x$ is a "cutpoint" if the removal of $x$ increases the number of connected components of the graph. A connected graph $G$ is "nonseparable" if $G$ has no cutpoints. A "block" of graph $G$ is a maximal nonseparable subgraph of $G$. Now a "BCT (block-cutpoint tree)" is defined for a given graph $G$ as follows: $T = (Z, W)$ is a block-cutpoint graph of $G$ (denoted as $bc(G)$) if: (1) $Z = C \cup B$ is a set of vertices where $C = \{c_1, \ldots, c_n\}$ is a set of all cutpoints of $G$ and $B = \{B_1, \ldots, B_m\}$ is a set of all blocks of $G$ ($B_i$ signifies a set of vertices of a block) and (2) $W = \{w_1, \ldots, w_l\} = \{(B_i,c_j)\ B_i \in B, c_j \in C, c_j \in B_i\}$. In other words, a graph is a $bc(G)$ if the set of vertices is the union of a cutpoint set and a block set, and all edges have one terminal in a cutpoint set and the other in a block set where the cutpoint is contained in the block. A $bc(G)$ has the following properties: (1) it is a bigraph, (2) it has only one central vertex, (3) it is a tree, and (4) terminal vertices are blocks. A $bc(G)$ is called a block-cutpoint tree (BCT) because of property 3; we use "BCT" hereafter. Cutpoints and blocks of a graph are illustrated in Figure 1.

**2. BCT Representation.** Figure 2 shows the structure of cholesterol. The upper representation is the usual one and the lower is its BCT representation. It is shown that the BCT is a bipartite tree with blocks (white circles) and cutpoints (black circles). A canonical form of the bipartite connection table of BCT (called a BCT table hereafter) is used as the data format for computer use. For a BCT with $m$ blocks and $n$ cutpoints, row numbers and column numbers of an $m \times n$ matrix correspond to the block and cutpoint identifiers, respectively. An element of a matrix $a_{ij} = 1$ means that $B_i$ is adjacent to $c_j$, and $a_{ij} = 0$ means that $B_i$ is not adjacent to $c_j$. An example of a BCT table is shown in Figure 3. A canonical form of this BCT table is obtained by the following procedure: (1) Generation of BCT canonical form.

(a) A central vertex is assigned to the root of BCT. (As there is only one central vertex, according to BCT property (2), the root is uniquely determined.)

0095-2338/80/1620-0023$01.00/0 

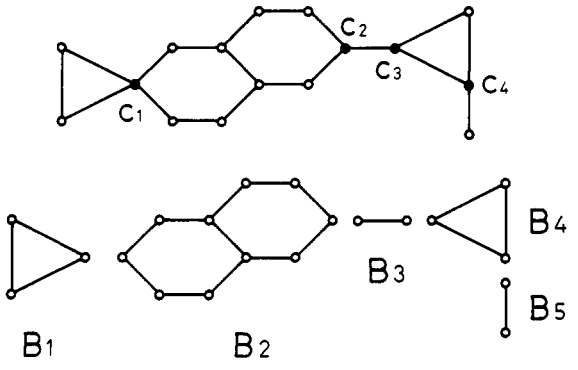


**Figure 1.** Blocks and cutpoints of a graph $G$. $\{B_1, B_2, B_3, B_4, B_5\}$ is a set of blocks and $\{C_1, C_2, C_3, C_4\}$ is a set of cutpoints.

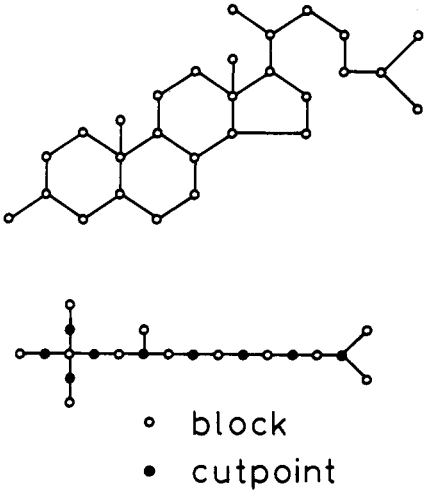


**Figure 2.** Structure representation of "cholesterol". The upper representation is a usual notation and the lower is a BCT representation.

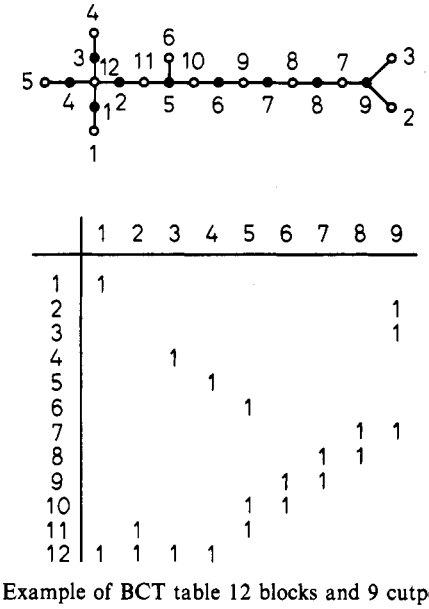


**Figure 3.** Example of BCT table 12 blocks and 9 cutpoints.

(b) A vertex $x$ belongs to level $i$ if $d(x, x_0) = i$, where $x_0$ is the root. (The root is on level 0, and no other vertex belongs to level 0.)

(c) Vertices in level $i$ are arranged by the following rules in ascending order of $i$.

*Rule 1.* Vertices with a common parent are collected into one group. Vertices in a group are arranged continuously in a level next to the parent.

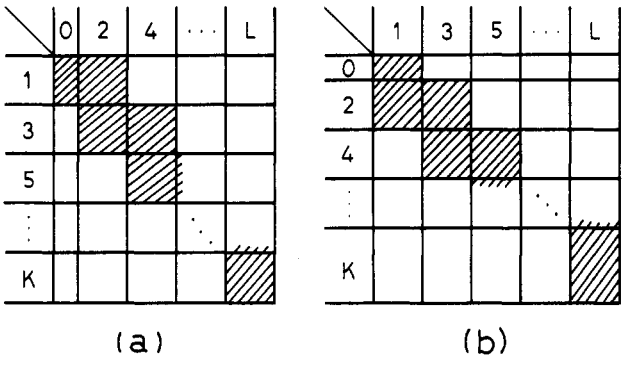


**Figure 4.** Arrangement of the levels: (a) BCT table with a cutpoint root; (b) BCT table with a block root.

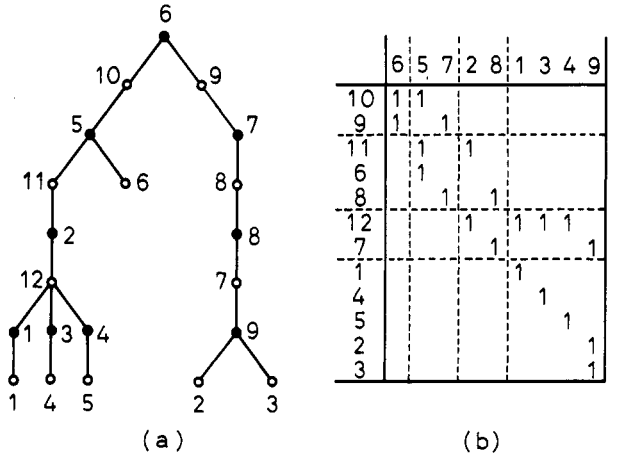


**Figure 5.** BCT canonical form (a) and canonical BCT table (b).

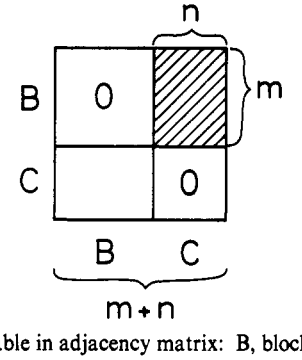


**Figure 6.** BCT table in adjacency matrix: B, block part; C, cutpoint part.

*Rule 2.* Vertices in a group are arranged in descending order of the number of descendants.

*Rule 3.* If two vertices in a group have the same number of descendants, subtrees whose roots are the two vertices are constructed according to rule 1 and rule 2. Comparing two sequences of descendants of two subtrees lexicographically, two subtrees (therefore two vertices corresponding to the two roots) are arranged in lexicographical descending order. If the two sequences are the same, the arrangement is arbitrary and the two subtrees are equivalent.

(2) BCT canonical form is converted to the canonical BCT table according to the following rules.

(a) Let $m$ and $n$ be the number of blocks and cutpoints, respectively; $m$ rows and $n$ columns are assigned to blocks and cutpoints, respectively.

(b) Rows and columns are divided among the levels. Arrangement of the levels is shown by Figure 4a or b.

(c) Arrangement in a level of the BCT table corresponds to the order of BCT canonical form obtained by (1).

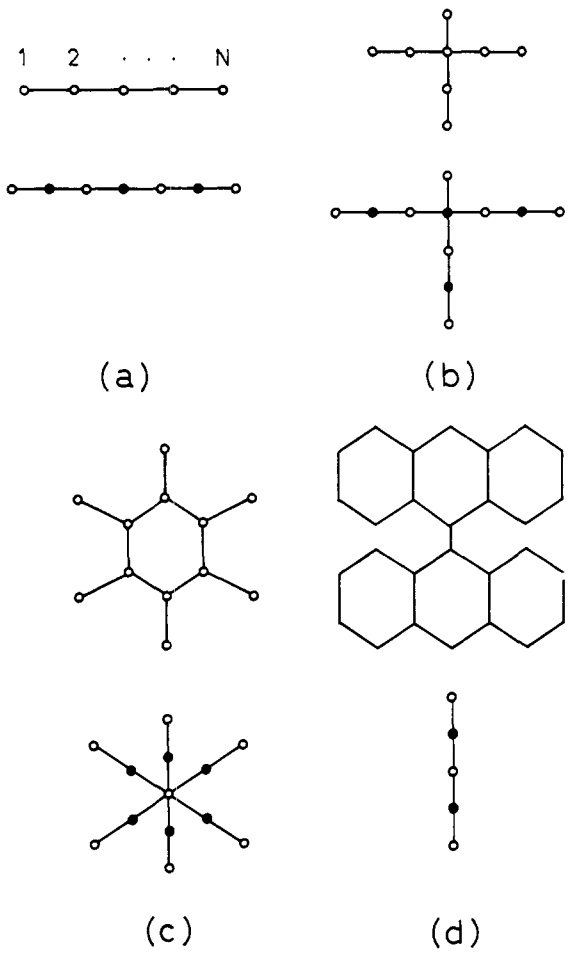


**Figure 7.** Graphs and their BCTs.

Examples of BCT canonical form and a canonical BCT table are shown in Figure 5. The size of the adjacency matrix of BCT is $(m + n) \times (m + n)$. On the other hand, the size of a BCT table is $m \times n$, illustrated by shading in Figure 6. Thus, the memory space occupied by the BCT table is $m{\cdot}n/(m + n)^2$ to that by the adjacency matrix. (If $m \approx n$, the BCT table occupies one-fourth of the memory space of the adjacency matrix.) If a given graph $G = (X, U)$ is a tree, the number of vertices of BCT derived from $G$ is twice the number of $|X|$. When $m \approx n \approx |X|$, the memory size of BCT ($m{\cdot}n = |X|^2$) is equal to the size of the adjacency matrix. Figure 7 shows several relationships between graphs and their BCTs concerning the number of vertices. As shown there, the number of vertices of BCT varies greatly, according to the type of original graph. The type which increases the number of vertices maximally is (a), and the storage needed is $m{\cdot}n = (N - 1)(N - 2) < N^2$, where $N = |x|$, so that it does not exceed the size of the adjacency matrix of the original graph. In case (d), the number of vertices of the BCT decreases remarkably, with great improvement in the efficency of memory use. As shown later, the information concerning the internal (ring) structure of the blocks is given by the block dictionary. On the other hand, the canonical tree representation by the Edmonds algorithm[14] needs just $N$ (i.e., it is a sequence of $N$ figures). Each figure requires several bits to represent the number of descendants, while the BCT table requires only *1* bit for *1* number. Therefore, the canonical representation of a tree by the Edmonds algorithm requires $k{\cdot}N$ bits, as compared with the BCT table which requires $m{\cdot}n$ bits, where $k$ is the number of bits necessary to represent one number. Figure 8 shows the behavior of the number of bits required in the case of $m = n$. If $k = 16$, memory size required by the BCT table is less than that of sequence representation for $N$

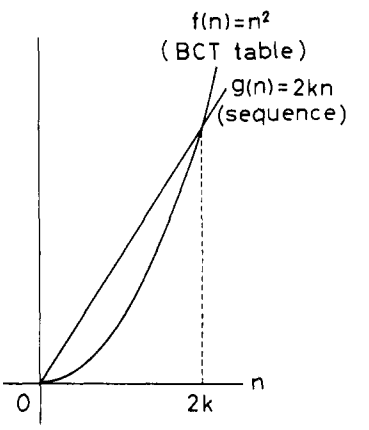


**Figure 8.** Number of bits occupied by a graph. $n$ = (number of BCT vertice)/$2k$ bits for a vertex.

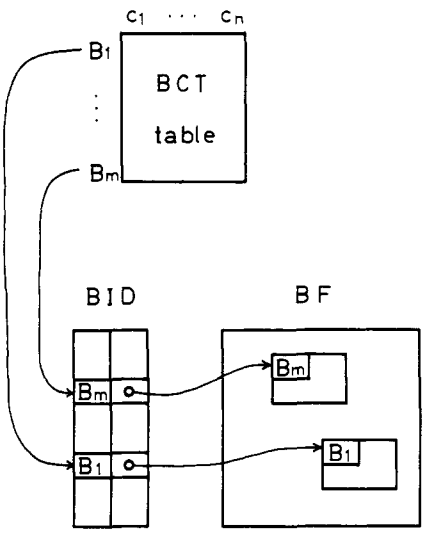


**Figure 9.** Relation between BCT table and block dictionary: BID, block identifier table; BF, block file.

< 64. This implies that the BCT table requires at most the same memory size as sequence representation for practical use. A BCT table has an advantage over other methods with respect to simplicity of manipulation, owing to the block dictionary. The relation between the BCT table and the block dictionary is shown in Figure 9. Blocks are represented by colors (i.e., identifiers), so that required memory size is economized greatly by this file organization. Thus the bipartite BCT table is selected as a data format. The internal structure of a block is the content of the block dictionary (see Figure 9), and it is represented in the form of a connection table.[15] Positions of cutpoints are specified by means of unique numbering of vertices in a block. This information is put in an interface file, independent of a block dictionary. The hierarchy of the structural information and the flexible manipulation in the hierarchy are justified by this file organization.

## III. ALGORITHM

In order to obtain a BCT table for a given graph, cutpoints and blocks of the graph must be found before the generation of a canonical BCT. An algorithm to find cutpoints of a graph has been previously presented by Harary,[16] who introduced the BCT concept to graph theory; we have developed and used the following algorithm.

**1. Cutpoint Finding.** A graph of interest is $G = (X, U)$, where $X = \{x_1, \ldots, x_n\}$, $U \subset X \times X = \{(x,y)|x \in X, y \in X\}$ (see II.1). We show the logical foundation of the algorithm to examine whether $p \in X$ is a cutpoint or not. This algorithm

examines all the vertices in $X$. A hypergraph $H$ is defined as follows:[17] $\mathcal{E} = (E_i|i = 1, 2, \ldots)$ generates hypergraph on $X$ if $\mathcal{E}$ satisfies the conditions:

$$1)\ E_i \neq \phi \qquad \text{for } i = 1, 2, \ldots$$

$$2)\ \bigcup_i E_i = X$$

$H = (X, \mathcal{E})$ is called a hypergraph, where $X = \{x_1, \ldots, x_n\}$ is a set of vertices and $E_i$ is an edge. A vertex $x$ is adjacent to a vertex $y$ if there exists an edge which contains both vertices $x$ and $y$. An edge $E_i$ is adjacent to an edge $E_j$ if $E_i \cap E_j \neq \phi$. A path of length $q$ is defined as a sequence $(x_1, E_1, x_2, E_2, \ldots, E_q, x_{q+1})$ satisfying the following conditions:

$$1)\ x_1, \ldots, x_{q+1} \in X$$

$$2)\ E_1, \ldots, E_q \in \mathcal{E}$$

$$3)\ x_k, x_{k+1} \in E_k \qquad \text{for } k = 1, \ldots, q$$

Suppose $x \sim y$ denotes that there exists a path from $x$ to $y$; $x \sim y$ is the equivalence relation. Equivalent classes by this equivalence relation are called connected components of a hypergraph. Now the following steps give the cutpoint finding algorithm ($p \in X$ is a vertex to be examined):

(1) $S^1 = (Z_1, \mathcal{E}_1)$ is a hypergraph, where

$$Z_1 = \{z|d(p,z) = 1, z \in X\}$$

$$\mathcal{E}_1 = U \cap (Z_1 \times Z_1) = (E_i|i = 1, 2, \ldots), \qquad |E_i| = 2$$

Classes $S_1^1, S_2^1, \ldots, S_l^1$ are obtained by applying the equivalence relation ($\sim$) to $Z_1$. (These are the connected components of $S^1$.)

(2) Suppose that classes $S_1^{J-1}, \ldots, S_l^{J-1}$ ($l \geq 2$) are obtained by classifying a set of vertices:

$$Z_{J-1} = \{z|d(p,z) = J - 1, z \in X\}$$

Then the procedure classifying a set of vertices:

$$Z_J = \{z|d(p,z) = J, z \in X\}$$

is as follows: $S^J = (Z_J, \mathcal{E}_J)$ is a hypergraph, where

$$\mathcal{T} = (T_1, \ldots, T_l),\ T_i = \{z|d(p,z) = J, a_{y,z} = 1 \text{ for } y \in S_i^{J-1}\}$$

$$\mathcal{E} = U \cap (Z_J \times Z_J) = (E_i|i = 1, 2, \ldots)$$

and

$$\mathcal{E}_J = \mathcal{T} \cup \mathcal{E}$$

where $a_{y,z}$ is an element of the adjacency matrix of $G$. Therefore $Z_J$ can be classified by the equivalence relation into classes $S_1^J, \ldots, S_k^J$ ($1 \leq k \leq l$) which are the connected components of $S^J$.

(3) The following criteria decide whether $p$ is a cutpoint or not: (1) If the number of connected components of $S^J$ ($J = 1, 2, 3, \ldots$) is one, $p$ is not a cutpoint. (2) If $T_i = \phi$ connected to $S_i^{J-1}$ ($i = 1, \ldots, l$, $l \geq 2$), $p$ is a cutpoint. These criteria are based on the following definition or properties of a cutpoint. A vertex $p$ is a cutpoint of a connected graph $G$ if and only if $G$ becomes disconnected by removal of $p$. Criterion 1 means a case that the number of connected components has been found to be only one at a distance $J$ from $p$ ($J = 1, 2, \ldots$, max distance). Criterion 2 means a case that if a connected component made of vertices within a distance $J - 1$ has been found, it is a component separated from another part of the graph.

**2. Block Finding.** A terminal vertex and its adjacent vertex always constitute a block, which here will be called a "T-block". Given an original graph $G_0 = (X_0, U_0)$ and a set of terminal vertices $T_0 \subset X_0$, a graph $G_1 = (X_1, U_1)$ and $T_1 \subset X_1$ are obtained by removal of $T_0$ where $T_1$ is a set of terminal vertices in $X_1$. If $G_i = G_{i+1}$, a graph $G_i$ contains no terminal vertices. T-blocks are picked out until the state reaches that of $G_i = G_{i+1}$ according to the procedure shown in Figure 10. The following procedure is applied to a graph $G = (X, U)$ which contains no terminals. Terminal vertices of a BCT are always blocks (BCT property 4). The procedure is based on the same idea as that for T-blocks, which picks out blocks from the exterior. But this BCT terminal does not appear in the adjacency matrix explicitly.

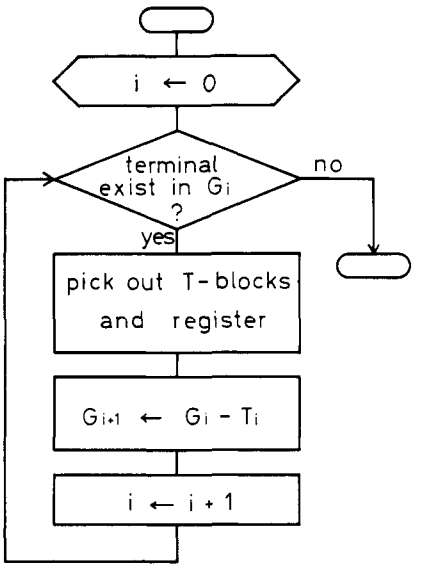


**Figure 10.** Procedure to pick out T-blocks.

(1) A cutpoint $c$ is selected.

(2) All vertices in $X$ are classified according to the distance from $c$:

$$X = X_0 \cup X_1 \cup \ldots \cup X_m,\ X_i \cap X_j = \phi\ (i \neq j)$$

$X_i$ is a set of vertices at a distance $i$ from $c$ ($X_0 = \{c\}$).

(3) The flow shown in Figure 11 gives the procedure to pick out blocks. The details of the procedure PICK is described as follows:

(a) Suppose $c_j$ is a cutpoint of $d(c,c_j) = J$, and a set of vertices $A$:

$$A = \{x|d(c,x) = J + 1, a_{x,cj} = 1\}$$

where $\alpha_{x,cj}$ denotes an element of the adjacency matrix of $G$. A hypergraph $H$ is defined as follows:

$$H = (A, \mathcal{E})$$

$$\mathcal{E} = U \cap (A \times A) = (E_i|\ i = 1, 2, \ldots), |E_i| = 2$$

Then the connected components $B_1, \ldots, B_L$ are obtained by classifying vertices of $A$ according to the equivalence relation ($\sim$). The number of connected components obtained by extending $B_1, \ldots, B_L$ to the exterior does not exceed $L$. And at least one block is contained in the connected components. (This is warranted because $G$ is a connected graph and $c_j$ is a cutpoint of $G$.)

(b) There always exists a vertex which is farther than $c_j$ from $c$, so let $k = 2$ initially.

(c) Connected components $B_1, \ldots, B_L$ are extended as follows: A family of sets

$$\mathcal{T} = (T_1, \ldots, T_L)$$

$$T_i = \{x|d(c,x) = J + k, a_{x,b} = 1, b \in B_i\}$$

is generated. If $T_i = \phi$ for some $i$, $B_i$ yields a block. If $\mathcal{T} = \phi$, the procedure ceases to extend, and $B_1, \ldots, B_L$ are blocks,

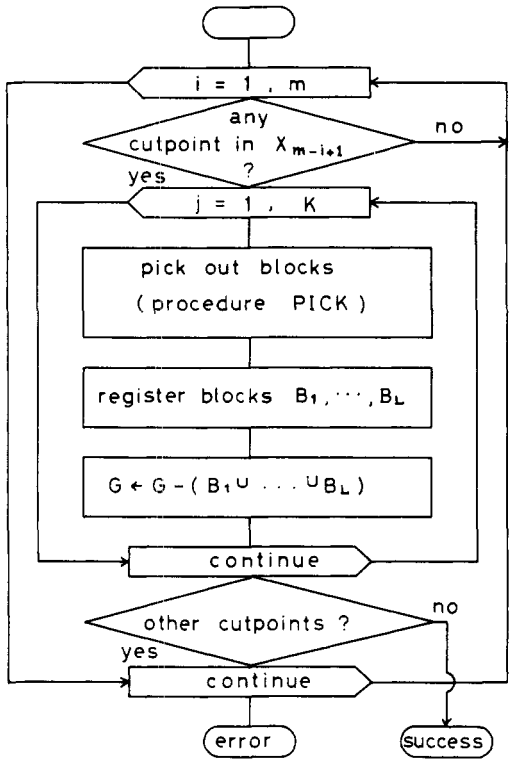


**Figure 11.** Procedure to pick out blocks.

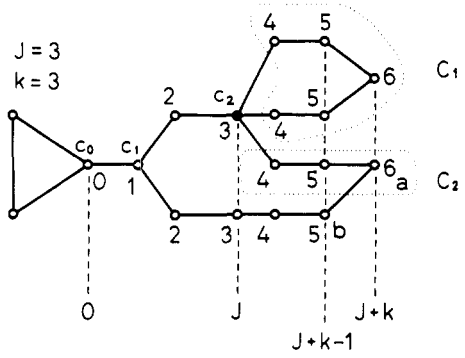


**Figure 12.** Class $C_2$ should be removed because vertex *a* is adjacent to vertex *b*.

respectively. Now suppose $\mathcal{T}$ contains no empty sets; a hypergraph $H = (A, \mathcal{E})$ is constituted:

$$A = (\overset{L}{\underset{i=1}{\cup}} B_i) \cup (\overset{L}{\underset{i=1}{\cup}} T_i)$$

$$\mathcal{E} = \mathcal{T} \cup \mathcal{B}, \mathcal{B} = (B_1, \ldots, B_L)$$

A set $A$ is classified into equivalent classes (connected components) according to the equivalence relation ($\sim$):

$$\mathcal{C} = (C_1, \ldots, C_k)$$

(d) Connectivity of $C_1, \ldots, C_k$ with the other part of $G$ is examined (see Figure 12). If $C_i$ (therefore $T_i$) is adjacent to the vertex of a distance $J + k - 1$ or $J + k$, the extension of $C_i$ does not yield a block. Therefore $C_i$ is removed from $\mathcal{C}$. After examination of this connectivity, a family of classes $\mathcal{C}$ gives new classes (subscript $L$ may be renewed):

$$\mathcal{B} = (B_1, \ldots, B_L)$$

(e) $k$ is increased by one: $k \leftarrow k + 1$. If $k$ exceeds maximal distance, $B_1, \ldots, B_L$ yield blocks, respectively. Otherwise, the procedure is repeated from step c).

## IV. EFFICIENCY OF CUTPOINT FINDING ALGORITHM

The final state of class generation is illustrated in Figure 13. The vertices enclosed by a dotted line constitute classes at this stage. Figures 13a–e are the cases that the reference vertex is not a cutpoint, and Figures 13f–h are the case of cutpoints. For example, case a shows that the number of classes is only one at a distance 1. Case f shows that the number of classes is three when one of the classes completes the connected component (which means that the removal of the reference vertex increases the number of connected components). As is shown in these figures, the processing time depends upon the number of vertices processed until the nearest connected component is found. Therefore, the vertices with degree 2 on a cycle are found to be non-cutpoint at a distance $n/2$ ($n$ even) or $(n - 1)/2$ ($n$ odd), where $n$ is the number of vertices containing the reference vertices. (Cases a, b, and d in Figure 13 are the examples when $n$ = 3, 6, and 6, respectively.) In turn, the degree of the reference vertex gives the upper limit of the number of classes, so it is almost always less than 5 if a graph is a chemical structure.

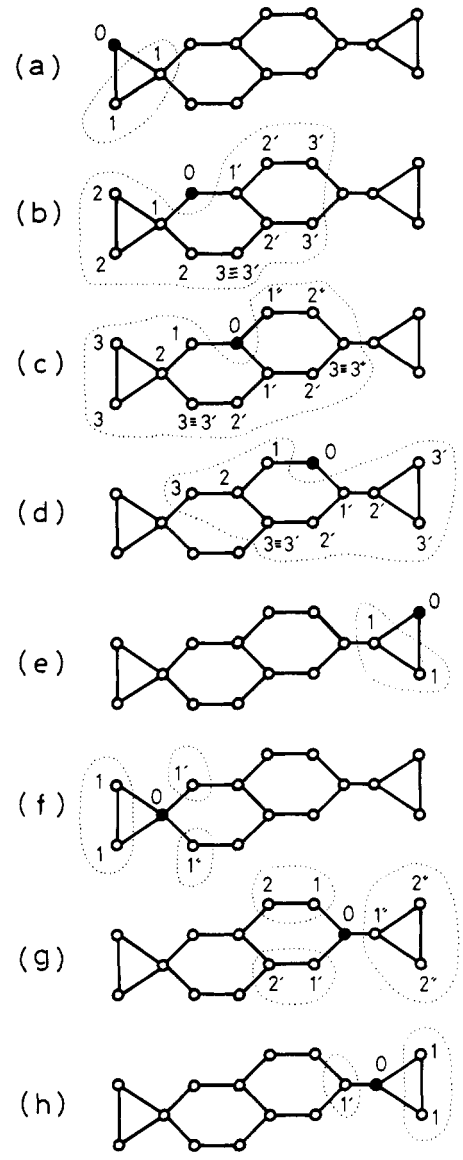


**Figure 13.** Final states of class generation. Numbers 1, 2, 3, etc., denote the distance from the reference vertex (black circle).

The efficiency of the block-finding algorithm is estimated similarly.

## V. CONCLUSION

A hierarchical representation of chemical structure is useful from the viewpoint of processing time as well as memory

requirements. A block-cutpoint tree (BCT) is used to represent chemical structures. The algorithm to construct a BCT was written in APL and implemented on the IBM 5100. Input data to this algorithm are in the form of an adjacency matrix of compounds. The compound file and block dictionary have been designed. The compound file is a set of BCT representations of compounds and the topological relationship among blocks is described. Further structural informations, such as atomic identification, bond types, steric relations, etc., are described in a block dictionary. Complicated structures need another file to describe substituted positions (i.e., cutpoints), symmetry, steric relations, and so on. These will be utilized in application systems, such as substructure search, automatic analysis of spectra, and molecular design, which are being developed in our laboratory.

## ACKNOWLEDGMENT

The authors wish to thank Dr. Masahiro Uchino for helpful discussions.

## REFERENCES AND NOTES

(1) J. E. Ash and E. Hyde, "Chemical Information Systems", Ellis Horwood, Chichester, England, 1975, Chapter 11.
(2) D. J. Gluck, "A Chemical Structure Storage and Search System Developed at DuPont", *J. Chem. Doc.*, **5**, 43 (1965).
(3) J. E. Rush, "Status of Notation and Topological Systems and Potential Future Trends", *J. Chem. Inf. Comput. Sci.*, **16**, 202 (1976).
(4) S. Rossler and A. Kolb, "The GREMAS Systems, an Integral Part of the IDC System for Chemical Documentation", *J. Chem. Doc.*, **10**, 128 (1970).
(5) E. G. Smith, "The Wiswesser Line-Formula Chemical Notation", McGraw-Hill, New York, 1968.
(6) IUPAC "Rules for IUPAC Notation for Organic Compounds", Wiley, New York, 1961.
(7) H. Skolnik, "A New Linear Notation Based on Combination of Carbon and Hydrogen", *J. Chem. Doc.*, **6**, 689 (1969).
(8) L. Quadnelli, V. Bareggi, and S. Spiga, "A New Linear Representation of Chemical Structures", *J. Chem. Inf. Comput. Sci.*, **18**, 37 (1978).
(9) Chi-Hsiung Lin, "SEFLIN-Separate Feature Linear Notation System for Chemical Compounds", *J. Chem. Inf. Comput. Sci.*, **18**, 41 (1978).
(10) R. G. Dromey, "A Simple Tree-Structured Line Formula Notation for Representing Molecular Topology", *J. Chem. Inf. Comput. Sci.*, **18**, 225 (1978).
(11) J. E. Dubois, "Principles of the DARC Topological System", *Entropie*, **25**, 5-13 (1969).
(12) Nicos Christofides, "Graph Theory, An Algorithmic Approach", Academic Press, New York, 1975.
(13) Frank Harary, "Graph Theory", Addison-Wesley, Reading, Mass., 1969.
(14) J. Edmonds, *J. Res. Natl. Bur. Std. B*, 69 (1965).
(15) Masahiro Uchino, "Array-Theoretic Approach to Basic Problems in Chemical Information: Unique Coding, Substructure Search and Perception of Topochromic Symmetry", The ACS/CSJ Chemical Congress, 1979.
(16) Ian C. Ross and Frank Harary, "Identification of the Liaison Person of an Organization Using the Structure Matrix", *Manage. Sci.*, **1** (1955).
(17) C. Berge, "Graphes et hypergraphes", Dunod, Paris, 1970.

# MOLY—An Interactive System for Molecular Analysis

THOMAS M. DYOTT, A. J. STUPER,* and G. S. ZANDER
Rohm and Haas Company, Spring House, Pennsylvania 19477

Received July 10, 1979

This paper describes an interactive computer system, which is used to assist in the design of biologically active compounds. The system, called MOLY, is capable of: providing rapid construction and manipulation of three-dimensional, chemically correct molecular images; providing visual comparisons between different molecules; performing detailed conformational analysis; estimating a molecule's lipophilicity; and providing a limited description of a molecule's electronic properties. The architecture of MOLY is such that additions and enhancements can be easily implemented.

## INTRODUCTION

The last decade has seen considerable progress toward understanding the relationships between chemical structure and biological activity. A number of advances in this area have resulted from the simultaneous growth in sophistication of the models which describe activity and the computers which perform these calculations.

A major advance in modeling has been the development of relations which predict changes in transport properties on the basis of structural variations occurring within a homologous series.[1] The ability to incorporate certain electronic and steric factors into the model has made development of Quantitative Structure Activity Relationships (QSAR) a powerful tool for rationalizing the effects of structural alteration upon biological activity. Unfortunately, such models are only applicable to changes within a homologous series. They cannot be used to indicate the structural requirements for compounds which deviate significantly from the original lead.

The problem of developing new classes of compounds which exhibit a desired activity remains the subject of intensive investigation. Much work remains to be done in understanding the effect structural alterations have on (1) chemical reactivity, (2) physiochemical properties, (3) electronic properties, and (4) steric interactions.

In recent years a great deal of effort has gone into developing theoretical tools which can shed some light on these aspects of the structure–activity puzzle.[2] Unfortunately, knowing any one of these properties alone is insufficient for developing active compounds. All of these properties work synergistically to cause a specific effect or effects.

This paper describes our efforts to develop a computer system which provides an arsenal of sophisticated techniques with which to attack problems in molecular design. This is not a simple collection of programs, but rather an integrated computer system capable of providing sophisticated forms of molecular analysis. The system, called MOLY, is capable of evolving as more and better methods for structure–activity analyses are developed.

Currently MOLY is capable of providing (1) construction and manipulation of three-dimensional chemically correct molecular images, (2) visual comparisons between different molecules, (3) detailed conformational analysis, (4) a limited description of a molecule's electronic properties, and (5) an estimate of a molecule's lipophilicity. The remaining sections of this paper detail the organization of MOLY and our plans for future growth.

## SYSTEM OVERVIEW

MOLY is an interactive computer graphics system with the following characteristics: (1) large (≃800K bytes), (2)

0095-2338/80/1620-0028$01.00/0 