



STRUCTURE GENERATION ON THE BASIS OF BCT
REPRESENTATION OF CHEMICAL STRUCTURES

by

Takashi Nakayama

and

Yuzuru Fujiwara *

July 22, 1981

INSTITUTE
OF
INFORMATION SCIENCES AND ELECTRONICS

UNIVERSITY OF TSUKUBA

Structure Generation on the Basis of BCT
Representation of Chemical Structures

abstract

A method of structure generation based on BCT (block-cutpoint tree) representation of chemical structure has been developed. The generation program is a part of the automatic structure analysis system of mass spectra(ASASMAS), and is used when a set of the inferred substructures are given as input data. The input substructures are represented by means of BCT.

INTRODUCTION

The major parts of the process of chemical structure analysis are inference of constituent substructures and structure generation by combining those inferred substructures. A method of structure generation from substructures already inferred is described in this paper. Various schemes for structure generation have been devised,¹⁻¹⁰ each of these methods is based on a method of representation of chemical structures, and it may be said that the method of representation of chemical structures determines the method of structure generation. All of these methods including the present one pursue the governing principle of reducing the number of combinations. In this paper, the chemical structure is represented in terms of BCT(block-cutpoint tree),¹¹ which clarifies the hierarchy in chemical structures wholly so that the idea of the connectivity stack³ and the superatom⁵ are included naturally in our method presented. Combinatorial problems that occur in the process of structure generation are thus partitioned into subproblems and are classified into strages.

Structure generation is regarded as a problem in combinatorial analysis. There must be neither duplications nor omissions in the generated structures, and the method of structure generation should be efficient; processing time should be short in practice. The processing time depends largely on the method of representation of chemical structures. The constituent unit of the BCT representation of chemical structures is a block (a biconnected component of a graph), and this makes it possible to reduce the number of combinations greatly by omitting atom-by-atom processing. Another feature of the method of structure generation described here is the availability of a graph database. The various graphs which appear in the process of structure generation are not generated for each case, but instead are retrieved from the graph database.

REPRESENTATION OF CHEMICAL STRUCTURES

Chemical structures are represented by means of graphs, and atoms and bonds of a chemical structure correspond to vertices and edges of a graph respectively. However, it is inefficient to process chemical structures at the atomic level, and sometimes a superatom/ring assembly is used as the processing unit. In ASASMAS, the concept of the superatom is extended to represent the chemical structures in a way that is consistent with BCT. The following is a brief description of BCT representation of chemical structures.¹²

Let $u \in V$ be a vertex of a connected graph $G = (V, E)$. A vertex u is a "cutpoint" if the removal of u yields the disconnectivity of the graph G . A "block" of a graph G is a maximal subgraph of G which contains no cutpoints. Now, $bc(G)$ (block-cutpoint graph of G) is defined: $T = (W, F)$ is a $bc(G)$ if (1) $W = A \cup B$ is a set of vertices where $A = \{ a_1, \dots, a_n \}$ is a set of all cutpoints of G , and $B = \{ b_1, \dots, b_m \}$ is a set of all blocks of G , and (2) $F = \{ f_1, \dots, f_l \} = \{ (a_i, b_j) \mid a_i \in b_j, a_i \in A, b_j \in B \}$, where $a_i \in b_j$ means that a cutpoint a_i is a member of a

set of vertices of block b_j . A $bc(G)$ has the following properties: (1) it is a bipartite graph of subsets A and B, (2) it is a tree regardless of the original structure, (3) terminal vertices correspond to blocks of G, and (4) the distance between any pair of terminal vertices is an even number. A $bc(G)$ is called a block-cutpoint tree (BCT) because of property (2). A tree is a BCT if and only if it possesses property (4), and this property is used for generating BCT. Examples of the BCT representation of chemical structures are shown in Figure 1. The internal structure of each block is filed in the block dictionary.

GRAPH DATABASE

The graph generation procedure which would usually be performed in the process of structure generation is replaced by the retrieval of object graphs from the graph database. As it is impossible to prepare all of the graphs, a graph which cannot be found in the graph database is generated when it is first required and this generated graph is also added to the database. There are five major files in ASASMAS: (1) a compound file, (2) a table of correspondence between substructures and mass spectral components (CSSC), (3) a block file, (4) a BCT file, and (5) a spectral data file. A compound file and a CSSC contain the chemical structures or substructures in the form of BCT representation, and are used for inferring substructures in the process of structure analysis. The internal structures of block in that BCT representation are stored in a block file. A BCT file is a set of BCTs which are organized according to parameters (m,n), where m is the number of blocks and n is the number of cutpoints of a BCT, and is used when needed during the process of structure generation. The records of a BCT file

are trees which possess the BCT's property (4) described previous section and furthermore satisfy the condition; $\text{deg } u \leq 4$, where u is a cutpoint of a BCT, because the degree of u cannot exceed the valence of u which is considered to be an atom such as carbon, oxygen, or nitrogen. The outline of the system ASASMAS (especially the relation between programs and these five data files) is shown in Figure 2.

METHOD OF STRUCTURE GENERATION

Outline of structure generation. Program GENERATION is initiated when a set of substructures b_1, \dots, b_m is given as input data, where b_1, \dots, b_m are simple blocks that have been inferred by program ANALYSIS. GENERATION generates all the possible combinations of blocks b_1, \dots, b_m with neither omission nor duplication. The outline of the procedure is: (1) all the BCTs of m blocks and n cutpoints are generated, where $1 \leq n \leq m-1$; (2) blocks b_1, \dots, b_m are assigned to block vertices of a generated BCT (labeling); (3) cutpoints are assigned to vertices in each block of the labeled BCT; and (4) object structures are obtained by connecting blocks according to the cutpoint assignment.

Generation of (m,n) BCT. A BCT of m blocks and n cutpoints is denoted by (m,n) BCT. The first step of structure generation which generates (m,n) BCT is for the most part accomplished by retrieving them from the BCT file. This reduces the processing time of structure generation. When m blocks are given, the number of cutpoints n varies from 1 to $m-1$, i.e., $1 \leq n \leq m-1$. The case of

$n=1$ corresponds to a "star" whose center is a cutpoint, and $n=m-1$ is the case that the degree of every cutpoint is equal to two.

The procedure of (m,n) BCT generation is to collect the trees that have m block vertices and n cutpoint vertices. Let N -tree denote the tree with N vertices, where $N=m+n$. An N -tree is generated from an $(N-1)$ -tree by adding one vertex. Now, let $T = \{ t_1, t_2, \dots, t_l \}$ be a set of all $(N-1)$ -trees, where t_i represents an $(N-1)$ -tree, and let T_i be a set of trees which are generated from $t_i \in T$ by adding one vertex, $T' = T_1 \cup T_2 \cup \dots \cup T_l$ is a set of N -trees. Therefore the generation procedure of N -trees is divided into two parts: the generation of T_i ($i=1, \dots, l$) and the generation of the union T' .

(1) Generation of T_i . Let $t_i = (V, E)$, $t_i \in T$. First a set of vertices V is divided into orbits of permutation group on V . The element of this permutation group is automorphism of t_i , that is to say, the permutation on a vertex set V which preserves the adjacency relationship among vertices:

$$V = V_1 \cup V_2 \cup \dots \cup V_r, \quad V_p \cap V_q = \phi \quad \text{for any } p \neq q.$$

The vertices in a subset V_j ($j=1, \dots, r$) are topologically equivalent. Second, an N-tree is generated from t_i by adding one vertex to the arbitrary vertex in V_j ($j=1, \dots, r$). Therefore the number of generated N-trees is the number of orbits, i.e., $|T_i| = r$.

(2) Generation of T' . $T' = T_1 \cup T_2 \cup \dots \cup T_r$ gives the total N-trees, but $T_i \cap T_j = \phi$ does not always hold. The duplications (i.e., the elements of $T_i \cap T_j$) should be eliminated. This problem is solved by introducing linear order into the set of N-trees; an N-tree is coded into numerical sequence by Edmonds' method,¹³ and is stored in the appropriate vertex of binary tree which should represent the set of N-trees.¹⁴ Duplication is detected when a generated N-tree is found in the vertex in which it should be stored. The set of N-trees is generated in this manner, but it consists of BCTs and non-BCTs. An N-tree is a BCT if and only if it satisfies the BCT's property (4) described in section 2. An N-BCT is a bipartite graph whose vertex set consists of block and

cutpoint vertices, and are classified according to the parameter (m,n) , where m and n are the numbers of block vertices and cutpoint vertices respectively. The BCTs which are stored in the BCT file are restricted to those in which the degrees of all cutpoint vertices do not exceed four, because the ceiling of the free valence may be set to four for most organic compounds. These are called valid BCTs. Table I shows the number of BCTs in each class corresponding to (m,n) . The valid $(6,n)$ BCTs ($n=2, \dots, 5$) are shown in Figure 3. The total number of valid $(6,n)$ BCTs is 20.

Labeling vertices of (m,n) BCT. A BCT is a bipartite graph whose vertex set consists of block and cutpoint vertices, and the terminal vertices are all block vertices. Therefore, it is easy to distinguish block vertices from cutpoint vertices. The second step of the structure generation is to assign the elements b_1, \dots, b_m of the given block set to the block vertices of the (m,n) BCT. This procedure is called labeling.

Now let C_1, \dots, C_k denote the colors of blocks, and let n_i be the number of blocks whose

color is C_i ($i=1, \dots, k$), where it is assumed that $n_1 \leq n_2 \leq \dots \leq n_k$ without loss of generality. Labeling is in block number order; the first is C_1 , the second C_2 , and so on.

(1) If $n_1 = n_2 = \dots = n_{k-1} = 1$, C_i ($i=1, \dots, k-1$) is assigned to the representative block vertex of a class partitioned on the basis of topochromatic equivalency. If block vertices of an (m, n) BCT are partitioned into l classes g_1, \dots, g_l , there are l ways of assignments of C_i . When C_i is assigned to the representative of a class, b_j , the following condition should be satisfied between the degree of the representative in the (m, n) BCT ($\deg b_j(C_i)$) and the number of the constituent vertices of the block C_i ($|C_i|$):

$$\deg b_j(C_i) \leq |C_i| \quad (A)$$

If there are no classes which satisfy the condition (A), it means that the assignment of C_1, \dots, C_{i-1} which resulted in the assignment of the C_i should be forbidden. After the assignment of C_1, \dots, C_{k-1} is finished, C_k is assigned to the remaining block vertices under condition (A).

This labeling procedure can be represented in the form of a tree, and an assignment of C_1, \dots, C_k corresponds to a path in the tree which connects the root with a terminal vertex. Therefore the number of labeling methods is equivalent to the number of terminal vertices of depth $(m-1)$ in the tree, and there are no duplicates. This tree is called a generation tree.

(2) When $N_j \geq 2$ for some $j < k$, the vertices

whose depth is from $\sum_{p=1}^{j-1} n_p$ to $(n_j - 1 + \sum_{p=1}^{j-1} n_p)$

for $j \geq 2$, or from 0 to $(n_j - 1)$ for $j = 1$, are the same color in the generation tree, so if the procedure described above is applied, there will be duplication. To avoid the duplication, color C_j is assigned n_j times to block vertices: First, the block vertices to which colors are not yet assigned are partitioned according to topochromatic equivalency. This partition is not limited to the simple classification based on orbits, but is further subdivided. Those classes are organized hierarchically. The subdivision is performed according to the degree of consanguinity among vertices. If the distance between vertices u and

v of the same orbit in a rooted tree is k , the degree of consanguinity between u and v is k . This is denoted by $\text{san}(u,v) = k$. The relation introduced into vertices of the same orbit as the degree of consanguinity within k ($\text{san}(u,v) \leq k$) is an equivalence relation (denoted by " \equiv "). The following apply if u , v and w are vertices which belong to the same orbit: 1. a reflexive law: $u \equiv u$ ($\text{san}(u,u) = 0$). 2. a symmetric law: if $u \equiv v$, then $v \equiv u$. 3. a transitive law: if $u \equiv v$ and $v \equiv w$, then $u \equiv w$. (The transitive law is shown as follows: assuming that $u \not\equiv w$, and x is the closest common ancestor of u and w , there should exist two paths which connect v with y or z ($\neq x$) which are on the path between u and w . Then there exists a cycle which contain vertices x , y and z , and this contradicts that x , y and z are the vertices of a tree.) If partitions of $\text{san}(u,v) \leq k_1$ and of $\text{san}(u,v) \leq k_2$ are obtained for k_1 and k_2 when $k_1 < k_2$, the former must be a subdivision of the latter. Therefore the hierarchical partitioning structure is obtained based on the relation of the degree of consanguinity (see Figure 4).

This partitioning structure can be represented in the form of a tree (called a partitioning tree, as in Figure 4). Then the labeling problem at this step is equivalent to enumerating all the ways of assignment of the same things (color C_j) to the rooms (vertices in a level) stepwise in order of depth. The capacity of each room (i.e., the maximum number of colors to be admitted) is determined for every depth. That is to say, the capacity of a room is equivalent to the sum of the numbers of members of classes corresponding to the descendant leaves of the vertex (a room).

After color C_j is assigned to the leaves of a partitioning tree, specific block vertices to which color C_j is to be assigned are selected arbitrarily from leaf classes by the number of assigned members. In the generation tree, labeling paths are grown by connecting N_j members selected serially with the parent. This problem is equivalent to the problem of partition of an integer under some restriction.

The example of labeling one of the (6,4)BCT is as follows: A (6,4)BCT and block colors C_1 ,

C_2 and C_3 which should be assigned to it are shown in Figure 5(a). Block vertices of the (6,4)BCT are numbered from 1 to 6. As $n_1=1$, $n_2=2$ and $n_3=3$ (i.e., $n_1 < n_2 < n_3$), C_1 , C_2 and C_3 are assigned in this order. And

$$\#node(C_1) = 3, \quad \#node(C_2) = 6, \quad \#node(C_3) = 2$$

and

$$\text{deg1} = \text{deg2} = \text{deg4} = \text{deg5} = \text{deg6} = 1; \quad \text{deg3} = 4$$

so colors C_1 and C_3 cannot be assigned to vertex 3 (condition (A) does not hold).

(1) Assignment of C_1 . Block vertices are partitioned into three orbits, $g_1 = \{ 3 \}$, $g_2 = \{ 1, 2 \}$ and $g_3 = \{ 4, 5, 6 \}$, but g_1 is rejected by condition (A). Therefore there are two kinds of assignment of C_1 : g_2 and g_3 . It is arbitrary to select representative vertices from each class, and the selecting vertex 1 from g_2 and vertex 4 from g_3 corresponds to the roots of generation trees T_1 and T_2 shown in Figure 5(b).

(2) Assignment of C_2 . In case of generation tree T_1 , the remaining vertices are partitioned into three orbits $g_1 = \{ 2 \}$, $g_2 = \{ 3 \}$ and

$g_3 = \{ 4, 5, 6 \}$. Since further subdivision based on the degree of consanguinity does not exist for these classes, n_2 (= two) of the color C_2 are assigned to g_1 , g_2 and g_3 . There are four sets of assignments, $\{ g_1, g_2 \}$, $\{ g_1, g_3 \}$, $\{ g_2, g_3 \}$ and $\{ g_3 \}$. The number of members assigned to each class (each element of a set calculated above) are $\{ 1, 1 \}$, $\{ 1, 1 \}$, $\{ 1, 1 \}$ and $\{ 2 \}$.

And finally, selection of specific members from each class is arbitrary, so we get $\{ 2, 3 \}$, $\{ 2, 4 \}$, $\{ 3, 4 \}$ and $\{ 4, 5 \}$ as examples.

Procedures for generation tree T_2 are similar and straightforward.

(3) Assignment of C_3 . This is the final color to be assigned, and the only way is to assign C_3 to all the remaining vertices. However, it should be noted that some paths are blocked by condition (A) upon assigning C_3 ; color C_1 and C_3 cannot be assigned to vertex 3, and this is shown by X in Figure 5(d). Final generation trees are obtained as in Figure 5(d) when C_3 's are assigned.

Specific labelings corresponding to paths connecting a root with leaves of depth 5 are shown in Figure 5(e)

Assignment of cutpoints. It is not clear how those labeled block vertices are connected at the atomic level at this stage. The detailed structures which specify the connectivity among blocks at the atomic level are generated by allocating cutpoints in each block of the labeled BCT. For example, in the (6,4)BCT in Figure 5(a), cutpoints a_1 from block 1 and 2, cutpoints a_1, a_2, a_3 and a_4 from block 3, cutpoint a_2 from block 4, cutpoint a_3 from block 5, and cutpoint a_4 from block 6 are selected and the connectivity among blocks at the atomic level is determined.

The procedure of selecting cutpoints a_1, \dots, a_p from a block b_i is as follows: let $j=1$.

- (1) The constituent vertices of b_i are partitioned into orbits: g_1, \dots, g_q .
- (2) An arbitrary vertex of g_k ($k=1, \dots, q$) is selected as a candidate for a_j . There are q kinds of selections of a_j for a given selection of a_1, \dots, a_{j-1} .
- (3) If $j=p$, then end. Otherwise, go to (4).
- (4) The selected vertex is regarded as a vertex which is given a new color a_j ; go to (1) with

$j + j + 1$.

This procedure is also implemented in the form of generation tree described in the previous labeling procedure for BCT.

The example of cutpoint assignment for a benzene ring is shown in Figure 6.

Connecting blocks according to cutpoint assignment. Using the method of cutpoint assignment for each block vertex of a labeled BCT, structures are generated by connecting blocks with other blocks in turn, fusing the cutpoints which should be the same atom in the generated structures.

If there are no symmetries in a labeled BCT (i.e., if there are no block vertices which are equivalent topochromatically), all the combinations of cutpoint assignments for block vertices give different assembled structures, i.e., there are no duplicates among them. The number of

combinations of cutpoint assignment is $\prod_{i=1}^m l_i$,

where there are l_i kinds of cutpoint assignment in block b_i . However, if there is a symmetry in a labeled BCT, there are duplicates in

$\prod_{i=1}^m \mathcal{Z}_i$ structures. To avoid these duplicates, the problem of connecting the blocks is considered in the same way as the problem of coloring block vertices. This is equivalent to selecting a kind of cutpoint assignment for each block vertex, where the kinds of colors are equivalent to the kinds of cutpoint assignment inherent to the blocks. Therefore, this problem is solved similarly to the problem of BCT labeling. The structures generated by connecting blocks of Figure 5(e) are shown in Figure 7. Six structures shown in the lowest part of Figure 7 correspond to the first and last three assignments for six-ring (C_2) shown above them. There is only one labeled BCT which could be connected at the atomic level. The other three BCTs are excluded by the condition that the degree of a cutpoint must not be greater than four.

Partitioning vertices into orbits. When block vertices of (m,n) BCT are labeled, or cutpoints are selected from constituent vertices of a block, it is necessary to partition these vertices into orbits according to topochromatic

equivalency. Vertices u and v of a labeled graph G are equivalent topochromatically if $f(u) = v$ for some automorphism f of G , where an automorphism of a graph G is a permutation on a set of vertices of G which preserves the adjacency among vertices. A set of all automorphisms of G forms a group. The orbits of this group are the classes partitioned according to topochromatic equivalency.

Therefore, if all the automorphisms are found, it is easy to partition vertices. In practice, it is easy to find all the automorphisms for a graph that is not highly symmetrical by using backtrack search. The procedure of finding automorphisms is as follows:

Let $V = \{ 1, 2, \dots, n \}$ be a set of vertices of G . The procedure checks whether the adjacency relationship among vertices is preserved for a permutation;

$$f = \begin{pmatrix} 1 & 2 & \dots & n \\ i_1 & i_2 & \dots & i_n \end{pmatrix}$$

Let $f(k) = (i_1, \dots, i_k)$ denote the left portion of a permutation f corresponding to k

vertices, i.e.,

$$f = \begin{pmatrix} & k+1 & \cdots & n \\ f(k) & & & \\ & i_{k+1} & \cdots & i_n \end{pmatrix}$$

Assuming that $f(k)$ is given, $f(k+1)$ is determined as follows: (1) $i_{k+1} \in V - \{i_1, \dots, i_k\}$. (2) Both $(k+1)$ and i_{k+1} belong to the same orbit. (3) The adjacency relationship formed by vertices $\{1, \dots, k, k+1\}$ should be formed by $\{i_1, \dots, i_k, i_{k+1}\}$, where it is guaranteed that the adjacency relationship formed by $\{1, \dots, k\}$ is formed by $\{i_1, \dots, i_k\}$, so it only has to be guaranteed that the adjacency between i_{k+1} and $\{i_1, \dots, i_k\}$ is the same as the one between $(k+1)$ and $\{1, \dots, k\}$. Automorphisms are obtained when the procedure reaches $f(n)$. If there are no vertices that satisfy the conditions (1), (2) and (3), then there are no automorphisms which contain $f(k)$ as a part of a permutation (see Figure 8).

It is not required necessarily to find all the automorphisms for partitioning vertices into orbits, because vertices can be classified into some classes (these are integrated to some orbits)

when an automorphism is found, and the automorphism that should be found next is conditioned that it has to contain at least one cycle that maps a vertex (member) in some class to another class's member. If there is no automorphism that satisfies the condition, the resulted classes give the orbits. The program that is implemented according to this algorithm is also prepared, and is useful for getting orbits for highly symmetrical graphs.

Partitioning vertices of a tree into orbits.

The programs that give partitioning of vertices are prepared for both highly and not-so-highly symmetrical graphs by finding automorphisms defined on a set of vertices. On the other hand, the particular partitioning procedure that does not use automorphisms explicitly is prepared for trees.

This partitioning is accomplished by using Edmonds' canonical form of a tree. Assuming that there is one "center" in a given tree. Then the only vertex that is topologically equivalent to the center is itself. Regarding this tree as a rooted tree whose root is the center vertex,

vertices u and v are equivalent if and only if they satisfy the following three conditions:

- (1) Vertices u and v are the same distance from the center.
- (2) The parents of u and v are the same vertex, or an equivalent one.
- (3) The subtrees whose roots are u or v are isomorphic, where a subtree is defined as a subgraph that consists of a vertex (u or v) and its all the descendants in the given rooted tree. Since these conditions are checked immediately in terms of Edmonds' canonical form of a tree, partitioning vertices is given by investigating vertices for every distance from the root to leaves of a maximal distance. The classes partitioned in this fashion agree with the orbits of the automorphism group described previously.

When there are two centers in a given tree, it can be coped with by inserting a dummy vertex between the two vertices, converting the case of one center (see Figure 9).

CONCLUSION

A method to generate chemical structures is presented, and it is based on BCT representation of chemical structures, on labeling nodes of bipartite graphs (BCT) corresponding to blocks and on allocating cutpoints in the blocks.

The programs were implemented in FORTRAN on a ACOS-900 computer, and the spectral data were edited from EPA/NIH Mass Spectral Database (1975 edition, about 11300 records). BCT file is organized according to parameter (m,n) as shown in Table I. The execution speed of structure generation depends on the times of orbit computation in the generation stages such as BCT labeling, cutpoint assignment and then connecting blocks. Apparently orbits should be computed for each node of a generation tree except leaves, so the number of orbit computation equals to the number of nodes (not leaves) of generation trees. The execution time of orbit computation depends on the number of vertices and symmetric degree of the graph, so we prepared three types of programs for orbit computation. This method of structure generation is useful for automatic structure elucidation.

tion, and constitutes a part of ASASMAS.

REFERENCES

- (1) Nelson, D.B.; Munk, M.E.; Gash, K.B.;
Herald, D.L.Jr. J.Org.Chem. 1969, 34, 3800.
- (2) Antosz, F.J.; Nelson, D.B.; Herald, D.L.Jr.;
Munk, M.E. J.Am.Chem.Soc. 1970, 92, 4933.
- (3) Shelly, C.A.; Munk, M.E. Anal.Chim.Acta.
1978, 103, 245.
- (4) Shelly, C.A.; Munk, M.E. J.Chem.Inf.Comput.
Sci. 1979, 19, 247.
- (5) Kudo, Y.; Sasaki, S. J.Chem.Doc. 1974, 14,
200.
- (6) Kudo, Y.; Sasaki, S. J.Chem.Inf.Comput.Sci.
1976, 16, 43.
- (7) Masinter, L.M.; Sridharan, N.S.; Lederberg, J.;
Smith, D.H. J.Am.Chem.Soc. 1974, 96, 7702.
- (8) Masinter, L.M.; Sridharan, N.S.; Carhart, R.E.;
Smith, D.H. J.Am.Chem.Soc. 1974, 96, 7714.
- (9) Carhart, R.E.; Smith, D.H.; Brown, H.;
Djerassi, C. J.Am.Chem.Soc. 1975, 97, 5755.
- (10) Carhart, R.E.; Smith, D.H.; Brown, H.;
Sridharan, N.S. J.Chem.Inf.Comput.Sci.
1975, 15, 124.
- (11) Nakayama, T.; Fujiwara, Y. J.Chem.Inf.Comput.
Sci. 1980, 20, 23.

- (12) Harary, F. " Graph Theory " ; Addison-Wesley: Reading, Massachusetts, 1969.
- (13) Busacker, R.G.; Saaty, T.L. " Finite Graphs and Networks: An Introduction with Applications " , McGraw-Hill, 1965; Chapter 6.
- (14) Aho, A.V.; Hopcroft, J.E.; Ullman, J.D. " The Design and Analysis of Computer Algorithm " , Addison-Wesley: Reading, Massachusetts, 1974.

FIGURE LEGENDS

- Figure 1. BCT representation of chemical structures.
- Figure 2. Block diagram of the automatic structure analysis system of mass spectra (ASASMAS); GENERATION is the program set for structure generation.
- Figure 3. Valid (m,n)BCTs for $m=6$.
- Figure 4. Partitioning of vertices of a tree including the degree of consanguinity.
- Figure 5. (a) A skeleton of a (6,4)BCT and blocks.
(b) Assignment of C_1 to (6,4)BCT and roots of corresponding generation trees T_1 and T_2 .
(c) Growth of generation trees through assignment of C_2 .
(d) Completion of generation tree T_1 and T_2 .
(e) Four kinds of labeling corresponding to four paths of generation trees.
- Figure 6. (a) Partitioning of vertices.
(b) Assignment of C_1 and following

partitioning.

(c) Assignment of C_2 and following
partitioning.

(d) Assignment of C_3 .

Figure 7. Connecting blocks at the atomic level.

Figure 8. Example of vertex partitioning. There
are 32 automorphisms on the set of
vertices of this graph.

Figure 9. Example of tree vertex partitioning.
There are $9!$ automorphisms on the set
of vertices of this tree. In practice,
it is impossible and unnecessary to get
all of them for the purpose of parti-
tioning vertices.

Table I. Number of BCTs classified according to
parameter (m,n).

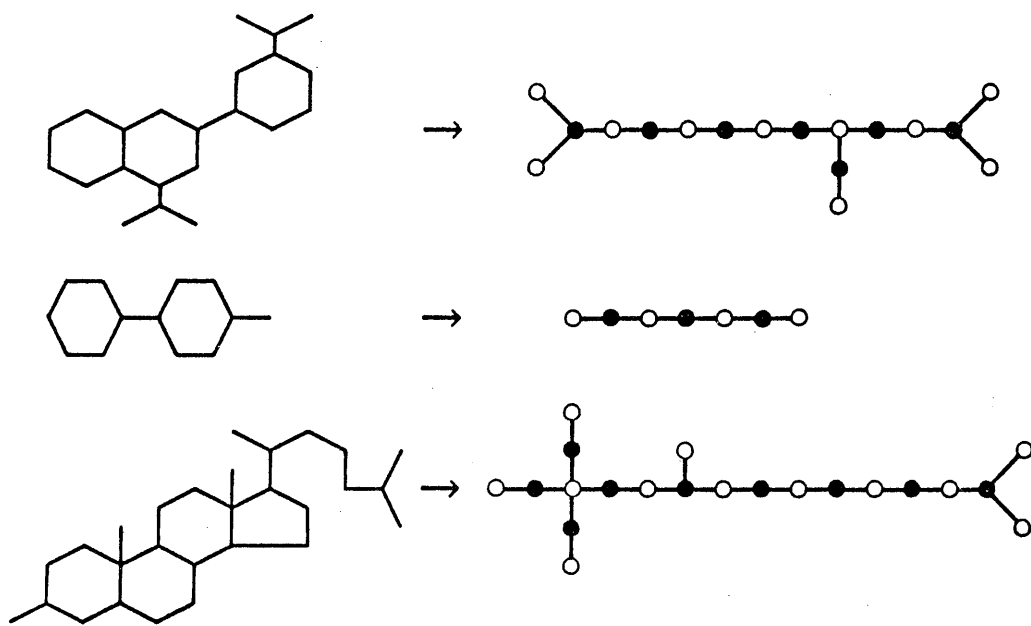


Figure 1

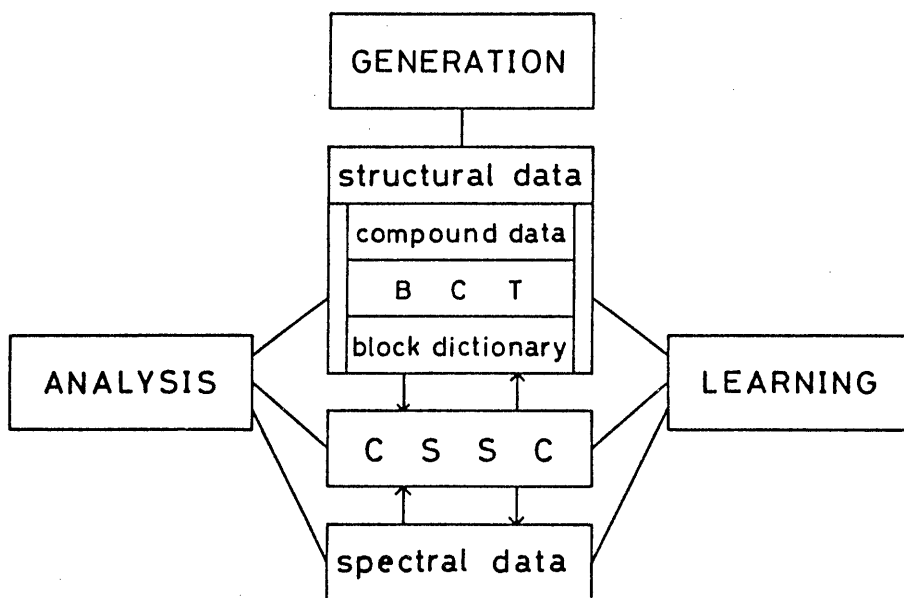


Figure 2

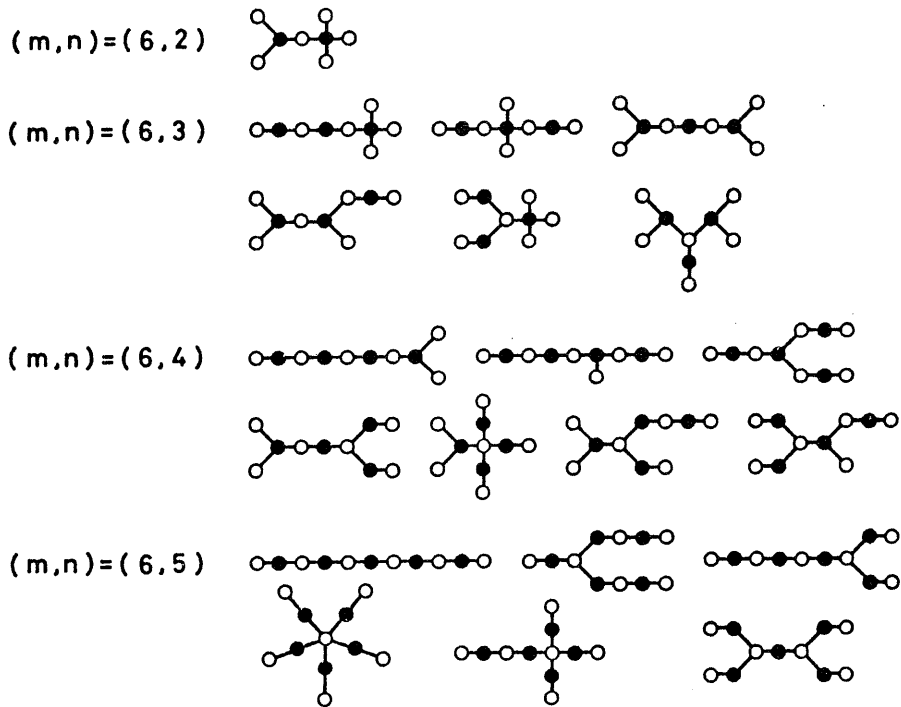
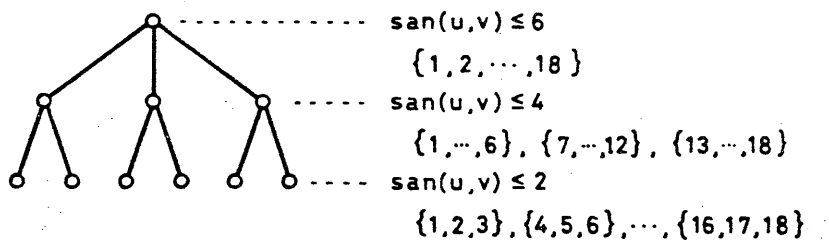
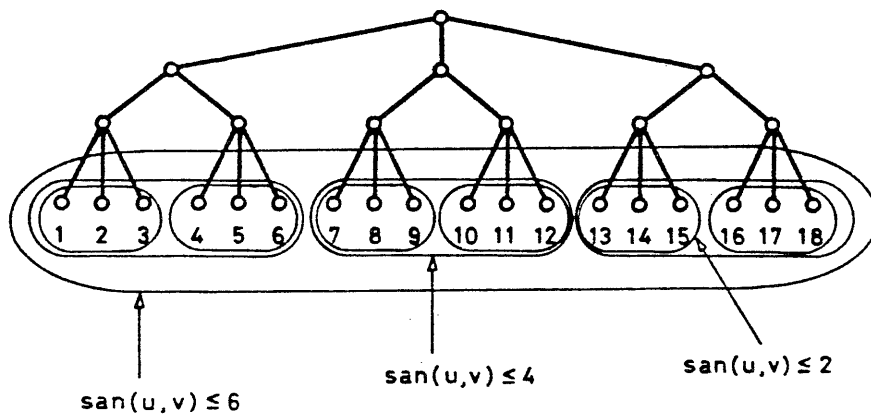


Figure 3



tree representation of partitioning structure

Figure 4

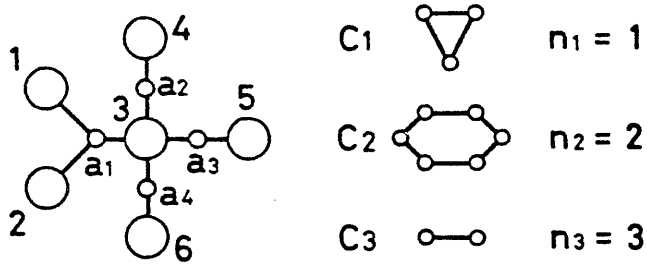


Figure 5(a)

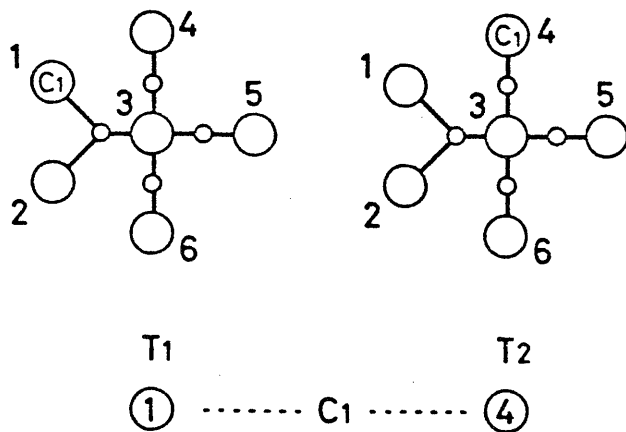


Figure 5(b)

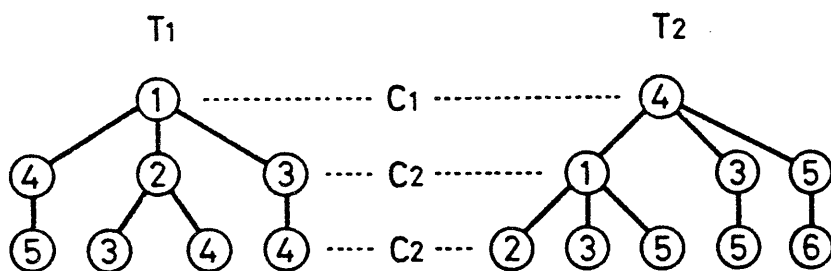


Figure 5(c)

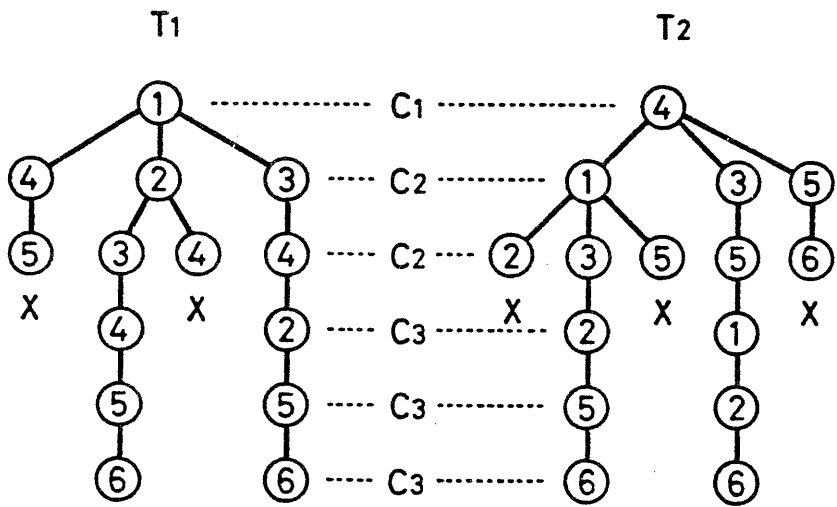


Figure 5(d)

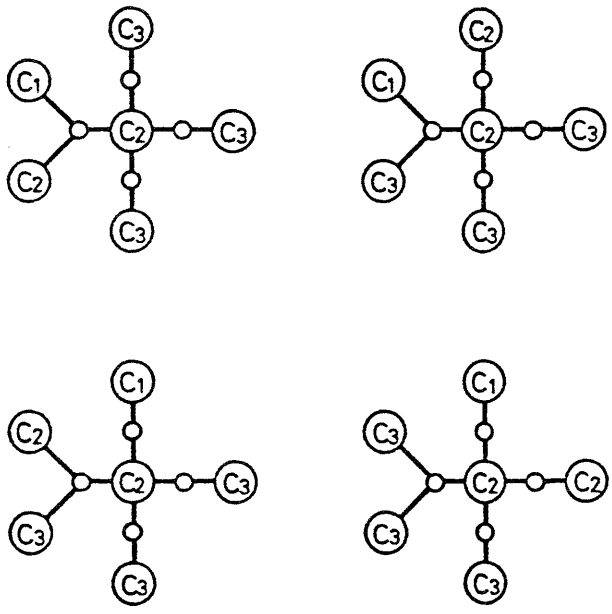


Figure 5(e)

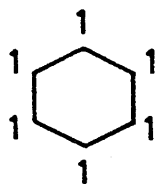


Figure 6(a)

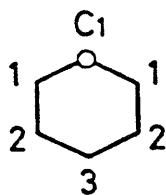


Figure 6(b)

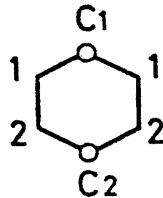
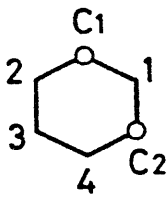
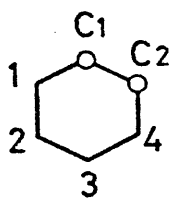


Figure 6(c)

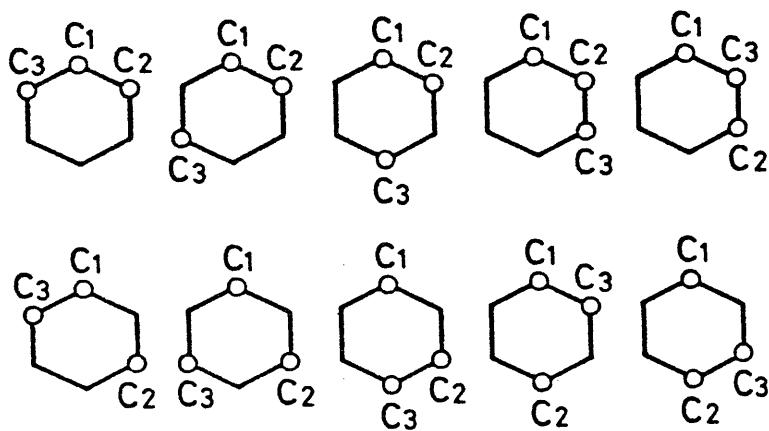


Figure 6(d)

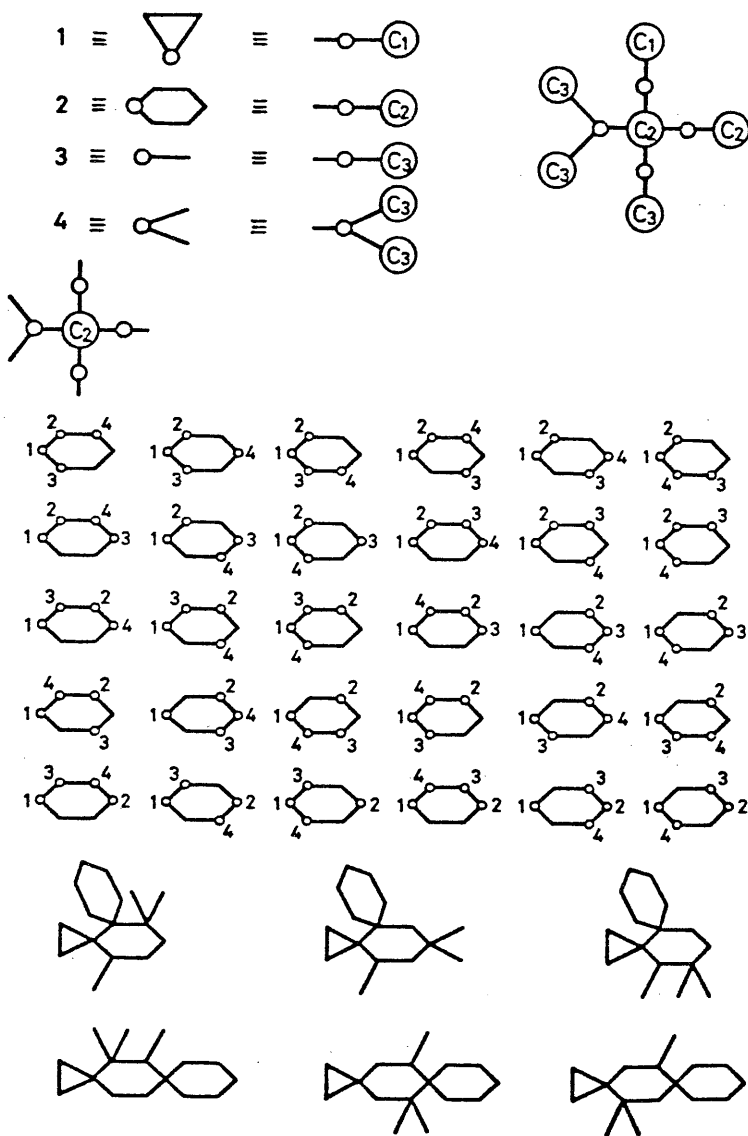


Figure 7

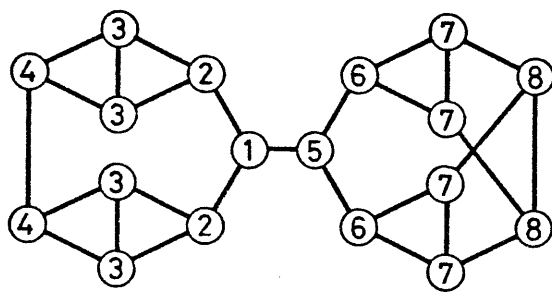
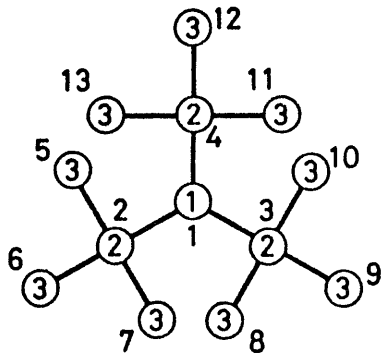


Figure 8



node 1 2 5 6 7 3 8 9 10 4 11 12 13
 code 1 3 4 1 1 1 4 1 1 1 4 1 1 1
 s1 s2 s3

$$s1 = s2 = s3 \Rightarrow 2 = 3 = 4$$

similarly

$$5 = 6 = 7 = 8 = 9 = 10 = 11 = 12 = 13$$

Figure 9

m \ n	1	2	3	4	5	6	7
2	1						
3	1	1					
4	1	1	2				
5	1	2	3	3			
6	1	2	6	7	6		
7	1	3	9	17	18	11	
8	1	3	13	30	51	44	23
9	1	4	17	53	109	148	
10	1	4	23	79	213		
11	1	5	28	119			
12	1	5	35				
13	1	6					
14	1						

Table I. Number of BCTs classified according to parameter (m,n).

INSTITUTE OF INFORMATION SCIENCES AND ELECTRONICS
UNIVERSITY OF TSUKUBA
SAKURA-MURA, NIIHARI-GUN, IBARAKI 305 JAPAN

REPORT DOCUMENTATION PAGE	REPORT NUMBER ISE-TR-81-21
TITLE Structure Generation on the Basis of BCT Representation of Chemical Structures	
AUTHOR(s) Yuzuru Fujiwara [Institute of Information Sciences, University of Tsukuba] Takashi Nakayama [Central Research Laboratory, Kuraray Co. Ltd.]	
REPORT DATE July 22, 1981	NUMBER OF PAGES 37
MAIN CATEGORY 3.7 Information Retrieval	CR CATEGORIES 3.79
KEY WORDS Graph database/graph generation/BCT graph/ASASMAS/learning system	
ABSTRACT A method of structure generation based on BCT (block cutpoint tree) representation of chemical structure has been developed. The generation program is a part of the automatic structure analysis system of mass spectra (ASASMAS), and is used when a set of the inferred substructures are given as input data. The input substructures are represented by means of BCT.	
SUPPLEMENTARY NOTES	