match no. 27 pp. 153-174 1992

THE GENERATION OF MOLECULAR GRAPHS FOR QSAR STUDIES BY THE ACYCLIC FRAGMENT COMBINING

O. A. Lomova, D. V. Sukhachev, M. I. Kumskov,
V. A. Palyulin, S. S. Tratch, N. S. Zefirov*
Department of Chemistry, Moscow State University,
Moscow, 119899, U. S. S. R.

(received: March 1992)

Abstract. An algorithm for generation of acyclic combinations of fragments (taken from the given basic set) is suggested. It can be used for construction of molecular graphs for the computer-assisted search of the compounds with definite properties by means of quantitative structure-property and structure-biological activity models. The formal description of fragment types and operations for them is given. The generation algorithm is considered in detail. The sets of generated structures and the possible ways to reduce them including the additional selection criteria are discussed. The WLN code was used for the realization of the algorithm of this kind. The computer program GOLD for IBM PC AT based on the suggested algorithm allows the comparatively fast generation (*ca.* 50 structures per second) and storage on a hard disk of 10^4-10^6 structures.

INTRODUCTION

During the last decade the problem of computer-assisted designing of organic structures with definite properties attracts the increasing attention. The search of structure - biological activity and structure - property relationships (QSAR and QSPR, respectively) became a fruitful and extensively developing field of chemistry [1,2]. The solution of these problems can be subdivided into two steps. At the first step the set of compounds with known properties (usually belonging to the comparatively narrow class) is used to find the quantitative relationships of property/biological activity with the structure of the compounds; the structure is usually characterized by various descriptors [3]. At the second step the regularities found are used for predicting of properties for a series of new compounds. In this connection the necessity arises to generate such series of organic structures for the prediction of properties and selection of structures with necessary properties. It should be noted that the structures generated should usually belong to the definite class (the same as used at the first step).

A variety of methods to generate chemical structures is used in chemistry. Most of them are oriented to the generation of structures from their molecular formula [4-11]. Computer generation of chemical structures from fragments for the analysis of ^{13}C NMR data has been also discussed [12], and recently the generation of structures by random combination of known fragments was considered [13]. In the present paper the algorithm for generation of acyclic combinations of fragments from the given set is suggested, the molecular formula in

this case is not kept constant for the generated structures.

The following problems are solved to realize this type of structure generation.

1. The generation of nonisomorphic structures. The algorithm described allows to prevent the appearance of duplicates (which could appear owing to the symmetry of some of the fragments).

2. The restriction of the number of structures generated. As far as the number of possible fragment combinations increases rapidly with the increase of the initial number of fragments used in the generation procedure, the allowed combinations of the fragments should be strongly limited to obtain the necessary fragment combinations in reasonable time.

The suggested algorithm provides the generation of nonisomorphic molecular graphs corresponding to correct chemical structures. Furthermore the algorithm includes the logical functions (predicates) for selection of a comparatively small number of structures from the broad class of allowed acyclic combinations of the given fragments.

GENERAL STATEMENT OF THE PROBLEM

Let define the fragment as an ordered triple (G, U_G, E_G) (see Fig.1) where:

G is a connected nondirected graph with labeled vertices and multiple edges (in fact, it is a molecular subgraph which can contain cyclic and branched parts);

U_G is an one-element ($U_G=\{u_G\}$) or empty ($U_G=\emptyset$) set, which contains an arc (directed edge) having the sink in G (the vertex in a graph G

where the arc u_G enters) and not having the source in G (the vertex in G from which the arc exits); such an arc and corresponding vertex will be named the fragment's sink and designated on figures as $\longrightarrow\bullet$;

E_G is a nonempty or empty set of arcs having the sources in a graph G and not having the sink in G; such arcs (and corresponding vertices) will be named the sources of the fragment and designated as $\bullet\longrightarrow$ in figures (the source of several arcs from the same vertex in G is permissible).

The arcs u_G and $e_G \in E_G$ can be multiple. Note, that one of the sets U_G and E_G must be necessarily nonempty.

We stress here, that a fragment is not a graph (for the corresponding graphs *vide infra*), because it contains the sink and/or sources corresponding to "free valences" (entering and exiting arcs each of them connected with a single vertex). We define the degree p of a fragment as the number of its sources $|E_G|$.

Let us introduce the classification of the fragments. We define the central fragment (denoted as C) as a fragment not having a sink u_G (U_G is empty) and having nonempty set of sources E_G (for the example see Fig.1b). Other fragments considered here have a single sink u_G and are subdivided into the following types:

1) terminal fragments, t ($p = 0$);

2) linear fragments, l ($p = 1$);

3) branched fragments, b ($p > 1$).

We shall also differentiate the elementary fragments (EFs) which are given and the composite fragments which are constructed as combinations of EFs. The composite fragments in our algorithm can be only terminal (denoted as TCFs) or linear.

For example, let the set Z of EFs (Fig.1a) and the central fragment C (Fig.1b) be given. The set Z consists of one terminal, one branched and one linear fragment. The structures are generated in the following way. Firstly, the sets $\hat{T}$ of TCFs are obtained by means of all permissible acyclic combinations of the EFs from the set Z. The example of a set $\hat{T}$ is presented on Fig.1c. Secondly, the generation of the required molecular graphs (Fig.1d) needs all permissible replacements of the corresponding sources of the central

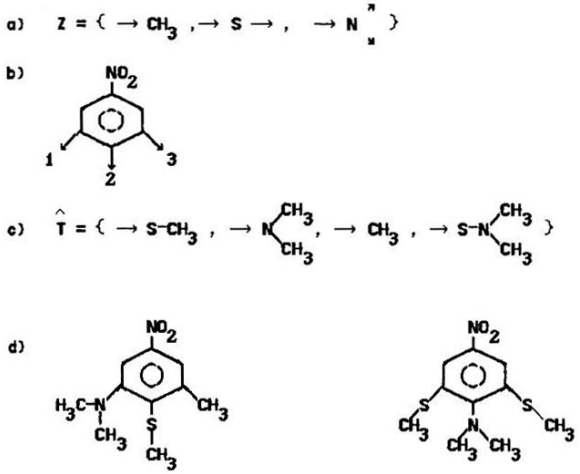


Fig.1. General scheme of generation of molecular graphs:

(a) The set Z of EFs.

(b) The central fragment C.

(c) The set $\hat{T}$ of generated TCFs.

(d) The examples of generated molecular graphs.

fragment C by TCFs from $\hat{T}$ to be performed. The multiplicities of TCFs' sinks and those of the corresponding sources of C should be taken into account.

Note, that owing to the symmetry of branched EFs from a set Z the identical TCFs could in principle be generated. Similarly, the symmetry of a central fragment C could also lead to the generation of identical resulting graphs. The symmetry of branched EFs as well as of the fragment C is, however, taken into account in our algorithm to avoid the generation of identical results.

OPERATIONS WITH FRAGMENTS

Let us define the combining operation for a fragment x of a degree p with the ordered sequence of fragments $S = (s_1,\ldots,s_p)$ and denote it as x&S. This operation is defined if for each pair (e_i, s_i), $i=\overline{1,p}$, the multiplicity of an ith source e_i of a fragment x and that of the single sink of the ith fragment s_i coincide.

For description of the algorithm of the generation of TCFs from the given set of EFs it is sufficient to define the operation x&S for three different combinations of fragments:

linear - linear, $l\&(l_i)$;

linear - terminal, $l\&(t_i)$;

branched EF - sequence of terminal fragments, b&S.

Let us consider each of the cases in detail.

1. Combining of linear fragments with linear ones, $l_1\&(l_2)$.

For two linear fragments the combining operation results in the sink-source linkage leading to the formation of linear composite

fragment. For example:

l_1: ->CO->; l_2: ->CH_2->; l_1&(l_2): ->CO-CH_2->

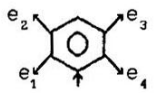
2. Combining of linear fragments with terminal ones, l&(t).

As in the previous case, for these kinds of fragments the combining operation results in the sink-source linkage; the terminal composite fragment (TCF) is obtained in this case. For example:

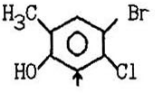
l= ->CO->; t = ->OH; l&(t) = ->CO-OH

3. Combining of the branched EF with the ordered sequence of terminal fragments, b&S, S = (t_1,...,t_p), p being the degree of the branched EF b.

For these kinds of fragments the operation consists in combining all sources from the set E_b of a fragment b with the sinks corresponding to a sequence S of identical or nonidentical terminal fragments. The linkage of p sink-source pairs (e_i,t_i), $i=\overline{1,p}$, obviously, leads to the formation of the TCF. The example is given for the branched EF b of degree 4 and the sequence S consisting of 4 nonidentical terminal fragments, S = (OH, CH_3, Br, Cl):

b: 

t_1: ->OH
t_2: ->CH_3
t_3: ->Br
t_4: ->Cl

b&S: 

Note that this operation can result in the formation of identical TCFs due to the symmetry of a fragment b . We really overcome this problem by the account of the symmetry of a fragment b, *vide infra*.

THE ALGORITHM FOR GENERATION OF COMPOSITE TERMINAL FRAGMENTS

The main idea of the algorithm is as follows: firstly, the linear EFs are combined into chains, i.e. into composite linear fragments (see Fig. 2a). Secondly, the resulting composite linear fragments are combined with the terminal EFs thus forming the set $\hat{T}$. The terminal EFs are also included into $\hat{T}$ (see Fig. 2b). The resulting fragments from $\hat{T}$ are sequentially combined with branched and then linear fragments, and the TCFs obtained at this stage are also added into the

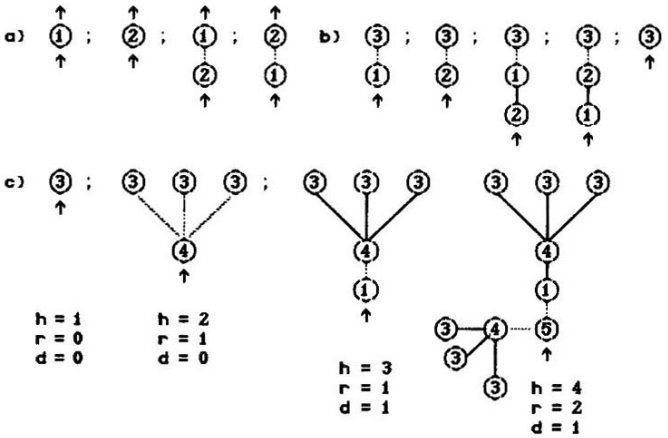


Fig.2. General scheme of generation of TCFs:

a) Generation of linear composite fragments from linear EFs.

b) Generation of TCFs from linear composite fragments and terminal EFs (in the set $\hat{T}$, the terminal EF is included).

c) The sequential generation of TCFs by their combining with branched and linear EFs.

set $\hat{T}$ (see Fig.2c). In Fig.2 the combining of fragments at each stage is designated by dotted lines.

Fig.2 shows, that each TCF can be represented by the acyclic graph with vertices corresponding to EFs and edges corresponding to the linkages between EFs formed at different generation stages.

Let us introduce for such graphs (and for corresponding TCFs) the following numerical characteristics:

1) height (h) is the number of vertices (number of EFs) in the longest path connecting the root of acyclic graph with a leaf (the terminal EF); the vertex having the sink is thought to be the root of an acyclic graph.

2) rank (r) is the maximal number of branched EFs in any of paths, connecting the root with leaves. All paths must be taken into account.

3) disperse (d) is a maximal number of adjacent linear EFs in a branched acyclic graph. In the case of unbranched acyclic graphs the disperse value is equated to zero.

Let $Z = T \cup L \cup B$, with $T=\{t_i\}$, $L=\{l_j\}$, $B=\{b_k\}$ be the sets of terminal, linear, and branched EFs. We shall construct the set of TCFs (denoted as $\hat{T}$) consisting of TCFs with $r \leq M$; $d \leq N$; $h \leq (N+1)*(M+1)$. The values of M and N are the given integers.

Let X and F be the sets of fragments. We shall denote as X&F the set of composite fragments obtained by the allowed combining operations applied to all fragments $x \in X$ and all possible ordered sequences $S=(f_1,\ldots,f_p)$, consisting of p fragments from F , p denotes the degree of any fragment x, for which the combining operation is allowed.

The following algorithm is used for the generation of TCFs:

Stage I. The generation of all possible linear fragments:

$$L_1=L; \quad L_i=L_{i-1} \cup (L \& L_{i-1}); \quad i=\overline{2,N}.$$

Stage II. The generation of the TCFs which do not contain branched EFs:

$$T_1=T \cup (L_N \& T);$$

At this stage all TCFs with $h \leq N+1$, $d=0$, $r=0$ are obtained.

Stage III. The sequential combining of TCFs with branched EFs and with linear fragments from L_N:

For $i=1,\ldots,M$ perform steps 1-3, starting with the set T_1, obtained at the stage II:

Step 1. Combine branched EFs with TCFs from T_i and form the auxiliary set V_i :

$$V_i = B \& T_i.$$

To avoid the repeated generation of the same fragments only nonequivalent sequences S, which contain at least one TCF with $r=i-1$, are allowed at this step. The set V_i contains all the TCFs with $h \leq (N+1)*i+1$; $r=i$; $d \leq N$.

Step 2. Combine the linear composite fragments from the set L_N with TCFs from V_i just obtained and form the auxiliary set W_i:

$$W_i = L_N \& V_i .$$

The set W_{i+1} contains all the TCFs with $h \leq (N+1)*(i+1)$; $r=i$; $d \leq N$.

Step 3. $$T_{i+1} = T_i \cup V_i \cup W_i.$$

Thus, the resulting set T_{i+1} contains all possible TCFs with $h \le (N+1)*(i+1)$; $r \le i$; $d \le N$.

After M steps the desired set $T_{M+1}=\hat{T}$ is formed which contains all TCFs with $h \le (N+1)*(M+1)$; $r \le M$; $d \le N$.

Note 1. For generation of nonidentical fragments it is enough to take into account only the symmetry of the branched EFs at step 1; the details will be discussed in the next section.

Note 2. The algorithm can lead to generation of an extremely large number of composite fragments (especially at the stage III) even for moderate values of M and N. Thus, the restrictions for the fragment combinations should be introduced. The restrictions can be written in form of logical functions (predicates) R(x,S), where x and S are a fragment and a sequence, involved in the combining operation, respectively.

The algorithm allows to define three different predicates R which relate to stage I and steps 1 and 2 of stage III. Thus, the number of generated TCFs in the set $\hat{T}$ can be really strongly diminished.

Note 3. The additional predicates R(C,S) can be defined for the generation procedure which produces molecular graphs from the central fragment C and all permissible sequences S of the TCFs from the set $\hat{T}$.

THE ACCOUNT OF SYMMETRY OF BRANCHED ELEMENTARY FRAGMENTS

For the symmetry analysis of the branched EFs, we shall consider the corresponding "enlarged" graphs $\hat{G}$ (see Fig.3). In a graph $\hat{G}$, corresponding to the fragment b of degree p, the set of vertices consists of the set V of vertices of the fragment's graph G, p-element

set of vertices V_1, corresponding to the set of sources E_G, and one-element set of vertices V_2, corresponding to the sink of the fragment b.

The set of edges of $\hat{G}$ consists of the set E of edges of the graph G, p-element set E_1 of edges, connecting vertices of V_1 with the corresponding sources of G and one-element set E_2 of edges, connecting a single vertex of V_2 with corresponding sink of G. Thus, all the vertices of the sets V_1 and V_2 have the degree 1. Note, that the vertices from V_1 and V_2 must be considered as differently labeled; the labels of corresponding vertices (squares and triangle, see Fig.3c) should not be identical to the labels of the vertices of the set V.

Evidently, the symmetry of the graph $\hat{G}$ differs from that of the fragment's graph G, and describes the symmetry of the fragment b at the same time. Strictly speaking, the symmetry of $\hat{G}$ is characterized by its automorphism group $\Gamma(\hat{G})=\langle\gamma_1\rangle$; this group consists of those permutations of the direct sum of symmetric groups $S_{|V|} \oplus S_{|V_1|} \oplus S_{|V_2|}$ which conserve all graph adjacencies and all labels.

The automorphism group of a fragment b will be defined as the restriction of the group Γ on the set V_1, this group will be denoted as Aut(b). In fact, Aut(b) is the action group, which can contain identical permutations, if they differ in Γ only by their action on V. The duplicates are thought to be removed, however, from the group Aut(b).

Let us consider the combining operation b&S at the step 1 of the stage III of the algorithm. To any $S=(t_1,\ldots,t_p)$, $t_j \in T_i$, $j=\overline{1,p}$, the definite mapping $\varphi=V_1 \Rightarrow T_i$ from the set V_1 into the set T_i actually corresponds. The permutation $\pi \in Aut(b)$ transforms φ into the equivalent

mapping $\varphi' = \pi(\varphi)$. Let Φ be a set of mappings, corresponding to all allowed sequences S. Then the group Aut(b) induces a new group (denoted as $E^{Aut(b)}$), acting on Φ (this group is a special case of a power group, introduced by Harary [14]).

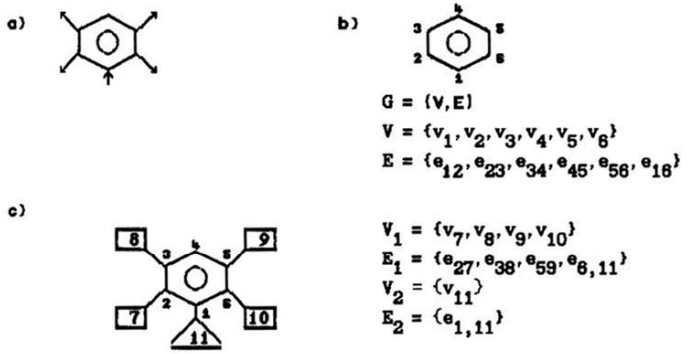


Fig. 3. The formation of "enlarged" graph $\hat{G}$ for a branched EF b:

a) a branched EF b;

b) the graph G of a fragment b;

c) the "enlarged" graph $\hat{G}$ of a fragment b

(the vertices of V_1 are denoted by ▫, the vertices of V_2 are denoted by △).

Thus, for the generation of all nonidentical fragments at step 1 it is necessary to find the system of orbit representatives (transversals) of the induced group $E^{Aut(b)}$, acting on Φ.

Let us consider four possible situations (see Fig. 4):

a) for the fragments similar to those represented in Fig. 4a the group Aut(b) is a symmetric group ($Aut(b)=S_2$ and $Aut(b)=S_3$, respectively). In this case for the generation of nonequivalent mappings φ it is sufficient to introduce the linear order on T_i and to select only those mappings φ for which $v<v' \Rightarrow \varphi(v)\leq\varphi(v')$; v and v' are thought to be arbitrary vertices of the set V_1.

b) for the fragments similar to those represented in Fig. 4b Aut(b) is the direct sum of symmetric groups ($Aut(b)=S_2+S_3$ in Fig. 4b). In this case the knowledge of orbits of Aut(b) is sufficient, each of the orbits can be treated as in the previous case.

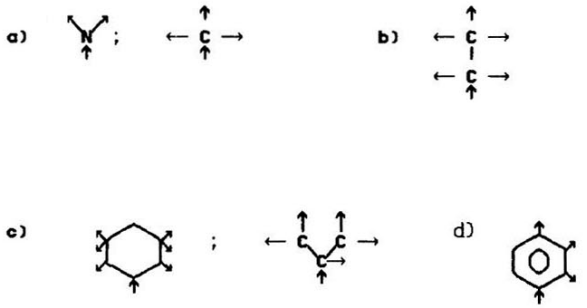


Fig. 4. Possible types of symmetry of a branched EF b:
a) symmetric groups (S_2 and S_3);
b) direct sum of symmetric groups ($S_2 + S_3$);
c) the general case: arbitrary groups;
d) identity group.

c) In the general case Aut(b) is an arbitrary group (see examples of Fig.4c). For such fragments the automorphism group Aut(b) must be explicitly known and for each mapping φ the equivalent mappings $\pi(\varphi)$ must be actually constructed. If $\pi(\varphi)$ is lexicographically smaller than φ (with respect to the order, introduced on T_i), then the mapping φ must be removed (because the equivalent mapping $\pi(\varphi)$ has been generated before).

d) In a trivial case the fragment b is unsymmetric (see Fig. 4d for an example) and the group Aut(b) is an identity group. In this special case all mappings are nonequivalent, and no checks are needed.

ACCOUNT OF THE SYMMETRY OF A CENTRAL FRAGMENT.

The methodology applied for the symmetry account of a central fragment is very similar to that for branched fragments.

For the central fragment C the "enlarged" graph $\hat{G}_C$ with the vertex set $V \cup V_1$ and the edge set $E \cup E_1$ must be constructed; the examples are shown in Fig.5. In this graph, it is possible to differentiate the vertices of the set V_1, and this makes it possible to associate the different sets of EFs to different sources of C. Thus, the number of orbits of automorphism groups $Aut(\hat{G}_C)$ depends not only on the symmetry of graph $\hat{G}_C$ of a fragment C, but also on the sets of EFs related to different sources of C.

For example in the case shown on Fig.5a the same set of EFs corresponds to each source of C and all four sources are labeled by the same label. The symmetry of $\hat{G}_C$ depends only on the symmetry of the fragment C, and the automorphism group of $\hat{G}_C$ is a symmetric group S_4

in this case.

In the case shown in Fig.5b the set Z of EFs is thought to consist of two disjoint sets Z_1 and Z_2 ($Z = Z_1 \cup Z_2$) which correspond to source vertices $\{v_1, v_3, v_5\}$ and $\{v_2, v_4, v_6\}$, respectively. For this reason, the group $Aut(G_C)$ is isomorphic to S_3 and consists only of 6 permutations.

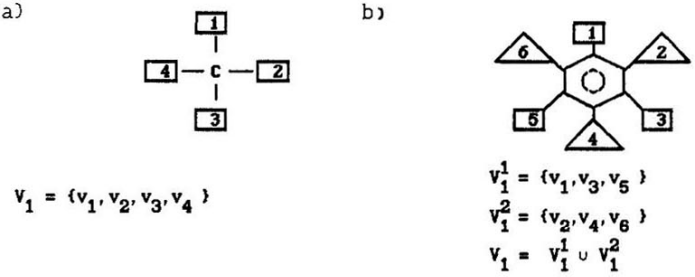


Fig 5. Two typical situations for a central fragment C:

a) one set of EFs is assosiated with all source vertices of $\hat{G}_C$;

b) two different sets of EFs are associated with the source vertices of $\hat{G}_C$

(the vertices of V_1, corresponding to Z_1, are denoted by □, the vertices of V_1, corresponding to Z_2, are denoted by △).

We stress here, that all four cases discussed in the preceding section can also take place in the case of $\hat{G}_C$ graphs. This means that only for arbitrary groups $Aut(\hat{G}_C)$ (case c in previous section) the permutations must be explicitly known.

COMPUTER REALIZATION OF THE ALGORITHM.

The described algorithm has been implemented on IBM PC. The computer program GOLD consists of the following blocks:

1. The block of a dialog choice of the central fragment C and of the sets of fragments T, L, B. The fragments are represented in the form of adjacency matrix only for graphical representation of them in the process of dialog and in the form of modified WLN code for the generation process. The matrices, which contain information about restricted automorphism groups of branched and central fragments, are formed for the symmetry account. The user also inputs the desired values of the maximal rank (MD), disperse (ND) and multiplicities n_j of linear EFs $l_j \in L$, $j=\overline{1,|L|}$.

2. The block of generation of molecular graphs. The result of its work is the text file, consisting of the WLN codes [15]. Two restricting predicates R_1 and R_2, are taken into account in the generation process.

The restricting predicate R_1 for stage I states that the linear EFs $l_j \in L$, $j=\overline{1,|L|}$, are included into the composite linear fragments with the multiplicity not exceeding n_j; the value of n_j is introduced by the user.

The restricting predicate R_2 for the step 1 of stage III determines the maximum number for each of branched EFs in the TCF; this number (also introduced by user) is typically small (1 or 2) to avoid the combinatorial explosion.

The generation block calculates the *a priori* estimation of the

expected number of molecular graphs which should be generated without account of symmetry.

3. The block decoding the set of generated structures into the adjacency matrices written in the text format. This block includes the originally developed program of decoding WLN codes.

REPRESENTATION OF THE FRAGMENTS.

For representation of the fragments in the computer memory and for realization of combining operations, we have chosen the WLN codes [15], which were slightly modified to represent the sinks and sources. Such a choice is thought to make the computer program easier and faster as compared with the use of adjacency matrices, because the combining of the fragments in form of WLN codes consists of the inserting and merging of text lines (except the alkyl numbers). Furthermore, the WLN-codes are rather compact (up to 130 symbols in most cases), and this allows to save a large number of generated structures on a hard disk.

It is necessary to mention that in the general case the account of the symmetry for the branched EFs and central fragments does not guarantee the generation of nonisomorphic graphs because the different combinations of multiatomic EFs can lead to just the same results.

The example of generation of identical linear fragments is shown below:

Elementary fragments:	WLN code:
->CO-CH_2->	V1
->CH_2-O->	10
->CO-CH_2-CH_3->	V2
->O->	0;
Composite fragments:	**WLN code:**
->CO-CH_2···CH_2-O->	V1 + 10 => V20
->CO-CH_2-CH_2···O->	V2 + 0 => V20.

For successful deleting such fragments and structures, the canonicity property of WLN codes is used: the equal WLN codes correspond to just the same structures, and such codes are really excluded after lexicographical sorting of WLN codes of composite fragments and generated structures.

In our algorithm, the WLN rules are used to introduce the order on the sets T_i to avoid the generation of the identical TCFs when the combining operations for branched fragments (at the step 1 of the stage III) are performed. In this combining operation only those mappings φ are chosen for which the sequences S satisfy the WLN canonicity rules for acyclic structures. The same rules are applied for the combining operations with the central fragment C being involved. The WLN codes obtained in such a procedure are not necessarily canonical ones, because they are formed in the generation process from WLN codes of EFs, while for generation of the canonical code the structure should be known as a whole. However, such "partial canonicity" permits to exclude identical structures generated from different combinations of EFs. Our program of WLN decoding checks only

syntax rules of WLN code and, hence, decodes all the generated structures correctly.

CONCLUSION

The suggested algorithm of the structure generation belongs to the class of "fragmentary" generators. The managing of the generation process allows not only to select the desired structures, but also to study the influence of various fragments and their combinations on the activity under investigation; the mathematical models for the series of compounds are used for that purpose.

The important moment of the generation process is the prohibition of the formation of cyclic combinations of fragments (rings). The formation of rings could lead to generation of structures considerably different from those used in a training set (for which the mathematical model to predict activity was constructed), the predictions for the structures based on that model could be incorrect in this case.

Any mathematical model of activity has the limitations on the classes of compounds for which the activity could be predicted. Such limitations are typically formulated as the limitations for the definite kinds of fragments and their combinations in the structures generated to predict the activity. These limitations can be obtained from the analysis of the training set of structures used for the construction of the mathematical model. Thus, the generation of the sets of structures is an important part in the investigation process consisting of the following steps: the treatment of the training set

of experimentally studied compounds - the elaboration of activity model - the generation of structures - the selection of potentially active structures.

The realization of this sequence in the form of the software allows the researcher to solve the inverse problems (for discussions see [6,16]) making possible the searching for active structures similar to those from the training set.

REFERENCES

1. A. Stuper, W. Bruger, P. Jurs, "Computer Assisted Studies of Chemical Structure and Biological Function", John Wiley & Sons, New York, 1979.
2. L.B. Kier and L.H. Hall, "Molecular Connectivity in Structure-Activity Analysis", Research Studies Press Ltd., Letchworth, 1986.
3. "Chemical Applications of Topology and Graph Theory" ed. by R.B. King, ELSEVIER, Amsterdam, Oxford, New York, Tokyo, 1983.
4. I.A. Faradzhev, in "Algorithmic Studies in Combinatorics", NAUKA, Moscow, 1978, pp. 11-19.
5. S.G. Molodtsov, V.N. Piottukh-Peletskii, *Vychislitelnye Sistemy*, v. 103, Novosibirsk, 1984, p. 51.
6. V. Kvasnička, J. Pospichal, *J.Chem.Inf.Comput.Sci.*, 30, 99 (1990).
7. I.P. Bangov, *MATCH*, 4, 235 (1983).
8. I.P. Bangov, K.D. Kanev, *J.Math.Chem.*, 2, 31 (1988).
9. A. Kerber , R. Laue , D. Loser, *Anal.Chim.Acta* 235, 221 (1990).
10. R. Carhart, D. Smith, H. Brown, C. Djerassi, *J. Am. Chem. Soc.*, 97, 5755 (1975).

11. B. Novak, L. Szotyory, *Textes Conf. Cadre Congr. Int. Contrib. Calc. Electron Dev. Genie Chim. Chim. Ind.*, V. A, 78 (1978).

12. M. Razinger, J. Zupan, M. Novic, *Mikrochim. Acta, II*, 411 (1986).

13. R. Nilakantan, N. Bauman, R. Venkataraghavan, *J. Chem. Inf. Comput. Sci.* 31, 527 (1991).

14. F. Harary and E. M. Palmer, *J. Comput. Theory*, 1, 157 (1966).

15. E. G. Smith, "The Wisswesser Line Formula Chemical Notation", McGraw Hill, New. York., 1968.

16. I. I. Baskin, E. V. Gordeeva, R. O. Devdariani, N. S. Zefirov, V. A. Palyulin, M. I. Stankevich, *Dokl. Akad. Nauk SSSR*, **307**, 613, (1989).