# Information

## Application of the electronegativity indices of organic molecules to tasks of chemical informatics*

*M. I. Trofimov*[★] *and E. A. Smolenskii*

*N. D. Zelinsky Institute of Organic Chemistry, Russian Academy of Sciences, 47 Leninskii prosp., 119991 Moscow, Russian Federation. Fax: +7 (495) 135 5328. E-mail: mtrofimov@online.ru, smolensk@cacr.ioc.ac.ru*

An efficient structure filtration method for the operation with chemical databases containing information on the structures and properties of organic molecules was proposed. The technique involves the use of electronegativity indices for generation of identification keys and for isomorphism tests of the molecular graphs corresponding to the structural formulas. The test set for the method proposed included a total of 95,000,000 molecules containing up to sixty carbon atoms. Tests revealed a high discriminating capability of the electronegativity indices and high efficiency of the method for solving both general problems (recognition of chemical structures, chemical database management systems) and specific tasks (generation of molecular graphs, *etc.*) in chemical informatics.

**Key words:** chemical informatics, databases, generation of graphs, isomorphism of graphs, computer chemistry, recognition of chemical structures, topological index, electronegativity.

Filtration of data containing information on the structures of organic molecules is necessary when solving a broad spectrum of tasks in chemical informatics. For instance, when processing a database filling query, the database management system should first of all retrieve from the database those molecular structures that are isomorphic to the structures in the query. Similar operations are also required in processing a query for search for molecular structures in the database. Generation of molecular structures using a generator program or a computer synthesis program is associated with the need for clearance of the data generated from duplicate structures, *etc*. It is well known that the structural formula of a compound can be graphically represented in different manner. Different chemical nomenclatures take into account specific features of human perception and provide a compromise between the "strictness" and "convenience". Such a compromise is admissible for human practice but not always appropriate for computer implementation. For instance,

* Dedicated to Academician N. S. Zefirov on the occasion of his 70th birthday.

if any database containing information on the physico-chemical properties is filled using an arbitrary atomic numbering scheme in the structural formula and corresponding arbitrary encoding of the structure, then it may appear that the same compound will be mapped on several disconnected records in the database, one record concerning, *e.g.*, the "melting point" property and the other record concerned with the "density" property. As a result, a user has the risk of missing records containing information of the properties of this compound only because atoms were enumerated in different manner compared to the procedure employed in the course of database filling. In the early stage of development of chemical informatics the user had the sole responsibility for identical representation of structural information in databases. To this end, a complex, intricate, and sometimes contradictory systems of rules for encoding of chemical structures in formulating database queries were designed. Often, such a system appeared to be incomplete, because it was impossible to allow for the variety of structures of organic compounds in the database design stage. At present, data mining is automatically performed by computers using various approaches. However, all of them are far from being perfect, and even the mathematical background of this field of research are still to be clarified. In this work we studied a possible unified approach to solving similar problems.

The flow chart for the approach proposed is shown in Fig. 1.

Structural information is represented in the form of molecular graphs (MGs) sent by the source of MGs to the receiver of MGs, which in turn constructs the MG master table, assigning each MG a unique key (master keys). If the receiver can not create a unique key for a MG that is non-isomorphic to all the MGs already had been placed in the MG master table, it assigns this MG a unique additional key and places this MG to the additional MG table. The receiver does not add an MG to the tables if this MG is isomorphic to another MG already placed in the tables.

A program implementing this scheme creates the MG tables. They can be stored dependences in main storage as well as in auxiliary storage (*e.g.*, hard disks) for further processing and use by another program or they can be processed simultaneously with their generation either in synchronous or asynchronous mode. If the MG tables were generated earlier, the database query source can act

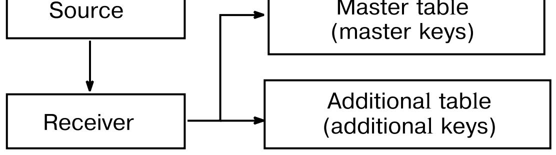
Source
Master table (master keys)
Receiver
Additional table (additional keys)

**Fig. 1.** Flow chart illustrating filtration of molecular graphs.

as a source. In this work we use simple MG generators as a source.

## Algorithms used

Keys are formed using the electronegativity indices (EIs) derived from the atomic electronegativities (AEs) in a molecule calculated using our method.[1,2] Unlike many pure topological indices, the AEs characterize not only the topological structure but also the composition of the chemical compound. At the same time, the AEs are easier to calculate compared to calculations of indices using quantum chemistry or molecular mechanics methods. The method for AE calculations for organic molecules is the development of the method for AE calculations of inorganic molecules.[3] Its usage is based on the correspondence between the AE and the key physico-chemical properties of a broad range of inorganic compounds including the most part of the elements of the periodic table. These features of the AE permit efficient solution of specific tasks, *e.g.*, search for structure—property relations for sets of organic compounds belonging to different chemical classes. In particular, dependences between the AEs and the energy[1] and geometric[4] parameters of molecules and the NMR spectroscopy data[2] were studied.

The atomic electronegativity is calculated using the relation[1]

$$S_i = \sqrt[v_i+1]{S_i^0 \prod_{t=1}^{v_i} S_t}\,, \qquad (1)$$

where $v_i$ is the number of atoms adjacent to the *i*th atom, $S_i^0$ is the standard electronegativity of a given type of atoms (chemical element); and $S_t$ is the electronegativity of the adjacent atom.

If no additional distinctions for the multiple bonds are introduced, the parameter $v_i$ can be treated as a vertex degree of an MG. In this case the multiple bonds are allowed for automatically, *e.g.*, the vertex degrees in the carbon skeletons of cyclohexane and benzene are equal to 4 and 3, respectively. It is important that from this point of view all carbon atoms in benzene and all bonds between them are equivalent. If an MG contains rings, special corrections should be made. In accordance with specific features of the problem posed we changed the algorithm proposed earlier[2] for the detection of smallest rings (Appendix 1, Algorithm 1).

The set of smallest rings is a subset of the set of all rings of an MG, produced as a result of the operation of algorithm 1. In this work we do not introduce any other definition of the smallest ring, because here the use of the smallest rings is reduced to routine calculations of the ring correction.

Taking the logarithm of relation (1) and having introduced the ring correction, we get

$$x_i = \frac{1}{v_i + 1}\sum_t x_t + b_i,$$

where $x_i = \ln S_i$, $b_i = \dfrac{\ln S_i^0}{v_i + 1} + c_i$, $c_i = \dfrac{1}{q}\sum_p \dfrac{r_{ip}}{p}$,

$q$ is the number of the vertices included in the rings of size $p$ (see Appendix 1).

Having written similar relations for all $m$ vertices of the MG, we get the system of $m$ linear equations

$$A \cdot \bar{X} = \bar{B}, \qquad (2)$$

where $\bar{B} = (b_i)$ is the column vector of absolute terms, $\bar{X} = (x_i)$ is the vector of the unknown quantities, $A = E - M \cdot N$, where $E$ is a unity matrix of size $m \times m$, and $M = (m_{ij})$ is the adjacency matrix of the MG,

$$N = (n_{ij}), \qquad \text{where } n_{ij} = \begin{cases} \dfrac{1}{v_i + 1}, & i = j \\ 0, & i \neq j. \end{cases}$$

Let us specify the accuracy of calculations ε. Then the set, $\{x_i\}$, of solutions of the system (2) can be divided into blocks in such a fashion that both $x_i$ and $x_j$ belong to the same $U_k$ block if $|x_i - x_j| < \varepsilon$. Let us sort the solution vector of the system (2) in non-descending order in accordance with the following rule for the partition $U$:

$$U_k < U_l, \text{ if } (|U_k| < |U_l|) \vee ((|U_k| = |U_l|) \wedge (x_k < x_l)), \qquad (3)$$

where $\{x_i\} = \bigcup\limits_{k=1}^{u} U_k$, $u$ is the number of blocks in the partition, $x_i \in U_k$.

Components of the solution vector $\bar{Y}$ thus sorted are the local (vertex) EIs. Let us denote this EI by EI1. Note that in accordance with the rule (3) for the $U$ and $U'$ partitions of the sets of solutions $\{x_i\}$ and $\{x_i'\}$ one has

$$U_k = U_k', \text{ if } (x_k = x_i') \wedge (|U_k| = |U_k'|),$$

where $x_k$, $x_k'$ are elements of the blocks $U_k$ and $U_k'$, respectively.

Let us define $U = U'$, if $\forall U_k = U_k'$, where $k = 1, ..., u$.

Let us define two more EIs, namely, EI2 and EI3 (Appendix 2). The electronegativity indices EI2 and EI3 are the integral superindices composed of the integral (Wiener index) and differential indices (EI1, vertex eccentricity) and the molecular parameters (empirical formula, number of hydrogen atoms, smallest rings). Thus, the EI2 and EI3 indices are ordered records of molecular indices and parameters in the form of strings. The former, EI2, is a universal index. However, *e.g.*, for the set of, structures with the general formula $C_xH_y$, where $x$ is a constant, the EI2 index is obviously redundant. For such structures, it is more appropriate to use a specific, more "economic" index EI3. The representation method of the EI2 and EI3 indices is not specified in their definitions and may depend on implementation. The representation used in this work has the form of ANSI (Pascal) strings of arbitrary length (*string* type); however, *e.g.*, null-terminated strings can be used instead. The encoding method facilitates program debugging, but more "economic" methods may also be possible. However, in any case the preset accuracy of calculations (ε) should be taken into account.

By expressing the check sum of the EI2 (or EI3) index through the 32-bit Circular Redundancy Check (CRC-32)[5] widely used, in particular, for verification of files, we get the EI4 index, which will be used as the master key. In addition to the master key, we will also use an additional key (see Appendix 2, 2.2). The CRC-32 is calculated using a highly efficient procedure, namely, a single loop whose length equals that of the string being processed. The loop body contains a single assignment statement, which assigns the value of an expression containing the choice of tabulated values and their combination using fast logical operations ("and", exclusive "or", shift). This gives a 32-bit integer, which varies between 0 and ($2^{32} - 1 = 4\,294\,967\,295$).

Let us define the following characteristics of the additional MG table. The maximum $g$ value (see Appendix 2, 2.2) among all additional keys of the database will be called the maximum EI4 degeneration. The number of additional keys used in the database will be called the general EI4 degeneration. Note also that the master and additional keys are different; therefore, a practical implementation can allow the master table and the additional table to be combined into a single table. The keys described are employed for MG filtration (see Fig. 1) using algorithm 2 (Appendix 3).

High optimizability of the algorithms proposed should be emphasized. This important property is exemplified in Appendix 3. Of course, not all problems concerning the optimization of these algorithms can be solved as simply as in the example listed in Appendix 3. For instance, some problems required the use of assembler instead of high-level programming languages. In particular, an efficient library was created, which manipulates sets of any cardinal number and makes it possible to avoid the Pascal limitations on the base set type (256-element size). In spite of our efforts on optimization, we believe that the potentialities of the approach proposed are far from being exhausted. In some cases, optimum solutions were deliberately not chosen in order to simplify tests and to follow principles of safe programming. Therefore, the experimental estimates of performance should be considered as preliminary, rough results which can be improved.

The discriminating capability of the EI1 index can be exemplified as follows. AE calculations for a hypothetical

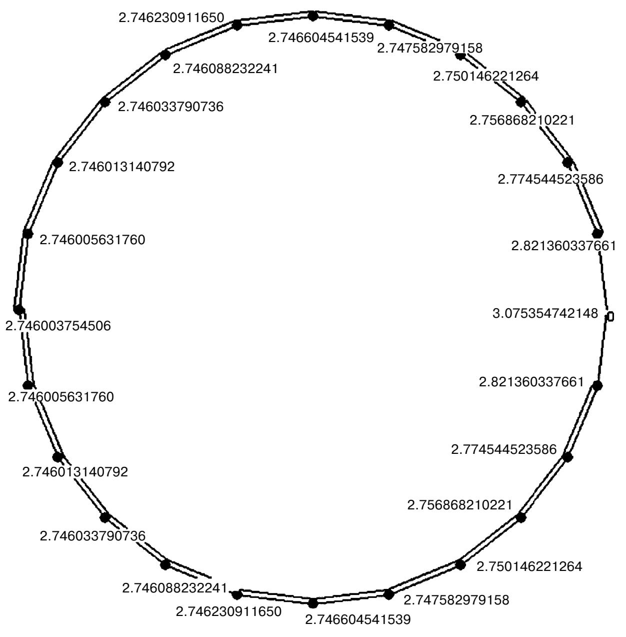

**Fig. 2.** Atomic electronegativities in hypothetical molecule $C_{23}O$.

molecule $C_{23}O$ at $S_i^0$ = 2.746 (C) and 3.654 (O) and without use of ring correction (Fig. 2) show that the AEs are redistributed from the most electronegative atom (O) to less electronegative atoms (C) and that the carbon atoms arranged at shorter distances from the oxygen atom are characterized by a larger increase in electronegativity compared to the more distant carbons. The AEs are distributed in such a fashion that the equivalent atoms have equal electronegativities. This example demonstrates a general pattern we observed in preliminary studies for different classes of organic compounds, namely, the presence of a heteroatom is responsible for small-size partition into groups of equivalent atoms, thus simplifying structure recognition, and the multiple bonds present no additional difficulties. Therefore, we will further restrict ourselves to consideration of saturated hydrocarbons.

AE calculations with the use of ring correction for the cuneane MG (Fig. 3) give a three-block partition in which two blocks have two and one block has four equivalent vertices. In contrast to the preceding example, in this case the key role for correct partition is played by the ring correction.

Noteworthy is a group of algorithms[6,7] based on the splitting of the eigenvalues of the modified adjacency matrices and solution of the systems of linear equations, which determine the inverse matrices. This group looks similar to algorithm 3 (see Appendix 3). These algorithms take into account the vertex degrees but explicitly ignore (in contrast to the approach proposed in this work) such important topological features of graphs, as rings. Vectors of the absolute terms of the systems of linear equations have the type (0, ..., 0, 1, 0, ..., 0). As a result, the algorithm development efforts are explicitly related to the number, $m$, of graph vertices and estimated at $O(m^4)$ and $O(m^5)$. Yet another algorithm was called[6] "heuristic"; a similar algorithm[7] is of limited use with respect to graph classes. For comparison, mention may also be made of the Corneil—Gotlieb classical algorithm[8] and the algorithms presented in review articles.[9,10]

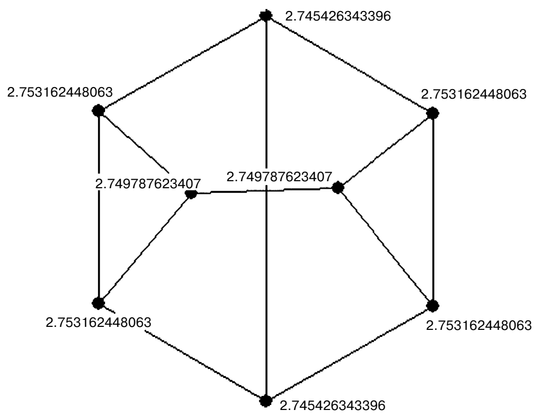

**Fig. 3.** Atomic electronegativities in cuneane ($C_8H_8$).

## Computational Experiment

Both third-party software and specially designed programs were used to test the approach proposed. The test programs for practical implementation of the MG filtration flow chart (see Fig. 1) were assembled from the specially designed MT2 enFilter package (Borland Delphi 7 environment using library functions for random number generation, CRC-32 computation, and a built-in assembler). An enumerative graph generator with duplications (GenGraph), a generator of random graphs with preset number of vertices (GenRandom), a random vertex enumerator, and the GenReg generator of regular graphs were used as the source.[11] Tests were carried out with a personal computer based on an Intel® Pentium® 4 CPU (Prescott core operated at a frequency of 3.2 GHz, 800 MHz front side bus) and the ASUS P4P800 SE motherboard with dual-channel memory access (1 GBytes RAM) and Intel® Hyper-Threading Technology support. The operating system was Microsoft® Windows® XP SP 1.

Tests were performed on connected graphs characterized by vertex degrees of at most 4, *i.e.*, the skeletal graphs (SGs) corresponding to the carbon skeletons of organic molecules. Pendant vertices of degree 1 corresponding to hydrogen atoms were added to the skeleton vertices of degrees less than 4 in order to obtain all the skeleton vertex degrees equal to 4. As a result, MG sets corresponding to saturated hydrocarbons of the general formula $C_xH_y$ were generated. The MGs characterized by $y = 0$ were excluded as chemically uninteresting. Such a filtration of the skeleton graphs, which meets these conditions, and eventual formation of MGs will be called the primary filtration (PF).

In solving the system (2), in order to calculate the EI1 indices for the vertices corresponding to carbon (hydrogen) atoms, the absolute terms were assigned the values $b_C$ ($b_H$). Based on the consistency considerations in order to make our method compatible with the AE calculation procedures proposed earlier, we used the Sanderson scale, according to which $S_i^0$ = 2.746 (C), 2.592 (H), and 4.000 (F). Thus, $b_C$ = ln($S_C^0$) = 1.0101... and $b_H$ = ln($S_H^0$) = 0.9524.... The initial modified weight $d_0$ (see Appendix 3) should be larger than any initial weight; therefore, we used a value that exceeds $S_F^0$, namely, $d_0$ = 5.0 and $\Delta d$ = 0.1. In principle, algorithm 3 can also be implemented with other values, although not all of them may appear to be appropriate. The necessary accuracy of calculations, ε, was determined experimentally, being equal to $10^{-12}$. We found non-isomorphic graphs with the EI3 indices coinciding at a lower accuracy (*i.e.*, at larger ε); a higher accuracy only causes unreasonable increase in the computational cost. Intermediate calculations were carried out in the Extended format for the Intel Pentium® 4 processors (ten-byte real numbers with 63-bit mantissa). The GenGraph program executes following a simple algorithm listed in Appendix 4.

As a result, GenGraph produces a collection of graphs with $m$ + 1 vertices. This collection includes a complete set of SGs and isomorphic SGs. Note that the cardinal number of this set is much smaller than that of the complete set of isomorphic SGs created following, *e.g.*, algorithm 5 (Appendix 5) but it is sufficient for evaluation purposes. The SGs generated using this algorithm are subject to PF and the "correct" MGs are sent to the receiver, which selects reiterated MGs isomorphic to the MGs arrived earlier using algorithm 2. The MG tables eventually obtained include a complete set of non-isomorphic MGs with $m$ + 1 vertices. The SGs of these MGs are used for the next generation following algorithm 4. As the start collection we used the tabulated collection $G_3$, which includes only two graphs (Fig. 4). No other connected graphs with $m$ = 3 exist.[11,12]

Correctness of the operation of the program can be checked using a trivial algorithm of complete enumeration of all adjacency matrices $M$ of size $m \times m$ (see Appendix 5). Graphs corresponding to the matrices obtained using this algorithm should be checked for connectivity, subject to PF, and sent to the receiver.

The MGs obtained as a result of the execution of the algorithms 4 and 5 coincide with one another to the accuracy to isomorphism. Check-up was performed up to $m$ = 8. Then, algorithm 5 appeared to be too slow for the personal computers used and we had to balance the situation with algorithm 4. Namely, we decided to generate only a number of rather than all possible isomorphic graphs. Table 1 lists the GenGraph evaluation results for $m$ = 4, ..., 12. The total number of the MGs filtrated by the receiver coincides with the values calculated by the MolGen Version 3.5 program.[13] In order to provide a self-check, algorithm 3 included the **check2** procedure. Graph isomorphism was independently controlled by a program based on the Nauty[14] Version 2.2 beta 6 module; all graphs with degenerated EI4 indices were checked; further tests were controlled identically. Algorithms 4 and 5 are modifications of the well-known graph generation algorithms, which have been well documented (*e.g.*, a non-heuristic Faradzhev method,[15] special issues of journals,[16,17] and studies[11,13,18]); they are presented here solely to understand the details of the tests performed.

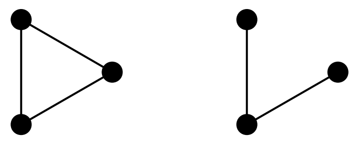
**Fig. 4.** Collection of graphs $G_3$.

## Results and Discussion

The results presented in Table 1 allow the EI3 degeneration to be estimated experimentally. A total of 6,250 non-isomorphic MGs with the same EI4 value were found, or ~0.08% of the total number of MGs selected (7,642,074). The EI4 index becomes degenerate when the number of the MG vertices becomes equal to 10, which is an exclusive result compared to the individual indices. For instance, degenerations of the highly popular Hosoya index manifest themselves already for the six-vertex MGs.[19] However, this comparison should be interpreted very carefully, because the EI4 index, in contrast to the Hosoya index, belongs to the composite super-

**Table 1.** Generation of complete sets of graphs with preset number of vertices

| $m$ | $N_g$ | $N_f$ | $P_{total}$ | $P_{max}$ | $K$ | $R_{max}$ | $I_{max}$ | τ |
|---|---|---|---|---|---|---|---|---|
| 4 | 14 | 6 | 0 | 0 | 8 | 2 | 2 | 0:00:00 |
| 5 | 90 | 20 | 0 | 0 | 77 | 3 | 3 | 0:00:00 |
| 6 | 422 | 77 | 0 | 0 | 421 | 4 | 4 | 0:00:00 |
| 7 | 2572 | 351 | 0 | 0 | 2639 | 5 | 5 | 0:00:00 |
| 8 | 15832 | 1923 | 0 | 0 | 16537 | 5 | 5 | 0:00:05 |
| 9 | 116407 | 12191 | 0 | 0 | 120703 | 5 | 5 | 0:00:51 |
| 10 | 934368 | 89343 | 15 | 1 | 965497 | 5 | 10 | 0:07:43 |
| 11 | 8439569 | 739070 | 144 | 1 | 8664659 | 6 | 10 | 1:16:30 |
| 12 | 83651542 | 6799093 | 6091 | 3 | 85507864 | 7 | 16 | 14:42:59 |
| Total | 93160816 | 7642074 | 6250 | | 95278405 | | | |

*Note*: $m$ is the number of vertices of SGs; $N_g$ is the number of MGs generated; $N_f$ is the number of filtrated MGs; $P_{total}$ is the total degeneration of EI4; $P_{max}$ is the maximum degeneration of EI4; $K$ is the number of external calls of the IsoTest procedure (Algorithm 3); $R_{max}$ is the maximum recursion depth for the IsoTest procedure; $I_{max}$ is the maximum iteration number for the IsoTest procedure; and τ/h:m:s is the program execution time. Designation 0:00:00 means that the program execution took less than 1 s.

indices which are characterized[19] by a much higher discriminating capability than the individual (non-composite) indices. While testing, we found that the EI4 index can be improved by, *e.g.*, using the "Y modification" of the Hosoya index proposed in our earlier study[20] instead of the Wiener index. Unfortunately, this improvement of the discriminating capability of the EI4 index could cause a catastrophic decrease in performance, that is, an exponential complexity for calculating the modified Hosoya index instead of polynomial complexity for calculating the Wiener index. At the same time, exclusion of the Wiener index and vertex eccentricities from the EI4 index markedly deteriorated its discriminating capability. Both indices (Wiener index and vertex eccentricities) are calculated in parallel based on the distance matrix obtained using the Floyd algorithm[21] after $O(m^3)$ steps. Therefore, the compromise solution implemented in the EI4 index seems to be worthwhile. Probably, the use of other indices, *e.g.*, the 'extended" Wiener index,[22] will show better results.

**Table 2.** Comparison of pair sets in 10,000 of random graphs with preset numbers of vertices

| $n$ | $m$ | $R_{max}$ | $I_{max}$ |
|---|---|---|---|
| 1 | 13 | 4 | 4 |
| 2 | 14 | 4 | 4 |
| 3 | 15 | 4 | 4 |
| 4 | 16 | 4 | 4 |
| 5 | 17 | 4 | 4 |
| 6 | 18 | 5 | 5 |
| 7 | 19 | 5 | 5 |
| 8 | 20 | 6 | 6 |
| 9 | 21 | 5 | 5 |
| 10 | 22 | 5 | 5 |
| 11 | 23 | 5 | 5 |
| 12 | 24 | 5 | 5 |
| 13 | 25 | 5 | 5 |
| 14 | 26 | 5 | 5 |
| 15 | 27 | 5 | 5 |
| 16 | 28 | 5 | 5 |
| 17 | 29 | 5 | 5 |
| 18 | 30 | 5 | 5 |
| 19 | 31 | 5 | 5 |
| 20 | 32 | 8 | 8 |
| 21 | 33 | 6 | 6 |
| 22 | 34 | 5 | 5 |
| 23 | 35 | 6 | 6 |
| 24 | 36 | 6 | 6 |
| 25 | 37 | 6 | 6 |
| 26 | 38 | 6 | 6 |
| 27 | 39 | 7 | 7 |
| 28 | 40 | 6 | 6 |

*Note*: $n$ is the pair number; for other notations, see Note to Table 1.

The degenerated indices are distributed quite uniformly, which allowed no more than three additional keys to be used for each degeneration in all the graphs tested (see Table 1, maximum degeneration of EI4). In the course of our tests, in the worst situations at most four external calls of the IsoTest procedure (see algorithm 3) were done for each new MG.

The program elapsed time is presented in the "hh:mm:ss" format (see Table 1). The results of further tests listed in Tables 2—4 provide no elapsed time measurement data, because specific input—output procedures and other service operations took a rather long time. However, all subsequent tests took much shorter time compared to the time taken by the GenGraph generator test.

**Table 3.** Comparison of complete pair sets of regular graphs with preset number of vertices and a vertex degree of 3

| $m$ | $N$ | $R_{max}$ | $I_{max}$ |
|---|---|---|---|
| 14 | 509 | 7 | 7 |
| 16 | 4060 | 8 | 8 |
| 18 | 41301 | 8 | 8 |
| 20 | 510489 | 9 | 17 |
| Total number of pairs | 556359 | | |

*Note*: $N$ is the number of MGs; for other notations, see Note to Table 1.

**Table 4.** Comparison of pair sets in 5,000 regular graphs with a preset number of vertices and a vertex degree of 3*

| $n$ | $m$ | $R_{max}$ | $I_{max}$ |
|---|---|---|---|
| 1 | 22 | 8 | 8 |
| 2 | 24 | 9 | 9 |
| 3 | 26 | 9 | 9 |
| 4 | 28 | 10 | 10 |
| 5 | 30 | 10 | 10 |
| 6 | 32 | 11 | 11 |
| 7 | 34 | 11 | 11 |
| 8 | 36 | 12 | 12 |
| 9 | 38 | 12 | 12 |
| 10 | 40 | 13 | 13 |
| 11 | 42 | 13 | 13 |
| 12 | 44 | 14 | 14 |
| 13 | 46 | 14 | 14 |
| 14 | 48 | 15 | 15 |
| 15 | 50 | 15 | 15 |
| 16 | 52 | 16 | 16 |
| 17 | 54 | 16 | 16 |
| 18 | 56 | 17 | 17 |
| 19 | 58 | 18 | 18 |
| 20 | 60 | 18 | 18 |

* For notations, see Notes to Tables 1 and 2.

The GenRandom generator uses common principles of random graph generation;[22] therefore, we will only briefly outline them. The Renumber procedure results in random renumbering of the vertices of a graph while the MakePair procedure constructs a non-identical (but isomorphic) copy of the graph using the Renumber. The Sort procedure is a usual procedure for sorting of records with integer keys by constructing a non-balanced binary tree; having $m$ random numbers at the input, we get a random tree with $m$ vertices. Then, randomly chosen pairs of vertices (with vertex degrees less than 4) in the tree were connected by edges; this operation was repeated with a random number of repetitions. The random graph thus obtained was sent to the makePair procedure and then this graph and its isomorphic copy were subject to PF and sent to the receiver. Our tests involved generation of sets including each of 10,000 isomorphic pairs of random graphs with 13 to 40 SG vertices. Table 2 presents the results obtained. It should be noted that, theoretically, pairs of the graphs generated by the GenRandom procedure can be doubled; however, the probability of such coincidences is negligible.

The operation of the random vertex enumerator is also based on the use of the MakePair procedure. Namely, the enumerator reads a graph from a data file generated by the GenReg generator of regular graphs and creates an isomorphic copy of this graph. Next, both graphs are subject to PF and then sent to the receiver. As above, we are interesting in the MGs corresponding to saturated hydrocarbons of the general formula $C_xH_y$, with $y > 0$. This holds for regular SGs with vertex degrees of 2 and 3, the former case corresponding to a trivial situation where for each $x$ there is a single regular SG including a ring of size $x$. This case can also be ignored in tests, so SGs with a vertex degree of 3 are to be tested. Generation of complete sets of SGs with a preset number of vertices faces no problems with allowance for the PF and software used up to 20-vertex regular SGs (the number of the SGs exceeds 500,000; see Table 3). Further tests were carried out with incomplete sets of regular SGs including a total of 5,000 pairs of isomorphic graphs in each set up to 60-vertex SGs (see Table 4). The last-named case corresponds to molecules of the formula $C_{60}H_{60}$, containing a total of 120 atoms. Regular graphs are thought to be the most difficult for many algorithms including the graph isomorphism algorithms. However, our tests revealed no fundamental difficulties for this class of graphs. In this case the recursion depth and the number of iterations also depend only slightly on the number of vertices, being characterized by the same small numerical values. Moreover, for some irregular SGs these values can be higher than for the regular graphs with a large number of vertices.

Noteworthy is a trend observed experimentally using graph generators, *viz.*, the proportion of regular SGs decreases as the number of vertices ($x$) increases (Table 5).

**Table 5.** Proportion of regular graphs

| $x$ | $G$ | $R_3$ | $Y_R$ (%) |
|---|---|---|---|
| 4 | 6 | 1 | 33.3333 |
| 5 | 20 | 0 | 5.0000 |
| 6 | 77 | 2 | 3.8961 |
| 7 | 351 | 0 | 0.2849 |
| 8 | 1923 | 5 | 0.3120 |
| 9 | 12191 | 0 | 0.0082 |
| 10 | 89343 | 19 | 0.0224 |
| 11 | 739070 | 0 | 0.0001 |
| 12 | 6799093 | 85 | 0.0013 |

*Note*: $x$ is the number of vertices in the graph; $G$ is the number of graphs; $R_2$ is the number of regular graphs with a vertex degree of 2, $R_2 = 1$; $R_3$ is the number of regular graphs with a vertex degree of 3; and $Y_R$ is the proportion of regular graphs.

For instance, at $x = 4$ there are two regular SGs (one with vertices of degree 2 and one with vertices of degree 3), or 33.3% of the total number of graphs. This proportion decreases to 0.02% at $x = 10$ and to 0.001% at $x = 12$. One can assume that the trend will also be observed on further increase in $x$. Thus, one can expect that the proportion of complex (in the framework of the approach proposed) structures will decrease with increasing $x$. In particular, if database querying is simulated using a random graph generator, we get the situation illustrated by Table 2. Here, the maximum recursion depth and number of iterations are smaller than for the values listed in Tables 3 and 4, because the most difficult cases were not included in the random sets and the probability for difficult cases to be included in the random sets seems to decrease with an increase in $x$. However, this model is sometimes unrealistic, because certain regular SGs, *e.g.*, cyclobutane, cyclopentane, and cyclohexane, cubane, dodecahedrane, *etc.* play significant roles in chemistry. One can assume that a server managing a database for a rather long time period can be queried with a query distribution pattern similar to the random model. But these periods will be alternated by shorter but intense periods of high demand for information concerning a small group of structures; the complexity of the processing of such queries by the server will exceed some averaged characteristics. This can be due to, *e.g.*, sensations concerned with the synthesis of a compound from this group. For instance, a pioneering synthesis similar in significance to the synthesis of dodecahedrane will undoubtedly attract the major attention of chemists, which will unavoidably affect the performance of our speculative server. Therefore, in the course of tests particular attention was given to the regular MGs in spite of a small proportion of such graphs among all possible MGs. The total number of generated graphs $G$ was 93,160,816 (Table 1) + 560,000 (Table 2) + 1,112,718 (Table 3) + 200,000 (Table 4) = 95,033,534.

Some auxiliary operations including the examples presented in this study were carried out using the GraphEdit graph editor program,* with plug-in modules for calculations of indices and the MG format conversions. This is a server program that interacts with client programs using the ActiveX technology. The database management system of the MT2 GraphsDB 2004, Version 1.2.3 chemical database corresponding to the flow chart shown in Fig. 1 was implemented as a client program. The GraphsDB system is an implementation of the relational database model running the widely used Corel Paradox database format and manipulating SQL queries.

Particular attention should be given to expandability of the approach proposed. For instance, a number of valuable additional features appear when this strategy is combined with the method[24] for encoding of structural formulas of organic compounds. The concept of the method, formulated[25] as long ago as 1962, belongs to one of the first successful applications of graph theory in organic chemistry. Since then, progress in modern science (mathematics, physics, chemistry) and technology (first of all, computer technology) made the method much more significant and valuable. The method allows one to minimize the MG representation to a theoretically possible level (*i.e.*, to reduce the redundancy to zero). This permits a more efficient use of computer memory. Besides, certain important classes of MGs (*e.g.*, trees) can be identified faster than using the algorithms listed above; this improves the efficiency of the method from the viewpoint of average performance.

The authors express their gratitude to Academician N. S. Zefirov (N. D. Zelinsky Institute of Organic Chemistry, Russian Academy of Sciences), M. Meringer (University of Bayreuth, Germany), and M. S. Molchanova (N. D. Zelinsky Institute of Organic Chemistry, Russian Academy of Sciences) for helpful discussion of some MG generation problems, S. S. Trach (M. V. Lomonosov Moscow State University) for discussion of the manuscript, L. G. Vetrov (Moscow State Technical University) for help in mathematical analysis of the properties of the EI1 index, and to M. P. Trofimova for help in performing the computational experiment and preparation of the manuscript.

## Appendix 1

***Algorithm 1***

```
1. Assign zero values to all elements of matrix $R = (r_{ij})$ and vector $\bar{C} = (c_i)$.
2. For all edges $(i, j)$ of graph $G$, do steps 3—6.
3. $G' := G$.
4. Remove edge $(i, j)$ in $G'$.
5. Calculate the length, $l$, of the shortest path from $i$ to $j$.
6. $r_{ip} := r_{ip} + 1$; $r_{jp} = r_{jp} + 1$, where $p = l + 1$.
7. $q := 0$.
8. For $i$, $i \in [1, m]$ do
9. $h := 0$.
10. For $p$, $p \in [3, m]$ do
11. If $r_{ip} \neq 0$ then $h := h + r_{ip}/p$.
12. End of loop 10.
13. If $h \neq 0$ then $c_i := c_i + h$, $q := q + 1$.
14. End of loop 8.
15. For $i$, $i \in [1, m]$ do
16. $c_i := c_i/q$.
17. End of loop 15. ■*
```

***Note.*** The dimension of the matrix $R = (r_{ip})$ is $m \times m$ and that of the vector $\bar{C}$ is $m$, where $m$ is the number of vertices of graph $G$. Execution of steps 2—6 causes the matrix $R$ to include number of the smallest rings $r_{ip}$ of size $p$ for the vertex $i$. In step 10, only the rings of size from 3 to $m$ are considered. Therefore, if, *e.g.*, the vertex $j$ is a pendant vertex (*i.e.*, after removing the edge $(i, j)$ in step 4 it is impossible to find the shortest path in step 5) then $l = 0$, and in step 11 $h = 0$. Step 11 is executed $m$ times in the loop 8—14 and each vertex included in one or more rings in graph $G$ increases the $q$ value by 1, pendant vertices do not change the $q$ value. Edges in step 2 can be enumerated in any order; therefore, the result is independent of enumeration of vertices. Then each vertex is assigned the index $c_i$

$$c_i = \frac{1}{q}\sum_p \frac{r_{ip}}{p},$$

where $q$ is the number of vertices included in the rings.

Search for the shortest path can be performed using well-known algorithms, *e.g.*,[21] with the complexity estimated at $O(m + e)$, where $e$ is the number of edges. Therefore, the complexity of steps 2—6 can be estimated at $O(e^2 + em)$. The complexity of steps 8—14 is estimated by two nested loops (8, 10) at $O[m(m - 3)]$ and the complexity of steps 15—17 is estimated by one loop at $O(m)$. Neglecting the lowest terms, the total complexity of the algorithm can be estimated at $O(e^2 + m^2)$.

## Appendix 2

### 2.1. Definition of the EI2 and EI3 electronegativity indices

We will use the Backus—Naur notations that are widely used for the description of formal languages.[26] Let us define the EI2 and EI3 indices as strings:

```
<EI2>::=<EI2 header><list of vertex indices>
<EI3>::=<EI3 header><list of vertex indices>
<EI2 header>::=<Wiener index><empirical formula>;
<EI3 header>::=<Wiener index>H<number of hydrogen
   atoms>;
<list of vertex indices>::=<vertex index
   list>|<vertex index list><list of vertex
   indices>
<vertex index list>::=<EI1>,<vertex
   eccentricity>,<list of smallest rings>;
<list of smallest rings>::=<smallest ring>
   |<smallest ring><list of smallest rings>
<smallest ring>::=<ring size>*<number of rings>
```

* M. I. Trofimov, *Program MT2 GraphEdit 2004, Version 1.8.1*, E-mail: mtrofimov@online.ru.

* From this point on the end of the algorithm is denoted by "■".

Here symbols ",", ";", "*", and "H" are terminal symbols; *list of vertex indices* and *list of smallest rings* are ordered lists; *i.e.*, for two vertices $i$ and $i + 1$ the *list of vertex indices* is ..., $s_i$, $s_{i+1}$ if substring $s_i$ (*vertex i index list*) is smaller or equal to substring $s_{i+1}$ (*vertex* $i + 1$ *index list*); analogously for two rings of size $p$ and $p + 1$ the *list of smallest rings* is ..., $s_p$, $s_{p+1}$, because $s_p < s_{p+1}$, meanwhile the *list of smallest rings* being added to the *vertex index list* only if the molecule contains rings. For each vertex only those smallest rings are enumerated in which the vertex is included.

### 2.2. Additional key

The additional key is formed as a string:

`<additional key>::=<EI4>N<g>`

Here "N" is the terminal symbol, $g$ is the ordinal number of the additional key for a given EI4 value (*i.e.*, for a given EI4 value the first additional key corresponds to $g = 1$, the second additional key corresponds to $g = 2$, *etc.*).

## Appendix 3

***Algorithm 2***

For each molecular graph $G$ sent from the source to the receiver (see Fig. 1), the receiver calculates the master key EI4 and looks for the key in the master table. If the search for $G$ gave a negative result, then $G$ is placed in the master table and assigned the calculated key. Otherwise, an isomorphism test is performed for $G$ and $G'$ found in the master table. If the graphs are not isomorphic, search is carried out in the additional table for all keys $k$ that include the EI4 index found and $g = 1, 2, ..., k$. Each MG found as a result of this search is also tested for isomorphism with $G$. If isomorphism is found, the search is terminated; otherwise, $G$ is placed in the additional table and assigned the key with $g = k + 1$. ■

Isomorphism tests are performed for weighted graphs. To this end, each vertex of graphs $G$ and $G'$ is assigned a weight initially equal to the EI1 index of this vertex. The isomorphism test uses algorithm 3, where the variables $U$, $u$, and $m$ have the same sense as earlier and the initial modified weight $d_0$ is a constant (for the choice of $d_0$, see the Computational Experiment Section).

***Algorithm 3***

*Procedure IsoTest*

```
1. Compare EI2 (or EI3) for $G$ and $G'$, if they are
   different then the graphs are not isomorphic, exit
2. Iteration counter $t$:=0, recursion depth
   counter $r$:=0, modified weight $d$:=$d_0$.
3. Call **check**.
```

*Recursive Procedure* ***check***

```
Input parameters: $G$, $G'$, $d$, $U$, $U'$.
1. If isomorphism was found earlier then exit
2. $r$:=$r$+1.
3. If $U \neq U'$ then exit else
4. If $u = m$ then call **check2**($G$, $G'$, $U$, $U'$).
   If isomorphism is found then exit
   else
5. Take a vertex of graph $G$ from the smallest block $U_j$
   such that $|U_j| > 1$, replace the weight of the vertex
   by $d$.
   Calculate EI1 for $G$, sort solutions and
   obtain a new partition $U$.
6. Vertex enumeration loop for graph $G'$:
   $t$:=$t$+1. Take the next vertex from block $U_j'$ and
   replace its weight by $d$. Compute EI1 for $G'$,
   sort solutions and obtain a new
   partition $U'$. If the solutions of the systems coincide
   then recursive call **check**($G$, $G'$, $d + \Delta d$, $U$, $U'$).
   If there are no more vertices in block $U_j'$ then
   exit.
7. exit **check**.
```

*Procedure* ***check2***

```
Input parameters: $G$, $G'$, $U$, $U'$.
   Check for correspondence between the vertices of the
   graphs $G$ and $G'$ having the same weights:
   if connectivity is retained on mapping of $G$ onto $G'$
   then isomorphism is found. ■
```

To explain this approach, we will now consider the properties of the EI1 index. Two graphs, $G$ and $G'$, with the adjacency matrices $M$ and $M'$, respectively, are isomorphic if and only if there exists the permutation matrix $P = (p_{ij})$,

$$\text{where } p_{ij} = \begin{cases} 1, & i \to j \\ 0, & \end{cases}$$

such that $M' = P^{-1} \cdot M \cdot P$ (see Ref. 12). The properties of the matrix $P$ are as follows:

1) $P$ is an orthogonal matrix, *i.e.*, $P^{-1} = P^T$;

2) if graph $G$ is isomorphic to graph $G'$ and the vertex ($i$) of $G$ maps into the vertex ($j$) of $G'$, then

$$\bar{e}_i = P \cdot \bar{e}_j \text{ and } E_{ii} = P \cdot E_{jj} \cdot P^{-1},$$

where all coordinates of the vector $\bar{e}_i$ are equal to 0 except for the $i$th coordinate equal to 1 and the element on the crossing of the $i$th row and $i$th column of the matrix $E_{ii}$ is equal to 1, while all other elements of $E_{ii}$ are equal to zero.

The degree of a vertex is calculated using the adjacency matrix $M = (m_{ij})$ as follows

$$v_i = \sum_j m_{ij}.$$

Let us denote vectors with the coordinates $v_i$ and $v_i'$ by $\bar{v}$ and $\bar{v}'$, respectively. In the matrix form one can write $\bar{v} = M \cdot \bar{e}$ and $\bar{v}' = M' \cdot \bar{e}$, where all coordinates of the vector $\bar{e}$ are equal to 1. From this it follows that $\bar{v}' = P^{-1} \cdot \bar{v}$. Let us define the matrices $D$ and $D'$as the diagonal matrices with the diagonal elements equal to $v_i$ and $v_i'$, respectively. Then one gets

$$N = (E + D)^{-1} \text{ и } N' = (E + D')^{-1},$$

where $E$ is the unity matrix of corresponding size.

In turn, the $D$ and $D'$ matrices can be respectively expressed through the $\bar{v}$ and $\bar{v}'$ vectors:

$$D = \sum_i E_{ii} \cdot \bar{v} \cdot \bar{e}_i^T, \qquad D' = \sum_j E_{jj} \cdot \bar{v}' \cdot \bar{e}_j^T.$$

Consider two systems of linear equations (2):

$$A \cdot \bar{X} = \bar{B}, \qquad A' \cdot \bar{X}' = \bar{B}'.$$

*Assertion.* Let a graph $G$ be isomorphic to graph $G'$ and $P$ be the permutation matrix. Then, if $\bar{B}' = P^{-1} \cdot \bar{B}$ then $\bar{X}' = P^{-1} \cdot \bar{X}$.

*Proof.* Let us find a relation between the matrices $A$ and $A'$:

$$P^{-1} \cdot D \cdot P = P^{-1} \cdot (\sum_i E_{ii} \cdot \bar{v} \cdot \bar{e}_i^T) \cdot P = \sum_i P^{-1} \cdot E_{ii} \cdot \bar{v} \cdot \bar{e}_i^T \cdot P =$$

$$= \sum_i P^{-1} \cdot E_{ii} \cdot P \cdot P^{-1} \cdot \bar{v} \cdot (P^{-1} \cdot \bar{e}_i)^T = \sum_j E_{jj} \cdot \bar{v}' \cdot \bar{e}_j^T = D'.$$

From here we get

$$N = (E + D)^{-1} = (P^{-1} \cdot E \cdot P + P^{-1} \cdot D' \cdot P)^{-1} =$$

$$= [P^{-1} \cdot (E + D') \cdot P]^{-1} = P^{-1} \cdot (E + D')^{-1} \cdot P = P^{-1} \cdot N' \cdot P.$$

Similarly:

$$A = E - M \cdot N = P^{-1} \cdot E \cdot P - P^{-1} \cdot M' \cdot P \cdot P^{-1} \cdot N' \cdot P =$$

$$= P^{-1} \cdot (E - M' \cdot N') \cdot P = P^{-1} \cdot A' \cdot P,$$

$$A^{-1} = P^{-1} \cdot A'^{-1} \cdot P.$$

Finally, we get

$$\bar{X}' = A'^{-1} \cdot \bar{B}' = P^{-1} \cdot A^{-1} \cdot P \cdot P^{-1} \cdot \bar{B} = P^{-1} \cdot A^{-1} \cdot \bar{B} = P^{-1} \cdot \bar{X},$$

which was to be proved.

Note that if all coordinates of the $\bar{B}$ vector are equal to 1 (*i.e.*, $\bar{B} = \bar{e}$), the solution of the system (coordinates of the $\bar{X}$ vector) gives the vertex degrees of the graph after subtraction of unity from all these coordinates, *i.e.*, $\bar{X} - \bar{e} = \bar{v}$.

The assertion proved suggests that

1. If graphs $G$ and $G'$ are isomorphic, for any $i$ there exists $j$ such that the solutions of the systems $A \cdot \bar{X} = \bar{e}_i$ and $A' \cdot \bar{X}' = \bar{e}_j$ coincide to an accuracy of permutation of the coordinates of the vectors $\bar{X}$ and $\bar{X}'$.

2. If, for isomorphic graphs, at certain ($i$) and ($j$) the solutions $\bar{X}$ and $\bar{X}'$ of the systems $A \cdot \bar{X} = \bar{e}_i$ and $A' \cdot \bar{X}' = \bar{e}_j$ are obtained by mutual permutation and have all different coordinates ($X(k) \neq X(l)$, $\forall k \neq l$), the corresponding permutation of the coordinates of the vectors $\bar{X}$ and $\bar{X}'$ gives isomorphism of the graphs $G$ and $G'$.

3. If, for certain $i$ at any $j$, the solutions of the systems $A \cdot \bar{X} = \bar{e}_i$ and $A' \cdot \bar{X}' = \bar{e}_j$ are different to an accuracy of permutation of the coordinates of the vectors $\bar{X}$ and $\bar{X}'$, these graphs are not isomorphic.

The recursion depth and the number of iterations in algorithm 3 depend on the number and size of partition blocks for the sets of the solutions of the systems of linear equations for the vertex weights obtained by consecutive replacement of the initial weights by the modified ones ($d_0$, $d_0 + \Delta d$, $d_0 + 2\Delta d$, ...), being only slightly dependent on the number of vertices $m$, just like the number of identical solutions of the system of linear equations of the type (2) is not simply related to the number, $m$, of the unknown quantities. The number and sizes of the partition blocks depend on the initial weights and degrees of vertices and on the discriminating capability of the index employed for the weights. They also depend, in a complex fashion, on the symmetry of the system of equations. In turn, this symmetry is directly related to the symmetry of the adjacency matrix of the corresponding graph. Each replacement of the initial weight of a vertex by the modified weight leads to a strong lowering of the symmetry of the system of equations, thus dividing the partition into progressively decreasing blocks and minimizing the enumeration.

Each iteration requires solution of two systems of linear equations of the type (2) in order to calculate the modified weights of the vertices of the tested graphs $G$ and $G'$. Here, only the absolute term vectors $B$ change from one iteration to another, whereas the coefficient matrices $A$ remain unchanged. In this case, a standard approach is as follows. In the first solution of the systems (2) one should calculate the inverse matrices $A^{-1}$ and $A'^{-1}$ and then search for solutions in the form:

$$\bar{X} = A^{-1} \cdot \bar{B} \text{ and } \bar{X}' = A'^{-1} \cdot \bar{B}.$$

The inverse matrix can be found simultaneously with the solution of the system of linear equations by the Gauss—Jordan method.[27] The complexity of such computations can be estimated at $O(m^3)$ and subsequent calculations of $\bar{X}$ and $\bar{X}'$ at $O(m^2)$ for each system. Solution of the system of linear equations according by the Gauss—Jordan method is more time-consuming than the classical Gauss procedure. Therefore, it is appropriate to use the approach described above only if it is necessary to solve a rather large number of systems of linear equations with identical matrices $A$. In the case of the experimental set described above (see Computational Experiment Section) minimization of enumeration is so efficient that this number becomes insufficient and the classical Gauss method gives better results.

The adjacency matrices (but not the inverse of the adjacency matrices) are sparse matrices; therefore, one can expect that the use of the sparse matrix approach can accelerate the solution of the system (2). It should also be noted that the current stage of development of microprocessors allows the real performance (execution of identical, in particular, matrix operations) to be substantially improved on the microprogramming level owing to all kinds of tricks similar to instruction pipelining, *etc.* Other tricks are possible on the low-level (assembler) and high-level programming languages. For instance, *e.g.*, many operations in the algorithms presented in this work consist in enumeration of a set in order to extract a next element. The classical Pascal, contrary to other programming languages, offers a unique feature of explicit representation of the set types, which makes it possible to solve this problem as follows:

```
for i:=1 to n do
    if i in s then
        begin s :=s-[i];...; end;
```

A rough estimate of the complexity of this code fragment is given by $O(n)$, which ignores the complexity of the loop body.

However, for small sets *s* containing at most 15 elements and modern extended Pascal language (*e.g.*, Object Pascal Delphi) it is possible to find a tabulated solution:

```
repeat
        i := table1[word(s)];
        s := table2[word(s)];
        ...;
until s=[];
```

The tables, table1 and table2, contain all possible $2^{16}$ = 65 536 variants, being initialized once with the program startup. These tables are arranged in a 128 KByte memory block, which is more than modest taking into account the operating characteristics of modern personal computers. For highly filled sets the gain from this optimization can seem to be insignificant, but in the case of a MG where the set *s* corresponds to, *e.g.*, adjacent vertices, the gain can be very high. Such operations allowed the performance of the GenGraph generator (see below) designed for complete enumeration of MGs including less than fifteen vertices to be improved by more than a dozen times. In spite of the fact that this result is surely expected, its explanation in terms of formal estimates of computational complexity faces problems. Criticism has been repeatedly expressed in the graph theory studies, concerning practical limitations of the method of estimates of the hard-to-solve problems and the notion of NP-completeness (see., *e.g.*, Ref. 28). It was also pointed[29] that the hard solvability of NP-complete problems was not rigorously established despite a commonly accepted, among mathematicians, opinion that it is impossible to design efficient algorithms for solving these problems. Detailed consideration of these issues goes far beyond the scope of this study; therefore, we will only mention that real performance has as much in common with the theoretical complexity as ideal gas with real gas or as famous MIX ideal computer by Knuth[30] with Pentium® 4. Of course, more and more appropriate models are elaborated. For instance, one cannot but agree with the statement that an equally accessible address machine (it can serve as a basis for the introduction of the notion of NP-completeness) is a good approximation to real computers.[31] We only want to emphasize the words "model" and "approximation" (intrinsic in any model) in order to characterize the complexity and ambiguity of this problem, which is unseen from the outside. Here we cannot but repeat the first sentence, which traditionally opens the vast majority of the isomorphism studies, namely, up to date it remains unclear whether or not is the problem NP-complete.[23]

## Appendix 4

***Algorithm 4***

```
1. Take a next graph $G$ from $G_m$ (the set of all
   non-isomorphic graphs with $m$ vertices).
   Let $W$ be a subset of all vertices $G$ of degree
   less than 4.
2. For $\forall v_i \in W$ do:
   1) add a vertex $v_{m+1}$ and an edge $(v_i, v_{m+1})$ to $G$,
   2) output the resulting graph $G^i_{m+1}$.
3. For $\forall v_j \in W \backslash \{v_i\}$ do:
   1) add an edge $(v_j, v_{m+1})$ to $G^i_{m+1}$.
   2) output the resulting graph $G^j_{m+1}$.
4. For $\forall v_k \in W \backslash \{v_i, v_j\}$ do:
   1) add an edge $(v_k, v_{m+1})$ to $G^j_{m+1}$.
   2) output the resulting graph $G^k_{m+1}$.
5. For $\forall v_l \in W \backslash \{v_i, v_j, v_k\}$ do:
   1) add an edge $(v_l, v_{m+1})$ to $G^k_{m+1}$.
   2) output the resulting graph $G^l_{m+1}$. ■
```

## Appendix 5

***Algorithm 5***

```
1. Assign zero values to all elements of matrix $M = (m_{ij})$
   and the $(m^2 - m)/2$-bit integer variable $u$
2. $u$:=$u$+1.
3. $k$:=0.
4. For $i$ from 1 to $m - 1$ do
5. For $j$ from $i + 1$ to $m$ do
6. $k$:=$k$+1
7. Considering $u$ as a binary number, for bit $t_k$ do:
   $m_{ji} := \begin{cases} 1, & \text{if } t_k = 1 \\ 0, & \text{if } t_k = 0, \end{cases}$
   $m_{ij} := m_{ji}$.
8. End of loop 5.
9. End of loop 4.
10. Output of the matrix $M$.
11. If $m_{ij} = 1$ ($i \neq j$) then exit
    else go to 2. ■
```

## References

1. N. S. Zefirov, M. A. Kirpichenok, F. F. Izmailov, and M. I. Trofimov, *Dokl. Akad. Nauk SSSR*, 1987, **296**, 883 [*Dokl. Chem.*, 1987 (Engl. Transl.)].
2. M. I. Trofimov and E. A. Smolenskii, *Izv. Akad. Nauk. Ser. Khim.*, 2000, 401 [*Russ. Chem. Bull., Int. Ed.*, 2000, **49**, 402].
3. R. T. Sanderson, *Chemical Bonds and Bond Energy*, Acad. Press, New York, 1976, 218 pp.
4. N. S. Zefirov, M. A. Kirpichenok, and M. I. Trofimov, *Dokl. Akad. Nauk SSSR*, 1989, **304**, 887 [*Dokl. Chem.*, 1989 (Engl. Transl.)].
5. M. R. Nelson, *Dr. Dobb´s J.*, May, 1992.
6. A. B. Prolubnikov and R. T. Faizulin, *Matematicheskie struktury i modelirovanie* [*Mathematical Structures and Modelling*], Omsk, Omsk Gos. Univ., 2002, Issue 9, 1 (in Russian).
7. A. B. Prolubnikov and R. T. Faizulin, *Matematicheskie struktury i modelirovanie* [*Mathematical Structures and Modelling*], Omsk, Omsk Gos. Univ., 2003, Issue 11, 28 (in Russian).
8. D. G. Corneil and C. C. Gotlieb, *J. ACM*, 1970, **17**, 51.
9. R. C. Read and D. G. Corneil, *J. Graph Theory*, 1977, **1**, 339.
10. G. Gati, *J. Graph Theory*, 1979, **3**, 95.
11. M. Meringer, *J. Graph Theory*, 1999, **30**, 137.
12. F. Harary, *Graph Theory*, Addison-Wesley, Massachusetts, 1969].
13. A. Kerber, R. Laue, T. Grüner, and M. Meringer, *MATCH*, 1998, **37**, 205.
14. B. D. McKay, *Congressus Numerantium*, 1981, **30**, 45.

15. I. A. Faradzhev, in *Algoritmicheskie issledovaniya v kombinatorike* [*Algorithmic Studies in Combinatorics*], Nauka, Moscow, 1978, 3 (in Russian).
16. *MATCH*, 1992, **27**.
17. *MATCH*, 1998, **37**.
18. M. S. Molchanova, V. V. Shcherbukhin, and N. S. Zefirov, *J. Chem. Inf. Comput. Sci.*, 1996, **36**, 888.
19. D. Bonchev, O. Mekenyan, and N. Trinajstić, *J. Comp. Chem.*, 1981, **2**, 127.
20. M. I. Trofimov, *J. Math. Chem.*, 1991, **8**, 327.
21. W. Lipski, *Kombinatoryka dla Programistow*, Wydawnictwa Naukowo-Techniczne, Warzawa, 1982.
22. S. S. Tratch, M. I. Stankevitch, and N. S. Zefirov, *J. Comp. Chem.*, 1990, **11**, 899.
23. R. Sedgewick, *Algorithms in C. Part 5. Graph Algorithms*, 3rd ed., Addison-Wesley, Boston, 2003.
24. E. A. Smolenskii, *Dokl. Akad. Nauk*, 2001, **380**, 60 [*Dokl. Chem.*, 2001 (Engl. Transl.)].
25. E. A. Smolenskii, *Zh. Vychisl. Matem. Matem. Fiz.* [*J. Comput. Math. Math. Phys.*], 1962, **2**, 371 (in Russian).
26. V. J. Rayward-Smith, *A First Course in Formal Language Theory*, Blackwell Scientific Publications, Oxford, 1983.
27. I. N. Bronshtein and K. A. Semendyaev, *Spravochnik po matematike dlya inzhenerov i uchashchikhsya vuzov* [*Handbook of Mathematics for Engineers and Students*], 13th ed., Nauka, Moscow, 1986, 492, 493 (in Russian).
28. A. A. Zykov, *Osnovy teorii grafov* [*Fundamentals of Graph Theory*], Vuz. Kniga, Moscow, 2004, 9 (in Russian).
29. M. I. Nechepurenko, V. K. Popkov, S. M. Mainagashev, S. B. Kaul´, V. A. Proskuryakov, V. A. Kokhov, and A. B. Gryzunov, *Algoritmy i programmy resheniya zadach na grafakh i setyakh* [*Algorithms and Programs for Solving Problems on Graphs and Networks*], Nauka, Novosibirsk, 1990, 16 (in Russian).
30. D. E. Knuth, *The Art of Computer Programming*, **1**, *Fundamental Algorithms*, Addison-Wesley, Massachusetts, 1968.
31. V. N. Kas´yanov and V. A. Evstigneev, *Grafy v programmirovanii: obrabotka, vizualizatsiya i primenenie* [*Graphs in Programming: Processing, Visualization, and Applications*], BHV-Petersburg, St.-Petersburgh, 2003, 1015 (in Russian).

*Received April 20, 2005;*
*in revised form October 6, 2005*