communications in mathematical
match
and in computer chemistry

32 (1995), 19-25

EXHAUSTIVE GENERATION OF ORGANIC ISOMERS FROM BASE 2 AND BASE 4 NUMBERS

R. Barone, F. Barberis and M. Chanon

Laboratoire AM3 - URA CNRS 1411
Faculté des Sciences de St Jérôme
13397 Marseille cedex 20 - France

ABSTRACT

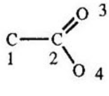
A program which generates organic isomers from arrays of base 2 and base 4 numbers is described. The algorithm is very simple, but slow. It was used to check the result of several programs which gave different results for the generation of isomers.

INTRODUCTION

Enumeration of organic isomers is still a developing field mainly for structure elucidation. The recent Contreras et al.' paper [1] describing their new approach and comparing their results with other approaches showed differences with other programs. These results prompted us to use the approach that we have developed for the invention of new reactions with the purpose of isomers generation [2] in order to decide between different results. With little modification it has been possible to use our program for the exhaustive generation of isomers.

PROGRAM

In this approach, the isomers were generated by using arrays of base 2 and base 4 numbers. If one considers the following compound :



The bond table which describes it is :

Bond	Endpoints		Bond order
1	1	2	1
2	2	3	2
3	2	4	1

Since the value of a bond cannot exceed 3, **the array of the bond orders (1 2 1) may be seen as an array of a base 4 number**. So, the generation of all the isomers consists simply in generating all arrays of base 4 numbers. For doing that, we construct a bond table with all the possible bonds between all the atoms. For example, for the formula C_2O_2 (hydrogens are not counted at this stage of the program) one obtains the following bond table:

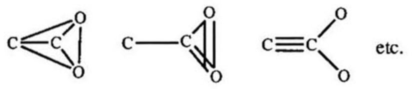
Bond	Endpoints		Bond order
1	1	2	0
2	1	3	0
3	1	4	0
4	2	3	0
5	2	4	0
6	3	4	0

The bond order is set to 0 at the beginning of the program. Then the program generates the arrays of base 4 numbers for the bond order using the following algorithm :

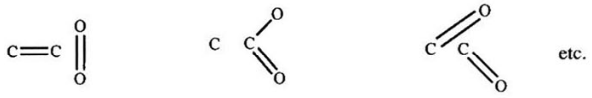
```
Base = 4
Number_of_bond = 6
FOR i = 1 to Number_of_bond
  BOND(i) = 0
NEXT i

DO
  N = Number_of_bond
  DO
    BOND_ORDER(N)=BOND_ORDER(N)+1
    IF BOND_ORDER(N) < BASE THEN
      OK = 1
      ' a new number is generated
```

```
        ' the array BOND contains
        ' the order of each bond
      ELSE
        BOND_ORDER(N) = 0
        N = N -1
        OK = 0
      END IF
    LOOP WHILE OK = 0
  LOOP WHILE N > 0
```

This approach has the disadvantage of generating unrealistic structures such as :



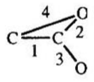
or bond tables including disjoint molecules :



These impossible structures are eliminated by the program which checks each solution in terms of rules of valency and which discards solutions which contain a disconnected structure. It therefore generates a great number of solutions : for 4 atoms, there are 6 bonds and 4^6 (=4096) solutions. For n atoms, the number of bonds is $k = n*(n-1)/2$ and the number of solutions = 4^k. On the other hand, the idea of thinking in terms of base 4 numbers makes the programming so straightforward that it was worth making it operational. The number of solutions increases however very dramatically . For 6 atoms the number of possible bonds is : $k = (6 * 5) / 2 = 15$ and the number of combinations is : $4^{15} = 1.073742 * 10^9$! So, even if the programming is very simple, such an approach is not directly utilizable.To reduce the number of solutions in keeping the same principle, we divide the problem into two steps :

In the first step the program generates only the skeletons of all the isomers : in this case the nature of the bonds may be only 0 or 1, so the bond order array is a base 2 number and the number of generated isomers is "only" 2^k . For 6 bonds the number of generated skeletons decreases to : $2^{15} = 32768$. Each skeleton is analyzed and bad solutions (valence violation, disconnected molecule, incorrect number of hydrogens (see below)) are rejected. In this step the nature of the atoms is defined in the skeleton.

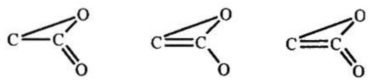
In the second step, for each saved skeleton the program generates the corresponding isomers by computing the base 4 numbers. This step generates final isomers with double and triple bonds. But not all the base 4 numbers are needed because in this case the skeleton is defined and it is possible to calculate for each bond its maximum order. For example, let us take the following skeleton generated during the first step :



At the beginning the binary number describing the bond array of the substructure is 1 1 1 1. We may calculate the maximum order for each bond, for example the connectivity of the central carbon is 3, so the bond between the two carbon atoms (bond 1) cannot exceed 2. We obtain :

Bond number :	1 2 3 4
Maximum bond order :	2 1 2 1

And the only isomers which are generated, besides 1 1 1 1, are : 1 1 2 1, 2 1 1 1, 2 1 2 1, that is :



Finally the different steps of the program are :

1 : Input of the formula. The user draws the skeleton of a structure having the right formula. The hydrogen atoms are not taken into account. From the drawing the program generates the bond table and the array of bond orders is set to zero. Then the computer asks the user to input the number of hydrogens (NBH).

2 : Generation of base 2 numbers from the algorithm given above. Each corresponding connectivity table is analyzed, 3 tests are worked out , a solution is rejected if :

a) The valence of an atom is wrong.

b) The number of molecules in the bond table > 1.

c) The number of H is incorrect (From the skeleton the program adds hydrogens to complete the valency of each atome and calcultates the total number of hydrogens (TH) for this saturated structure, if TH < NBH then the solution is rejected).

Since identical structures may be generated, one more step is needed for the elimination of duplicate structures. To do that, each structure is canonicalized by an algorithm described by Moreau [3].

3 : From the skeletons created in the previous step the base 4 numbers are generated. In this step again identical strcutures may be generated and the elimination of duplicate structures is performed.

To save time all the isomers are kept in dynamic memory.

RESULTS

The number of isomers found by CAMGEC, the program of Contreras' group, is different from the other approaches (AEGIS [4], DENDRAL [5], ASSEMBLE [6], GEN [7], MOLGRAPH [8]), for two formulas : C_4H_7NO and $C_6H_{10}O$. For C_4H_7NO CAMGEC generated 767 isomers whereas the other programs generated 764 isomers and for $C_6H_{10}O$ CAMGEC generated 748 isomers whereas the other programs generated 747 isomers.

Our program, called GI (Generator of Isomers), does not confirm the results of CAMGEC. It confirms the results of the other programs : 764 isomers for C_4H_7NO and 747 isomers for $C_6H_{10}O$.

The analysis of the characteristics of the isomers shows where the differences take place :

Structure	C4H7NO		C6H10O	
System	CAMGEC	GI	CAMGEC	GI
Two double bonds	174	174	151	151
One double bond + one ring	387	387	375	375
One triple bond	60	60	47	47
Two rings	**146**	**143**	**175**	**174**
Total	**767**	**764**	**748**	**747**

The aim of our approach was to decide between different programs because of its simplicity and its "fool proof" character. Our program cannot be used to generate isomers for large structures. This blind, brute force approach is very time consuming. The program was developed in Microsoft Basic 7 for a 386 IBM/PC.

For 6 atoms other than hydrogen (C_4H_7NO) the 32768 base 2 numbers are generated in 23 minutes. Among them only 6426 are valid (one structure and the valence of atoms and the number of H are correct). Elimination of duplicate structures leads to 328 unique skeletons.

The 328th isomer is generated for the 19768th combination. The second part of the program is faster : from the 328 skeletons, GI generates 7408 structures in 3 minutes. Among them 807 are valid and 764 are unique.

For $C_6H_{10}O$ (7 atoms : 21 possible bonds)) the time needed to generate the 2^{21} = 2 097 152 combinations was 21 hours. Among them only 308 were unique. From the 308 unique skeletons, part two of the program generated the 747 isomers in 6 minutes. The base 4 generator created 22539 structures, among them 849 are valid and 747 are unique.

In order to improve the program we added a simple test to avoid the generation of all the combinations. The program takes into account the valence of the atoms. Atom number 1 is given to O : this atom must have only one or two bonds. The connectivity table which is generated at the beginning of the program looks like :

Bond	Endpoints	
1	1	2
2	1	3
3	1	4
4	1	5
5	1	6
6	1	7

etc.

The six first digits of the binary number are at the beginning : 0 0 0 0 0 0. These digits must contain only one or two 1. A number like 0 0 1 1 0 1 is forbidden, and all the numbers which could be generated from this one are not generated. With this improvment "only" 680 076 structures are generated instead of the 2 097 152. Among them 147 308 are valid and 308 unique. The time decreases to 13 hours and 26 minutes; it is still too large for making the program operational for large structures.

Recently Hendrickson described an approach to generate carbon skeletons. It was designed to create rigorously all possible adjacency matrices by generating all one/zero entries for each successive row [9] . This approach is similar to the one used in GI, but the algorithm is different. Rules are used to avoid the generation of all the combinations. This approach could be generalized to skeletons with heteroatoms. Coupled with the base 4 part of our program it could then produce a fast program to generate isomers.

CONCLUSION

In order to test several programs which differ in the enumeration of isomers, we used a very simple program to create all the isomers from a molecular formula by generating base 2 and base 4 numbers. This approach is very simple and safe. It allowed us to confirm the results of other programs. It is very simple but very slow because it generates a great number of combinations. It could be improved by using several rules avoiding the generation of all the combinations. Nevertheless, in spite of its slowness this approach is interesting because it can be easily programmed and it can be used for medium sized structures. It could offer new insights in this field or in related ones as shown by the recent Elk's report describing a simplified algorithm using base 5 numbers to assign canonical names to cata-condensed polybenzenes [10].

REFERENCES

1. M.L Contreras, R. Rozas and R. Valdivia, *J. Chem. Inf. Comput. Sci.*, 1994, **34**, 610-616.
2. R. Barone, M. Arbelot, A. Baldy, M. Chanon, R. Gallo, *Rev. Roumaine Chim.*, 1991, **36**, 581-598.
3. G. Moreau, *Nouv. J. Chim.*, 1980, **4**, 17.
4. H.J. Luinge, *MATCH*, 1992, **27**, 175-189.
5. R.K. Lindsay, B.G. Buchanan, E.A. Feigenbaum, J. Lederberg, *Application of artificial intelligence for organic chemistry,* McGraw-Hill, New York, 1980.
6. C.A. Shelley, M.E. Munk, *Anal. Chim. Acta*, 1981, **133**, 507-516.
7. S. Bohanec, J. Zupan, *J. Chem. Inf. Comput. Sci.*, 1991, **31**, 531-540.
8. A. Kerber, R. Lane, D. Mose, *Anal. Chim. Acta*, 1990, **235**, 221-228.
9. J. B. Hendrickson, C. A. Parks, *J. Chem. Inf. Comput. Sci.*, 1991, **31**, 101-107.
10. S. B. Elk, *J. Chem. Inf. Comput. Sci.*, 1994, **34**, 637-640.